JBoss Enterprise Application Platform 6.3

Guide de sécurité

À utiliser dans Red Hat JBoss Enterprise Application Platform 6

Last Updated: 2017-10-27

JBoss Enterprise Application Platform 6.3 Guide de sécurité

À utiliser dans Red Hat JBoss Enterprise Application Platform 6

Notice légale

Résumé

Cet ouvrage est un guide pour sécuriser Red Hat JBoss Enterprise Application Platform 6, qui inclut des correctifs publiés.

Table des matières

PARTIE I. SÉCURITÉ DANS RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM 6

CHAPITRE 1. INTRODUCTION

1.1. RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM 6

Red Hat JBoss Enterprise Application Platform 6 (JBoss EAP 6) est une plate-forme middleware construite sur la base de standards ouverts et compatibles avec Java Enterprise Edition 6. Elle intègre JBoss Application Server 7 avec un clustering de haute disponibilité, une messagerie, une mise en cache distribuée et autres technologies.

JBoss EAP 6 comprend une nouvelle structure modulaire qui permet aux services d'être activés seulement si nécessaire, améliorant ainsi la vitesse de démarrage.

La console de gestion et l'interface CLI rendent la modification des fichiers de configuration XML inutile et rajoutent la capacité d'encoder et d'automatiser des tâches.

En plus, JBoss EAP 6 comprend des frameworks de développement et des API pour développer rapidement des applications de Java EE sécurisées et évolutives.

Rapporter un bogue

1.2. SÉCURISER JBOSS ENTERPRISE APPLICATION PLATFORM 6

La Sécurité informatique est le terme général donné au domaine de technologies d'informations chargées de sécuriser les environnements virtuels à l'ère du numérique. Cela peut inclure la protection et l'intégrité des données, la sécurité des applications, l'évaluation du risque et de la vulnérabilité, et les protocoles d'authentification et d'autorisation.

Les données informatiques représentent un atout important pour la plupart des organisations. La protection des données est essentielle pour la plupart des entreprises. JBoss EAP fournit une approche multi couches de sécurité pour s'occuper des données à tous les niveaux.

Les systèmes vraiment sécurisés sont ceux conçus avec la sécurité comme principale caractéristique de haut en bas. Ces systèmes utilisent le principe de Sécurité by Design. Dans ces systèmes, les infiltrations et les attaques malveillantes sont acceptées comme partie intégrante de l'apparatus de sécurité normal et les systèmes sont conçus pour les contourner.

La sécurité peut être appliquée au niveau système d'exploitation, middleware ou application. Pour plus d'informations sur la sécurité au niveau du système d'exploitation telle qu'elle s'applique à RHEL, reportez-vous au document Red Hat Enterprise Linux Security Guide.

Dans les prochains chapitres, vous lirez des informations sur les différents niveaux et couches de sécurité de JBoss EAP 6. Ces couches fournissent l'infrastructure pour toutes les fonctionnalités de sécurité de la plate-forme.

Rapporter un bogue

PARTIE II. SÉCURISER LA PLATE-FORME

CHAPITRE 2. JAVA SECURITY MANAGER

2.1. JAVA SECURITY MANAGER

Java Security Manager

> Java Security Manager est une classe qui gère la limite extérieure de la sandbox Java Virtual Machine (JVM), contrôlant ainsi comment le code exécuté dans la JVM peut interagir avec les ressources extérieures à la machine virtuelle Java. Lorsque le gestionnaire de sécurité Java est activé, l'API Java vérifie avec le gestionnaire de sécurité pour approbation avant d'exécuter une vaste gamme d'opérations potentiellement dangereuses.

Le Java Security Manager utilise une police de sécurité pour déterminer si une action sera permise ou refusée.

Rapporter un bogue

2.2. POLICES DU JAVA SECURITY MANAGER

Security Policy

> Un jeu d'autorisations définies pour les différentes classes du code. Le Java Security Manager examine les requêtes en provenance des applications en fonction de la stratégie de sécurité. Si une action est autorisée par la stratégie, le Java Security Manager permettra que l'action ait lieu. Si l'action n'est pas autorisée par la stratégie, le Java Security Manager refusera cette action. La stratégie de sécurité peut définir des autorisations basées sur l'emplacement du code ou sur les principaux du sujet.

Le Java Security Manager et la stratégie de sécurité utilisés sont configurés à l'aide des options Java Virtual Machine **`java.security.manager`** et **`java.security.policy`**.

Informations de base

Une entrée de police de sécurité consiste en les éléments de configuration suivants connectés au **`policytool`** :

CodeBase

> L'emplacement de l'URL (à l'exclusion des informations sur l'hôte ou le domaine) d'où viennent les codes. Ce paramètre est en option.

SignedBy

> L'alias est utilisé dans le fichier de clés pour référencer le signataire dont la clé privée a été utilisée pour signer le code. Cela peut correspondre à une valeur unique ou à une liste séparée par des virgules. Ce paramètre est facultatif. Si omis, la présence ou l'absence de signature n'a aucun impact sur le Java Security Manager.

Principaux

> Une liste de paires ***`principal_type`***/***`principal_name`***, qui doivent être présentes au sein de l'ensemble principal du thread en cours d'exécution. L'entrée de principaux est facultative. Si elle est est omise, cela signifie que les principaux du thread en cours n'auront aucun impact sur le Java Security Manager.

Permissions

Une permission est l'accès qui est accordé au code. De nombreuses autorisations sont fournies dans le cadre de la spécification Java Enterprise Edition 6 (Java EE 6).

Rapporter un bogue

2.3. EXÉCUTER JBOSS EAP 6 DANS LE JAVA SECURITY MANAGER

Pour spécifier une stratégie Java Security Manager, vous devez modifier les options Java transmises à l'instance de serveur ou de domaine lors du processus d'amorçage. Pour cette raison, vous ne pouvez passer les paramètres en option aux scripts **`domain.sh`** ou **`standalone.sh`**. La procédure suivante va vous guider à travers les étapes de configuration de votre instance pour exécuter au sein d'une stratégie Java Security Manager.

Pré-requis

- Avant de suivre cette procédure, vous devrez rédiger une stratégie de sécurité, en utilisant la commande **`policytool`** comprise dans votre Java Development Kit (JDK). Cette procédure assume que votre stratégie se trouve dans ***`EAP_HOME`*`/bin/server.policy`**. Sinon, vous pouvez écrire la stratégie de sécurité à l'aide 'un éditeur de texte et la sauvegarder comme ***`EAP_HOME`*`/bin/server.policy`**.
- Le domaine ou le serveur autonome doivent être tout à fait arrêtés avant d'éditer un fichier de configuration quelconque.

Procéder à la procédure suivante pour chaque hôte physique ou pour chaque instance de votre domaine, si vous avez des membres de domaine éparpillés dans des systèmes multiples.

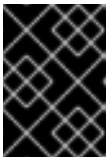
Procédure 2.1. Configurer le gestionnaire de sécurité dans JBoss EAP 6

1. **Ouvrir le fichier de configuration.**
   Ouvrir le fichier de configuration pour le modifier. Ce fichier se trouve dans un de ces emplacements, suivant que vous utilisiez un domaine géré ou un serveur autonome. Il ne s'agit pas du fichier exécutable utilisé pour démarrer le serveur ou le domaine.
   - **Domaine géré**
     - Dans Linux: ***`EAP_HOME`*`/bin/domain.conf`**
     - Dans Windows: ***`EAP_HOME`*`\bin\domain.conf.bat`**
   - **Serveur autonome**
     - Dans Linux: ***`EAP_HOME`*`/bin/standalone.conf`**
     - Dans Windows: ***`EAP_HOME`*`\bin\standalone.conf.bat`**
2. **Ajouter les options Java au fichier.**
   Pour s'assurer que les options Java soient bien utilisées, ajouter les au blic de code qui commence par :

   ```
   if [ "x$JAVA_OPTS" = "x" ]; then
   ```

   Vous pouvez modifier la valeur de **`-Djava.security.policy`** pour indiquer l'emplacement

exact de votre police de sécurité. Cela ne doit prendre qu'une seule ligne, sans saut de ligne. Utiliser **==** quand vous définissez la propriété **`-Djava.security.policy`** indique que le gestionnaire de sécurité utilisera *uniquement* le fichier de police spécifié. Utiliser **=** indique que le gestionnaire de sécurité utilisera la police spécifiée *en combinaison à* la police définie dans la section **`policy.url`** de ***`JAVA_HOME`*`/lib/security/java.security`**.



IMPORTANT

Les versions de JBoss Enterprise Application Platform à partir de 6.2.2 requièrent que la propriété système **`jboss.modules.policy-permissions`** soit sur *true*.

Exemple 2.1. domain.conf

```
JAVA_OPTS="$JAVA_OPTS -Djava.security.manager -
Djava.security.policy==$PWD/server.policy -
Djboss.home.dir=/path/to/EAP_HOME -Djboss.modules.policy-
permissions=true"
```

Exemple 2.2. domain.conf.bat

```
set "JAVA_OPTS=%JAVA_OPTS% -Djava.security.manager -
Djava.security.policy==\path\to\server.policy -
Djboss.home.dir=\path\to\EAP_HOME -Djboss.modules.policy-
permissions=true"
```

Exemple 2.3. standalone.conf

```
JAVA_OPTS="$JAVA_OPTS -Djava.security.manager -
Djava.security.policy==$PWD/server.policy -
Djboss.home.dir=$JBOSS_HOME -Djboss.modules.policy-
permissions=true"
```

Exemple 2.4. standalone.conf.bat

```
set "JAVA_OPTS=%JAVA_OPTS% -Djava.security.manager -
Djava.security.policy==\path\to\server.policy -
Djboss.home.dir=%JBOSS_HOME% -Djboss.modules.policy-
permissions=true"
```

3. **Démarrer le domaine ou le serveur.**
   Démarrer le domaine ou le serveur en tant que normal.

Rapporter un bogue

2.4. ÉCRIRE UNE STRATÉGIE POUR LE JAVA SECURITY MANAGER

Introduction

Introduction

Il y a une application nommée **policytool** dans la plupart des distributions JDK et JRE, ayant pour but la modification ou la création de polices de sécurité pour le Java Security Manager. Vous trouverez des informations sur **policytool** dans http://docs.oracle.com/javase/6/docs/technotes/tools/.

Procédure 2.2. Définir une stratégie pour le Java Security Manager

1. **Démarrez policytool.**
   Démarrez l'outil **policytool** d'une des façons suivantes.
   - **Red Hat Enterprise Linux**
     À partir de votre GUI ou invite de commande, exécutez **/usr/bin/policytool**.
   - **Microsoft Windows Server**
     Exécutez **policytool.exe** à partir du menu de Démarrage (Start) ou à partir de **bin** de votre installation Java. L'emplacement peut varier.
2. **Créer une stratégie.**
   Pour créer une stratégie, sélectionnez **Add Policy Entry**. Ajouter les paramètres dont vous aurez besoin, et cliquez sur **Done**.
3. **Modifier une stratégie existante**
   Sélectionnez la police à partir d'une liste de stratégies existantes, et sélectionnez le bouton **Edit Policy Entry**. Modifiez les paramètres suivant les besoins.
4. **Supprimer une stratégie existante.**
   Sélectionnez la stratégie à partir d'une liste de stratégies existantes, et sélectionnez le bouton **Remove Policy Entry**.

Rapporter un bogue

2.5. IBM JRE ET JAVA SECURITY MANAGER

IBM JRE utilise un fournisseur de stratégies par défaut, qui ne fonctionne pas correctement avec la stratégie de sécurité de la plateforme JBoss Enterprise Application Platform. Vous devez modifier la configuration pour utiliser le fournisseur de stratégies standard, si vous souhaitez utiliser IBM JRE pour héberger la JBoss Enterprise Application Platform avec le Java Security Manager actif.

Pour configurer la configuration JRE, pour IBM JRE, modifez le fichier **JAVA_HOME/jre/lib/security/java.security**, et définir la valeur de **policy.provider** à **sun.security.provider.PolicyFile**.

```
policy.provider=sun.security.provider.PolicyFile
```

Rapporter un bogue

2.6. DÉBOGAGE DES STRATÉGIES DU GESTIONNAIRE DE SÉCURITÉ

Vous pouvez activer les informations de débogage pour vous aider à résoudre les problèmes liés aux stratégies de sécurité. L'option **java.security.debug** configure le niveau des informations liées à la sécurité qui ont été reportées. La commande **java -Djava.security.debug=help** vous produira de

l'aide avec l'ensemble complet des options de débogage. Définir le niveau de débogage à **`all`** est utile quand on résout un échec lié à la sécurité dont la cause est inconnue, mais en général, cela produit trop d'informations. Une valeur par défaut utile et raisonnable est **`access:failure`**.

Procédure 2.3. Activer le débogage général

- **Cette procédure permettra un bon niveau de sécurité général pour les informations de débogage liées à la sécurité.**
  Ajouter la ligne suivante au fichier de configuration du serveur.
  - Si l'instance de JBoss EAP 6 exécute sur une domaine géré, la ligne sera ajoutée au fichier **`bin/domain.conf`** de Linux ou au fichier **`bin\domain.conf.bat`** de Windows.
  - Si l'instance de JBoss EAP exécute sur un domaine autonome, la ligne sera ajoutée au fichier **`bin/standalone.conf`** de Linux ou au fichier **`bin/standalone.conf.bat`** de Windows.

`Linux`

```
JAVA_OPTS="$JAVA_OPTS -Djava.security.debug=access:failure"
```

`Windows`

```
set "JAVA_OPTS=%JAVA_OPTS% -Djava.security.debug=access:failure"
```

Résultat

Un niveau général d'informations de débogage lié à la sécurité a été activé.

Rapporter un bogue

CHAPITRE 3. DOMAINES DE SÉCURITÉ

3.1. DOMAINES DE SÉCURITÉ

Un *domaine de sécurité* est une série de mappages entre les utilisateurs et les mots de passe, les utilisateurs et les rôles. Les domaines de sécurité représentent un mécanisme permettant d'ajouter l'authentification et l'autorisation à vos applications Web et EJB. JBoss EAP 6 fournit deux domaines de sécurité par défaut :

- **`ManagementRealm`** stocke les informations d'authentification pour l'API de gestion, qui fournit les fonctionnalités pour l'interface CLI et la Console de gestion sur le web. Il fournit un système d'authentification pour gérer JBoss EAP 6 . Vous pouvez également utiliser le **`ManagementRealm`** si votre application a besoin des mêmes règles commerciales que vous utilisez pour l'API de gestion, lors de son authentification.
- **`ApplicationRealm`** stocke l'utilisateur, le mot de passe et les informations de rôle pour les applications Web et les EJB.

Chaque domaine est stocké dans un certain nopmbre de fichiers du système de fichiers :

- ***`REALM`*`-users.properties`** stocke les mots de passe et les mots de passe hachés.
- ***`REALM`*`-roles.properties`** stocke les mappages user-to-role.
- **`mgmt-groups.properties`** stocke le fichier de mappage user-to-group pour **`ManagementRealm`**. Utilisé uniquement quand RBAC (Role-based Access Control) est activé.

Les fichiers de propriété sont stockés dans les répertoires **`domain/configuration/`** et **`standalone/configuration/`**. Les fichiers sont inscrits simultanément par la commande **`add-user.sh`** ou **`add-user.bat`**. Quand vous exécutez la commande, la première décision est de décider dans quel domaine ajouter votre premier utilisateur.

Rapporter un bogue

3.2. AJOUT D'UN DOMAINE DE SÉCURITÉ

1. **Exécuter l'interface CLI**
   Démarrez par la commande **`jboss-cli.sh`** ou **`jboss-cli.bat`** et connectez-vous au serveur.

2. **Créer le nouveau domaine de sécurité lui-même.**
   Exécutez la commande suivante pour créer un nouveau domaine de sécurité nommé **`MyDomainRealm`** sur un contrôleur de domaine ou sur un serveur autonome.

   ```
   /host=master/core-service=management/security-
   realm=MyDomainRealm:add()
   ```

3. **Créer les références du fichier de propriétés qui stocke les informations sur le nouveau rôle.**
   Exécutez la commande suivante pour créer un pointeur au fichier nommé **`myfile.properties`**, qui contiendra les propriétés attachées au nouveau rôle.

NOTE

Le fichier de propriétés nouvellement créées n'est pas géré par les scripts **add-user.sh** et **add-user.bat** inclus. Il devra être administré en externe.

```
/host=master/core-service=management/security-
realm=MyDomainRealm/authentication=properties:add(path=myfile.proper
ties)
```

Résultat

Votre nouveau domaine de sécurité sera créé. Lorsque vous ajoutez des utilisateurs et des rôles à ce nouveau domaine, l'information va être stockée dans un fichier séparé des domaines de sécurité par défaut. Vous pouvez gérer ce nouveau fichier à l'aide de vos propres applications ou procédures.

Rapporter un bogue

3.3. AJOUT D'UN UTILISATEUR À UN DOMAINE DE SÉCURITÉ

1. **Éxécuter la commande `add-user.sh` ou `add-user.bat`.**
   Ouvrez un terminal (shell) et changez de répertoire ***`EAP_HOME`*`/bin/`**. Si vous exécutez Red Hat Enterprise Linux ou un autre système d'exploitation style UNIX, exécutez **`add-user.sh`**. Si vous exécutez sur un serveur Microsoft Windows, exécutez **`add-user.bat`**.

2. **Choisir d'ajouter un utilisateur de gestion ou un utilisateur d'application.**
   Pour cette procédure, saisir **b** pour ajouter un utilisateur d'application.

3. **Choisir le domaine dans lequel l'utilisateur sera ajouté.**
   Par défaut, le seul domaine disponible est **`ApplicationRealm`**. Si vous avez ajouté un domaine personnalisé, vous pouvez saisir son nom à la place.

4. **Saisir le nom d'utilisateur, le mot de passe et les rôles lorsque vous y serez invité.**
   Saisir le nom d'utilisateur, le mot de passe et les rôles lorsque vous y serez invité. Vérifiez votre choix en tapant **`yes`**, ou **no** pour annuler les changements. Les changements sont inscrits dans les fichiers de propriétés du domaine de sécurité.

Rapporter un bogue

CHAPITRE 4. ENCODER LE TRAFFIC RÉSEAU

4.1. INDIQUER L'INTERFACE DE RÉSEAU QUE JBOSS EAP 6 UTILISE

Aperçu

Isolez les services afin qu'ils soient accessibles uniquement aux clients qui en ont besoin augmente la sécurité de votre réseau. JBoss EAP 6 comprend deux interfaces dans sa configuration par défaut, qui se lient à l'IP adresse **`127.0.0.1`** ou **`localhost`**, par défaut. Une des interfaces est appelée **`gestion`** et est utilisée par la Console de gestion, CLI et API. L'autre est appelé **`public`** et est utilisé pour déployer des applications. Ces interfaces ne sont pas spéciales ou importantes, mais sont fournies comme point de départ.

L'interface **`management`** utilise les ports **`9990`** et **`9999`** par défaut, et l'interface **`public`** utilise le port **`8080`**, ou le port **`8443`** avec HTTPS.

Vous pouvez changer l'adresse IP de l'interface de gestion, de l'interface publique ou bien les deux à la fois.

AVERTISSEMENT

Si vous exposez les interfaces de gestion à d'autres interfaces de réseau non accessibles par les hôtes distants, vérifiez les implications au niveau de la sécurité. Le plus souvent, il n'est pas conseillé de donner un accès à distance aux interfaces de gestion.

1. **Stopper le serveur JBoss EAP 6**
   Stoppez JBoss EAP 6 en envoyant une interruption de manière appropriée à votre système d'exploitation. Si vous exécutez JBoss EAP 6 comme une application de premier plan, il suffit d'appuyer sur **`Ctrl+C`**.

2. **Démarrer JBoss EAP 6 à nouveau, en spécifiant l'adresse de liaison.**
   Utilisez l'argument de ligne de commande **`-b`** pour démarrer JBoss EAP 6 sur une interface particulière.

   Exemple 4.1. Indiquez l'interface publique.

   ```
   EAP_HOME/bin/domain.sh -b 10.1.1.1
   ```

   Exemple 4.2. Indiquez l'interface de gestion.

   ```
   EAP_HOME/bin/domain.sh -bmanagement=10.1.1.1
   ```

   Exemple 4.3. Indiquez des adresses différentes pour chaque interface.

```
EAP_HOME/bin/domain.sh -bmanagement=127.0.0.1 -b 10.1.1.1
```

Exemple 4.4. Liez l'interface publique à toutes les interfaces de réseau.

```
EAP_HOME/bin/domain.sh -b 0.0.0.0
```

Il est possible de modifier votre fichier de configuration XML directement, pour changer les adresses de liaison par défaut. Toutefois, si vous faites cela, vous ne serez plus en mesure d'utiliser l'argument de ligne de commande **-b** pour spécifier une adresse IP en cours d'exécution, donc ce n'est pas recommandé. Si vous décidez de le faire, n'oubliez pas de stopper JBoss EAP 6 complètement avant d'éditer le fichier XML.

Rapporter un bogue

4.2. CONFIGURER LES PARE-FEUX DE RÉSEAU POUR QU'ILS FONCTIONNENT DANS JBOSS EAP 6

Résumé

La plupart des environnements de production utilisent des pare-feux dans le cadre d'une stratégie globale de sécurité réseau. Si vous avez besoin que plusieurs instances de serveur puissent communiquer entre eux ou avec des services externes tels que des serveurs web ou des bases de données, votre pare-feu doit en tenir compte. Un pare-feu bien géré ouvre uniquement les ports qui sont nécessaires à l'opération et limite l'accès aux ports pour des adresses IP, des sous-réseaux et protocoles réseau spécifiques.

Une discussion plus complète sur les pare-feux est au delà du dessein de cette documentation.

Pré-requis

- Déterminez les ports que vous souhaitez ouvrir.
- Vous devez avoir une bonne compréhension de vos logiciels de pare-feux. Cette procédure utilise la commande **system-config-firewall** de Red Hat Enterprise Linux 6. Microsoft Windows Server inclut un pare-feu intégré, et plusieurs solutions de pare-feux de tierce partie existent pour chaque plate-forme. Sur un serveur Microsoft Windows, vous pouvez utiliser PowerShell pour configurer le pare-feu.

Hypothèses

Cette procédure configure un pare-feu dans un environnement qui comprend les hypothèses suivantes :

- Le système d'exploitation est Red Hat Enterprise Linux 6
- JBoss EAP 6 exécute sur l'hôte **10.1.1.2**. En option, le serveur peut avoir son propre pare-feu.
- Le serveur du pare-feu de réseau exécute sur l'hôte **10.1.1.1** sur l'interface **eth0**, et possède une interface externe **eth1**.
- Le trafic de réseau du port **5445** (port utilisé par JMS) devra être renvoyé sur JBoss EAP 6. Aucun autre trafic doit pouvoir transiter par le pare-feu du réseau.

Procédure 4.1. Gérer les pare-feux de réseau pour qu'ils soient opérationnels dans JBoss EAP 6

1. **Connectez-vous à la Console de management.**
   Connectez-vous dans la Console de gestion. Par défaut, elle exécute sur http://localhost:9990/console/.

2. **Déterminez les liaisons de socket utilisées par le groupe de liaisons de socket.**

   a. Cliquez sur l'étiquette **`Configuration`** qui se trouve en haut de la console de gestion.

   b. Étendre le menu **`General Configuration`**. Sélectionnez **`Socket Binding`** (liaison de sockets).

   c. L'écran **`Socket Binding Declarations`** apparaît. Au départ, le groupe **`standard-sockets`** apparaît. Choisissez un autre groupe en le sélectionnant à partir de la case de combo sur la droite.

   NOTE

   Si vous utilisez un serveur autonome, il ne possédera qu'un seul groupe de liaisons de socket.

   La liste de noms de sockets et des ports apparaît, avec huit valeurs par page. Vous pourrez naviguer entre les pages grâce à la flèche de navigation en dessous du tableau.

3. **Déterminez les ports que vous souhaitez ouvrir.**
   Suivant la fonction d'un port particulier, et suivant les besoins de votre environnement, certains ports devront sans doute être ouverts sur votre pare-feu.

4. **Configurez votre pare-feu pour redirigez le trafic réseau vers la plateforme JBoss EAP 6.**
   Procédez à ces étapes de configuration de votre pare-feu de réseau pour permettre au trafic de se diriger vers le port désiré.

   a. Connectez-vous au pare-feu de votre machine, et accéder à cette commande, en tant qu'utilisateur root.

   b. Saisir la commande **`system-config-firewall`** pour lancer l'utilitaire de configuration du pare-feu. Un GUI ou Utilitaire de ligne de commande opérera selon la façon dont vous êtes connecté au système de pare-feu. Cette tâche assume que vous êtes connecté via SSH et que vous utilisez l'interface de ligne de commande.

   c. Utiliser la clé **`TAB`** de votre clavier pour naviguer vers le bouton **`Customize`**, puis appuyer sur la clé **`ENTER`**. L'écran**`Trusted Services`** apparaîtra.

   d. Ne changez aucune valeur, mais utilisez la clé **`TAB`** pour naviguer vers le bouton **`Forward`**, puis, appuyer sur **`ENTER`** pour aller vers le prochain écran. L'écran **`Other Ports`** apparaîtra.

   e. Utiliser la clé **`TAB`** pour naviguer vers le bouton **`<Add>`**, puis appuyer sur la clé **`ENTER`**. L'écran **`Port and Protocol`** apparaîtra.

   f. Saisir **`5445`** dans le champ **`Port / Port Range`**, puis utiliser la clé **`TAB`** pour vous rendre dans le champ **`Protocol`**, puis saisir **`tcp`**. Utiliser la clé **`TAB`** pour naviguer vers le bouton **`OK`**, puis appuyer sur **`ENTER`**.

g. Utiliser la clé **TAB** pour naviguer vers le bouton **Forward** jusqu'à atteindre l'écran **Port Forwarding**.

h. Utiliser la clé **TAB** pour naviguer vers le bouton **<Add>**, puis appuyer sur la clé **ENTER**.

i. Remplir les valeurs suivantes pour définir la redirection de port vers port **5445**.

- Interface source : **eth1**
- Protocole : **tcp**
- Port / Plage de port : **5445**
- Destination IP address : **10.1.1.2**
- Port / Plage de port : **5445**

Utiliser la clé **TAB** pour naviguer vers le bouton **OK**, puis appuyer sur la clé **ENTER**.

j. Utiliser la clé **TAB** pour naviguer vers le bouton **Close**, puis appuyer sur la clé **ENTER**.

k. Utiliser la clé **TAB** pour naviguer vers le bouton **OK**, puis appuyer sur **ENTER**. Pour appliquer les changements, lire la notice d'avertissement, puis appuyer sur **Yes**.

5. **Configurer un pare-feu sur votre hôte de plateforme JBoss EAP 6.**
Certaines organisations choisissent de configurer un pare-feu sur le serveur JBoss EAP 6 lui-même et de fermer tous les ports qui ne sont pas utiles à son fonctionnement. Voir Section 4.3, « Ports de réseau utilisés par JBoss EAP 6 » pour déterminer quels ports ouvrir, puis fermer le reste. La configuration par défaut de Red Hat Enterprise Linux 6 ferme tous les ports sauf **22** (utilisé pour Secure Shell (SSH) et **5353** (utilisé pour la multi-diffusion DNS). Si vous configurez les ports, assurez-vous que vous avez un accès physique à votre serveur pour ne pas, par inadvertance, vous verrouiller vous-même.

Résultat

Votre pare-feu est configuré pour renvoyer le trafic vers votre serveur JBoss EAP 6 interne, de la façon dont vous avez spécifié dans la configuration de votre pare-feu. Si vous avez choisi d'activer un pare-feu sur votre serveur JBoss Enterprise Application Platform 6, tous les ports seront fermés sauf ceux nécessaires à l'exécution de vos applications.

Procédure 4.2. Configurer le pare-feu dans Microsoft Windows avec PowerShell

1. Désactiver le pare-feu pour déterminer si le comportement de réseau actuel est lié à la configuration du pare-feu à des fins de débogage.

```
Start-Process "$psHome\powershell.exe" -Verb Runas -ArgumentList '-
command "NetSh Advfirewall set allprofiles state off"'
```

2. Autoriser les connexions UDP sur le port 23364. Par exemple :

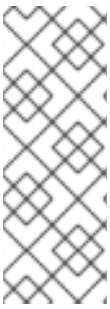
```
Start-Process "$psHome\powershell.exe" -Verb Runas -ArgumentList '-
command "NetSh Advfirewall firewall add rule name="UDP Port 23364"
dir=in  action=allow protocol=UDP localport=23364"'
Start-Process "$psHome\powershell.exe" -Verb Runas -ArgumentList '-
command "NetSh Advfirewall firewall add rule name="UDP Port 23364"
dir=out action=allow protocol=UDP localport=23364"'
```

Rapporter un bogue

4.3. PORTS DE RÉSEAU UTILISÉS PAR JBOSS EAP 6

Les ports utilisés par la configuration JBoss EAP 6 par défaut sont liés à plusieurs facteurs :

- Le fait que vos groupes de serveurs utilisent le groupe de liaisons de sockets par défaut, ou un groupe de liaisons de sockets personnalisé.
- Les exigences de vos déploiements individuels.



NOTE

Un décalage de port numérique peut être configuré pour atténuer les conflits de ports lorsque vous exécutez plusieurs serveurs sur un même serveur physique. Si votre serveur utilise un décalage de port numérique, ajoutez la valeur de décalage au numéro de port par défaut pour le groupe de liaisons de sockets de son groupe de serveurs. Par exemple, si le port HTTP du groupe de liaisons de socket est **`8080`** et si votre serveur utilise un décalage de port de **`100`**, son port HTTP sera **`8180`**.

À moins d'instruction particulière, les ports utilisent le protocole TCP.

Groupes de liaison de sockets par défaut

- **`full-ha-sockets`**
- **`full-sockets`**
- **`ha-sockets`**
- **`standard-sockets`**

Tableau 4.1. Référence aux Groupes de liaisons de sockets par défaut

Nom	Port	Port multi-diffusion	Description	full-ha-sockets	full-sockets	ha-socket	standard-socket
`ajp`	8009		Protocole Apache JServ. Utilisé pour le clustering HTTP et pour l'équilibrage des charges.	Oui	Oui	Oui	Oui
`http`	8080		Le port par défaut des applications déployées.	Oui	Oui	Oui	Oui

Nom	Port	Port multi-diffusion	Description	full-ha-sockets	full-sockets	ha-socket	standard-socket
https	8443		Connexion cryptée-SSL entre les applications déployées et les clients.	Oui	Oui	Oui	Oui
jacorb	3528		Services CORBA pour les transactions JTS et autres services dépendants-ORB.	Oui	Oui	Non	Non
jacorb-ssl	3529		Services CORBA cryptés-SSL.	Oui	Oui	Non	Non
jgroups-diagnostics		7500	Multidiffusion. Utilisée pour découvrir des pairs dans les groupements HA. Non configurable par les Interfaces de gestion.	Oui	Non	Oui	Non
jgroups-mping		45700	Multidiffusion. Utilisée pour découvrir l'appartenance de groupe d'origine dans un cluster HA.	Oui	Non	Oui	Non
jgroups-tcp	7600		Découverte de pairs unicastes dans les groupements HA avec TCP.	Oui	Non	Oui	Non
jgroups-tcp-fd	57600		Utilisé pour la détection des échecs en TCP.	Oui	Non	Oui	Non
jgroups-udp	55200	45688	Découverte de pairs unicastes dans les groupements HA avec UDP.	Oui	Non	Oui	Non

Nom	Port	Port multi-diffusion	Description	full-ha-sockets	full-sockets	ha-socket	standard-socket
jgroups-udp-fd	54200		Utilisé pour la détection des échecs par UDP.	Oui	Non	Oui	Non
messaging	5445		Service JMS.	Oui	Oui	Non	Non
messaging-group			Référencé par la diffusion HornetQ JMS et les groupes Discovery	Oui	Oui	Non	Non
messaging-throughput	5455		Utilisé par JMS Remoting.	Oui	Oui	Non	Non
mod_cluster		23364	Port de multidiffusion pour la communication entre JBoss EAP 6 et l'équilibreur de charges HTTP.	Oui	Non	Oui	Non
osgi-http	8090		Utilisé par les composants internes qui utilisent le sous-système OSGi. Non configurable par les Interfaces de gestion.	Oui	Oui	Oui	Oui
remoting	4447		Utilisé pour l'invocation EJB.	Oui	Oui	Oui	Oui
txn-recovery-environment	4712		Gestionnaire de recouvrement des transactions JTA.	Oui	Oui	Oui	Oui
txn-status-manager	4713		Gestionnaire des transactions JTA / JTS.	Oui	Oui	Oui	Oui

Ports de gestion

En plus des groupes de liaisons de socket, chaque contrôleur d'hôte ouvre deux ports supplémentaires pour la gestion :

- **`9990`** - Le port de Console de gestion web
- **`9999`** - Le port utilisé par la Console de gestion et par le Managemetn API

De plus, si HTTPS est activé pour la console de gestion, 9443 sera également ouvert en tant que valeur de port par défaut.

Rapporter un bogue

4.4. CRYPTAGE

Le *cryptage* désigne le brouillage des informations sensibles en appliquant des algorithmes mathématiques. Le cryptage est une des fondations de la sécurisation de votre infrastructure pour les violations de données, les pannes de système et autres risques.

Le cryptage peut être appliqué aux données de chaînes simples, comme les mots de passe. Il peut également être appliqué aux flux de communication de données. Le protocole HTTPS, par exemple, crypte toutes les données en en transférant une partie à une autre. Si vous vous connectez d'un serveur à un autre en utilisant le protocole Secure Shell (SSH), l'ensemble de votre communication sera envoyé vers un *tunnel* crypté.

Rapporter un bogue

4.5. CRYPTAGE SSL

SSL (Secure Sockets Layer) crypte le trafic réseau entre deux systèmes. Le trafic entre les deux systèmes est crypté à l'aide d'une clé bidirectionnelle, générée au cours de la phase de connexion de protocole ou *handshake* et qui n'est connue que par ces deux systèmes.

Pour les échanges bidirectionnels, SSL utilise PKI (de l'anglais Public Key Infrastructure), une méthode d'encodage qui utilise une *key pair* (paire de clés). Une paire de clés est composée de deux clés cryptographiques distinctes, mais correspondantes - une clé publique et une clé privée. La clé publique est partagée avec les autres et est utilisée pour crypter les données et la clé privée est tenue secrète et est utilisée pour déchiffrer des données qui ont été chiffrées à l'aide de la clé publique.

Lorsqu'un client fait une demande de connexion sécurisée, une phase de protocole de transfert prend place avant que la communication sécurisée puisse commencer. Pendant la négociation SSL, le serveur transmet sa clé publique au client sous la forme d'un certificat. Le certificat contient l'identité du serveur (son URL), la clé publique du serveur et une signature numérique valide le certificat. Ensuite, le client valide le certificat et prend une décision quant à savoir si le certificat doit être approuvé ou non. Si le certificat est approuvé, le client génère la clé de cryptage bidirectionnel pour la connexion SSL, la chiffre à l'aide de la clé publique du serveur et la renvoie sur le serveur. Le serveur décrypte la clé de cryptage bidirectionnel à l'aide de sa clé privée, et ensuite, la communication entre les deux machines est cryptée via cette connexion par la clé de cryptage bidirectionnel.

Rapporter un bogue

4.6. IMPLÉMENTATION DU CRYPTAGE SSL POUR LE SERVEUR DE JBOSS EAP 6.

Introduction

De nombreuses applications web requièrent une connexion cryptée-SSL entre les clients et le serveur, connue sous le nom de connexion **HTTPS**. Vous pouvez utiliser cette procédure pour activer **HTTPS** sur votre serveur ou groupe de serveurs.

Pré-requis

- Un ensemble de clés de cryptage SSL et un certificat de cryptage SSL. Vous pourrez vous les procurer par l'intermédiaire d'une autorité de signature de certificats. Pour générer les clés de cryptage par les utilitaires de Red Hat Enterprise Linux, voir Section 4.7, « Générer une clé de cryptage SSL et un certificat ».
- Informations utiles sur votre environnement et sur votre installation :
  - Le nom complet du répertoire où les fichiers de certificats sont stockés.
  - Le mot de passe de cryptage pour vos clés de cryptage.
- Exécuter l'interface CLI et le connecter à votre contrôleur de domaine ou à votre serveur autonome.

NOTE

Cette procédure utilise des commandes appropriées à la configuration de JBoss EAP 6, qui utilise un domaine géré. Si vous utilisez un domaine autonome, modifier les commandes de Management CLI en supprimant **`/profile=default`** du début d'une commande de Management CLI.

Procédure 4.3. Configurer le JBoss Web Server pour qu'il puisse utiliser HTTPS

1. **Ajoutez un nouveau connecteur HTTPS.**
   Exécutez la commande de Management CLI suivante, en changeant le profil comme il se doit. Cela va créer un nouveau connecteur sécurisé, nommé **HTTPS**, qui utilise le protocole **`https`**, la liaison de socket **`https`** (ayant comme valeur par défaut **8443**), et qui est définie pour être sécurisée.

   Exemple 4.5. Commande de Management CLI

   ```
   /profile=default/subsystem=web/connector=HTTPS/:add(socket-
   binding=https,scheme=https,protocol=HTTP/1.1,secure=true)
   ```

2. Éxécutez la commande de Management CLI suivante pour définir le protocole à **`TLSv1`**.

   Exemple 4.6. Commande de Management CLI

   ```
   /profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:
   write-attribute(name=protocol,value=TLSv1)
   ```

3. **Sélectionnez les suites de cryptage qui conviennent**
   Il existe un certain nombre de primitives cryptographiques disponibles, utilisées comme blocs de

base pour former des suites de chiffrement. Le premier tableau répertorie les primitives cryptographiques recommandées. Le deuxième énumère les primitives cryptographiques qui, bien qu'elles puissent être utilisées pour assurer la compatibilité avec les logiciels existants, ne sont pas considérées comme aussi sûres que celles qui sont recommandées.

AVERTISSEMENT

Red Hat recommande de faire une liste blanche sélective d'un ensemble d'algorithmes de chiffrage puissants à utiliser pour la **`cipher-suite`**. L'activation de systèmes de chiffrage faibles est un risque important de sécurité. Consultez la documentation du fournisseur de votre JDK avant de statuer sur les suites de chiffrement particulières à utiliser, car il peut y avoir des problèmes de compatibilité.

Tableau 4.2. Primitives cryptographiques recommandées

RSA avec clés 2048 bit et OAEP
AES-128 en mode CBC
SHA-256
HMAC-SHA-256
HMAC-SHA-1

Tableau 4.3. Autres primitives cryptographiques

RSA avec des clés qui dépassent 1024 et taille de remplissage héritée
AES-192
AES-256
3DES (triple DES, avec deux ou trois clés de 56 bit)
RC4 (fortement déconseillé)
SHA-1
HMAC-MD5

Pour obtenir une liste complète des paramètres que vous pouvez définir pour les propriétés SSL du connecteur, voir Section 4.8, « Référence de connecteur SSL ».

4. **Configurer le certificat de cryptage SSL et les clés.**
   Exécutez la commande CLI suivante pour configurer votre certificat SSL, en remplaçant vos propres valeurs par celles de l'exemple. Cet exemple suppose que le keystore est copié dans le répertoire de configuration du serveur, qui est ***EAP_HOME*/domain/configuration/** pour un domaine géré.

   Exemple 4.7. Commande de Management CLI

   ```
   /profile=default/subsystem=web/connector=HTTPS/ssl=configuration:a
   dd(name=https,certificate-key-
   file="${jboss.server.config.dir}/keystore.jks",password=SECRET,
   key-alias=KEY_ALIAS, cipher-suite=CIPHERS)
   ```

5. **Déployer une application.**
   Déployez une application dans un groupe de serveurs qui utilise le profil que vous avez configuré. Si vous utilisez un serveur autonome, déployer une application sur votre serveur. Les demandes HTTPS en sa direction utilisent la nouvelle connexion cryptée-SSL.

Rapporter un bogue

4.7. GÉNÉRER UNE CLÉ DE CRYPTAGE SSL ET UN CERTIFICAT

Pour utiliser une connexion chiffrée SSL HTTP (HTTPS), ainsi que d'autres types de communication cryptée-SSL, vous avez besoin d'un certificat de cryptage signé. Vous pouvez acheter un certificat d'une autorité de certification (AC), ou vous pouvez utiliser un certificat auto-signé. Les certificats auto-signés ne sont pas considérés dignes de confiance par de nombreux tiers, mais conviennent à des fins de test internes.

Cette procédure vous permet de créer un certificat auto-signé lié à des utilitaires disponibles dans Red Hat Enterprise Linux.

Pré-requis

- La commande **keytool** doit être disponible. Elle est fournie par le Java Development Kit. Dans Red Hat Enterprise Linux, OpenJDK installe cette commande à l'emplacement suivant **/usr/bin/keytool**.
- Comprendre la syntaxe et les paramètres de la commande **keytool**. Cette procédure utilise des instructions extrêmement standard, car des discussions plus sophistiquées sur les spécificités des certificats SSL ou sur la commande **keytool** sont hors de portée de cette documentation.

Procédure 4.4. Générer une clé de cryptage SSL et un certificat

1. **Générer un keystore avec des clés privées et des clés publiques.**
   Exécutez la commande suivante pour générer un keystore nommé **server.keystore** ayant comme alias **jboss** dans votre répertoire actuel.

   ```
   keytool -genkeypair -alias jboss -keyalg RSA -keystore
   server.keystore -storepass mykeystorepass --dname
   "CN=jsmith,OU=Engineering,O=mycompany.com,L=Raleigh,S=NC,C=US"
   ```

Le tableau suivant décrit les paramètres utilisés avec la commande « keytool ».

Paramètre	Description
-genkeypair	La commande **`keytool`** qui génère une paire de clés contenant une clé publique et une clé privée.
-alias	L'alias est pour le keystore. Cette valeur est arbitraire, mais l'alias **`jboss`** est la valeur par défaut utilisée par le serveur JBoss Web.
-keyalg	L'algorithme de création de paires de clés. Dans ce cas, c'est **RSA**.
-keystore	Le nom et l'emplacement du fichier keystore. L'emplacement par défaut est le répertoire en cours. Le nom que vous choisissez est arbitraire. Dans ce cas, il s'agit du fichier nommé **`server.keystore`**.
-storepass	Ce mot de passe est utilisé pour authentifier le keystore, et pour que la clé puisse être lue. Le mot de passe doit contenir au moins 6 caractères de long et doit être fourni quand on accède au keystore. Dans un tel cas, on utilise **`mykeystorepass`**. Si vous omettez ce paramètre, on vous demandera de le saisir quand vous exécuterez la commande.
-keypass	Il s'agit du mot de passe pour la clé.  **NOTE** À cause d'une limitation d'implémentation, il doit correspondre à celui du mot de passe du store.

Paramètre	Description
--dname	Une chaîne avec des guillemets qui décrit le nom distinct de la clé, comme par exemple : "CN=jsmith,OU=Engineering,O=mycompany.com,L=Raleigh,C=US". Cette chaîne est une compilation des composants suivants : • **CN** - Le nom commun ou le nom d'hôte. Si le nom d'hôte est "jsmith.mycompany.com", le **CN** sera "jsmith". • **OU** - L'unité organisationnelle, par exemple "Engineering" • **O** - Le nom de l'organisation, par exemple "mycompany.com". • **L** - La localité, par exemple "Raleigh" ou "London" • **S** - L'état ou la province, par exemple "NC". Ce paramètre est optionnel. • **C** - Les 2 lettres d'un code pays, par exemple "US" ou "UK".

Quand vous exécuterez la commande ci-dessus, on vous demandera les informations suivantes :

- Si vous n'utilisiez pas le paramètre ***-storepass*** sur la ligne de commande, on vous demandera de saisir le mot de passe du keystore. Saisir le nouveau mot de passe à la seconde invite.
- Si vous n'utilisez pas le paramètre **-keypass** sur la ligne de commande, on vous demandera de saisir le mot de passe de la clé. Appuyez sur **Enter** pour le définir à la même valeur que celle du mot de passe du keystore.

Quand la commande s'achèvera, le fichier **server.keystore** contiendra la clé unique avec l'alias **jboss**.

2. **Vérifier la clé.**
   Vérifiez que la clé fonctionne correctement en utilisant la commande suivante.

   ```
   keytool -list -keystore server.keystore
   ```

   On vous demande le mot de passe du keystore. Les contenus du keystore sont affichés (dans ce cas, il s'agit d'une simple clé nommée **jboss**). Notez le type de la clé **jboss**, qui est **PrivatekeyEntry**. Cela indique que le keystore contient à la fois une entrée publique et une entrée privée pour cette clé.

3. **Créer une demande de signature de certificat.**
   Exécutez la commande suivante pour générer une demande de signature de certificat en utilisant la clé publique du keystore que vous avez créée dans la 1ère étape.

```
keytool -certreq -keyalg RSA -alias jboss -keystore server.keystore
-file certreq.csr
```

On vous demandera le mot de passe pour pouvoir authentifier le keystore. La commande **`keytool`** crée alors une nouvelle demande de signature de certificat nommée **`certreq.csr`** dans le répertoire en cours d'utilisation.

4. **Tester la demande de signature de certificat nouvellement générée.**
Tester les contenus du certificat avec la commande suivante :

```
openssl req -in certreq.csr -noout -text
```

Les détails du certificat apparaissent.

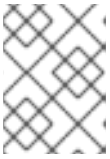
5. **En option : soumettre votre demande de signature de certificat à une autorité de certification (AC).**
Une Autorité de Certification (AC) authentifie votre certificat pour qu'il soit considéré de confiance par des clients de tierce partie. L'AC vous a produit un certificat signé, et en option, vous a peut être fourni un ou plusieurs certificats intermédiaires.

6. **Option : exporter un certificat auto-signé du keystore.**
Si vous n'en avez besoin que dans un but de test ou en interne, vous pourrez utiliser un certificat auto-signé. Vous pourrez en exporter un, créé dans la première étape, en provenance du keystore, comme suit :

```
keytool -export -alias jboss -keystore server.keystore -file
server.crt
```

On vous demande un mot de passe pour pouvoir s'authentifier au keystore. Un certificat auto-signé, intitulé **`server.crt`**, a été créé dans le répertoire en cours d'utilisation.

7. **Importer le certificat signé avec tout certificat intermédiaire.**
Importer chaque certificat, dans l'ordre dans lequel l'AC vous le demande. Pour chaque AC que vous importez, remplacer **`intermediate.ca`** ou **`server.crt`** par le nom du fichier. Si vos certificats ne sont pas fournis dans des fichiers séparés, créer un fichier séparé pour chaque certificat, et coller leur contenu dans le fichier.



NOTE

Votre certificat signé et les clés de ce certificat sont des ressources de valeur. Soyez vigilant sur la façon dont vous les transporterez entre les serveurs.

```
keytool -import -keystore server.keystore -alias intermediateCA -
file intermediate.ca
```

```
keytool -importcert -alias jboss -keystore server.keystore -file
server.crt
```

8. **Testez que vos certificats soient bien importés avec succès.**
Exécutez la commande suivante, et saisissez le mot de passe de keystore quand on vous le demandera. Les contenus de votre keystore seront affichés, et les certificats feront partie de la liste.

```
keytool -list -keystore server.keystore
```

Résultat

Votre certificat signé est maintenant inclus dans votre keystore et est prêt à l'utilisation pour crypter les connexions SSL, y compris les communications au serveur web HTTPS.

Rapporter un bogue

4.8. RÉFÉRENCE DE CONNECTEUR SSL

Les connecteurs JBoss Web peuvent inclure les attributs de configuration SSL suivants. Les commandes CLI fournies sont conçues pour un domaine géré à l'aide du profil **`par défaut`**. Changez le nom du profil à celui que vous souhaitez configurer, pour un domaine géré, ou omettre la portion **`/profile=default`** de la commande, pour un serveur autonome.

Tableau 4.4. Attributs de connecteurs SSL

Attribut	Description	Commande CLI
name	Le nom d'affichage du connecteur SSL	L'attribute **`name`** est en lecture seule.

Attribut	Description	Commande CLI

verify-client	Les valeurs possibles de **`verify-client`** diffèrent selon qu'il s'agisse d'un connecteur HTTP/HTTPS ou d'un connecteur natif APR qui est utilisé.
	
	Connecteur HTTP/HTTPS
	
	Les valeurs possibles sont **`true`**, **`false`**, ou **`want`**. Définir à **`true`** pour obtenir une chaîne de certificat valide de la part d'un client avant d'accepter une connexion. Définir à **`want`** si vous voulez que la pile SSL demande un certificat de client, mais ne pas l'échouer si celui-ci n'est pas présenté. Définir à **`false`** (la valeur par défaut) si vous ne souhaitez pas demander de chaîne de certificat, à moins que le client ne demande une ressource protégée par une contrainte de sécurité qui utilise l'authentification de **`CLIENT-CERT`**.
	
	Connecteur APR natif
	
	Les valeurs possibles sont **`optional`**, **`require`**, **`optionalNoCA`**, et **`none`** (ou tout autre string, ce qui aura le même effet que **`none`**). Ces valeurs déterminent si un certificat est optionnel, requis, ou optionnel sans Autorité de certification, ou encore, non repuis. La valeur par défaut est **`none`**, ce qui signifie que le client n'a pas la possibilité de soumettre un certificat.
		La commande du premier exemple utilise le connecteur HTTPS.
		
		`/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=verify-client,value=want)`
		
		La commande du second exemple utilise le connecteur APR.
		
		`/profile=default/subsystem=web/connector=APR/ssl=configuration/:write-attribute(name=verify-client,value=require)`

verify-depth	Le nombre maximal d'émetteurs de certificats intermédiaires vérifiés avant de décider que les clients n'ont pas de certificat valide. La valeur par défaut est **`10`**.	`/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=verify-depth,value=10)`

Attribut	Description	Commande CLI
certificate-key-file	Le chemin d'accès complet et nom de fichier du keystore où le certificat de serveur signé est stocké. Avec le cryptage JSSE, ce fichier de certificat sera le seul, tandis que OpenSSL utilise plusieurs fichiers. La valeur par défaut est le fichier **`.keystore`** dans le répertoire home de l'utilisateur exécutant JBoss EAP 6. Si votre **`keystoreType`** n'utilise pas de fichier, définissez le paramètre en chaîne vide.	`/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=certificate-key-file,value=../domain/configuration/server.keystore)`
certificate-file	Si vous utilisez un cryptage OpenSSL, définissez la valeur de ce paramètre au chemin d'accès du fichier qui contient le certificat de serveur.	`/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=certificate-file,value=server.crt)`
password	Le mot de passe pour le trustore et le keystore à la fois. Dans l'exemple suivant, remplacez *PASSWORD* par votre propre mot de passe.	`/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=password,value=PASSWORD)`
protocol	Version du protocole SSL à utiliser. Les valeurs prises en charge dépendent de l'implémentation sous-jacente (JSSE ou OpenSSL). Consultez Java SSE Documentation.	`/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=protocol,value=ALL)`

Attribut	Description	Commande CLI
cipher-suite	Une liste des cryptages autorisés. Pour la syntaxe JSSE, cela devra corrrespondre à une liste séparée par des virgules. Pour la syntaxe OpenSSL, cela devra correspondre à une liste séparée par deux-points. La valeur par défaut est **`HIGH:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5`**. L'exemple ne mentionne que deux cryptages possibles, mais les exemples de la vie courante en utilisent d'autres.  **IMPORTANT** L'utilisation de cryptages faibles posent des risques de sécurité importants. Voir http://www.nist.gov/manuscript-publication-search.cfm?pub_id=915295 pour obtenir des informations NIST sur les suites de cryptage. Pour obtenir une liste de cryptage OpenSSL, consultez https://www.openssl.org/docs/apps/ciphers.html#CIPHER_STRINGS. Notez que ce qui suit n'est pas pris en charge : **`@SECLEVEL`**, **`SUITEB128`**, **`SUITEB128ONLY`**, **`SUITEB192`**.	`/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=cipher-suite, value="TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA")`
key-alias	L'alias utilisé pour le certificat de serveur dans le keystore. Dans l'exemple suivant, remplacez *KEY_ALIAS* par l'alias de votre certificat.	`/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=key-alias,value=KEY_ALIAS)`

Attribut	Description	Commande CLI
truststore-type	Le type de truststore. Il existe différents types de truststores, dont **PKCS12** et le **JKS** standard de Java.	`/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=truststore-type,value=jks)`
keystore-type	Le type de keystore. Il existe différents types de keystores, dont **PKCS12** et le **JKS** standard de Java.	`/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=keystore-type,value=jks)`
ca-certificate-file	Le fichier contenant les certificats CA. Il s'agit du **`truststoreFile`**, dans le cas de JSSE, et il utilise le même mot de passe que le keystore. Le fichier **`ca-certificate-file`** est utilisé pour valider les certificats de clients.	`/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=certificate-file,value=ca.crt)`
ca-certificate-password	Le mot de passe de certificat pour le **`ca-certificate-file`**. Dans l'exemple ci-dessous, remplacez *MASKED_PASSWORD* par votre propre mot de passe masqué.	`/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=ca-certificate-password,value=`*`MASKED_PASSWORD`*`)`
ca-revocation-url	Un fichier ou URL qui contient la liste de révocations. Se réfère au **`crlFile`** pour JSSE ou au **`SSLCARevocationFile`** pour SSL.	`/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=ca-revocation-url,value=ca.crl)`

Attribut	Description	Commande CLI
session-cache-size	La taille du cache SSLSession. Cet attribut s'applique uniquement aux connecteurs JSSE. La valeur par défaut est **0**, qui indique une taille de cache illimitée.	`/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=session-cache-size,value=100)`
session-timeout	Le nombre de secondes avant qu'une SSLSession n'expire. Cet attribut s'applique uniquement aux connecteurs JSSE. La valeur par défaut est **86400** secondes, ce qui correspond à 24 heures.	`/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=session-timeout,value=43200)`

Rapporter un bogue

4.9. ENCODAGE SE CONFORMANT À FIPS 140-2

4.9.1. Conformité FIPS 140-2

FIPS (Federal Information Processing Standard) 140-2 (FIPS 140-2) est un standard de sécurité informatique gouvernemental US pour l'accréditation des modules informatiques cryptographiques. Le standard FIPS 140-2 est souvent un pré-requis pour les systèmes informatiques des agences gouvernementales et pour le secteur commercial privé.

JBoss EAP 6 utilise le chiffrement de modules externes et peut être configuré pour pouvoir utiliser un module de cryptage compatible FIPS 140-2.

Rapporter un bogue

4.9.2. Cryptographie compatible FIPS 140-2 dans JDK IBM

Dans JDK IBM, le fournisseur IBMJCEFIPS IBM® JCE (Java™ Cryptographic Extension) et le module IBM JSSE (Java Secure Sockets Extension) FIPS 140-2 Cryptographic Module (IBMJSSEFIPS) pour plateformes multiples fournissent une cryptographie compatible FIPS 140-2.

Pour plus d'informations sur le fournisseur IBMJCEFIPS, voir la documentation the IBM Documentation for IBM JCEFIPS, et NIST IBMJCEFIPS – Security Policy.

Stockage des clés
Notez que le JCE IBM ne procure pas de keystore. Les clés sont stockées sur l'ordinateur et n'en sortent pas. Si les clés sont déplacées d'un ordinateur à l'autre, elles doivent être encryptées.

Pour exécuter **keytool** en mode compatible FIPS, utilisez l'option **-providerClass** sur chaque commande, comme suit :

-

```
keytool -list -storetype JCEKS -keystore mystore.jck -storepass mystorepass
-providerClass com.ibm.crypto.fips.provider.IBMJCEFIPS
```

Examinez les informations de fournisseur FIPS
Pour examiner les informations sur le IBMJCEFIPS utilisé par le serveur, activer la journalisation debug-level en ajoutant **`-Djavax.net.debug=true`** à **`standalone.conf`** ou à **`domain.conf`**. Les informations sur le fournisseur FIPS seront journalisées dans **`server.log`**, comme par exemple :

```
04:22:45,685 INFO  [stdout] (http-/127.0.0.1:8443-1) JsseJCE:  Using
MessageDigest SHA from provider IBMJCEFIPS version 1.7
04:22:45,689 INFO  [stdout] (http-/127.0.0.1:8443-1) DHCrypt:  DH
KeyPairGenerator  from provider from init IBMJCEFIPS version 1.7
04:22:45,754 INFO  [stdout] (http-/127.0.0.1:8443-1) JsseJCE:  Using
KeyFactory DiffieHellman from provider IBMJCEFIPS version 1.7
04:22:45,754 INFO  [stdout] (http-/127.0.0.1:8443-1) JsseJCE:  Using
KeyAgreement DiffieHellman from provider IBMJCEFIPS version 1.7
04:22:45,754 INFO  [stdout] (http-/127.0.0.1:8443-1) DHCrypt:  DH
KeyAgreement  from provider IBMJCEFIPS version 1.7
04:22:45,754 INFO  [stdout] (http-/127.0.0.1:8443-1) DHCrypt:  DH
KeyAgreement  from provider from initIBMJCEFIPS version 1.7
```

Rapporter un bogue

4.9.3. Mots de passe conformes FIPS 140-2

Un mot de passe conforme FIPS doit avoir les caractéristiques suivantes :

1. Une longueur minimale de sept (7) caractères.
2. Il doit inclure des caractères d'au moins trois (3) des classes de caractères suivantes :
    - chiffres ASCII,
    - minuscules ASCII,
    - majuscules ASCII,
    - non alphanumériques ASCII, et
    - non-ASCII.

Si le premier caractère du mot de passe est en lettre majuscule ASCII, il ne comptera pas comme une lettre majuscule ASCII pour la restriction 2.

Si le dernier caractère du mot de passe est un chiffre ASCII, il ne comptera pas comme chiffre ASCII pour la restriction 2.

Rapporter un bogue

4.9.4. Activer la Cryptography FIPS 140-2 pour SSL dans Red Hat Enterprise Linux 6

Cette tâche décrit comment configurer le conteneur web (JBoss Web) de JBoss EAP 6 pour que la cryptographie soit conforme à FIPS 140-2 pour SSL. Cette tâche ne couvre que les étapes spécifiques à Red Hat Enterprise Linux 6.

Cette tâche utilise la bibliothèque Mozilla NSS en mode FIPS pour cette fonctionnalité.

Pré-requis

- Red Hat Enterprise Linux 6 doit déjà être configuré pour être configuré en conformité avec FIPS 140-2. Voir https://access.redhat.com/knowledge/solutions/137833.

Procédure 4.5. Voir Conformité Cryptographie FIPS 140-2 pour SSL

1. **Créez la base de données**
   Créez la base de données NSS dans un répertoire qui appartienne à l'utilisateur **`jboss`**.

   ```
   $ mkdir -p  /usr/share/jboss-as/nssdb
   $ chown jboss /usr/share/jboss-as/nssdb
   $ modutil -create -dbdir /usr/share/jboss-as/nssdb
   ```

2. **Créez un fichier de configuration NSS**
   Créez un nouveau fichier texte ayant comme nom **`nss_pkcsll_fips.cfg`** dans le répertoire **`/usr/share/jboss-as`** avec le contenu suivant :

   ```
   name = nss-fips
   nssLibraryDirectory=/usr/lib64
   nssSecmodDirectory=/usr/share/jboss-as/nssdb
   nssModule = fips
   ```

   Le fichier de configuration NSS doit spécifier :

   - un nom,
   - le répertoire où se trouve la bibliothèque, et
   - le répertoire où la base de données NSS a été créée selon l'étape 1.

   Si vous n'êtes pas sur une version 64bit de Red Hat Enterprise Linux 6, alors définissez **`nssLibraryDirectory`** à **`/usr/lib`** à la place de **`/usr/lib64`**.

3. **Activez le fournisseur SunPKCS11**
   Modifiez le fichier de configuration **`java.security`** de votre JRE (**`$JAVA_HOME/jre/lib/security/java.security`**) et ajoutez la ligne suivante :

   ```
   security.provider.1=sun.security.pkcs11.SunPKCS11  /usr/share/jboss-
   as/nss_pkcsll_fips.cfg
   ```

   Notez que le fichier de configuration spécifié sur cette ligne est le fichier créé à l'étape 2.

   Toute autre ligne **`security.provider.`*`X`*** de ce fichier devra posséder la valeur *X* +1 pour que la priorité soit donnée à ce fournisseur.

4. **Activez le mode FIPS pour la bibliothèque NSS**
   Exécutez la commande **`modutil`** comme indiqué pour activer le mode FIPS :

   ```
   modutil -fips true -dbdir /usr/share/jboss-as/nssdb
   ```

   Notez que le répertoire indiqué ici est le répertoire créé à l'étape 1.

Vous aurez sans doute une erreur de bibliothèque à ce niveau, ce qui vous oblige à régénérer les signatures de bibliothèques sur certains des objets NSS partagés.

5. **Modifiez le mot de passe du token FIPS**
   Définissez le mot de passe du token FIPS par la commande suivante. Notez que le nom du token doit correspondre à **`NSS FIPS 140-2 Certificate DB`**.

```
modutil -changepw "NSS FIPS 140-2 Certificate DB" -dbdir
/usr/share/jboss-as/nssdb
```

   Le mot de passe utilisé pour le token FIPS doit être un mot de passe conforme FIPS.

6. **Créez le certificat grâce aux outils NSS**
   Saisissez la commande suivante pour créer un certificat par les outils NSS.

```
certutil -S -k rsa -n jbossweb  -t "u,u,u" -x -s "CN=localhost,
OU=MYOU, O=MYORG, L=MYCITY, ST=MYSTATE, C=MY" -d /usr/share/jboss-
as/nssdb
```

7. **Configurez le connecteur HTTPS pour utiliser le keystore PKCS11**
   Ajoutez un connecteur HTTPS par la commande suivante dans JBoss CLI :

```
/subsystem=web/connector=https/:add(socket-
binding=https,scheme=https,protocol=HTTP/1.1,secure=true)
```

   Puis, ajoutez la configuration SSL par la commande suivante, en remplaçant PASSWORD par le mot de passe conforme FIPS de l'étape 5.

```
/subsystem=web/connector=https/ssl=configuration:add(name=https,pass
word=PASSWORD,keystore-type=PKCS11,
cipher-
suite="SSL_RSA_WITH_3DES_EDE_CBC_SHA,SSL_DHE_RSA_WITH_3DES_EDE_CBC_S
HA,
TLS_RSA_WITH_AES_128_CBC_SHA,TLS_DHE_DSS_WITH_AES_128_CBC_SHA,
TLS_DHE_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA,
TLS_DHE_DSS_WITH_AES_256_CBC_SHA,TLS_DHE_RSA_WITH_AES_256_CBC_SHA,
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA,TLS_ECDH_ECDSA_WITH_AES_128_CBC
_SHA,
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_ECDSA_WITH_3DES_EDE_CB
C_SHA,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_ECDSA_WITH_AES_256_CB
C_SHA,
TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA,TLS_ECDH_RSA_WITH_AES_128_CBC_SHA
,
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SH
A,
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_256_CBC_SH
A,
TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA,TLS_ECDH_anon_WITH_AES_128_CBC_S
HA,
TLS_ECDH_anon_WITH_AES_256_CBC_SHA")
```

8. **Vérifier**

Vérifiez que la JVM puisse lire la clé privée du keystore PKCS11 en exécutant la commande suivante :

```
keytool -list -storetype pkcs11
```

Exemple 4.8. Configuration XML du connecteur HTTPS avec conformité FIPS 140-2

```
<connector name="https" protocol="HTTP/1.1" scheme="https" socket-
binding="https" secure="true">
  <ssl name="https" password="*****"
      cipher-
suite="SSL_RSA_WITH_3DES_EDE_CBC_SHA,SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA,
         TLS_RSA_WITH_AES_128_CBC_SHA,
TLS_DHE_DSS_WITH_AES_128_CBC_SHA,
         TLS_DHE_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA,

TLS_DHE_DSS_WITH_AES_256_CBC_SHA,TLS_DHE_RSA_WITH_AES_256_CBC_SHA,

TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA,TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
,

TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SH
A,

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SH
A,

TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA,TLS_ECDH_RSA_WITH_AES_128_CBC_SHA,

TLS_ECDH_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA,

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,

TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA,TLS_ECDH_anon_WITH_AES_128_CBC_SHA,
         TLS_ECDH_anon_WITH_AES_256_CBC_SHA"
      keystore-type="PKCS11"/>
</connector>
```

Notez que l'attribut **`cipher-suite`** a des sauts de ligne insérés pour faciliter la lecture.

Rapporter un bogue

CHAPITRE 5. SÉCURISER LES INTERFACES DE GESTION

Le scénario courant est d'exécuter JBoss EAP 6 sans sécurité sur les interfaces de gestion pour permettre des changements de configuration rapides.

Dans un déploiement en production, sécurisez les interfaces de gestion par une des méthodes suivantes au moins :

- Section 4.1, « Indiquer l'interface de réseau que JBoss EAP 6 utilise »
- Section 4.2, « Configurer les pare-feux de réseau pour qu'ils fonctionnent dans JBoss EAP 6 »

De plus, un mode silencieux d'authentification local présent permet aux clients locaux (sur la machine du serveur) de se connecter au Management CLI sans exiger un nom d'utilisateur ou un mot de passe. Ce comportement est pratique pour les utilisateurs locaux et les scripts de Management CLI. Pour désactiver cette fonction, voir la procédure décrite dans la rubrique Section 5.4, « Supprimer l'authentification silencieuse du domaine de sécurité par défaut. ».

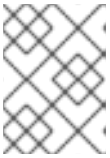
Rapporter un bogue

5.1. CONFIGURATION DE LA SÉCURITÉ UTILISATEUR PAR DÉFAUT

Introduction

Toutes les interfaces de gestion de JBoss EAP 6 sont sécurisées par défaut. Cette sécurité existe sous deux formes :

- Les interfaces locales sont sécurisées par un contrat SASL entre des clients locaux et le serveur auquel ils se connectent. Ce mécanisme de sécurité repose sur la capacité du client à accéder au système de fichiers local. C'est parce que l'accès au système de fichiers local permettait au client d'ajouter un utilisateur ou encore de modifier la configuration pour déjouer les autres mécanismes de sécurité. Cela est conforme au principe selon lequel si l'accès physique au système de fichiers est possible, les autres mécanismes de sécurité sont alors superflus. Le mécanisme passe par quatre étapes :

  NOTE

  L'accès HTTP est considéré comme éloigné, même si vous vous connectez à l'hôte local par HTTP.

  1. Le client envoie un message au serveur incluant une requête pour authentifier le mécanisme SASL local.
  2. Le serveur génère un token unique, qu'il écrit dans un fichier unique, et envoie un message au client avec le chemin d'accès fichier complet.
  3. Le client lit le token dans le fichier et l'envoie au serveur, pour vérifier qu'il ait bien un accès local au système de fichiers.
  4. Le serveur vérifie le token et efface le fichier.

- Les clients distants, y compris HTTP, utilisent une sécurité basée domaine. Le domaine par défaut, qui comprend les autorisations pour configurer l'instance JBoss EAP 6 à distance en utilisant les interfaces de gestion, est **`ManagementRealm`**. Un script est fourni pour vous permettre d'ajouter des utilisateurs à ce domaine (ou à des domaines que vous créez). Pour

plus d'informations sur la façon d'ajouter des utilisateurs, voir le chapitre *Getting Started* de *JBoss EAP 6 Installation Guide*. Pour chaque utilisateur, le nom d'utilisateur, un mot de passe haché sont stockés dans un fichier.

Domaine géré

***EAP_HOME*/domain/configuration/mgmt-users.properties**

Serveur autonome

***EAP_HOME*/standalone/configuration/mgmt-users.properties**

Même si les contenus de **mgmt-users.properties** sont masqués, le fichier doit toujours être considéré comme un fichier sensible. Il est recommandé qu'il soit défini sur le mode de fichier **600**, qui ne donne aucun accès autre que l'accès en lecture et écriture au propriétaire du fichier.

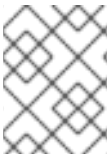
Rapporter un bogue

5.2. APERÇU GÉNÉRAL DE LA CONFIGURATION DE L'INTERFACE DE GESTION AVANCÉE

La configuration de l'interface de gestion ***EAP_HOME*/domain/configuration/host.xml** ou ***EAP_HOME*/standalone/configuration/standalone.xml** contrôle l'interface réseau à laquelle se relie le processus contrôleur hôte, les types d'interfaces de gestion qui sont disponibles à tous, et quel type de système d'authentification est utilisé pour authentifier les utilisateurs sur chaque interface. Cette rubrique explique comment configurer les interfaces de gestion en fonction de votre environnement.

Le sous-système de gestion est formé d'un élément **<management>** avec les quatre éléments enfant configurables suivants. Les domaines de sécurité et les connexions sortantes sont définies chacune pour commencer, puis appliquées aux interfaces de gestion en tant qu'attributs.

- <security-realms>
- <outbound-connections>
- <management-interfaces>
- <audit-log>



NOTE

Voir la section *Management Interface Audit Logging* du guide *Administration and Configuration Guide* pour obtenir plus d'informations sur la journalisation d'auditing.

Domaines de sécurité

Le domaine de sécurité est chargé de l'authentification et de permettre aux utilisateurs autorisés d'administrer JBoss EAP 6 via l'API de gestion, l'interface CLI ou la Console de gestion basée-web.

Il existe deux domaines de sécurité basés fichier différents, qui existent dans l'installation par défaut : **ManagementRealm** et **ApplicationRealm**. Chacun de ces domaines de sécurité utilise un fichier **-users.properties** pour stocker les utilisateurs et les mots de passe de hachage, et **-roles.properties** pour stocker les correspondances entre les utilisateurs et les rôles. Un support est également inclus pour le domaine de sécurité activé-LDAP.

NOTE

Les domaines de sécurité peuvent également être utilisés pour vos propres applications. Les domaines de sécurité dont il s'agit sont particuliers aux interfaces de gestion.

Les connexions de sortie

Certains domaines de sécurité se connectent à des interfaces externes, comme un serveur LDAP. Une connexion sortante définit la manière d'établir cette connexion. Un type de connexion prédéfinie, la connexion **`ldap-connection`**, définit tous les attributs obligatoires et facultatifs pour se connecter au serveur LDAP et vérifier les informations d'identification.

Pour obtenir plus d'informations sur la façon de configurer l'authentification LDAP, voir Section 5.11.2, « Utiliser LDAP pour vous authentifier auprès des interfaces de gestion ».

Interfaces de gestion

Une interface de gestion comprend des propriétés sur la façon de connecter et configurer JBoss EAP. Ces informations comprennent l'interface réseau nommée, le port, le domaine de sécurité et autres informations configurables sur l'interface. Deux interfaces sont incluses dans une installation par défaut :

- **`http-interface`** est la configuration de la console de gestion basée-web.
- **`native-interface`** est la configuration pour la Management CLI en ligne de commande et l'API de gestion Rest-like.

Chacun des quatre principaux éléments configurables du sous-système de gestion de l'hôte sont étroitement liés. Un domaine de sécurité fait référence à une connexion sortante et une interface de gestion à un domaine de sécurité.

On peut trouver des informations associées dans Chapitre 5, *Sécuriser les interfaces de gestion*.

Rapporter un bogue

5.3. DÉSACTIVER L'INTERFACE DE GESTION HTTP

Dans un domaine géré, vous avez seulement besoin d'un accès à l'interface HTTP sur le contrôleur de domaine, plutôt que sur des serveurs membres de domaine. En outre, sur un serveur de production, vous pouvez finalement décider de désactiver la Console de gestion basée-web.

NOTE

Les autres clients, tels que JBoss Operations Network, opèrent également par l'interface HTTP. Si vous souhaitez utiliser ces services ou tout simplement désactiver la gestion de la Console elle-même, vous pouvez définir l'attribut **`console-enabled`** de l'interface HTTP à **`false`**, au lieu de désactiver l'interface complètement.

```
/host=master/core-service=management/management-interface=http-interface/:write-attribute(name=console-enabled,value=false)
```

Pour désactiver l'interface HTTP, ce qui désactive également l'accès à la Console de gestion basée-web, vous pouvez finalement supprimer l'interface HTTP.

La commande JBoss CLI suivante vous permettra de lire le contenu actuel de votre interface HTTP, si vous décidez de l'ajouter à nouveau.

Exemple 5.1. Lire la configuration de l'interface HTTP

```
/host=master/core-service=management/management-interface=http-
interface/:read-resource(recursive=true,proxies=false,include-
runtime=false,include-defaults=true)
{
    "outcome" => "success",
    "result" => {
        "console-enabled" => true,
        "interface" => "management",
        "port" => expression "${jboss.management.http.port:9990}",
        "secure-port" => undefined,
        "security-realm" => "ManagementRealm"
    }
}
```

Pour supprimer l'interface HTTP, lancez la commande suivante :

Exemple 5.2. Supprimer l'interface HTTP

```
/host=master/core-service=management/management-interface=http-
interface/:remove
```

Pour activer l'accès à nouveau, lancez la commande suivante pour recréer l'Interface HTTP avec les valeurs par défaut.

Exemple 5.3. Recréer l'Interface HTTP

```
/host=master/core-service=management/management-interface=http-
interface:add(console-
enabled=true,interface=management,port="${jboss.management.http.port:999
0}",security-realm=ManagementRealm)
```

Rapporter un bogue

5.4. SUPPRIMER L'AUTHENTIFICATION SILENCIEUSE DU DOMAINE DE SÉCURITÉ PAR DÉFAUT.

Résumé

L'installation par défaut de JBoss EAP 6 contient une méthode d'authentification silencieuse pour un utilisateur de Management CLI local. Cela donne à l'utilisateur local la possibilité d'accéder au Management CLI sans authentification par nom d'utilisateur ou mot de passe. Cette fonctionnalité est activée dans un but pratique et pour faciliter l'exécution des scripts de Management CLI sans exiger l'authentification des utilisateurs locaux. Cette méthode est considérée comme une fonctionnalité utile étant donné que l'accès à la configuration locale donne généralement aussi à l'utilisateur la possibilité d'ajouter ses propres informations d'utilisateur ou sinon la possibilité de désactiver les contrôles de sécurité.

L'avantage de l'authentification silencieuse des utilisateurs locaux peut être désactivée quand un plus

grand contrôle de sécurité est exigé. Ceci peut être réalisé en supprimant l'élément **`local`** dans la section **`security-realm`** du fichier de configuration. S'applique à la fois à **`standalone.xml`** pour une instance de serveur autonome, ou à **`host.xml`** pour un domaine géré. Vous ne devez envisager de supprimer l'élément **`local`** que si vous comprenez l'impact que ceci peut avoir sur la configuration de votre serveur particulier.

La meilleure méthode pour désactiver l'authentification silencieuse est d'utiliser l'interface CLI, qui retire l'élément **`local`** visible dans l'exemple suivant.

Exemple 5.4. Exemple d'élément `local` dans `security-realm`

```
<security-realms>
    <security-realm name="ManagementRealm">
        <authentication>
            <local default-user="$local"/>
            <properties path="mgmt-users.properties" relative-
to="jboss.server.config.dir"/>
        </authentication>
    </security-realm>
    <security-realm name="ApplicationRealm">
        <authentication>
            <local default-user="$local" allowed-users="*"/>
            <properties path="application-users.properties" relative-
to="jboss.server.config.dir"/>
        </authentication>
        <authorization>
            <properties path="application-roles.properties" relative-
to="jboss.server.config.dir"/>
        </authorization>
    </security-realm>
</security-realms>
```

Conditions préalables9

- Démarrer l'instance de JBoss EAP 6
- Lancer l'interface CLI.

Procédure 5.1. Supprimer l'authentification silencieuse du domaine de sécurité par défaut.

- **Supprimer l'authentification silencieuse par l'interface CLI**
  Supprimer l'élément **`local`** du Domaine de gestion et du Domaine d'applications comme requis.

  a. Supprimer l'élément **`local`** du Domaine de gestion.

  - **Pour les serveurs autonomes**

    ```
    /core-service=management/security-
    realm=ManagementRealm/authentication=local:remove
    ```

  - **Pour les domaines gérés**

```
/host=HOST_NAME/core-service=management/security-
realm=ManagementRealm/authentication=local:remove
```

b. Supprimer l'élément **`local`** du Domaine d'applications.

- **Pour les serveurs autonomes**

```
/core-service=management/security-
realm=ApplicationRealm/authentication=local:remove
```

- **Pour les domaines gérés**

```
/host=HOST_NAME/core-service=management/security-
realm=ApplicationRealm/authentication=local:remove
```

Résultat

L'authentification silencieuse est retirée du **`ManagementRealm`** et de l'**`ApplicationRealm`**.

Rapporter un bogue

5.5. DÉSACTIVER L'ACCÈS À DISTANCE DU SOUS-SYSTÈME JMX

Une connectivité à distance JMX vous permet de déclencher JDK et les opérations de gestion d'applications. Afin de garantir une installation, désactivez cette fonction. Vous pouvez faire cela soit en supprimant la configuration de la connexion à distance, soit en supprimant le sous-système JMX entièrement. Les commandes du JBoss CLI référencent le profil par défaut dans une configuration de domaine géré. Pour modifier un profil différent, modifiez la partie de la commande **`/profil=default`**. Pour un serveur autonome, retirez complètement cette partie de la commande.

NOTE

Dans un domaine géré, le connecteur distant est retiré du sous-système JMX par défaut. Cette commande est fournie pour votre information, au cas où vous souhaiteriez l'ajouter en cours de développement.

Exemple 5.5. Supprimez le connecteur distant du sous-système JMX

```
/profile=default/subsystem=jmx/remoting-connector=jmx/:remove
```

Exemple 5.6. Supprimez le Sous-système JMX

Exécutez cette commande pour chaque profil que vous utilisez, si vous utilisez un domaine géré.

```
/profile=default/subsystem=jmx/:remove
```

Rapporter un bogue

5.6. CONFIGURER LES DOMAINES DE SÉCURITÉ DES INTERFACES DE GESTION

Les interfaces de gestion utilisent des domaines de sécurité pour contrôler l'authentification et l'accès aux mécanismes de configuration de JBoss EAP 6. Un domaine de sécurité ressemble à un groupe Unix. Il s'agit en fait d'une base de données de noms d'utilisateurs et de mots de passe qui peuvent être utilisés pour authentifier les utilisateurs.

Domaine de gestion par défaut

Les interfaces de gestion sont configurées pour pouvoir utiliser le domaine de sécurité **`ManagementRealm`** par défaut. Le ManagementRealm stocke ses combinaisons d'utilisateurs et de mots de passe dans le fichier **`mgmt-users.properties`**.

Exemple 5.7. ManagementRealm par défaut

```
/host=master/core-service=management/security-
realm=ManagementRealm/:read-
resource(recursive=true,proxies=false,include-runtime=false,include-
defaults=true)
{
    "outcome" => "success",
    "result" => {
        "authorization" => undefined,
        "server-identity" => undefined,
        "authentication" => {"properties" => {
            "path" => "mgmt-users.properties",
            "plain-text" => false,
            "relative-to" => "jboss.domain.config.dir"
        }}
    }
}
```

Créer un nouveau domaine de sécurité

Les commandes suivantes créent un nouveau domaine de sécurité nommé **`TestRealm`** et définissent le répertoire du fichier de propriétés qui conviennent.

Exemple 5.8. Créer un nouveau domaine de sécurité

```
/host=master/core-service=management/security-realm=TestRealm/:add
/host=master/core-service=management/security-
realm=TestRealm/authentication=properties/:add(path=TestUsers.properties
, relative-to=jboss.domain.config.dir)
```

Configurer l'authentification dans un domane de sécurité par un domaine de sécurité déjà existant.

Pour utiliser un domaine de sécurité pour s'authentifier à des interfaces de gestion :

Il vous faudra, tout d'abord, créer un domaine de sécurité. Puis, le spécifier comme valeur de l'attribut **`security-realm`** de l'interface de gestion :

Exemple 5.9. Spécifier un domaine de sécurité à utiliser pour l'interface de gestion HTTP

```
/host=master/core-service=management/management-interface=http-
interface/:write-attribute(name=security-realm,value=TestRealm)
```

Rapporter un bogue

5.7. CONFIGURER LA CONSOLE DE GESTION POUR HTTPS

Configurer la console de gestion JBoss EAP pour communication **uniquement** via HTTPS offre une sécurité accrue. Tout le trafic réseau entre le client (navigateur web) et la console de gestion est encodé, ce qui réduit le risque d'attaque de sécurité comme une attaque man-in-the-middle. Toute personne qui administre une instance de JBoss EAP a des autorisations supérieures sur cette instance par rapport aux utilisateurs non privilégiés, et l'utilisation du protocole HTTPS permet de protéger l'intégrité et la disponibilité de cette instance.

Dans cette procédure, les communications encodées avec le domaine ou l'instance autonome de JBoss EAP sont désactivées. Les mots de passe qui utilisent la fonctionnalité **vault**, et les mots de passe utilisés dans les fichiers de configuration sont masqués.

Cette procédure s'applique à la fois aux modes de configuration en **`standalone`** ou en **`domain`**. En mode **`domain`**, donnez le préfixe de nom d'hôte aux commandes CLI, par exemple **`/host=master`**.

Procédure 5.2.

1. **Créez un keystore pour sécuriser la console de gestion.**

   NOTE

   Ce keystore doit être en format JKS car la console de gestion n'est pas compatible avec les keystores en format JCEKS.

   Dans un émulateur de terminal, saisissez la commande suivante. Pour les paramètres ***`alias`***, ***`keypass`***, ***`keystore`***, ***`storepass`*** et ***`dname`***, remplacez les valeurs en exemple par les valeurs de votre choix.

   Le paramètre ***`validity`*** indique le nombre de jours pendant la clé est valide. Une valeur de 730 correspondra à deux ans.

   ```
   keytool -genkeypair -alias appserver -storetype jks -keyalg RSA -
   keysize 2048 -keypass password1 -keystore
   EAP_HOME/standalone/configuration/identity.jks -storepass password1
   -dname "CN=appserver,OU=Sales,O=Systems Inc,L=Raleigh,ST=NC,C=US" -
   validity 730 -v
   ```

2. **Veillez à ce que la console de gestion soit bien reliée à HTTPS**

   - **Mode autonome**
     Veillez à ce que la Console de gestion soit reliée à **`HTTPS`** pour son interface en ajoutant la configuration **`management-https`** et en supprimant la configuration **`management-http`**.

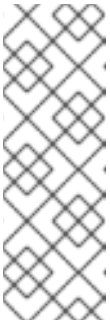
Assurez vous bien que l'instance JBoss EAP est en cours d'exécution, et saisir les commandes CLI de gestion suivantes :

```
/core-service=management/management-interface=http-
interface:write-attribute(name=secure-socket-binding,
value=management-https)
```

```
/core-service=management/management-interface=http-
interface:undefine-attribute(name=socket-binding)
```

La sortie de cette commande sera comme suit :

```
{"outcome" => "success"}
```



NOTE

À cet point dans le temps, le journal de JBoss EAP affichera sans doute le message d'erreur suivant. C'est normal car la configuration SSL n'est pas encore terminée.

```
JBAS015103: A secure port has been specified for the
HTTP interface but no SSL configuration in the realm.
```

- **Mode Domaine**
  Modifiez l'élément de socket de la section management-interface en ajoutant secure-port et en supprimant la configuration de port.

  Assurez vous bien que l'instance JBoss EAP est en cours d'exécution, et saisir les commandes CLI de gestion suivantes :

```
/host=master/core-service=management/management-interface=http-
interface:write-attribute(name=secure-port,value=9443)
/host=master/core-service=management/management-interface=http-
interface:undefine-attribute(name=port)
```

NOTE

À cet point dans le temps, le journal de JBoss EAP affichera sans doute le message d'erreur suivant. C'est normal car la configuration SSL n'est pas encore terminée.

```
JBAS015103: A secure port has been specified for the
HTTP interface but no SSL configuration in the realm.
```

3. **Option : personnalisation du groupe de socket-binding**
   Si vous utilisez un groupe ***`socket-binding`*** personnalisé, veillez à ce que la liaison ***`management-https`*** soit définie (présente par défaut, liée au port **`9443`**). Modifiez le fichier de configuration du master - par exemple **`standalone.xml`** - pour qu'il puisse correspondre à ce qui suit.

```
 <socket-binding-group name="standard-sockets" default-
interface="public" port-offset="${jboss.socket.binding.port-
offset:0}">
        <socket-binding name="management-native"
interface="management" port="${jboss.management.native.port:9999}"/>
        <socket-binding name="management-http"
interface="management" port="${jboss.management.http.port:9990}"/>
        <socket-binding name="management-https"
interface="management" port="${jboss.management.https.port:9443}"/>
```

4. **Créer un nouveau domaine de sécurité**
   Exécutez la commande suivante pour créer un nouveau domaine de sécurité nommé **`MyDomainRealm`**.

```
/host=master/core-service=management/security-
realm=ManagementRealmHTTPS/:add
/host=master/core-service=management/security-
realm=ManagementRealmHTTPS/authentication=properties/:add(path=Manag
ementUsers.properties, relative-to=jboss.domain.config.dir)
```

5. **Configurez l'interface de gestion avec le nouveau domaine de sécurité.**
   Saisir les commandes suivantes :

```
/host=master/core-service=management/management-interface=http-
interface/:write-attribute(name=security-
realm,value=ManagementRealmHTTPS)
```

6. **Configurez la console de gestion pour qu'elle puisse utiliser le keystore.**
   Saisissez la commande CLI suivante. Pour les paramètres ***`file`***, ***`password`*** et ***`alias`***, leurs valeurs doivent être copiées à partir de l'étape *Create a keystore to secure the management console* (Créer un keystore pour sécuriser la console de gestion).

```
/core-service=management/security-realm=ManagementRealmHTTPS/server-
identity=ssl:add(keystore-path=identity.jks,keystore-relative-
to=jboss.server.config.dir, keystore-password=password1,
alias=appserver)
```

   La sortie de cette commande sera comme suit :

```
{
    "outcome" => "success",
    "response-headers" => {
        "operation-requires-reload" => true,
        "process-state" => "reload-required"
    }
}
```

7. **Démarrez à nouveau le serveur JBoss EAP.**
   Au redémarrage du serveur, le journal doit contenir ce qui suit, juste avant le texte qui indique le nombre de services démarrés. La console de gestion est maintenant en écoute sur le port 9443, ce qui confirme que la procédure a réussi.

```
14:53:14,720 INFO  [org.jboss.as] (Controller Boot Thread)
```

```
JBAS015962: Http management interface listening on
https://127.0.0.1:9443/management
14:53:14,721 INFO  [org.jboss.as] (Controller Boot Thread)
JBAS015952: Admin console listening on https://127.0.0.1:9443
```

NOTE

Pour des raisons de sécurité, nous recommandons de masquer le mot de passe du kestore. Voir les détails ici Section 7.1, « Système d'archivage sécurisé de mots de passe ».

Rapporter un bogue

5.8. UTILISER DES INTERFACES DISTINCTES POUR LES CONNEXIONS HTTP ET HTTPS POUR L'INTERFACE DE GESTION.

L'interface de gestion est à l'écoute des connexions HTTP et HTTPS sur des interfaces distinctes. Un des scénarios pour ceci est d'écouter le traffic encrypté sur un réseau externe, et d'utiliser un traffic encrypté sur un réseau interne.

L'attribut **`secure-interface`** indique l'interface de réseau sur laquelle le socket de l'hôte doit être ouvert pour la communication de ka gestuib HTTPS, si une interface différente de celle spécifiée par l'attribut **`interface`** doit être utilisée. Si non spécifiée, l'interface spécifiée par l'attribut **`interface`** sera utilisée.

L'attribut **`secure-interface`** n'a aucun effet si l'attribut **`secure-port`** n'est pas défini.

Notez que lorsque le serveur reçoit le trafic HTTP ou HTTPS sur la *même* interface, les requêtes HTTPS reçues par le listener HTTP sont automatiquement redirigées vers le port HTTPS. Lorsque les interfaces *distinctes* sont utilisées pour le trafic HTTP et HTTPS, aucune redirection n'est effectuée lorsque le listener HTTP reçoit une demande HTTPS.

Voici un exemple de configuration ***`EAP_HOME`*`/domain/configuration/host.xml`** qui définit l'attribut **`secure-interface`** pour qu'il écoute le traffic HTTPS sur une interface différente du traffic HTTP :

```
<?xml version='1.0' encoding='UTF-8'?>

<host name="master" xmlns="urn:jboss:domain:3.0">

    <management>
        <security-realms>
            <security-realm name="ManagementRealm">
                <authentication>
                    <local default-user="$local" />
                    <properties path="mgmt-users.properties" relative-
to="jboss.domain.config.dir"/>
                </authentication>
            </security-realm>
        </security-realms>
        <management-interfaces>
            <native-interface security-realm="ManagementRealm">
                <socket interface="management"
port="${jboss.management.native.port:9999}"/>
```

```
            </native-interface>
            <http-interface security-realm="ManagementRealm">
                <socket interface="management"
port="${jboss.management.http.port:9990}" secure-
port="${jboss.management.https.port:9943}" secure-interface="secure-
management"/>
            </http-interface>
        </management-interfaces>
    </management>

    <domain-controller>
        <local/>
        <!-- Alternative remote domain controller configuration with a
host and port -->
        <!-- <remote host="${jboss.domain.master.address}"
port="${jboss.domain.master.port:9999}" security-realm="ManagementRealm"/>
-->
    </domain-controller>

    <interfaces>
        <interface name="management">
            <inet-address
value="${jboss.bind.address.management:127.0.0.1}"/>
        </interface>
        <interface name="secure-management">
            <inet-address value="${jboss.bind.address:10.10.64.1}"/>
        </interface>
    </interfaces>
</host>
```

Rapporter un bogue

5.9. AUTHENTIFICATION SSL 2-WAY DANS L'INTERFACE DE GESTION ET DANS LE CLI

L'authentification SSL 2-way, également connu sous le nom *authentification du client*, authentifie le client et le serveur à l'aide de certificats SSL. Cela garantit que le serveur soit non seulement qui il déclare être, mais le client également.

Dans cette section, on utilisera les conventions suivantes :

HOST1

Le nom d'hôte du serveur JBoss. Par exemple ; **`jboss.redhat.com`**

HOST2

Un nom qui convient au client. Par exemple : **`myclient`**. Notez qu'il ne s'agit par forcément d'un nom d'hôte.

CA_HOST1

Le DN (distinguished name) utilisé pour le certificat *HOST1*. Par exemple **`cn=jboss,dc=redhat,dc=com`**.

CA_HOST2

Le DN (distinguished name) utilisé pour le certificat *HOST2*. Par exemple **cn=myclient,dc=redhat,dc=com**.

Pré-requis

- Si vous souhaitez utiliser l'archivage de mots de passe pour stocker les mots de passe de keystore et de truststore (recommandé), l'archivage de mots de passe doit déjà avoir été créé. Voir Section 7.1, « Système d'archivage sécurisé de mots de passe ».

Procédure 5.3.

1. Générez les stores :

```
keytool -genkeypair -alias HOST1_alias -keyalg RSA -keysize 1024 -
validity 365 -keystore host1.keystore.jks -dname "CA_HOST1" -keypass
secret -storepass secret
```

```
keytool -genkeypair -alias HOST2_alias -keyalg RSA -keysize 1024 -
validity 365 -keystore host2.keystore.jks -dname "CA_HOST2" -keypass
secret -storepass secret
```

2. Exportez les certificats :

```
keytool -exportcert  -keystore HOST1.keystore.jks -alias HOST1_alias
-keypass secret -storepass secret -file HOST1.cer
```

```
keytool -exportcert  -keystore HOST2.keystore.jks -alias HOST2_alias
-keypass secret -storepass secret -file HOST2.cer
```

3. Importe des certificats dans les trust stores opposés :

```
keytool -importcert -keystore HOST1.truststore.jks -storepass secret
-alias HOST2_alias -trustcacerts -file HOST2.cer
```

```
keytool -importcert -keystore HOST2.truststore.jks -storepass secret
-alias HOST1_alias -trustcacerts -file HOST1.cer
```

4. Définir un CertificateRealm dans la configuration de votre installation (**host.xml** ou **standalone.xml**) et y pointez l'interface :

   Cela peut se faire en modifiant le fichier de configuration manuellement (non recommandé) ou en utilisant les commandes suivantes :

```
/core-service=management/security-realm=CertificateRealm:add()
```

```
/core-service=management/security-realm=CertificateRealm/server-
identity=ssl:add(keystore-path=/path/to/HOST1.keystore.jks,keystore-
password=secret, alias=HOST1_alias)
```

```
/core-service=management/security-
realm=CertificateRealm/authentication=truststore:add(keystore-
path=/path/to/HOST1.truststore.jks,keystore-password=secret)
```

IMPORTANT

Les commandes données ne s'appliquent qu'en mode autonome. Pour le mode domaine, ajoutez **/host=master** avant chaque commande.

5. Modifiez le ***security-realm*** de l'interface native du domaine de sécurité.

```
/host=master/core-service=management/management-interface=native-
interface:write-attribute(name=security-
realm,value=CertificateRealm)
```

6. Ajoutez la configuration SSL au CLI, utilisant le fichier ***EAP_HOME*/bin/jboss-cli.xml** pour les configurations. Utilisez soit un archivage de mots de passe pour stocker les mots de passe de keystore ou de truststore (recommandé), ou les stocker en texte brut :

   - Pour stocker les mots de passe de keystore et de truststore dans un archivage de mots de passe :

     Modifiez ***EAP_HOME*/bin/jboss-cli.xml** et ajoutez la configuration SSL (en utilisant les valeurs appropriées pour les variables). Ajoutez également la configuration de l'archivage, en remplaçant chaque valeur par celle de votre archivage.

```
<ssl>
  <vault>
    <vault-option name="KEYSTORE_URL" value="path-
to/vault/vault.keystore"/>
    <vault-option name="KEYSTORE_PASSWORD" value="MASK-
5WNXs8oEbrs"/>
    <vault-option name="KEYSTORE_ALIAS" value="vault"/>
    <vault-option name="SALT" value="12345678"/>
    <vault-option name="ITERATION_COUNT" value="50"/>
    <vault-option name="ENC_FILE_DIR" value="path-to/jboss-
eap/vault/"/>
  </vault>
  <alias>$HOST2alias</alias>
  <key-store>/path/to/HOST2.keystore.jks</key-store>
  <key-store-password>VAULT::VB::cli_pass::1</key-store-password>
  <key-password>VAULT::VB::cli_pass::1</key-password>
  <trust-store>/path/to/HOST2.truststore.jks</trust-store>
  <trust-store-password>VAULT::VB::cli_pass::1</trust-store-
password>
  <modify-trust-store>true</modify-trust-store>
</ssl>
```

   - Pour stocker les mots de passe de keystore et de truststore en texte brut :

     Modifiez ***EAP_HOME*/bin/jboss-cli.xml** et ajoutez la configuration SSL (en utilisant les valeurs appropriées pour les variables) :

```
<ssl>
  <alias>$HOST2alias</alias>
  <key-store>/path/to/HOST2.keystore.jks</key-store>
  <key-store-password>secret</key-store-password>
  <trust-store>/path/to/HOST2.truststore.jks</trust-store>
  <trust-store-password>secret</trust-store-password>
  <modify-trust-store>true</modify-trust-store>
</ssl>
```

Rapporter un bogue

5.10. SÉCURISER LES INTERFACES DE GESTION VIA JAAS

Pour utiliser JAAS pour s'authentifier à des interfaces de gestion :

Tout d'abord, créez un domaine de sécurité avec le module de connexion UserRoles :

```
/subsystem=security/security-domain=UsersLMDomain:add(cache-type=default)
/subsystem=security/security-
domain=UsersLMDomain/authentication=classic:add
/subsystem=security/security-
domain=UsersLMDomain/authentication=classic/login-module=UsersRoles:add()
```

Puis, créez un domaine de sécurité avec l'authentification JAAS :

```
/core-service=management/security-realm=SecurityDomainAuthnRealm:add
/core-service=management/security-
realm=SecurityDomainAuthnRealm/authentication=jaas:add(name=UsersLMDomain)
```

L'attribut **`assign-groups`** détermine si les informations chargées d'appartenance à un groupe de l'utilisateur en provenance du domaine de sécurité sont utilisées pour l'allocation à un groupe dans le domaine de sécurité. Si défini sur **`true`**, cette allocation de groupe sera utilisée pour le RBAC (Role-Based Access Control).

L'attribut **`assign-groups`** peut être défini à true par la commande CLI :

```
/core-service=management/security-
realm=ManagementRealm/authentication=jaas:write-attribute(name=assign-
groups,value=true)
```

Rapporter un bogue

5.11. LDAP

5.11.1. LDAP

Lightweight Directory Access Protocol (LDAP) est un protocole pour le stockage et la distribution d'informations en provenance de répertoires à travers un réseau. Ces informations de répertoires incluent des informations sur les utilisateurs, les périphériques physiques, les rôles d'accès et de restrictions et autres informations.

Certaines de implémentations communes de LDAP incluent OpenLDAP, Microsoft Active Directory, IBM Tivoli Directory Server, Oracle Internet Directory, et autres.

La plateforme JBoss EAP 6 comprend plusieurs modules d'authentification et d'autorisation qui vous permettent d'utiliser un serveur LDAP comme autorité d'authentification et d'autorisation pour vos applications Web et EJB.

Rapporter un bogue

5.11.2. Utiliser LDAP pour vous authentifier auprès des interfaces de gestion

Pour utiliser un serveur de répertoire LDAP comme source d'authentification pour la console de gestion, l'interface CLI ou l'API de gestion, vous devez effectuer les procédures suivantes :

1. Créer une connexion sortante au serveur LDAP.
2. Créer un domaine de sécurité activé-LDAP.
3. Référencer le nouveau domaine de sécurité dans l'interface de Gestion.

Créer une Connexion sortante au serveur LDAP

La connexion sortante LDAP autorise les attributs suivants :

Tableau 5.1. Attributs d'une connexion sortante LDAP

Attribut	Requis	Description
`url`	oui	L'adresse URL du serveur de répertoire.
`search-dn`	non	Le nom distinctif (DN) de l'utilisateur autorisé à effectuer des recherches.
`search-credentials`	non	Le mot de passe de l'utilisateur autorisé à effectuer des recherches.
`initial-context-factory`	non	L'usine de contexte initiale à utiliser quand on établit une connexion. Valeur par défaut **`com.sun.jndi.ldap.LdapCtxFactory`**.
`security-realm`	non	Domaine de sécurité à référencer pour obtenir un **`SSLContext`** configuré pour établir la connexion.

Exemple 5.10. Ajouter une connexion sortante LDAP

Cet exemple ajoute une connexion sortante par le jeu de propriétés suivant :

- Search DN: **`cn=search,dc=acme,dc=com`**
- Search Credential: **`myPass`**
- URL: **`ldap://127.0.0.1:389`**

La première commande ajoute le domaine de sécurité.

```
/host=master/core-service=management/security-
realm=ldap_security_realm:add
```

La seconde commande ajoute la connexion LDAP.

```
/host=master/core-service=management/ldap-
connection=ldap_connection/:add(search-
credential=myPass,url=ldap://127.0.0.1:389,search-
dn="cn=search,dc=acme,dc=com")
```

Créer un domaine de sécurité activé-LDAP

Les interfaces de gestion peuvent s'authentifier sur le serveur LDAP au lieu des domaines de sécurité basés fichiers de propriété et configurés par défaut. L'authentificateur LDAP fonctionne en établissant tout d'abord une connexion vers le serveur de répertoires distant. Il effectue ensuite une recherche en utilisant le nom d'utilisateur que l'utilisateur a transmis au système d'authentification, afin de trouver le nom distinctif complet (DN) du dossier LDAP. Une nouvelle connexion est alors établie, utilisant le DN de l'utilisateur comme information d'identification et mot de passe fournis par l'utilisateur. Si cette authentification au serveur LDAP réussit, le DN est considéré comme valide.

Le domaine de sécurité LDAP utilise les attributs de configuration suivants :

connection

Le nom de la connexion défini dans ***`outbound-connections`*** à utiliser pour se connecter au répertoire LDAP.

advanced-filter

Le filtre complet utilisé pour rechercher un utilisateur basé sur l'ID d'utilisateur fourni. Le filtre doit contenir une variable suivant le format suivant : **`{0}`**. Sera plus tard remplacé par le nom d'utilisateur fourni par l'utilisateur.

base-dn

Le nom distinctif (DN) du contexte pour commencer à chercher l'utilisateur.

recursive

Indique si la recherche doit être récursive dans toute l'arborescence de répertoires LDAP, ou si l'on doit rechercher uniquement le contexte spécifié. La valeur par défaut est **`false`**.

user-dn

Attribut de l'utilisateur qui contient le nom distinctif (DN). Utilisé par la suite pour tester l'authentification. Valeur par défaut **dn**.

username-attribute

Le nom de l'attribut à rechercher pour l'utilisateur. Ce filtre effectue une recherche simple où le nom d'utilisateur entré par l'utilisateur correspond à l'attribut spécifié.

allow-empty-passwords

Cet attribut détermine si un mot de passe vide est accepté. La valeur par défaut pour cet attribut est **`false`**.

Soit `username-filter` ou `advanced-filter`, devra être spécifié.

L'attribut ***`advanced-filter`*** contient une recherche par filtre dans la syntaxe standard LDP, par exemple :

```
(&(sAMAccountName={0})(memberOf=cn=admin,cn=users,dc=acme,dc=com))
```

Exemple 5.11. XML représentant un domaine de sécurité activé-LDAP

Cet exemple utilise les paramètres suivants :

- connection - **`ldap_connection`**
- base-dn - **`cn=users,dc=acme,dc=com`**.
- username-filter - **`attribute="sambaAccountName"`**

```
<security-realm name="ldap_security_realm">
   <authentication>
      <ldap connection="ldap_connection" base-
dn="cn=users,dc=acme,dc=com">
         <username-filter attribute="sambaAccountName" />
      </ldap>
  </authentication>
</security-realm>
```

AVERTISSEMENT

Il est important de veiller à ne pas autoriser les mots de passe LDAP. À moins que vous désiriez les avoir dans votre environnement, ils représentent un problème de sécurité sérieux.

EAP 6.1 inclut un correctif pour CVE-2012-5629, qui définit l'option allowEmptyPasswords des modules de connexion LDAP à false si l'option n'est pas déjà configurée. Pour les versions plus anciennes, cette option devra être configurée manuellement.

Exemple 5.12. Ajout d'un domaine de sécurité LDAP

La commande ci-dessous ajoute une authentification LDAP au domaine de sécurité et définit ses attributs pour un master nommé par l'hôte dans le domaine.

```
/host=master/core-service=management/security-
realm=ldap_security_realm/authentication=ldap:add(base-
```

```
dn="DC=mycompany,DC=org", recursive=true, username-
attribute="MyAccountName", connection="ldap_connection")
```

Appliquer le nouveau domaine de sécurité à l'Interface de gestion

Après avoir créé un domaine de sécurité, vous devez le référencer dans la configuration de votre interface de gestion. L'interface de gestion utilisera le domaine de sécurité pour l'authentification HTTP digest.

Exemple 5.13. Ajouter le domaine de sécurité à l'interface HTTP

Une fois que la configuration est en place, et que vous aurez démarré à nouveau le contrôleur d'hôtes, la console de gestion basée-web utilisera LDAP pour authentifier ses utilisateurs.

```
/host=master/core-service=management/management-interface=http-
interface/:write-attribute(name=security-
realm,value=ldap_security_realm)
```

Exemple 5.14. Ajouter le domaine de sécurité à l'interface native

Utiliser la commande suivante pour appliquer les mêmes paramètres à l'interface native :

```
/host=master/core-service=management/management-interface=native-
interface/:write-attribute(name=security-
realm,value=ldap_security_realm)
```

Rapporter un bogue

5.11.3. Utiliser la sortie LDAP avec SSL 2-ways dans l'interface de gestion et le CLI

JBoss EAP 6 peut être configuré pour utiliser une connexion sortante vers un serveur LDAP en utilisant un SSL 2-way pour l'authentification dans l'interface de gestion et le CLI.

Pré-requis

- On doit créer un domaine de sécurité activé-LDAP. Voir Section 5.11.2, « Utiliser LDAP pour vous authentifier auprès des interfaces de gestion » pour obtenir des informations supplémentaires sur la façon de créer le domaine de sécurité.

Procédure 5.4. Configurer la sortie LDAP avec un SSL 2-way

1. Configurer lekeystore et le truststore du domaine de sécurité. Le domaine de sécurité doit contenir un keystore configuré avec la clé que le serveur JBoss EAP 6 utilise pour s'authentifier auprès du serveur LDAP. Le domaine de sécurité doit également contenir un truststore configuré avec les certificats du serveur LDAP. Voir Section 5.9, « Authentification SSL 2-way dans l'Interface de gestion et dans le CLI » pour obtenir des instructions sur la façon de configurer les keystores et les truststores.

2. Ajouter une connexion sortante dans le serveur LDAP, spécifiant le domaine de sécurité configuré :

```
/core-service=management/ldap-
connection=LocalLdap:add(url="ldaps://LDAP_HOST:LDAP_PORT")

/core-service=management/ldap-connection=LocalLdap:write-
attribute(name=security-realm,value="LdapSSLRealm")
```

3. Configurer l'authentification LDAP dans le domaine de sécurité et les interfaces de gestion comme indiqué dans Section 5.11.2, « Utiliser LDAP pour vous authentifier auprès des interfaces de gestion ».

Rapporter un bogue

CHAPITRE 6. ACTIVER LES INTERFACES DE GESTION PAR LE CONTRÔLE D'ACCÈS BASÉ RÔLE

6.1. LES RBAC (ROLE-BASED ACCESS CONTROL)

Le Contrôle d'accès basé sur rôle (RBAC) est un mécanisme permettant de spécifier un jeu de permissions pour la gestion par les utilisateurs. Il permet à plusieurs utilisateurs de partager la responsabilité de gérer des serveurs JBoss EAP 6.3 sans qu'aucun d'entre eux n'ait besoin d'un accès illimité. En fournissant « une séparation des fonctions » pour la gestion des utilisateurs, JBoss EAP 6.3 facilite la tâche à une organisation de répandre la responsabilité entre individus ou groupes, sans octroi de privilèges inutiles. Cela garantit la sécurité maximale de vos serveurs et de vos données, tout en offrant une souplesse de configuration, de déploiement et de gestion.

Les RBAC (Role-Based Access Control) dans JBoss EAP 6.3 fonctionnent par une combinaison de permissions et de contraintes associées à des rôles.

Il y a sept rôles prédéfinis associés à des autorisations fixes différentes. Les rôles prédéfinis sont : Surveillant (Monitor), Opérateur (Operator), Mainteneur (Maintainer), Déployeur (Deployer), Auditeur (Auditor), Administrateur (Administrator) et Superutilisateur (Superuser). A chaque utilisateur de gestion est attribué un ou plusieurs rôles, qui définissent ce que l'utilisateur est autorisé à faire pour la gestion du serveur.

Rapporter un bogue

6.2. LES RBAC (ROLE-BASED ACCESS CONTROL) DANS LA CONSOLE DE GESTION ET LE CLI

Lorsque le contrôle d'accès basé sur les rôles (RBAC) est activé, le rôle assigné à un utilisateur déterminera les ressources auxquelles il aura accès et les opérations qu'il pourra effectuer avec des attributs de ressources.

La console de gestion

Dans la console de gestion, certains contrôles et vues sont désactivés (en gris) ou invisibles selon les permissions du rôle assignées à l'utilisateur.

Si vous n'avez pas de permission de lecture pour un attribut de ressource, cet attribut apparaîtra en blanc sur la console. Ainsi, la plupart des rôles ne peuvent pas lire les champs Nom d'utilisateur ou Mot de passe des sources de données.

Si vous n'avez pas de permission d'écriture sur un attribut de ressource, cet attribut sera désactivé (grisé) dans le formulaire de modification de la ressource. Si vous n'avez pas de permission d'écriture sur la ressource, le bouton de modification de la ressource ne s'affichera pas.

Si un utiliisateur ne possède pas les permissions nécessaires pour accéder à une ressource ou à un attribut (c'est « non adressable » ce pour rôle), ce n'apparaîtra pas dans la console. L'accès contrôle système lui-même uniquement visible par quelques rôles par défaut en est un exemple.

l'interface CLI ou l'API

Les utilisateurs de l'outil **`jboss-cli.sh`** ou de l'API rencontreront un comportement légèrement différent dans l'API lorsque RBAC est activé.

Les ressources et les attributs qui ne peuvent être lus sont filtrés des résultats. Si les éléments filtrés

sont adressables par le rôle, leurs noms seront répertoriés en tant que **`filtered-attributes`** dans la section **`response-headers`** du résultat. Si une ressource ou un attribut n'est pas adressable par le rôle, il n'apparaîtra pas.

Toute tentative d'accès à une ressource non adressable résultera par l'erreur **`resource not found`**.

Si un utilisateur tente d'écrire ou de lire une ressource qu'il peut adresser, mais qu'il n'a pas les permissions de lecture ou d'écriture qui conviennent, une erreur **`Permission Denied`** apparaîtra.

Rapporter un bogue

6.3. SCHÉMAS D'AUTHENTIFICATION SUPPORTÉS

Les RBAC (Role-Based Access Control) fonctionnent avec les fournisseurs d'authentification standards inclus dans JBoss EAP 6.3, comme : **`username/password`**, **`client certificate`**, et **`local user`**.

Username/Password

Les utilisateurs sont authentifiés par une combinaison de nom d'utilisation et de mot de passe qui sont ensuite vérifiés par le fichier **`mgmt-users.properties`**, ou via un serveur LDAP.

Client Certificate

Utilisation du Trust Store

Local User

`jboss-cli.sh` authentifie automatiquement en tant qu'utilisateur local si le serveur exécute sur la même machine. Par défaut, l'utilisateur local est un membre du groupe **`SuperUser`**.

Quel que soit le fournisseur utilisé, JBoss EAP est responsable de l'attribution des rôles aux utilisateurs. Toutefois, lors de l'authentification avec le fichier **`mgmt-users.properties`** ou par un serveur LDAP, ces systèmes peuvent fournir des informations de groupes d'utilisateur. Cette information peut également servir à JBoss EAP pour attribuer des rôles aux utilisateurs.

Rapporter un bogue

6.4. LES RÔLES STANDARD

JBoss EAP 6 fournit sept rôles prédéfinis par l'utilisateur : Monitor, Operator, Maintainer, Deployer, Auditor, Administrator, et SuperUser. Chacun de ces rôles possède un ensemble de permissions différentes conçues pour les cas d'utilisation spécifiques. Les rôles Monitor, Operator, Maintainer, Deployer, Auditor, Administrator, et SuperUser se superposent, avec chaque rôle possédant plus d'autorisations que le précédent. Les rôles Auditor et Deployer sont semblables aux rôles Monitor et Maintainer respectivement mais ont leurs propres permissions et restrictions supplémentaires.

Monitor

Les utilisateurs du rôle Monitor ont le plus petit nombre de permissions et ne peuvent lire que la configuration ou l'état du serveur en cours. Ce rôle est destiné aux utilisateurs qui ont besoin de vérifier et reporter la performance du serveur.

Les Monitors ne peuvent pas modifier la configuration du serveur, et ne peuvent pas accéder aux données ou opérations confidentielles.

Operator

Le rôle Operator étend le rôle Monitor en ajoutant la possibilité de modifier l'état d'exécution du serveur. Cela signifie que les opérateurs peuvent recharger et arrêter le serveur, ainsi que suspendre et reprendre les destinations JMS. Le rôle Operator est idéal pour les utilisateurs responsables des hôtes physiques ou virtuels du serveur d'applications, puisqu'ils peuvent s'assurer ainsi que les serveurs puissent être arrêtés, redémarrés ou corrigés si nécessaire.

Le opérateurs ne peuvent pas modifier la configuration du serveur ou accéder à des données ou opérations confidentielles.

Maintainer

Le rôle Maintainer a le droit de visualiser et modifier l'état d'exécution et toute la configuration à l'exception des données sensibles et des opérations. Le rôle du mainteneur est un rôle d'usage général qui n'a pas accès aux données sensibles et aux opérations. Le rôle Maintainer permet aux utilisateurs de bénéficier d'un accès presque total d'administration du serveur, sans donner à ces utilisateurs l'accès aux mots de passe ou autres informations sensibles.

Le mainteneurs ne peuvent pas accéder à des données ou opérations sensibles.

Administrator

Le rôle Administrator a un accès illimité aux ressources et opérations sur le serveur sauf sur le système d'enregistrement d'audit. Le rôle d'administrateur a accès à des données sensibles et aux opérations. Ce rôle peut également configurer le système de contrôle d'accès. Le rôle Administrator n'est requis que lorsque vous manipulez des données sensibles ou configurez les utilisateurs et les rôles.

Les administrateurs ne peuvent pas accéder au système de journalisation d'auditing et ne peuvent pas se changer eux-mêmes en rôles Auditor ou SuperUser.

SuperUser

Le rôle SuperUser a un accès illimité aux ressources et opérations sur le serveur, y compris au système d'enregistrement d'audit. Ce rôle est équivalent aux rôles d'utilisateurs administrateurs des versions plus anciennes de JBoss EAP 6 (6.0 et 6.1). Si RBAC est désactivé, tous les utilisateurs de management auront une permission équivalente à celle d'un rôle SuperUser.

Deployer

Le rôle Deployer a les mêmes permissions que Monitor, mais peut modifier la configuration et l'état des déploiements ou n'importe quel autre type de ressource activée en tant que ressource d'application.

Auditor

Le rôle Auditor possède toutes les permissions du rôle Monitor et peut également voir (mais pas modifier) les données sensibles. Il a aussi accès au système de journalisation de l'auditing. Le rôle Auditor est le seul rôle en dehors de SuperUser qui puisse accéder au système de journalisation de l'auditing.

Les auditeurs ne peuvent pas modifier les données ou les ressources sensibles. Seul l'accès lecture est permis.

Rapporter un bogue

6.5. LES PERMISSIONS DE RÔLE

Ce que chaque rôle peut faire est déterminé par ses permissions. Chaque rôle ne possède que quelques permissions. Seul le SuperUser possède toutes les permissions et le Monitor n'en possède que très peu.

Chaque permission peut donner un accès lecture et/ou écriture à une seule catégorie de ressource.

Les catégories sont les suivantes : l'état d'exécution, la configuration du serveur, les données sensibles, audit log, et le système de contrôle d'accès.

Tableau 6.1, « Matrice de permissions de rôle » résume les permissions de chaque rôle.

Tableau 6.1. Matrice de permissions de rôle

	Monitor	Operator	Maintainer	Deployer	Auditor	Admin	SuperUser
Lecture Config et État	X	X	X	X	X	X	X
Lire Données sensibles [2]					X	X	X
Modifier Données sensibles [2]						X	X
Lire/Modifer Audit Log					X		X
Modifier État Runtime		X	X	X[1]		X	X
Modifier Config persistante			X	X[1]		X	X
Lire/Modifier Contrôle d'accès						X	X

[1] les permissions sont limitées aux ressources d'applications.

[2] les ressources considérées comme "sensibles" sont configurées par les contraintes de sensibilité.

Rapporter un bogue

6.6. CONTRAINTES

Les contraintes sont des jeux de configuration de contrôle d'accès désignés correspondant à une liste de ressources. Le système RBAC utilise la combinaison des contraintes et des permissions de rôle pour déterminer si un utilisateur peut exécuter une action de gestion.

Les contraintes sont divisées en trois classifications : application, sensibilité et expression d'archivage.

Contraintes d'application

Les contraintes d'application définissent des ensembles de ressources et d'attributs accessibles aux utilisateurs du rôle chargé du déploiement (rôle Deployer). Par défaut, la seule Contrainte d'application activée est la principale qui inclut des déploiements, et des superpositions de déploiement. Les contraintes d'application sont également incluses (mais pas activées par défaut) pour les sources de données, connexion, mail, messagerie, nommage, adaptateurs de ressources et sécurité. Ces contraintes permettent aux utilisateurs Deployer de non seulement de déployer des applications, mais également de configurer et de maintenir les ressources requises par ces applications.

La configuration de contraintes d'applications se trouve dans l'API de gestion à l'adresse suivante : **`/core-service=management/access=authorization/constraint=application-classification`**.

Contraintes de sensibilité

Les contraintes de confidentialité définissent des ressources considérées comme « sensibles ». Une ressource sensible est généralement de nature secrète, comme un mot de passe, ou une information pouvant avoir de graves répercussions sur le fonctionnement du serveur, comme le networking, la configuration de la JVM ou les propriétés système. Le système de contrôle d'accès lui-même est aussi considéré comme sensible.

Les seuls rôles autorisés à écrire dans les ressources sensibles sont Administrateur et Superutilisateur. Le rôle Auditor est seulement capable de lire les ressources confidentielles. Aucun autre rôle ne peut avoir accès.

La configuration de contraintes de sensibilité se trouve dans l'API de gestion à l'adresse suivante : **`/core-service=management/access=authorization/constraint=sensitivity-classification`**.

Contrainte d'expression d'archivage sécurisé

Une contrainte d'expression d'archivage sécurisé détermine si la lecture ou l'écriture d'expressions d'archivage sécurisé est considéré comme une opération sensible. Par défaut, les deux sont sensibles.

La configuration de contraintes d'expression d'archivage sécurisé se trouve dans l'API de gestion à l'adresse suivante : **`/core-service=management/access=authorization/constraint=vault-expression`**.

Les contraintes ne peuvent pas être configurées dans la console de gestion actuellement.

Rapporter un bogue

6.7. JMX ET RBAC (ROLE-BASED ACCESS CONTROL)

RBAC s'applique à JMX de trois façons :

1. L'API de gestion de JBoss EAP 6 est exposé comme JMX Management Beans. Ces Management Beans sont appelés « core mbeans » et leur accès est contrôlé et filtré exactement de la même façon que l'API de gestion sous-jacent lui-même.

2. Le sous-système JMX est configuré avec des permissions d'écriture « sensibles ». Cela signifie que seuls les utilisateurs ayant pour rôle Administrateur et Superutilisateur peuvent apporter des modifications à ce sous-système. Les utilisateurs avec le rôle Auditeur peuvent aussi lire cette configuration du sous-système.

3. Par défaut, les beans de gestion enregistrés par des applications déployées et des services (non-core mbeans) sont accessibles par tous les utilisateurs de gestion, mais seuls les utilisateurs ayant un rôle Maintenance, Opérateur, Administrateur, Superutilisateur peuvent écrire dessus.

Rapporter un bogue

6.8. CONFIGURER LE RBAC (ROLE-BASED ACCESS CONTROL)

6.8.1. Aperçu des tâches de configuration RBAC

Quand RBAC est activé, seuls les utilisateurs qui possèdent le rôle Administration ou SuperUtilisateur peuvent voir ou modifier le système de contrôle d'accès.

La console de gestion fournit une interface pour les tâches RBAC standard suivantes :

- Voir et configurer les rôles assignés (ou exclus) pour chaque utilisateur
- Voir et configurer les rôles assignés (ou exclus) pour chaque groupe
- Voir Appartenance Utilisateur et Groupe par Rôle.
- Configurer l'appartenance par défaut d'un rôle.
- Créer un scoped rôle

Le CLI donne accès au système de contrôle d'accès total. Cela signifie que tout ce qui peut être fait à partir de la console de gestion doit être fait à cet endroit, mais qu'un nombre de tâches supplémentaires peut être fait via le CLI à défaut du système de contrôle d'accès.

Les tâches supplémentaires suivantes peuvent être faites par le CLI :

- Activer et désactiver RBAC
- Modifier la police de combinaison de permissions
- Configurer les contraintes de sensibilité des ressources et des ressources d'applications

Rapporter un bogue

6.8.2. Activer le RBAC (Role-Based Access Control)

Le système de contrôle d'accès basé sur les rôles (RABC) est désactivé par défaut. Il est activé en

changeant l'attribut provider en le faisant passer de **simple** à **rbac**. Cela peut être fait en utilisant l'outil **jboss-cli.sh**, ou en modifiant la configuration du serveur XML du fichier si le serveur est hors ligne. Quand RBAC est désactivé ou activé sur un serveur en cours d'exécution, la configuration du serveur doit être rechargée avant de prendre effet.

Une fois activé, ne peut être désactivé que par un utilisateur ayant les rôles Adminstrateur ou Superutilisateur. Par défaut, le **jboss-cli.sh** est exécuté par le rôle **SuperUser** s'il est exécuté dans la même machine que le serveur.

Procédure 6.1. Activer RBAC

- Pour activer RBAC avec **jboss-cli.sh** utiliser l'opération **write-attribute** de la ressource d'autorisation d'accès, pour définir la valeur de l'attribut provider à **rbac**

```
/core-service=management/access=authorization:write-
attribute(name=provider, value=rbac)
```

```
[standalone@localhost:9999 /] /core-
service=management/access=authorization:write-
attribute(name=provider, value=rbac)
{
    "outcome" => "success",
    "response-headers" => {
        "operation-requires-reload" => true,
        "process-state" => "reload-required"
    }
}
[standalone@localhost:9999 /] /:reload
{
    "outcome" => "success",
    "result" => undefined
}
```

Procédure 6.2. Désactiver RBAC

- Pour désactiver RBAC avec **jboss-cli.sh** utiliser l'opération **write-attribute** de la ressource d'autorisation d'accès pour définir la valeur de l'attribut provider à **simple**

```
/core-service=management/access=authorization:write-
attribute(name=provider, value=simple)
```

```
[standalone@localhost:9999 /] /core-
service=management/access=authorization:write-
attribute(name=provider, value=simple)
{
    "outcome" => "success",
    "response-headers" => {
        "operation-requires-reload" => true,
        "process-state" => "reload-required"
    }
}
[standalone@localhost:9999 /] /:reload
{
    "outcome" => "success",
```

```
    "result" => undefined
}
```

Si le serveur est hors ligne, la configuration XML peuvent être sur RBAC « activé » ou « désactivé ». Pour ce faire, modifiez l'attribut **`provider`** de l'élément de contrôle d'accès de l'élément **`access-control`**. Définissez la valeur **`rbac`** sur « activé » et **`simple`** sur « désactivé ».

```
<management>

        <access-control provider="rbac">
            <role-mapping>
                <role name="SuperUser">
                    <include>
                        <user name="$local"/>
                    </include>
                </role>
            </role-mapping>
        </access-control>

    </management>
```

Rapporter un bogue

6.8.3. Modifier la police de combinaison de permissions

La police de combinaison de permissions détermine comment les permissions sont définies quand un utilisateur possède plus d'un seul rôle. Peut être définie sur **`permissive`** ou **`rejecting`**. La valeur par défaut est **`permissive`**.

Quand elle est définie à **`permissive`**, si un rôle est assigné à l'utilisateur pour permettre une action, alors l'action sera autorisée.

Quand définie sur **`rejecting`**, si plusieurs rôles sont assignés à un utilissateur, alors aucune action ne sera requise. Lorsque la police est définie à **`rejecting`**, chaque utilisateur se verra assigner un rôle unique. Les utilisateurs ayant des rôles multiples ne seront pas en mesure d'utiliser l'outil de gestion de console ou **`jboss-cli.sh`** lorsque la police est définie sur **`rejecting`**.

La police de combinaison d'autorisations est configurée en affectant à l'attribut **`permission-combination-policy`** soit **`permissive`** ou **`rejecting`**. Cela peut être fait par l'outil **`jboss-cli.sh`**, ou en modifiant la configuration du serveur XML du fichier si le serveur est hors ligne.

Procédure 6.3. Définir la police de combinaison de permissions

- Effectuer l'opération **`write-attribute`** de la ressource d'autorisation d'accès pour définir l'attribut **`permission-combination-policy`** au nom de la police requise.

  ```
  /core-service=management/access=authorization:write-
  attribute(name=permission-combination-policy, value=POLICYNAME)
  ```

  Les noms de police valide sont « rejecting » ou « permissive ».

  ```
  [standalone@localhost:9999 /] /core-
  service=management/access=authorization:write-
  attribute(name=permission-combination-policy, value=rejecting)
  ```

```
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

Si le serveur est hors ligne, la configuration XML peut être modifiée pour changer la valeur de la politique de combinaison de permissions. Pour ce faire, modifiez l'attribut **`permission-combination-policy`** de l'élément de contrôle d'accès.

```
<access-control provider="rbac" permission-combination-policy="rejecting">
  <role-mapping>
    <role name="SuperUser">
      <include>
        <user name="$local"/>
      </include>
    </role>
  </role-mapping>
</access-control>
```

Rapporter un bogue

6.9. GESTION DES RÔLES

6.9.1. Appartenance à un rôle

Lorsque le contrôle d'accès basé sur les rôles (RBAC) est activé, ce qu'un utilisateur de gestion est autorisé à faire est déterminé par les rôles auxquels l'utilisateur est affecté. JBoss EAP 6.3 utilise un système d'inclusion et exclusion basé sur l'appartenance utilisateur et groupe pour déterminer à quel rôle un utilisateur appartient.

On considère qu'un rôle est assigné à un utilisateur si :

1. L'utilisateur est :
    - listé comme utilisateur à inclure dans le rôle, ou
    - un membre d'un groupe qui est listé pour être inclus dans le rôle.
2. L'utilisateur n'est pas :
    - listé comme utilisateur à exclure du rôle, ou
    - un membre d'un groupe qui est listé pour être exclus du rôle.

Les exclusions prennent la priorité sur les inclusions.

Les configurations d'exclusion et d'inclusion des utilisateurs ou groupes peuvent être effectuées par la console de gestion ou bien par le CLI.

Seuls les utilisateurs qui possèdent les rôles Superutilisateur ou Administrateur peuvent faire cette configuration.

Rapporter un bogue

6.9.2. Configurer le rôle d'utilisateur 'Assignment' (attribution de rôles)

Les rôles d'utilisateur à inclure ou à exclure peuvent être configurés dans la console de gestion et par le **`jboss-cli.sh`**. Cette section explique uniquement comment utiliser la console de gestion à cet effet.

Seuls les utilisateurs ayant les rôles **`SuperUser`** ou **`Administrator`** peuvent effectuer cette configuration.

La configuration des rôles Utilisateur de la console de gestion suit les étapes suivantes :

1. Connectez-vous à la console de gestion.
2. Cliquez sur l'onglet **Administration**.
3. Déployez le menu **`Access Control`** et sélectionner **Role Assignment**.
4. Sélectionnez l'onglet **USERS**.

Procédure 6.4. Créez une nouvelle attribution de rôle pour l'utilisateur

1. Connexion à la console de gestion.
2. Naviguez vers l'onglet **Users** (Utilisateurs) de la section **`Role Assignment`** (Attribution de rôles).
3. Cliquez sur le bouton **`Add`** en haut et à droite de la liste d'utilisateur. Le dialogue Ajouter Utilisateur (**`Add User`**) apparaîtra.

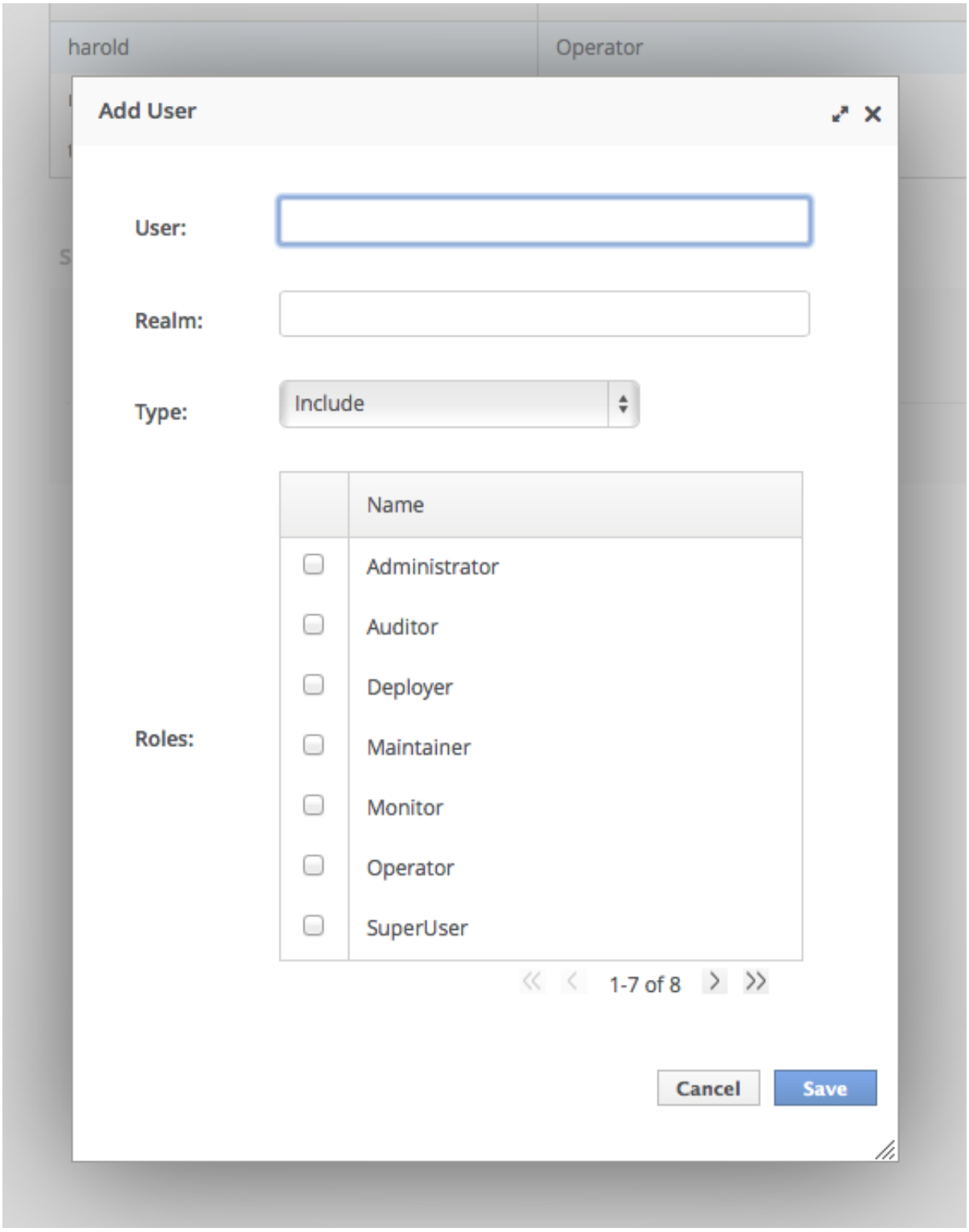


Figure 6.1. Boîte de dialogue 'Add User' (Ajouter Utilisateur)

4. Vous devez préciser un nom d'utilisateur, ou un domaine si possible.
5. Définissez le type de menu à inclure ou à exclure.
6. Cliquez la case des rôles à inclure ou à exclure. Vous pouvez utiliser la clé de contrôle (clé de commande OSX) pour cocher les divers items.
7. Cliquez sur le bouton **Save** pour terminer.

Si cela réussit, la boîte de dialogue 'Ajouter un utilisateur' (**`Add User`**) se ferme, et la liste des utilisateurs sera mise à jour pour refléter les modifications apportées. En cas d'échec, un message **`Impossible d'enregistrer l'attribution de rôle`** s'affichera.

Procédure 6.5. Mise à jour d'une attribution de rôle pour un utilisateur

1. Connexion à la console de gestion.
2. Naviguez vers l'onglet **Users** (Utilisateurs) de la section **`Role Assignment`** (Attribution de rôles).
3. Sélectionnez un utilisateur dans la liste.
4. Cliquez sur le bouton **Edit**. Le panneau de sélection est alors en mode Édit.

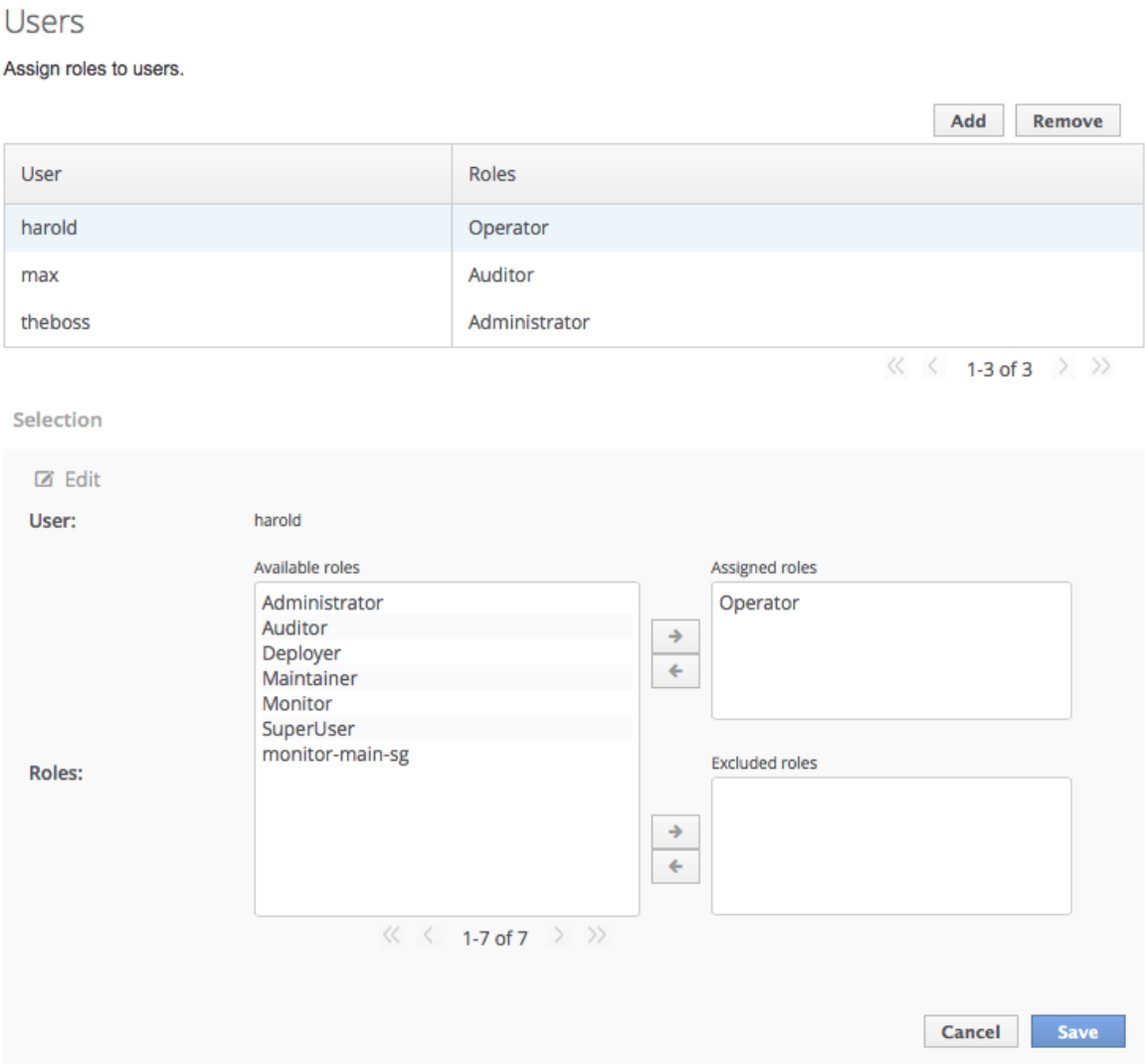


Figure 6.2. Sélectionnez Vue Édit

Vous pourrez ici ajouter ou supprimer les rôles assignés pour exclus pour l'utilisateur.

1. Pour ajouter un rôle assigné, sélectionnez le rôle requis de la liste de rôles disponibles sur la gauche et cliquez sur le bouton avec une flèche pointant sur la droite qui se trouve à côté de la liste de rôles assignés. Le rôle se déplacera alors depuis la liste de rôles disponibles vers la liste de rôles assignés.

2. Pour supprimer un rôle assigné, sélectionnez le rôle requis de la liste de rôles assignés sur la droite et cliquez sur le bouton avec une flèche pointant sur la gauche qui se trouve à côté de la liste de rôles assignés. Le rôle se déplacera alors depuis la liste de rôles assignés vers la liste de rôles disponibles.

3. Pour ajouter un rôle exclus, sélectionnez le rôle requis de la liste de rôles disponibles sur la gauche et cliquez sur le bouton avec une flèche pointant sur la droite qui se trouve à côté de la liste de rôles exclus. Le rôle se déplacera alors depuis la liste de rôles disponibles vers la liste de rôles exclus.

4. Pour supprimer un rôle exclus, sélectionnez le rôle requis de la liste de rôles assignés sur la droite et cliquer sur le bouton avec une flèche pointant sur la gauche qui se trouve à côté de la liste de rôles exclus. Le rôle se déplacera alors depuis la liste de rôles exclus vers la liste de rôles disponibles.

5. Cliquez sur le bouton **Save** pour terminer.

   Si cela réussit, la vue Édit se ferme, et la liste des utilisateurs sera mise à jour pour refléter les modifications apportées. En cas d'échec, un message **`Impossible d'enregistrer l'attribution de rôle`** s'affichera.

Procédure 6.6. Suppression d'une attribution de rôle pour un utilisateur

1. Connexion à la console de gestion.

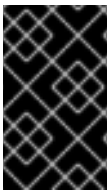
2. Naviguez vers l'onglet **Users** de la section Attribution de rôles.

3. Sélectionnez l'utilisateur dans la liste.

4. Cliquez sur le bouton **`Remove`**. L'invite de confirmation **`Remove Role Assignment`** apparaîtra.

5. Cliquez sur **Confirm**.

   Si cela réussit, l'utilisateur n'apparaîtra plus sur la liste d'attributions de rôle d'utilisateur.



IMPORTANT

Suppression de l'utilisateur de la liste d'attribution de rôles ne retire pas l'utilisateur du système, ni ne garantit qu'aucun rôle ne sera assigné à l'utilisateur. Les rôles peuvent toujours être assignés sur la base de l'appartenance à un groupe.

Rapporter un bogue

6.9.3. Configurer l'attribution de rôle utilisateur avec jboss-cli.sh

Les rôles d'utilisateur à inclure ou à exclure peuvent être configurés dans la console de gestion et par le **`jboss-cli.sh`**. Cette section explique uniquement comment utiliser **`jboss-cli.sh`**.

La configuration du mappage des utilisateurs et des groupes en rôles se situe dans l'API de gestion à : **`/core-service=management/access=authorization`** en tant qu'éléments **`role-mapping`**.

Seuls les utilisateurs qui possèdent les rôles SuperUser ou Administrator peuvent faire cette configuration.

Procédure 6.7. Affichage de la Configuration Attribution de rôles

1. Utiliser l'opération :read-children-names pour obtenir une liste complète des rôles configurés :

```
/core-service=management/access=authorization:read-children-
names(child-type=role-mapping)
```

```
[standalone@localhost:9999 access=authorization] :read-children-
names(child-type=role-mapping)
{
    "outcome" => "success",
    "result" => [
        "ADMINISTRATOR",
        "DEPLOYER",
        "MAINTAINER",
        "MONITOR",
        "OPERATOR",
        "SuperUser"
    ]
}
```

2. Utiliser l'opération **`read-resource`** d'un mappage de rôle (role-mapping) pour obtenir toutes les informations sur un rôle particulier :

```
/core-service=management/access=authorization/role-
mapping=ROLENAME:read-resource(recursive=true)
```

```
[standalone@localhost:9999 access=authorization] ./role-
mapping=ADMINISTRATOR:read-resource(recursive=true)
{
    "outcome" => "success",
    "result" => {
        "include-all" => false,
        "exclude" => undefined,
        "include" => {
            "user-theboss" => {
                "name" => "theboss",
                "realm" => undefined,
                "type" => "USER"
            },
            "user-harold" => {
                "name" => "harold",
                "realm" => undefined,
                "type" => "USER"
            },
            "group-SysOps" => {
                "name" => "SysOps",
                "realm" => undefined,
                "type" => "GROUP"
            }
        }
    }
}
[standalone@localhost:9999 access=authorization]
```

Procédure 6.8. Ajouter un nouveau rôle

Cette procédure montre comment ajouter un role-mapping pour un rôle. Cela doit être effectué avant que le rôle puisse être configuré.

- Utiliser l'opération **add** pour ajouter une nouvelle configuration de rôle.

```
/core-service=management/access=authorization/role-
mapping=ROLENAME:add
```

ROLENAME est le nom du rôle du nouveau mappage.

```
[standalone@localhost:9999 access=authorization] ./role-
mapping=AUDITOR:add
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

Procédure 6.9. Ajouter un Utilisateur comme inclus dans un rôle

Cette procédure montre comment ajouter un Utilisateur dans la liste d'inclusions d'un rôle.

S'il n'y a pas de configuration de rôle, alors vous devez commencer par la saisie du role-mapping.

- Utiliser l'opération **add** pour ajouter une entrée d'utilisateur dans la liste d'inclusions du rôle.

```
/core-service=management/access=authorization/role-
mapping=ROLENAME/include=ALIAS:add(name=USERNAME, type=USER)
```

ROLENAME est le nom du rôle en cours de configuration.

ALIAS est un nom unique pour ce mappage. Red Hat conseille d'utiliser une convention de mappage pour tous les alias comme **user-*USERNAME***.

USERNAME est le nom de l'utilisateur entrain d'être ajouté à la liste des inclusions.

```
 [standalone@localhost:9999 access=authorization] ./role-
mapping=AUDITOR/include=user-max:add(name=max, type=USER)
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

Procédure 6.10. Ajouter un Utilisateur comme exclus dans un rôle

Cette procédure montre comment ajouter un Utilisateur dans la liste d'exclusions d'un rôle.

S'il n'y a pas de configuration de rôle, alors vous devez commencer par la saisie du role-mapping.

- Utiliser l'opération **add** pour ajouter une entrée d'utilisateur dans la liste d'exclusions du rôle.

```
/core-service=management/access=authorization/role-
mapping=ROLENAME/exclude=ALIAS:add(name=USERNAME, type=USER)
```

ROLENAME est le nom du rôle en cours de configuration.

USERNAME est le nom de l'utilisateur ajouté à la liste des exclusions.

ALIAS est un nom unique pour ce mappage. Red Hat conseille d'utiliser une convention de mappage pour tous les alias comme **user-*USERNAME***.

```
[standalone@localhost:9999 access=authorization] ./role-
mapping=AUDITOR/exclude=user-max:add(name=max, type=USER)
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

Procédure 6.11. Configuration pour la suppression d'un rôle d'utilisateur d'inclusion

Cette procédure vous montre comment supprimer une entrée d'inclusion d'utilisateur d'un mappage de rôle.

- Utiliser l'opération **remove** pour supprimer la saisie.

```
/core-service=management/access=authorization/role-
mapping=ROLENAME/include=ALIAS:remove
```

ROLENAME est le nom du rôle en cours de configuration

ALIAS est un nom unique pour ce mappage. Red Hat conseille d'utiliser une convention de mappage pour tous les alias comme **user-*USERNAME***.

```
[standalone@localhost:9999 access=authorization] ./role-
mapping=AUDITOR/include=user-max:remove
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

Supprimer l'utilisateur de la liste d'inclusions ne permet pas de supprimer l'utilisateur du système, ni ne garantit que le rôle ne sera pas assigné à l'utilisateur. Le rôle peut être assigné sur la base de l'appartenance à un groupe.

Procédure 6.12. Configuration de la suppression d'un rôle d'utilisateur d'exclusion

Cette procédure vous montre comment supprimer une entrée d'exclusion d'utilisateur d'un mappage de rôle.

- Utiliser l'opération **remove** pour supprimer la saisie.

```
/core-service=management/access=authorization/role-
mapping=ROLENAME/exclude=ALIAS:remove
```

ROLENAME est le nom du rôle en cours de configuration.

ALIAS est un nom unique pour ce mappage. Red Hat conseille d'utiliser une convention de mappage pour tous les alias comme **user-*USERNAME***.

```
[standalone@localhost:9999 access=authorization] ./role-
mapping=AUDITOR/exclude=user-max:remove
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

Supprimer l'utilisateur de la liste d'exclusions ne permet pas de supprimer l'utilisateur du système, ni ne garantit que le rôle puisse être assigné à l'utilisateur. Les rôles peuvent être assignés sur la base de l'appartenance à un groupe.

Rapporter un bogue

6.9.4. Groupes Utilisateurs et Rôles

Les utilisateurs authentifiés à l'aide du fichier de **`mgmt-users.properties`** ou d'un serveur LDAP peuvent être membres des groupes d'utilisateurs. Un groupe d'utilisateurs est une étiquette arbitraire qui peut être assignée à un ou plusieurs utilisateurs.

Le système RBAC peut être configuré pour attribuer automatiquement des rôles aux utilisateurs selon les groupes d'utilisateurs auxquels ils appartiennent. Il peut également exclure des utilisateurs de rôles basés sur l'appartenance à un groupe.

Lorsque vous utilisez le fichier **`mgmt-users.properties`**, l'information groupe est stockée dans le fichier **`mgmt-groups.properties`**. Lorsque vous utilisez LDAP, l'information groupe est stockée dans le serveur LDAP et est maintenue par les responsables du serveur LDAP.

Rapporter un bogue

6.9.5. Configurer l'attribution de rôles de groupe

Des rôles peuvent être assignés à un utilisateur sur la base d'appartenance à un groupe d'utilisateurs.

Les groupes à inclure ou à exclure d'un rôle peuvent être configurés dans la console de gestion et par **`jboss-cli.sh`**. Cette section explique uniquement comment utiliser la console de gestion.

Seuls les utilisateurs ayant les rôles **`SuperUser`** ou **`Administrator`** peuvent effectuer cette configuration.

La configuration des rôles de groupe de la console de gestion suit les étapes suivantes :

1. Connectez-vous à la console de gestion.
2. Cliquez sur l'onglet **Administration**.
3. Déployez le menu **Access Control** et sélectionner **Role Assignment**.
4. Sélectionnez l'onglet **GROUPS**.

Procédure 6.13. Créez une nouvelle attribution de rôle pour un groupe

1. Connectez-vous à la console de gestion
2. Naviguez vers l'onglet **GROUPS** de la section **`Role Assignment`** (Attribution de rôles).
3. Cliquez sur le bouton **Add** en haut et à droite de la liste d'utilisateur. Le dialogue Ajouter Utilisateur (**`Add User`**) apparaîtra.

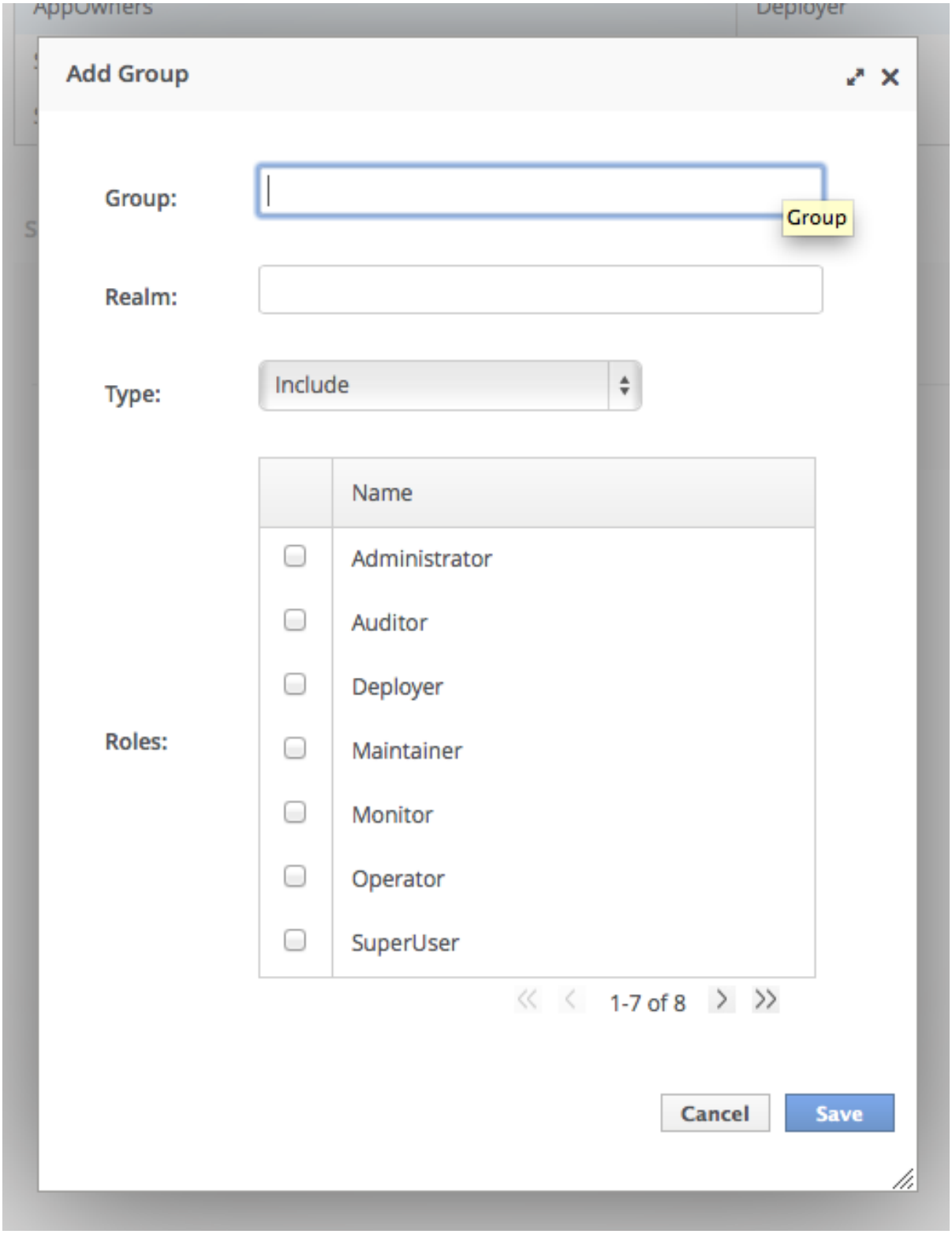


Figure 6.3. Ajouter le dialogue de groupe

4. Vous devez préciser un nom de groupe, et le domaine si possible.
5. Définissez le type de menu à inclure ou à exclure.
6. Cliquez la case des rôles à inclure ou à exclure. Vous pouvez utiliser la clé de contrôle (clé de commande OSX) pour cocher les divers items.
7. Cliquez sur le bouton **Save** pour terminer.

Si cela réussit, la boîte de dialogue Ajouter un groupe (**`Add Group`**) se fermera, et la liste des utilisateurs sera mise à jour pour refléter les modifications apportées. En cas d'échec, un message **`Impossible d'enregistrer l'attribution de rôle`** s'affichera.

Procédure 6.14. Mise à jour d'une attribution de rôle pour un groupe

1. Connexion à la console de gestion.
2. Naviguez vers l'onglet **GROUPS** de la section Attribution de rôles.
3. Sélectionnez un groupe dans la liste.
4. Cliquez sur le bouton Édit. La vue de sélection est alors en mode Édit.

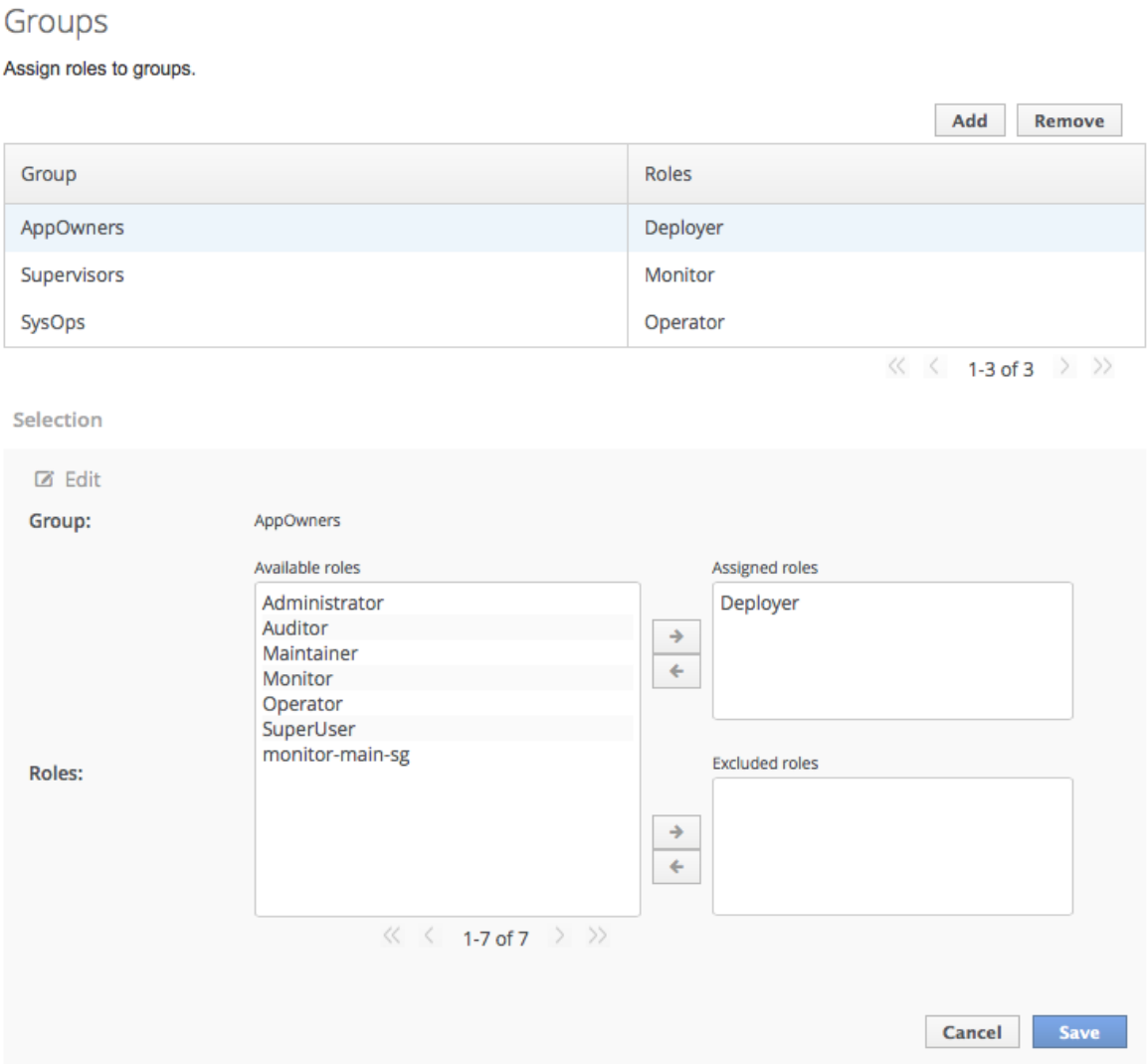


Figure 6.4. Sélection Vue Mode Édition

Vous pourrez ici ajouter ou supprimer les rôles assignés ou exclus du groupe :

- Pour ajouter un rôle assigné, sélectionnez le rôle que vous souhaitez de la liste de rôles disponibles sur la gauche et cliquez sur le bouton ayant une flèche pointant sur la droite qui se trouve à côté de la liste de rôles assignés. Le rôle se déplacera alors depuis la liste de rôles disponibles vers la liste de rôles assignés.
- Pour supprimer un rôle assigné, sélectionnez le rôle requis de la liste de rôles assignés sur

la droite et cliquez sur le bouton ayant une flèche pointant sur la gauche qui se trouve à côté de la liste de rôles assignés. Le rôle se déplacera alors depuis la liste de rôles assignés vers la liste de rôles disponibles.

- Pour ajouter un rôle exclus, sélectionnez le rôle requis de la liste de rôles disponibles sur la gauche et cliquez sur le bouton ayant une flèche pointant sur la droite qui se trouve à côté de la liste de rôles exclus. Le rôle se déplacera alors depuis la liste de rôles disponibles vers la liste de rôles exclus.
- Pour supprimer un rôle exclus, sélectionnez le rôle requis de la liste de rôles assignés sur la droite et cliquez sur le bouton ayant une flèche pointant sur la gauche qui se trouve à côté de la liste de rôles exclus. Le rôle se déplacera alors depuis la liste de rôles exclus vers la liste de rôles disponibles.

5. Cliquez sur le bouton **`Save`** pour terminer.

   Si cela réussit, la vue Édit se fermera, et la liste des groupes sera mise à jour pour refléter les modifications apportées. En cas d'échec, un message **Failed to save role assignment** (Impossible d'enregistrer l'attribution de rôle) s'affichera.

Procédure 6.15. Supprimez une attribution de rôle pour un groupe

1. Connexion à la console de gestion.
2. Naviguez vers l'onglet **GROUPS** de la section Attribution de rôles (**Role Assignment**).
3. Sélectionnez un groupe dans la liste.
4. Cliquez sur le bouton **`Remove`**. L'invite de confirmation **`Remove Role Assignment`** apparaîtra.
5. Cliquez sur **`Confirmer`**.

   Si cela réussit, le rôle n'apparaîtra plus sur la liste d'attributions de rôle de groupe.

   La suppression du groupe de la liste d'attribution de rôles ne retire pas l'utilisateur du système, ni ne garantit qu'aucun rôle ne sera assigné aux membres de ce groupe. Chaque membre du groupe peut encore avoir un rôle qui lui soit directement assigné.

Rapporter un bogue

6.9.6. Configurer l'attribution de rôles de groupe avec jboss-cli.sh

Les groupes à inclure ou à exclure d'un rôle peuvent être configurés dans la console de gestion et par le **`jboss-cli.sh`**. Cette section explique uniquement comment utiliser l'outil **`jboss-cli.sh`**.

La configuration du mappage des utilisateurs et des groupes en rôles se situe dans l'API de gestion à : **`/core-service=management/access=authorization`** en tant qu'éléments de mappage.

Seuls les utilisateurs ayant des rôles SuperUser ou Administrateur peuvent effectuer cette configuration.

Procédure 6.16. Affichage de la Configuration d'attribution de rôles de groupe

1. Utiliser l'opération **`read-children-names`** pour obtenir une liste complète des rôles configurés :

```
/core-service=management/access=authorization:read-children-
names(child-type=role-mapping)
```

```
[standalone@localhost:9999 access=authorization] :read-children-
names(child-type=role-mapping)
{
    "outcome" => "success",
    "result" => [
        "ADMINISTRATOR",
        "DEPLOYER",
        "MAINTAINER",
        "MONITOR",
        "OPERATOR",
        "SuperUser"
    ]
}
```

2. Utiliser l'opération **`read-resource`** d'un mappage de rôle (role-mapping) pour obtenir toutes les informations sur un rôle particulier :

```
/core-service=management/access=authorization/role-
mapping=ROLENAME:read-resource(recursive=true)
```

```
[standalone@localhost:9999 access=authorization] ./role-
mapping=ADMINISTRATOR:read-resource(recursive=true)
{
    "outcome" => "success",
    "result" => {
        "include-all" => false,
        "exclude" => undefined,
        "include" => {
            "user-theboss" => {
                "name" => "theboss",
                "realm" => undefined,
                "type" => "USER"
            },
            "user-harold" => {
                "name" => "harold",
                "realm" => undefined,
                "type" => "USER"
            },
            "group-SysOps" => {
                "name" => "SysOps",
                "realm" => undefined,
                "type" => "GROUP"
            }
        }
    }
}
[standalone@localhost:9999 access=authorization]
```

Procédure 6.17. Ajouter un nouveau rôle

Cette procédure montre comment ajouter un role-mapping pour un rôle. Cela doit être effectué avant que le rôle puisse être configuré.

- Utiliser l'opération **add** pour ajouter une nouvelle configuration de rôle.

```
/core-service=management/access=authorization/role-
mapping=ROLENAME:add
```

```
[standalone@localhost:9999 access=authorization] ./role-
mapping=AUDITOR:add
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

Procédure 6.18. Ajouter un Groupe comme inclus dans un rôle

Cette procédure montre comment ajouter un Groupe dans la liste d'inclusions d'un rôle.

S'il n'y a pas de configuration de rôle, alors vous devez commencer par la saisie du role-mapping.

- Utiliser l'opération **add** pour ajouter une entrée de Groupe dans la liste d'inclusions du rôle.

```
/core-service=management/access=authorization/role-
mapping=ROLENAME/include=ALIAS:add(name=GROUPNAME, type=GROUP)
```

ROLENAME est le nom du rôle en cours de configuration.

GROUPNAME est le nom du groupe entrain d'être ajouté à la liste des inclusions.

ALIAS est un nom unique pour ce mappage. Red Hat recommande d'utiliser une convention de mappage pour tous les alias comme **group-*GROUPNAME***.

```
[standalone@localhost:9999 access=authorization] ./role-
mapping=AUDITOR/include=group-investigators:add(name=investigators,
type=GROUP)
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

Procédure 6.19. Ajouter un groupe comme exclus dans un rôle

Cette procédure montre comment ajouter un groupe dans la liste d'exclusions d'un rôle.

S'il n'y a pas de configuration de rôle, alors vous devez commencer par la saisie du role-mapping.

- Utiliser l'opération **add** pour ajouter une entrée de Groupe dans la liste d'exclusions du rôle.

```
/core-service=management/access=authorization/role-
mapping=ROLENAME/exclude=ALIAS:add(name=GROUPNAME, type=GROUP)
```

ROLENAME est le nom du rôle en cours de configuration

GROUPNAME est le nom du groupe entrain d'être ajouté à la liste des inclusions

ALIAS est un nom unique pour ce mappage. Red Hat recommande d'utiliser une convention de mappage pour tous les alias comme **group-*GROUPNAME***.

```
[standalone@localhost:9999 access=authorization] ./role-
mapping=AUDITOR/exclude=group-supervisors:add(name=supervisors,
type=USER)
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

Procédure 6.20. Configuration de la suppression d'un rôle de groupe

Cette procédure vous montre comment supprimer une entrée d'inclusion de groupe d'un mappage de groupe.

- Utiliser l'opération **remove** pour supprimer la saisie.

```
/core-service=management/access=authorization/role-
mapping=ROLENAME/include=ALIAS:remove
```

ROLENAME est le nom du rôle en cours de configuration

ALIAS est un nom unique pour ce mappage. Red Hat recommande d'utiliser une convention de mappage pour tous les alias comme **group-*GROUPNAME***.

```
[standalone@localhost:9999 access=authorization] ./role-
mapping=AUDITOR/include=group-investigators:remove
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

Supprimer le groupe de la liste d'inclusions ne permet pas de supprimer le groupe du système, ni ne garantit que le rôle ne sera pas assigné à des utilisateurs de ce groupe. Le rôle peut être assigné à des utilisateurs du groupe individuellement.

Procédure 6.21. Supprimer une entrée d'exclusion de groupe d'utilisateurs

Cette procédure montre comment supprimer une entrée d'exclusion d'un mappage de rôle.

- Utiliser l'opération **remove** pour supprimer la saisie.

```
/core-service=management/access=authorization/role-
mapping=ROLENAME/exclude=ALIAS:remove
```

ROLENAME est le nom du rôle en cours de configuration.

ALIAS est un nom unique pour ce mappage. Red Hat recommande d'utiliser une convention de mappage pour tous les alias comme **group-*GROUPNAME***.

```
[standalone@localhost:9999 access=authorization] ./role-
mapping=AUDITOR/exclude=group-supervisors:remove
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

Supprimer le groupe de la liste d'exclusions ne permet pas de supprimer le groupe du système, ni ne garantit que le rôle ne va pas être assigné à des membres du groupe. Les rôles peuvent être exclus sur la base de l'appartenance à un groupe.

Rapporter un bogue

6.9.7. Autorisation et Chargement de groupes avec LDAP

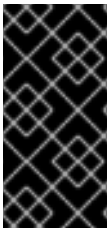
Un annuaire LDAP contient des entrées pour les comptes d'utilisateurs et de groupes, dont les références sont mises en correspondance par des attributs. Selon la configuration du serveur LDAP, une entité utilisatrice peut mapper les groupes à qui appartient l'utilisateur via les attributs **`memberOf`** ; une entité du groupe peut mapper les utilisateurs lui appartenant par les attributs **`uniqueMember`** ; ou les deux mappages peuvent être maintenus dans le serveur LDAP.

Il est également fréquent qu'un utilisateur soit authentifié auprès du serveur à l'aide d'un simple nom utilisateur. Pour les recherches d'informations d'appartenance de groupe selon le serveur de répertoires utilisé, les recherches d'utilisation peuvent être effectuées à l'aide de ce simple nom ou peuvent être réalisées en utilisant le nom unique d'entrée utilisateur du répertoire.

L'étape d'authentification d'un utilisateur qui se connecte au serveur a toujours lieu en premier. Une fois que l'utilisateur a bien été authentifié, le serveur charge un groupe d'utilisateurs. L'étape d'authentification et d'autorisation utilisent chacune une connexion au serveur LDAP. Le domaine contient une optimisation qui consiste à utiliser la connexion utilisée pour l'étape de chargement de groupe. Tel qu'il apparaît dans les étapes de configuration ci-dessous, il est possible de définir des règles au sein de la section autorisation pour convertir un nom d'utilisateur simple en nom unique d'utilisateur. Le résultat de la recherche « nom d'utilisateur à mappage de nom unique » en cours d'authentification est mis en cache et réutilisé lors de la demande d'autorisation quand l'attribut **`force`** est défini à «false». Quand **`force`** est défini à true, la recherche a lieu à nouveau en cours d'autorisation (pendant le chargement des groupes). Cela est normalement effectué quand des serveurs différents effectuent l'authentification et l'autorisation.

```
<authorization>
    <ldap connection="...">
     <!-- OPTIONAL -->
       <username-to-dn force="true">
           <!-- Only one of the following. -->
           <username-is-dn />
           <username-filter base-dn="..." recursive="..." user-dn-
attribute="..." attribute="..." />
           <advanced-filter base-dn="..." recursive="..." user-dn-
attribute="..." filter="..." />
        </username-to-dn>

       <group-search group-name="..." iterative="..." group-dn-
attribute="..." group-name-attribute="..." >
           <!-- One of the following -->
           <group-to-principal base-dn="..." recursive="..." search-
by="...">
               <membership-filter principal-attribute="..." />
           </group-to-principal>
           <principal-to-group group-attribute="..." />
       </group-search>
    </ldap>
</authorization>
```



IMPORTANT

Ces exemples indiquent des attributs qui utilisent des valeurs par défaut. Ces valeurs sont indiquées ici pour clarifier. Les attributs qui contiennent les valeurs par défaut sont supprimées de la configuration quand c'est persisté sur le serveur. L'exception est l'attribut **`force`**. Il est requis, même quand défini à la valeur par défaut **`false`**.

username-to-dn

L'élément **`username-to-dn`** indique comment mapper le nom d'utilisateur au nom distinctif de son entrée dans le répertoire LDAP. Cet élément n'est requis que lorsque les *deux* énoncés suivants sont véréfiées :

- Les étapes d'authentification et d'autorisation sont effectués sur deux serveurs LDAP distincts.
- La recherche de groupe utilise un nom distinctif.

1:1 username-to-dn

Ceci indique que le nom d'utilisateur saisi par l'utilisateur distant est le nom distinctif de l'utilisateur.

```
<username-to-dn force="false">
   <username-is-dn />
</username-to-dn>
```

Ceci définit un mappage 1:1, et il n'y a pas de configuration supplémentaire possible.

username-filter

La prochaine option est très semblable à la simple option décrite ci-dessus dans l'étape d'authentification. Un attribut est spécifié et on recherche une correspondance avec le nom d'utilisateur fourni.

```
<username-to-dn force="true">
    <username-filter base-dn="dc=people,dc=harold,dc=example,dc=com"
recursive="false" attribute="sn" user-dn-attribute="dn" />
</username-to-dn>
```

Les attributs pouvant être définis sont les suivants :

- **`base-dn`** : le nom distinctif du contexte pour commencer la recherche.
- **`recursive`** : indique si la recherche va s'étendre à des sous-contextes. La valeur par défaut est **`false`**.
- **`attribute`** : l'attribut de l'entrée de l'utilisateur à faire correspondre avec le nom d'utilisateur fourni. La valeur par défaut est **`uid`**.
- **`user-dn-attribute`** : l'attribut à lire pour obtenir les noms distinctifs d'utilisateurs. La valeur par défaut est **`dn`**.

advanced-filter

L'option finale est de spécifier un filtre avancé. Comme dans la section authentification, c'est l'opportunité d'utiliser un filtre personnalisé pour trouver le nom unique de l'utilisateur.

```
<username-to-dn force="true">
    <advanced-filter base-dn="dc=people,dc=harold,dc=example,dc=com"
recursive="false" filter="sAMAccountName={0}" user-dn-attribute="dn" />
</username-to-dn>
```

Pour les attributs qui correspondent à ceux du *username-filter*, le sens et les valeurs par défaut sont les mêmes. Cela laisse un nouvel attribut :

- **`filter`** : filtre personnalisé utilisé pour chercher une entrée d'utilisateur quand le nom d'utilisateur est substitué dans l'espace réservé **`{0}`**

IMPORTANT

Le code XML doit rester valide une fois que le filtre est défini donc si des caractères spéciaux comme **`&`** sont utilisés, assurez-vous que la forme qui convient soit utilisée. Par exemple, **`& amp ;`** pour le caractère **`&`**.

La recherche Groupe

Il existe deux styles différents qui puissent être utilisés lors de la recherche d'informations d'appartenance de groupe. Le premier style est quand l'entrée d'utilisateur contient un attribut qui référence les groupes dont l'utilisateur est membre. Le second style est quand le groupe contient un attribut référençant l'entrée des utilisateurs.

Lorsqu'il y a un choix sur le style à utiliser, Red Hat recommande que la configuration d'entrée utilisateur référençant le groupe soit utilisée. C'est parce qu'avec cette méthode, l'information de groupe peut être chargée par la lecture des attributs des noms uniques connus sans avoir à effectuer les recherches. L'autre approche nécessite des recherches extensives pour identifier les groupes qui référencent l'utilisateur.

Avant de décrire la configuration, voici quelques exemples LDIF pour illustrer cela.

Exemple 6.1. Principal à Groupe - Exemple LDIF

Cet exemple illustre un cas où nous avons un utilisateur **`TestUserOne`**, qui est membre de **`GroupOne`**, et **`GroupOne`** est à son tour membre de **`GroupFive`**. L'appartenance au groupe est démontrée par l'utilisation d'un attribut **`memberOf`** défini au nom unique du groupe dont l'utilisateur est membre.

Ce n'est pas affiché ici, mais un utilisateur peut avoir plusieurs attributs **`memberOf`** définis, un pour chaque groupe dont l'utilisateur est un membre direct.

```
dn: uid=TestUserOne,ou=users,dc=principal-to-group,dc=example,dc=org
objectClass: extensibleObject
objectClass: top
objectClass: groupMember
objectClass: inetOrgPerson
objectClass: uidObject
objectClass: person
objectClass: organizationalPerson
cn: Test User One
sn: Test User One
uid: TestUserOne
```

```
distinguishedName: uid=TestUserOne,ou=users,dc=principal-to-
group,dc=example,dc=org
memberOf: uid=GroupOne,ou=groups,dc=principal-to-group,dc=example,dc=org
memberOf: uid=Slashy/Group,ou=groups,dc=principal-to-
group,dc=example,dc=org
userPassword::
e1NTSEF9WFpURzhLVjc4WVZBQUJNbEI3Ym96UVAva0RTNlFNWUpLOTdTMUE9PQ==

dn: uid=GroupOne,ou=groups,dc=principal-to-group,dc=example,dc=org
objectClass: extensibleObject
objectClass: top
objectClass: groupMember
objectClass: group
objectClass: uidObject
uid: GroupOne
distinguishedName: uid=GroupOne,ou=groups,dc=principal-to-
group,dc=example,dc=org
memberOf: uid=GroupFive,ou=subgroups,ou=groups,dc=principal-to-
group,dc=example,dc=org

dn: uid=GroupFive,ou=subgroups,ou=groups,dc=principal-to-
group,dc=example,dc=org
objectClass: extensibleObject
objectClass: top
objectClass: groupMember
objectClass: group
objectClass: uidObject
uid: GroupFive
distinguishedName: uid=GroupFive,ou=subgroups,ou=groups,dc=principal-to-
group,dc=example,dc=org
```

Exemple 6.2. Groupe à Principal - Exemple LDIF

Cet exemple montre le même utilisateur **`TestUserOne`** qui est un membre de **`GroupOne`**, à son tour membre de **`GroupFive`** - cependant dans ce cas, c'est un attribut **`uniqueMember`** du groupe de l'utilisateur utilisé pour la référence croisée.

Encore une fois, l'attribut utilisé pour la mise en correspondance de l'appartenance à un groupe peut être répété, si vous regardez *GroupFive*, il y a également une référence à un autre utilisateur *TestUserFive* non visible ici.

```
dn: uid=TestUserOne,ou=users,dc=group-to-principal,dc=example,dc=org
objectClass: top
objectClass: inetOrgPerson
objectClass: uidObject
objectClass: person
objectClass: organizationalPerson
cn: Test User One
sn: Test User One
uid: TestUserOne
userPassword::
e1NTSEF9SjR0OTRDR1ltaHc1VVZQOEJvbXhUYjl1dkFVd1lQTmRLSEdzaWc9PQ==

dn: uid=GroupOne,ou=groups,dc=group-to-principal,dc=example,dc=org
```

```
objectClass: top
objectClass: groupOfUniqueNames
objectClass: uidObject
cn: Group One
uid: GroupOne
uniqueMember: uid=TestUserOne,ou=users,dc=group-to-
principal,dc=example,dc=org

dn: uid=GroupFive,ou=subgroups,ou=groups,dc=group-to-
principal,dc=example,dc=org
objectClass: top
objectClass: groupOfUniqueNames
objectClass: uidObject
cn: Group Five
uid: GroupFive
uniqueMember: uid=TestUserFive,ou=users,dc=group-to-
principal,dc=example,dc=org
uniqueMember: uid=GroupOne,ou=groups,dc=group-to-
principal,dc=example,dc=org
```

Recherche de groupe standard

Avant de chercher des exemples pour les deux approches montrées ci-dessus, nous devons tout d'abord définir les attributs communs aux deux approches.

```
<group-search group-name="..." iterative="..." group-dn-attribute="..."
group-name-attribute="..." >
    ...
</group-search>
```

- **`group-name`** : cet attribut est utilisé pour indiquer le formulaire qui doit être utilisé pour le nom de groupe retourné correspondant à la liste de groupes dont l'utilisateur est membre. Cela peut être sous la simple forme de nom du groupe ou de nom unique de groupe. Si le nom unique est nécessaire, cet attribut peut être défini à **`DISTINGUISHED_NAME`**. Valeur par défaut **`SIMPLE`**.

- **`itérative`** : cet attribut est utilisé pour indiquer si, après avoir identifié les groupes qui appartiennent à un utilisateur, on doit rechercher également de manière itérative basée sur les groupes afin d'identifier quels groupes appartiennent à quels groupes. Si la recherche itérative est activée, nous continuons jusqu'à ce que nous rejoignions un groupe qui ne soit pas membre si aucun autre groupe ou cycle n'est détecté. Par défaut, **`false`**.

L'appartenance à un groupe cyclique n'est pas un problème. Il existe un registre de chaque recherche pour empêcher les groupes qui ont déjà été fouillés d'être recherchés à nouveau.

IMPORTANT

Pour une recherche itérative, les entrées de groupe doivent ressembler aux entrées utilisateur. La même approche qui permet d'identifier les groupes auxquels appartient un utilisateur est alors utilisée pour identifier les groupes auxquels le groupe appartient. Ce ne serait pas possible si, une fois que nous parlons d'appartenance inter groupes, le nom de l'attribut utilisé pour la référence croisée changeait ou si la direction de la référence changeait.

- **`group-dn-attribute`**: sur une entrée pour un groupe dont l'attribut est son nom unique. La valeur par défaut est **`dn`**.
- **`group-name-attribute`**: sur une entrée pour un groupe dont l'attribut est son simple nom. La valeur par défaut est **`uid`**.

Exemple 6.3. Configuration d'exemple de Principal à Groupe

Basé sur l'exemple LDIF ci-dessus, voici un exemple de configuration chargeant itérativement un groupe d'utilisateurs où l'attribut utilisé pour référence est l'attribut **`memberOf`** de l'utilisateur.

```
<authorization>
    <ldap connection="LocalLdap">
        <username-to-dn>
            <username-filter base-dn="ou=users,dc=principal-to-
group,dc=example,dc=org" recursive="false" attribute="uid" user-dn-
attribute="dn" />
        </username-to-dn>
        <group-search group-name="SIMPLE" iterative="true" group-dn-
attribute="dn" group-name-attribute="uid">
            <principal-to-group group-attribute="memberOf" />
        </group-search>
    </ldap>
</authorization>
```

L'aspect le plus important de cette configuration est que l'élément **`principal-to-group`** a été ajouté avec un seul attribut.

- **`group-attribute`**: le nom de l'attribut sur l'entrée d'utilisateur qui correspond au nom unique du groupe qui appartient à l'utilisateur. La valeur par défaut est **`memberOf`**.

Exemple 6.4. Configuration d'exemple de Groupe à Principal

Cet exemple vous montre une recherche interactive pour l'exemple groupe à principal LDIF montré ci-dessus.

```
<authorization>
      <ldap connection="LocalLdap">
          <username-to-dn>
              <username-filter base-dn="ou=users,dc=group-to-
principal,dc=example,dc=org" recursive="false" attribute="uid" user-dn-
attribute="dn" />
          </username-to-dn>
          <group-search group-name="SIMPLE" iterative="true" group-dn-
attribute="dn" group-name-attribute="uid">
              <group-to-principal base-dn="ou=groups,dc=group-to-
principal,dc=example,dc=org" recursive="true" search-
by="DISTINGUISHED_NAME">
                  <membership-filter principal-attribute="uniqueMember"
/>
              </group-to-principal>
```

```
            </group-search>
        </ldap>
    </authorization>
```

Un élément **`group-to-principal`** est ajouté ici. Cet élément est utilisé pour définir comment les recherches de groupes qui référencent l'entrée de l'utilisateur seront exécutées. Les attributs suivants sont définis :

- **`base-dn`**: le nom unique du contexte à utiliser pour commencer la recherche.
- **`recursive`**: indique si les sous-contextes peuvent également être recherchés. La valeur par défaut est **`false`**.
- **`search-by`**: Le forme du nom de rôle utilisé dans les recherches. Valeurs valides**`SIMPLE`** et **`DISTINGUISHED_NAME`**. Valeur par défaut **`DISTINGUISHED_NAME`**.

Dans l'élément *group-to-principal*, il y a un élément *membership-filter* pour définir la correspondance.

- **`principal-attribute`** : le nom de l'attribut d'entrée de groupe qui référence l'entrée utilisateur. La valeur par défaut est **`member`**.

Rapporter un bogue

6.9.8. Scoped rôles

Les scoped rôles sont des rôles définis par l'utilisateur, qui donnent les permissions d'un des rôles standards, mais uniquement pour un ou plusieurs groupes ou hôtes spécifiés. Les scoped rôles permettent à la gestion d'utilisateurs d'octroyer des permissions limitées aux groupes de serveurs et aux hôtes requis.

Le scoped rôles peuvent être créés par des utilisateurs auxquels les rôles Administrateur ou SuperUtilisateur sont assignés.

Ils se résument par cinq caractéristiques :

1. Un nom unique.
2. Correspond aux rôles standards sur lesquels il est basé.
3. S'il s'applique au Groupes de serveurs ou aux Hôtes
4. La liste des groupes de serveurs ou hôtes auxquels il est limité.
5. Si tous les utilisateurs sont inclus automatiquement. La valeur par défaut sera false.

Une fois créé, un scoped rôle peut être assigné à des utilisateurs ou à des groupes de la même façon que les rôles standards.

La création d'un scoped rôle ne vous laisse pas définir de nouvelles permissions. Les scoped rôles vous permettent uniquement d'appliquer les permissions d'un rôle existant dans une portée limitée. Par exemple, vous pouvez créer un scoped rôle basé sur le rôle Deployer, qui est limité à un groupe de serveur unique.

Il n'y a que deux scopes auxquels les rôles peuvent être limités, hôte et groupe de serveur.

Rôles host-scoped

Un rôle non host-scoped limite les permissions de ce rôle à un ou plusieurs hôtes. Cela signifie que l'accès est donné aux arborescences de ressources **`/host=*/`** qui conviennent, mais les ressources spécifiques à d'autres hôtes sont cachées.

Rôles server-group-scoped

Un rôle server-group-scoped limite les permissions de ce rôle à un ou plusieurs groupes de serveurs. De plus, les permissions de rôle seront appliquées également au profil, groupe de liaison de socket, config du serveur et aux ressources de serveur associés avec les groupes de serveur spécifiés. Toutes les ressources secondaires à l'intérieur non liées de manière logique au groupe de serveurs ne seront pas visibles par l'utilisateur.

Les rôles sever-group-scoped et les hôtes ont les permissions du rôle Monitor pour le reste de la configuration du domaine partagé.

Rapporter un bogue

6.9.9. Création de rôles scoped

Les rôles scoped sont des rôles définis par l'utilisateur qui accordent les permissions d'un des rôles standard mais seulement pour un ou plusieurs groupes de serveurs ou d'hôtes. Cette rubrique montre comment créer des scoped roles.

Seuls les utilisateurs ayant les rôles **`SuperUser`** ou **`Administrator`** peuvent effectuer cette configuration.

La configuration de scoped roles de la console de gestion suit les étapes suivantes :

1. Connectez-vous à la console de gestion
2. Cliquez sur l'onglet **Administration**
3. Déployez le menu **Access Control** et sélectionnez **Role Assignment**.
4. Sélectionnez l'onglet **ROLES**, puis l'onglet **Scoped Roles** à l'intérieur.

La section **`Scoped Roles`** de la Console de gestion comprend deux parties principales, un tableau qui contient une liste de scoped roles configurés, et un panneau **`Selection`** qui affiche les détails du rôle actuellement sélectionné dans le tableau.

Les procédures suivantes vous montrent comment effectuer des tâches de configuration de scoped roles.

Procédure 6.22. Ajouter un nouveau scoped role

1. Connectez-vous à la console de gestion
2. Naviguer dans la partie **Scoped Roles** de l'onglet **Roles**.
3. Cliquer sur le bouton **Add**. Le dialogue **`Add Scoped Role`** apparaîtra.
4. Indiquer les détails suivants :

- **`Name`**, le nom distinctif du nouveau scoped role.
- **`Base Role`**, le rôle sur lequel ce rôle basera sa permission.
- **`Type`**, indique si ce rôle est limité à des hôtes ou à des groupes de serveurs.
- **`Scope`**, la liste d'hôtes ou de groupes de serveurs auxquels le rôle est limité. Vous pouvez sélectionner plusieurs entrées.
- **`Include All`**, indique si le rôle doit automatiquement inclure tous les utilisateurs. La valeur par défaut est non.

5. Cliquez sur le bouton de sauvegarde **Save**. La boîte de dialogue se fermera et le rôle nouvellement créé apparaîtra dans le tableau.

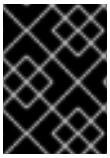
Procédure 6.23. Modifiez un scoped role.

1. Connectez-vous à la console de gestion
2. Naviguer dans la partie **Scoped Roles** de l'onglet **Roles**.
3. Cliquez sur le scoped role que vous souhaitez modifier dans le tableau. Les détails de ce rôle apparaissent dans le panneau **`Selection`** qui se trouve sous le tableau.
4. Cliquez sur **`Edit`** dans le panneau **`Selection`**. Le panneau **`Selection`** entre en mode d'édition.
5. Mettre à jour les informations que vous souhaitez modifier et cliquez sur le bouton de sauvegarde **`Save`**. Le panneau **`Selection`** retournera à son état précédent. Le panneau de sélection **`Selection`** et le tableau afficheront les informations nouvellement mises à jour.

Procédure 6.24. Affichez les membres scoped role

1. Connectez-vous à la console de gestion
2. Naviguez dans la partie **`Scoped Roles`** de l'onglet **Roles**.
3. Cliquez sur le scoped role du tableau dont vous souhaitez voir les **Members**, puis cliquez sur le bouton **Members**. La boîte de dialogue **Members of role** apparaîtra. Elle nous affichera les utilisateurs et les groupes qui sont inclus ou exclus du rôle.
4. Cliquez sur le bouton **Done** (terminé) quand vous aurez fini de consulter cette information.

Procédure 6.25. Supprimez un scoped role.



IMPORTANT

Un **Scoped Role** ne peut pas être supprimé si des utilisateurs ou groupes y sont assignés. Supprimez les assignations de rôle pour commencer, puis le supprimer.

1. Connectez-vous à la console de gestion
2. Naviguer dans la partie **Scoped Roles** de l'onglet **Roles**.
3. Sélectionnez le scoped role à supprimer du tableau.

4. Cliquez sur le bouton **Remove** button. Le dialogue **`Remove Scoped Role`** apparaîtra.

5. Cliquez sur le bouton **Confirm** (confirmer). La boîte de dialogue se fermera et le rôle sera supprimé.

Rapporter un bogue

6.10. CONFIGURER LES CONTRAINTES

6.10.1. Configurez les contraintes de sensibilité

Chaque contrainte de sensibilité définit un ensemble de ressources qui sont considérées comme « sensibles ». Une ressource sensible est généralement une ressource qui doit être tenue secrète, comme les mots de passe, ou une ressource qui aura de graves répercussions sur le serveur, comme la mise en réseau, la configuration de la JVM ou les propriétés système. Le système de contrôle d'accès lui-même est également considéré comme sensible. La sensibilité des ressources limite quels rôles sont capables de lire, écrire ou répondre à une ressource spécifique.

La configuration de contraintes de sensibilité se trouve dans l'API de gestion à l'adresse suivante : **`/core-service=management/access=authorization/constraint=sensitivity-classification`**.

Dans le modèle de gestion, chaque contrainte de ressource est identifiée en tant que **`classification`**. Les classifications sont ensuite regroupées en **`types`**. Il existe 39 catégories incluses qui sont regroupées en 13 types.

Pour configurer une contrainte de sensibilité, l'opération **`write-attribut`** permet de paramétrer les attributs **`configured-requires-read`**, **`configured-requires-write`**, ou **`configured-requires-addressable`**. Pour rendre ce type d'opération sensible, définir la valeur de l'attribut à **`true`**, sinon, pour le rendre non sensible, le définir à la valeur **`false`**. Par défaut, ces attributs ne sont pas définis et les valeurs **`default-requires-read`**, **`default-requires-write`**, et **`default-requires-addressable`** seront utilisées. Une fois que l'attribut est configuré, ce sera cette valeur qui sera utilisée à la place de la valeur par défaut. Impossible de modifier les valeurs par défaut.

Exemple 6.5. Comment rendre les propriétés système des opérations sensibles.

```
[domain@localhost:9999 /] cd /core-
service=management/access=authorization/constraint=sensitivity-
classification/type=core/classification=system-property
[domain@localhost:9999 classification=system-property] :write-
attribute(name=configured-requires-read, value=true)
{
    "outcome" => "success",
    "result" => undefined,
    "server-groups" => {"main-server-group" => {"host" => {"master" => {
        "server-one" => {"response" => {"outcome" => "success"}},
        "server-two" => {"response" => {"outcome" => "success"}}
    }}}}
}
[domain@localhost:9999 classification=system-property] :read-resource
{
    "outcome" => "success",
    "result" => {
        "configured-requires-addressable" => undefined,
```

```
        "configured-requires-read" => true,
        "configured-requires-write" => undefined,
        "default-requires-addressable" => false,
        "default-requires-read" => false,
        "default-requires-write" => true,
        "applies-to" => {
            "/host=master/system-property=*" => undefined,
            "/host=master/core-service=platform-mbean/type=runtime" =>
undefined,
            "/server-group=*/system-property=*" => undefined,
            "/host=master/server-config=*/system-property=*" =>
undefined,
            "/host=master" => undefined,
            "/system-property=*" => undefined,
            "/" => undefined
        }
    }
}
[domain@localhost:9999 classification=system-property]
```

Les rôles qui seront en mesure d'effectuer ces opérations en fonction de la configuration de ces attributs sont résumés dans Tableau 6.2, « Résultats des configurations de contraintes de sensibilité ».

Tableau 6.2. Résultats des configurations de contraintes de sensibilité

Valeur	requires-read	requires-write	requires-addressable
true	L'opération lecture est sensible. Seuls les rôles **Auditor**, **Administrator**, et **SuperUser** peuvent lire.	L'opération écriture est sensible. Seuls les rôles **Administrator** et **SuperUser** peuvent écrire.	L'opération Adressage est sensible. Seuls les rôles d'**Auditor**, **Administrator**, et **SuperUser** peuvent lire.
false	L'opération lecture n'est pas sensible. Tous les utilisateurs de gestion peuvent lire.	L'opération écriture n'est pas sensible. Les rôles **Maintainer**, **Administrator**, et **SuperUser** peuvent écrire. Les rôles **Deployers** peuvent également écrire dans une ressource d'applications.	L'opération Adressage n'est pas sensible. Tous les utilisateurs de gestion peuvent adresser.

Rapporter un bogue

6.10.2. Configurer les contraintes de ressources d'application

Chaque contrainte de ressource d'application définit un ensemble de ressources, d'attributs et d'opérations généralement associées avec le déploiement d'applications et de services. Lorsqu'une contrainte de ressource d'application est activée, les utilisateurs gestionnaires du rôle Déployeur peuvent

accéder aux ressources s'appliquant.

La configuration de contraintes d'applications se trouve dans le Modèle des gestion qui se trouve dans **`/core-service=management/access=authorization/constraint=application-classification/`**.

Dans le modèle de gestion, chaque contrainte de ressource d'application est identifiée comme une **`classification`**. Les classifications sont ensuite regroupées en **`types`**. Il existe 14 catégories incluses qui sont regroupées en 8 types. Chaque classification comporte un élément **`applies-to`** qui correspond à une liste de modèles de chemins d'accès de ressources s'appliquant à la configuration de la classification.

Par défaut, la seule classification de ressource d'application activée est **`core`**. Core inclut des déploiements, des superpositions de déploiement et des opérations de déploiement.

Afin d'activer une ressource d'application, l'opération **`write-attribute`** permet de définir l'attribut **`configured-application attribute`** de la classification sur **`true`**. Pour désactiver une ressource d'application, définir cet attribut sur **`false`**. Par défaut, ces attributs ne sont pas définis et la valeur **`default-application attribute`** sera utilisée. La valeur par défaut ne peut pas être modifiée.

Exemple 6.6. Activer la classification des ressources d'application logger-profile

```
[domain@localhost:9999 /] cd /core-
service=management/access=authorization/constraint=application-
classification/type=logging/classification=logging-profile
[domain@localhost:9999 classification=logging-profile] :write-
attribute(name=configured-application, value=true)
{
    "outcome" => "success",
    "result" => undefined,
    "server-groups" => {"main-server-group" => {"host" => {"master" => {
        "server-one" => {"response" => {"outcome" => "success"}},
        "server-two" => {"response" => {"outcome" => "success"}}
    }}}}
}
[domain@localhost:9999 classification=logging-profile] :read-resource
{
    "outcome" => "success",
    "result" => {
        "configured-application" => true,
        "default-application" => false,
        "applies-to" => {"/profile=*/subsystem=logging/logging-
profile=*" => undefined}
    }
}
[domain@localhost:9999 classification=logging-profile]
```

IMPORTANT

Les contraintes de ressources d'application s'appliquent à toutes les ressources qui correspondent à sa configuration. Par exemple, il n'est pas possible d'accorder à un utilisateur ayant la permission **`Deployer`** d'accéder à une ressource de source de données mais pas à une autre. Si ce niveau de séparation est nécessaire, il est recommandé de configurer les ressources dans différents groupes de serveurs et de créer les rôles **`Deployer`** d'étendue différente pour chaque groupe.

Rapporter un bogue

6.10.3. Configuration de contraintes d'expressions d'archivage sécurisé

Par défaut, la lecture et l'écriture d'expressions d'archivage sécurisé sont des opérations sensibles. La configuration de contraintes d'expression d'archivage sécurisé permet de définir l'une, ou ces deux opérations à être insensibles. Modifier cette contrainte permet à un plus grand nombre de rôles de lire et d'écrire des expressions d'archivage sécurisé.

La contrainte d'expression de sécurité se trouve dans le modèle de gestion de **`/core-service=management/access=authorization/constraint=vault-expression`**.

Pour configurer la contrainte d'expression d'archivage sécurisé, utilisez l'opération **`write-attribute`** pour définir les attributs de **`configured-requires-write`** et **`configured-requires-read`** à **`true`** ou **`false`**. Par défaut, celles-ci ne sont pas définies et les valeurs **`default-requires-read`** et **`default-requires-write`** sont utilisées. Impossible de modifier les valeurs par défaut.

Exemple 6.7. Comment rendre l'écriture d'expressions d'archivage une opération non sensible

```
[domain@localhost:9999 /] cd /core-
service=management/access=authorization/constraint=vault-expression
[domain@localhost:9999 constraint=vault-expression] :write-
attribute(name=configured-requires-write, value=false)
{
    "outcome" => "success",
    "result" => undefined,
    "server-groups" => {"main-server-group" => {"host" => {"master" => {
        "server-one" => {"response" => {"outcome" => "success"}},
        "server-two" => {"response" => {"outcome" => "success"}}
    }}}}
}
[domain@localhost:9999 constraint=vault-expression] :read-resource
{
    "outcome" => "success",
    "result" => {
        "configured-requires-read" => undefined,
        "configured-requires-write" => false,
        "default-requires-read" => true,
        "default-requires-write" => true
    }
}
[domain@localhost:9999 constraint=vault-expression]
```

Les rôles qui seront en mesure de lire et d'écrire dans les expressions d'archivage sécurisé en fonction de cette configuration sont résumés dans Tableau 6.3, « Résultats des configurations de contraintes d'expressions d'archivage sécurisé ».

Tableau 6.3. Résultats des configurations de contraintes d'expressions d'archivage sécurisé

Valeur	`requires-read`	`requires-write`
`true`	L'opération lecture est sensible. Seuls les rôles d'`Auditor`, `Administrator`, et `SuperUser` peuvent lire.	L'opération écriture est sensible. Seuls les rôles `Administrator`, et `SuperUser` peuvent écrire.
`false`	L'opération lecture n'est pas sensible. Tous les utilisateurs de gestion peuvent lire.	L'opération d'écriture n'est pas sensible. Les rôles `Monitor`, `Administrator`, et `SuperUser` peuvent écrire. Les rôles `Deployers` peuvent également écrire si l'expression d'archivage de sécurité est dans une ressource d'application.

Rapporter un bogue

6.11. RÉFÉRENCES DE CONTRAINTES

6.11.1. Références de contraintes de ressources d'application

Type: core

Classification: deployment-overlay

- default: true
- PATH: /deployment-overlay=*
- PATH: /deployment=*
- PATH: /

  Opération :

  upload-deployment-stream, full-replace-deployment, upload-deployment-url, upload-deployment-bytes

Type: datasources

Classification: datasource

- default: false
- PATH: /deployment=*/subdeployment=*/subsystem=datasources/data-source=*

- PATH: /subsystem=datasources/data-source=*
- PATH: /subsystem=datasources/data-source=ExampleDS
- PATH: /deployment=*/subsystem=datasources/data-source=*

Classification: jdbc-driver

- default: false
- PATH: /subsystem=datasources/jdbc-driver=*

Classification: xa-data-source

- default: false
- PATH: /subsystem=datasources/xa-data-source=*
- PATH: /deployment=*/subsystem=datasources/xa-data-source=*
- PATH: /deployment=*/subdeployment=*/subsystem=datasources/xa-data-source=*

Type: logging

Classification: logger

- default: false
- PATH: /subsystem=logging/logger=*
- PATH: /subsystem=logging/logging-profile=*/logger=*

Classification: logging-profile

- default: false
- PATH: /subsystem=logging/logging-profile=*

Type: mail

Classification: mail-session

- default: false
- PATH: /subsystem=mail/mail-session=*

Type: naming

Classification: binding

- default: false
- PATH: /subsystem=naming/binding=*

Type: resource-adapters

Classification: resource-adapters

- default: false
- PATH: /subsystem=resource-adapters/resource-adapter=*

Type: security

Classification: security-domain

- default: false
- PATH: /subsystem=security/security-domain=*

Rapporter un bogue

6.11.2. Références de contraintes de sensibilité

Type: core

Classification: access-control

- requires-addressable: true
- requires-read: true
- requires-write: true
- PATH: /core-service=management/access=authorization
- PATH: /subsystem=jmx ATTRIBUTE: non-core-mbean-sensitivity

Classification: credential

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=mail/mail-session=*/server=pop3 ATTRIBUTE: username , password
- PATH: /subsystem=mail/mail-session=*/server=imap ATTRIBUTE: username , password
- PATH: /subsystem=datasources/xa-data-source=* ATTRIBUTE: user-name, recovery-username, password, recovery-password
- PATH: /subsystem=mail/mail-session=*/custom=* ATTRIBUTE: username, password
- PATH: /subsystem=datasources/data-source=*" ATTRIBUTE: user-name, password
- PATH: /subsystem=remoting/remote-outbound-connection=*" ATTRIBUTE: username

- PATH: /subsystem=mail/mail-session=*/server=smtp ATTRIBUTE: username, password
- PATH: /subsystem=web/connector=*/configuration=ssl ATTRIBUTE: key-alias, password
- PATH: /subsystem=resource-adapters/resource-adapter=*/connection-definitions=*" ATTRIBUTE: recovery-username, recovery-password

Classification: domain-controller

- requires-addressable: false
- requires-read: false
- requires-write: true

Classification: domain-names

- requires-addressable: false
- requires-read: false
- requires-write: true

Classification: extensions

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /extension=*

Classification: jvm

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=platform-mbean/type=runtime ATTRIBUTE: input-arguments, boot-class-path, class-path, boot-class-path-supported, library-path

Classification: management-interfaces

- requires-addressable: false
- requires-read: false
- requires-write: true
- /core-service=management/management-interface=native-interface
- /core-service=management/management-interface=http-interface

Classification: module-loading

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=module-loading

Classification: patching

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=patching/addon=*
- PATH: /core-service=patching/layer=*"
- PATH: /core-service=patching

Classification: read-whole-config

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: / OPERATION: read-config-as-xml

Classification: security-domain

- requires-addressable: true
- requires-read: true
- requires-write: true
- PATH: /subsystem=security/security-domain=*

Classification: security-domain-ref

- requires-addressable: true
- requires-read: true
- requires-write: true
- PATH: /subsystem=datasources/xa-data-source=* ATTRIBUTE: security-domain
- PATH: /subsystem=datasources/data-source=* ATTRIBUTE: security-domain
- PATH: /subsystem=ejb3 ATTRIBUTE: default-security-domain

- PATH: /subsystem=resource-adapters/resource-adapter=*/connection-definitions=* ATTRIBUTE: security-domain, recovery-security-domain, security-application, security-domain-and-application

Classification: security-realm

- requires-addressable: true
- requires-read: true
- requires-write: true
- PATH: /core-service=management/security-realm=*

Classification: security-realm-ref

- requires-addressable: true
- requires-read: true
- requires-write: true
- PATH: /subsystem=remoting/connector=* ATTRIBUTE: security-realm
- PATH: /core-service=management/management-interface=native-interface ATTRIBUTE: security-realm
- PATH: /core-service=management/management-interface=http-interface ATTRIBUTE: security-realm
- PATH: /subsystem=remoting/remote-outbound-connection=* ATTRIBUTE: security-realm

Classification: security-vault

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=vault

Classification: service-container

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=service-container

Classification: snapshots

- requires-addressable: false

- requires-read: false
- requires-write: false
- PATH: / ATTRIBUTE: take-snapshot, list-snapshots, delete-snapshot

Classification: socket-binding-ref

- requires-addressable: false
- requires-read: false
- requires-write: false
- PATH: /subsystem=mail/mail-session=*/server=pop3 ATTRIBUTE: outbound-socket-binding-ref
- PATH: /subsystem=mail/mail-session=*/server=imap ATTRIBUTE: outbound-socket-binding-ref
- PATH: /subsystem=remoting/connector=* ATTRIBUTE: socket-binding
- PATH: /subsystem=web/connector=* ATTRIBUTE: socket-binding
- PATH: /subsystem=remoting/local-outbound-connection=* ATTRIBUTE: outbound-socket-binding-ref
- PATH: /socket-binding-group=*/local-destination-outbound-socket-binding=* ATTRIBUTE: socket-binding-ref
- PATH: /subsystem=remoting/remote-outbound-connection=* ATTRIBUTE: outbound-socket-binding-ref
- PATH: /subsystem=mail/mail-session=*/server=smtp ATTRIBUTE: outbound-socket-binding-ref
- PATH: /subsystem=transactions ATTRIBUTE: process-id-socket-binding, status-socket-binding, socket-binding

Classification: socket-config

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /interface=* OPERATION: resolve-internet-address
- PATH: /core-service=management/management-interface=native-interface ATTRIBUTE: port, interface, socket-binding
- PATH: /socket-binding-group=*
- PATH: /core-service=management/management-interface=http-interface ATTRIBUTE: port, secure-port, interface, secure-socket-binding, socket-binding

- PATH: / OPERATION: resolve-internet-address
- PATH: /subsystem=transactions ATTRIBUTE: process-id-socket-max-ports

Classification: system-property

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=platform-mbean/type=runtime ATTRIBUTE: system-properties
- PATH: /system-property=*
- PATH: / OPERATION: resolve-expression

Type: datasources

Classification: data-source-security

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=datasources/xa-data-source=* ATTRIBUTE: user-name, security-domain, password
- PATH: /subsystem=datasources/data-source=* ATTRIBUTE: user-name, security-domain, password

Type: jdr

Classification: jdr

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /subsystem=jdr OPERATION: generate-jdr-report

Type: jmx

Classification: jmx

- requires-addressable: false
- requires-read: false

- requires-write: true
- PATH: /subsystem=jmx

Type: mail

Classification: mail-server-security

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /subsystem=mail/mail-session=*/server=pop3 ATTRIBUTE: username, tls, ssl, password
- PATH: /subsystem=mail/mail-session=*/server=imap ATTRIBUTE: username, tls, ssl, password
- PATH: /subsystem=mail/mail-session=*/custom=* ATTRIBUTE: username, tls, ssl, password
- PATH: /subsystem=mail/mail-session=*/server=smtp ATTRIBUTE: username, tls, ssl, password

Type: naming

Classification: jndi-view

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=naming OPERATION: jndi-view

Classification: naming-binding

- requires-addressable: false
- requires-read: false
- requires-write: false
- PATH: /subsystem=naming/binding=*

Type: remoting

Classification: remoting-security

- requires-addressable: false
- requires-read: true

- requires-write: true
- PATH: /subsystem=remoting/connector=* ATTRIBUTE: authentication-provider, security-realm
- PATH: /subsystem=remoting/remote-outbound-connection=* ATTRIBUTE: username, security-realm
- PATH: /subsystem=remoting/connector=*/security=sasl

Type: resource-adapters

Classification: resource-adapter-security

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=resource-adapters/resource-adapter=*/connection-definitions=* ATTRIBUTE: security-domain, recovery-username, recovery-security-domain, security-application, security-domain-and-application, recovery-password

Type: security

Classification: misc-security

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=security ATTRIBUTE: deep-copy-subject-mode

Type: web

Classification: web-access-log

- requires-addressable: false
- requires-read: false
- requires-write: false
- PATH: /subsystem=web/virtual-server=*/configuration=access-log

Classification: web-connector

- requires-addressable: false
- requires-read: false

- requires-write: false
- PATH: /subsystem=web/connector=*

Classification: web-ssl

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=web/connector=*/configuration=ssl

Classification: web-sso

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=web/virtual-server=*/configuration=sso

Classification: web-valve

- requires-addressable: false
- requires-read: false
- requires-write: false
- PATH: /subsystem=web/valve=*

Rapporter un bogue

CHAPITRE 7. SÉCURISER LES MOTS DE PASSE ET LES AUTRES CHAÎNES SENSIBLES PAR UN ARCHIVAGE SÉCURISÉ.

7.1. SYSTÈME D'ARCHIVAGE SÉCURISÉ DE MOTS DE PASSE

JBoss EAP 6 possède un archivage sécurisé de mots de passe pour encrypter les strings sensibles dans un keystore crypté, et les décrypter pour les applications et les systèmes de vérification.

Les fichiers de configuration en texte brut, tels que les descripteurs de déploiement XML, doivent spécifier des mots de passe et autres informations sensibles.

Utiliser l'archivage de mots de passe sécurisé de JBoss EAP pour sécuriser vos strings sensibles dans des fichiers en texte brut.

Rapporter un bogue

7.2. CRÉER UN KEYSTORE JAVA POUR STOCKER DES STRINGS SENSIBLES

Pré-requis

- La commande **`keytool`** doit être disponible. Elle est fournie par le Java Runtime Environment (JRE). Chercher le chemin du fichier. Se trouve à l'emplacement suivant **`/usr/bin/keytool`** dans Red Hat Enterprise Linux.

AVERTISSEMENT

Les implémentations de keystore JCEKS diffèrent entre les fournisseurs Java. Vous devez générer le **`vault.keystore`** à l'aide du **`keytool`** du même fournisseur que le JDK que vous utilisez.

Si vous utilisez un archivage sécurisé généré par un **`keytool`** de JDK d'un fournisseur dans une instance EAP exécutant dans un JDK provenant d'un fournisseur différent, vous aurez l'exception suivante :

```
java.io.IOException:
com.sun.crypto.provider.SealedObjectForKeyProtector
```

Procédure 7.1. Installation du Java Keystore

1. **Créez un répertoire pour stocker votre keystore et autres informations cryptées.**
   Créez un répertoire qui contiendra votre keystore et autres informations pertinentes. Le reste de cette procédure assume que le répertoire est ***`EAP_HOME`*`/vault/`**. Comme le répertoire devra contenir des informations sensibles, il devra être accessible à un nombre restraint d'utilisateurs.

Au minimum, le compte d'utilisateur sous lequel JBoss EAP exécute requiert un accès en lecture-écriture.

2. **Déterminer les paramètres à utiliser avec `keytool`.**
   Déterminer les paramètres suivants :

   alias

   L'alias est un identificateur unique pour l'archivage sécurisé ou autres données stockées dans le keystore. L'alias dans l'exemple de commande à la fin de cette procédure est **`vault`** (archivage sécurisé). Les alias sont insensibles à la casse.

   keyalg

   L'algorithme à utiliser pour le cryptage. Dans cette procédure, l'exemple utilise **RSA**. Consultez la documentation de votre JRE et de votre système d'exploitation pour étudier vos possibilités.

   keysize

   La taille d'une clé de cryptage impacte sur la difficulté de décrypter au seul moyen de la force brutale. Dans cette procédure, l'exemple utilise **`1024`**. Pour plus d'informations sur les valeurs appropriées, voir la documentation distribuée avec **`keytool`**.

   keystore

   Le keystore est une base de données qui contient des informations chiffrées et des informations sur la façon de décrypter. Si vous ne spécifiez pas de keystore, le keystore par défaut à utiliser est un fichier appelé **`.keystore`** qui se trouve dans votre répertoire personnel. La première fois que vous ajoutez des données dans un keystore, il sera créé. L'exemple de cette procédure utilise le keystore **`vault.keystore`**.

   La commande du **`keytool`** a plusieurs options. Consultez la documentation de votre JRE ou de votre système d'exploitation pour obtenir plus d'informations.

3. **Déterminez les réponses aux questions que la commande `keystore` vous demandera.**
   Le **`keystore`** a besoin des informations suivantes pour remplir l'entrée du keytore :

   Mot de passe du keystore

   Lorsque vous créez un keystore, vous devez définir un mot de passe. Pour pouvoir travailler dans le keystore dans le futur, vous devez fournir le mot de passe. Créez un mot de passe dont vous vous souviendrez. Le keystore est sécurisé par son mot de passe, et par la sécurité du système d'exploitation et du système de fichiers où il se trouve.

   Mot de passe clé (en option)

   En plus du mot de passe du keystore, vous pouvez indiquer un mot de passe pour chaque clé contenue. Pour utiliser une clé, le mot de passe doit être donné à chaque utilisation. Normalement, cette fonction n'est pas utilisée.

   Prénom et nom de famille

   Cela, ainsi que le reste de l'information dans la liste, aide à identifier la clé de façon unique et à la placer dans une hiérarchie par rapport aux autres clés. Ne doit pas nécessairement correspondre à un nom, mais doit être composé de deux mots et doit être unique à une clé. L'exemple dans cette procédure utilise **`Admninistrateur Comptabilité`**. En terme de répertoires, cela devient le *nom commun* du certificat.

Unité organisationnelle

Il s'agit d'un mot unique d'identification qui utilise le certificat. Il se peut que ce soit l'application ou l'unité commerciale. L'exemple de cette procédure utilise **`enterprise_application_platform`**. Normalement, tous les keystores utilisés par un groupe ou une application utilisent la même unité organisationnelle.

Organisation

Il s'agit normalement d'une représentation de votre nom d'organisation en un seul mot. Demeure constant à travers tous les certificats qui sont utilisés par une organisation. Cet exemple utilise **`MyOrganization`**.

Ville ou municipalité

Votre ville.

État ou province

Votre état ou province, ou l'équivalent pour votre localité.

Pays

Le code pays en deux lettres.

Ces informations vont créer ensemble une hiérarchie de vos keystores et certificats, qui garantira qu'ils utilisent une structure de nommage consistante, et unique.

4. **Exécuter la commande `keytool`, en fournissant les informations que vous avez collectées.**

Exemple 7.1. Exemple d'entrée et de sortie de la commande `keystore`

```
$ keytool -genseckey -alias vault -storetype jceks -keyalg AES -
keysize 128 -storepass vault22 -keypass vault22 -validity 730 -
keystore EAP_HOME/vault/vault.keystore
Enter keystore password: vault22
Re-enter new password:vault22
What is your first and last name?
  [Unknown]:  Accounting Administrator
What is the name of your organizational unit?
  [Unknown]:  AccountingServices
What is the name of your organization?
  [Unknown]:  MyOrganization
What is the name of your City or Locality?
  [Unknown]:  Raleigh
What is the name of your State or Province?
  [Unknown]:  NC
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=Accounting Administrator, OU=AccountingServices,
O=MyOrganization, L=Raleigh, ST=NC, C=US correct?
  [no]:  yes

Enter key password for <vault>
        (RETURN if same as keystore password):
```

Résultat

Un fichier nommé **vault.keystore** est créé dans le répertoire ***EAP_HOME*/vault/**. Il stocke une clé simple, nommée **vault**, qui sera utilisée pour stocker des strings cryptés, comme des mots de passe, pour la plateforme JBoss EAP 6.

Rapporter un bogue

7.3. MASQUER LE MOT DE PASSE DU KEYSTORE ET INITIALISER LE MOT DE PASSE DE L'ARCHIVAGE DE SÉCURITÉ

Pré-requis

- Section 7.2, « Créer un Keystore Java pour stocker des strings sensibles »

1. **Exécuter la commande vault.sh.**
   Exécuter ***EAP_HOME*/bin/vault.sh**. Démarrer une nouvelle session interactive en tapant **0**.

2. **Saisir le nom du répertoire où les fichiers cryptés seront stockés.**
   Ce répertoire doit être accessible à des utilisateurs limités uniquement. Au minimum, le compte d'utilisateur sous lequel JBoss EAP exécute requiert un accès lecture-écriture. Si vous avez suivi Section 7.2, « Créer un Keystore Java pour stocker des strings sensibles », votre keystore se trouvera dans le répertoire nommé ***EAP_HOME*/vault/**.

   NOTE

   N'oubliez pas d'inclure la barre oblique finale dans le nom du répertoire. Soit **/** ou ****, selon votre système d'exploitation.

3. **Saisir le nom de votre keystore.**
   Saisir le nom complet vers le fichier de keystore. Cet exemple utilise ***EAP_HOME*/vault/vault.keystore**.

4. **Crypter le mot de passe du keystore.**
   Les étapes suivantes vous servent à crypter le mot de passe du keystore, afin que vous puissiez l'utiliser dans les applications et les fichiers de configuration en toute sécurité.

   a. **Saisir le mot de passe du keystore.**
      Quand vous y serez invité, saisir le mot de passe du keystore.

   b. **Saisir une valeur salt.**
      Entrez une valeur salt de 8 caractères. La valeur salt, ainsi que le nombre d'itérations (ci-dessous), sont utilisés pour créer la valeur de hachage

   c. **Saisir le nombre d'itérations.**
      Saisir un nombre pour le nombre d'itérations.

   d. **Notez les informations de mot de passe masqué.**
      Le mot de passe masqué, salt, et le nombre d'itérations sont imprimés en sortie standard. Prenez-en note dans un endroit sûr. Un attaquant pourrait les utiliser pour déchiffrer le mot de passe.

   e. **Saisir un alias pour l'archivage de sécurité.**

Quand on vous y invite, saisir un alias pour l'archivage de sécurité. Si vous suivez Section 7.2, « Créer un Keystore Java pour stocker des strings sensibles » pour créer votre archivage de sécurité, l'alias sera **`vault`**.

5. **Sortir de la console interactive.**
   Saisir **2** pour sortir de la console interactive.

Résultat

Votre mot de passe de keystore est masqué afin de pouvoir être utilisé dans les fichiers de configuration et de déploiement. De plus, votre archivage de sécurité est complètement configuré et prêt à l'utilisation.

Rapporter un bogue

7.4. CONFIGURER JBOSS EAP POUR QU'IL UTILISE L'ARCHIVAGE SÉCURISÉ DES MOTS DE PASSE

Aperçu

Avant de masquer les mots de passe et d'autres attributs sensibles dans les fichiers de configuration, vous devez sensibiliser JBoss EAP 6 à l'archivage sécurisé des mots de passe qui les stocke et les déchiffre. Suivez cette procédure pour activer cette fonctionnalité.

Pré-requis

- Section 7.2, « Créer un Keystore Java pour stocker des strings sensibles »
- Section 7.3, « Masquer le mot de passe du keystore et initialiser le mot de passe de l'archivage de sécurité »

Procédure 7.2. Assigner un mot de passe d'archivage sécurisé.

1. **Déterminer les valeurs qui conviennent pour la commande.**
   Déterminer les valeurs pour les paramètres suivants, qui sont déterminés par les commandes utilisées pour créer le keystore lui-même. Pour obtenir des informations sur la façon de créer un keystore, voir les sujets suivants : Section 7.2, « Créer un Keystore Java pour stocker des strings sensibles » et Section 7.3, « Masquer le mot de passe du keystore et initialiser le mot de passe de l'archivage de sécurité ».

Paramètre	Description
KEYSTORE_URL	Le chemin d'accès ou URI du fichier keystore, qui s'appelle normalement **`vault.keystore`**
KEYSTORE_PASSWORD	Le mot de passe utilisé pour accéder au keystore. Cette valeur devrait être masquée.
KEYSTORE_ALIAS	Le nom du keystore.
SALT	Le salt utilisé pour crypter et décrypter les valeurs de keystore.
ITERATION_COUNT	Le nombre de fois que l'algorithme de chiffrement est exécuté.

Paramètre	Description
ENC_FILE_DIR	Le chemin d'accès au répertoire à partir duquel les commandes de keystore sont exécutées. Normalement, le répertoire contient les mots de passe sécurisés.
hôte (domaine géré uniquement)	Le nom de l'hôte que vous configurez

2. **Utilisez l'interface CLI pour activer les mots de passe sécurisés.**
Exécutez une des commandes suivantes, selon que vous utilisez un domaine géré ou une configuration de serveur autonome. Substituez les valeurs de la commande par celles de la première étape de cette procédure.

NOTE

Si vous utilisez Microsoft Windows Server, remplacez chaque caractère `\` du chemin d'accès du répertoire par un caractère `\` supplémentaire dans la commande CLI. Par exemple, **`C:\\data\\vault\\vault.keystore`**. C'est parce qu'un simple caractère `\` est utilisé comme caractère d'échappement.

- **Domaine géré**

```
/host=YOUR_HOST/core-service=vault:add(vault-options=
[("KEYSTORE_URL" => "PATH_TO_KEYSTORE"), ("KEYSTORE_PASSWORD" =>
"MASKED_PASSWORD"), ("KEYSTORE_ALIAS" => "ALIAS"), ("SALT" =>
"SALT"),("ITERATION_COUNT" => "ITERATION_COUNT"), ("ENC_FILE_DIR"
=> "ENC_FILE_DIR")])
```

- **Serveur autonome**

```
/core-service=vault:add(vault-options=[("KEYSTORE_URL" =>
"PATH_TO_KEYSTORE"), ("KEYSTORE_PASSWORD" => "MASKED_PASSWORD"),
("KEYSTORE_ALIAS" => "ALIAS"), ("SALT" => "SALT"),
("ITERATION_COUNT" => "ITERATION_COUNT"), ("ENC_FILE_DIR" =>
"ENC_FILE_DIR")])
```

Ce qui suit est un exemple de la commande avec des valeurs hypothétiques :

```
/core-service=vault:add(vault-options=[("KEYSTORE_URL" =>
"/home/user/vault/vault.keystore"), ("KEYSTORE_PASSWORD" => "MASK-
3y28rCZlcKR"), ("KEYSTORE_ALIAS" => "vault"), ("SALT" =>
"12438567"),("ITERATION_COUNT" => "50"), ("ENC_FILE_DIR" =>
"/home/user/vault/")])
```

Résultat

JBoss EAP 6 est configuré pour décrypter les strings masqués par l'intermédiaire de l'archivage sécurisé de mots de passe. Pour ajouter des strings à l'archivage sécurisé et les utiliser dans votre configuration, voir la section suivante : Section 7.6, « Stocker et résoudre des strings sensibles cryptés du Keystore Java. ».

Rapporter un bogue

7.5. CONFIGURER JBOSS EAP POUR QU'IL UTILISE UNE IMPLÉMENTATION D'ARCHIVAGE SÉCURISÉ PERSONNALISÉ

Résumé

Vous pouvez utiliser votre propre implémentation de **`SecurityVault`** pour masquer les mots de passe, et les autres attributs sensibles dans les fichiers de configuration.

Procédure 7.3. Utilisez une implémentation sécurisée de l'archivage des mots de passe

1. Créez une classe qui implémente l'interface **`SecurityVault`**.
2. Créez un module contenant la classe de l'étape précédente, et spécifiez une dépendance sur **`org.picketbox`** où l'interface est **`SecurityVault`**.
3. Activez l'archivage de mots de passe sécurisé dans la configuration du serveur JBoss EAP en ajoutant l'élément d'archivage à l'aide des attributs suivants :

   code

   La nom complet de la classe qui implémente **`SecurityVault`**.

   module

   Le nom du module qui contient la classe personnalisée.

   Optionnellement, vous pouvez utiliser les paramètres **`vault-options`** pour initialiser la classe personnalisée d'un archivage de mots de passe. Par exemple :

```
/core-
service=vault:add(code="custom.vault.implementation.CustomSecurityVa
ult", module="custom.vault.module", vault-options=[("KEYSTORE_URL"
=> "PATH_TO_KEYSTORE"), ("KEYSTORE_PASSWORD" => "MASKED_PASSWORD"),
("KEYSTORE_ALIAS" => "ALIAS"), ("SALT" => "SALT"),("ITERATION_COUNT"
=> "ITERATION_COUNT"), ("ENC_FILE_DIR" => "ENC_FILE_DIR")])
```

Résultat

JBoss EAP 6 est configuré pour décrypter les chaînes masquées par l'intermédiaire d'une implémentation personnalisée de l'archivage des mots de passe.

Rapporter un bogue

7.6. STOCKER ET RÉSOUDRE DES STRINGS SENSIBLES CRYPTÉS DU KEYSTORE JAVA.

Résumé

En comptant les mots de passe et les autres strings sensibles, les fichiers de configuration en texte brut ne sont pas sécurisés. JBoss EAP 6 inclut la capacité à stocker et à utiliser les valeurs masquées dans les fichiers de configuration, et à utiliser ces valeurs masquées dans les fichiers de configuration.

Pré-requis

- Section 7.2, « Créer un Keystore Java pour stocker des strings sensibles »
- Section 7.3, « Masquer le mot de passe du keystore et initialiser le mot de passe de l'archivage de sécurité »
- Section 7.4, « Configurer JBoss EAP pour qu'il utilise l'archivage sécurisé des mots de passe »
- L'application ***EAP_HOME*/bin/vault.sh** doit pouvoir être accessible via l'interface de ligne de commande.

Procédure 7.4. Installation du Java Keystore

1. **Exécutez la commande `vault.sh`.**
   Exécutez ***EAP_HOME*/bin/vault.sh**. Démarrez une nouvelle session interactive en tapant **0**.

2. **Saisir le nom du répertoire où les fichiers cryptés seront stockés.**
   Si vous suivez Section 7.2, « Créer un Keystore Java pour stocker des strings sensibles », votre keystore sera dans un répertoire nommé **vault/** de votre répertoire de base. Dans la plupart des cas, il est logique de stocker toutes vos informations cryptées au même endroit dans le keystore. Cet exemple utilise le répertoire **/home/*USER*/vault/**.

   NOTE

   N'oubliez pas d'inclure la barre oblique finale dans le nom du répertoire. Soit `/` ou `\` selon votre système d'exploitation.

3. **Saisissez le nom de votre keystore.**
   Saisissez le nom complet vers le fichier de keystore. Cet exemple utilise **/home/*USER*/vault/vault.keystore**.

4. **Saisissez le mot de passe du keystore, le nom de l'archivage sécurisé, salt, et le nombre d'itérations.**
   Quand vous y êtes invité, saisissez le mot de passe du keystore, le nom de l'archivage sécurisé, salt, et le nombre d'itérations.

5. **Sélectionnez l'option de stockage d'un mot de passe.**
   Sélectionnez l'option **0** de stockage d'un mot de passe ou autre string sensible.

6. **Saisissez la valeur.**
   Une fois que vous y êtes invité, saisissez la valeur à deux reprises. Si les valeurs ne correspondent pas, vous serez invité à essayer à nouveau.

7. **Saisissez le bloc d'archivage sécurisé.**
   Saisissez le bloc d'archivage sécurisé, qui correspond à un conteneur pour les attributs ayant trait à la même ressource. Un exemple de nom d'attribut pourrait être **ds_ExampleDS**. Cela fera partie de la référence à la chaîne cryptée, dans votre source de données ou autre définition de service.

8. **Saissez le nom de l'attribut.**
   Saisissez le nom de l'attribut que vous stockez. Exemple de nom d'attribut **password**.

   Résultat

   Un message comme celui qui suit montre que l'attribut a été sauvegardé.

   ■

```
La valeur d'attribut sécurisé est stockée dans l'archivage sécurisé.
```

9. **Notez les informations pour ce string crypté.**
   Un message s'affiche sur la sortie standard, montrant le bloc d'archivage sécurisé, le nom de l'attribut, la clé partagée et des conseils sur l'utilisation du string dans votre configuration. Prendre note de ces informations dans un emplacement sécurisé. Voici un exemple de sortie.

```
*********************************************
Vault Block:ds_ExampleDS
Attribute Name:password
Configuration should be done as follows:
VAULT::ds_ExampleDS::password::1
*********************************************
```

10. **Utilisez le string crypté dans votre configuration.**
    Utilisez le string de l'étape de configuration précédente, à la place du string en texte brut. Une source de données utilisant le mot de passe crypté ci-dessus, est indiquée ci-dessous.

```
...
  <subsystem xmlns="urn:jboss:domain:datasources:1.0">
    <datasources>
      <datasource jndi-name="java:jboss/datasources/ExampleDS"
enabled="true" use-java-context="true" pool-name="H2DS">
        <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-
1</connection-url>
        <driver>h2</driver>
        <pool></pool>
        <security>
          <user-name>sa</user-name>
          <password>${VAULT::ds_ExampleDS::password::1}</password>
        </security>
      </datasource>
      <drivers>
         <driver name="h2" module="com.h2database.h2">
            <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-
datasource-class>
         </driver>
      </drivers>
    </datasources>
  </subsystem>
...
```

Vous pouvez utiliser un string crypté n'importe où dans votre fichier de configuration autonome ou de domaine pour lequel les expressions sont autorisées.

NOTE

Pour vérifier si les expressions sont utilisées dans un sous-système particulier, exécutez la commande CLI suivante sur ce sous-système :

```
/host=master/core-service=management/security-
realm=TestRealm:read-resource-description(recursive=true)
```

À partir du résultat de cette commande, cherchez la valeur du paramètre **`expressions-allowed`**. Si 'true', vous pourrez utiliser des expressions dans la configuration de ce sous-système particulier.

Une fois que vous aurez mis votre string dans le keystore, utilisez la syntaxe suivante pour remplacer tout string en texte claire par un texte crypté.

```
${VAULT::VAULT_BLOCK::ATTRIBUTE_NAME::ENCRYPTED_VALUE}
```

Voici un exemple de valeur réelle, où le bloc d'archivage sécurisé est **`ds_ExampleDS`** et l'attribut est **`password`**.

```
<password>${VAULT::ds_ExampleDS::password::1}</password>
```

Rapporter un bogue

7.7. STOCKER ET RÉSOUDRE DES STRINGS SENSIBLES DE VOS APPLICATIONS

Aperçu

Les éléments de configuration de la plate-forme JBoss EAP 6 prennent en charge la capacité à régler les chaînes cryptées en fonction des valeurs stockées dans Java Keystore, via le mécanisme Security Vault. Vous pouvez ajouter le support pour cette fonctionnalité à vos propres applications.

Tout d'abord, ajoutez le mot de passe dans votre Security Vault. En second lieu, remplacez le mot de passe de texte clair par celui qui est stocké dans le Security Vault. Vous pouvez utiliser cette méthode pour obscurcir les strings sensibles de votre application.

Pré-requis

Avant d'effectuer cette procédure, assurez-vous que le répertoire utilisé pour stocker vos fichiers dans le Security Vault existe bien. Qu'importe où vous les placez, tant que l'utilisateur qui exécute JBoss EAP 6 dispose de l'autorisation de lire et écrire des fichiers. Cet exemple situe le répertoire **`vault/`** dans le répertoire **`/home/`*`USER`*`/vault/`**. Le Security Vault lui-même correspond à un fichier nommé **`vault.keystore`** qui se trouve dans le répertoire **`vault/`**.

Exemple 7.2. Ajout du string de mot de passe au Security Vault

Ajoutez le string au Security Vault par la commande ***`EAP_HOME`*`/bin/vault.sh`**. La série de commandes et de réponses est incluse dans la session suivante. Les valeurs saisies par l'utilisateur apparaîtront clairement. Certaines sorties seront supprimées pour le formatage. Dans Microsoft Windows, le nom de la commande est **`vault.bat`**. Notez que dans Microsoft Windows, les chemins d'accès au fichier utilisent le caractère `\` comme séparateur de répertoire, et non pas le caractère `/`.

```
[user@host bin]$ ./vault.sh
*********************************
****  JBoss Vault ********
*********************************
Please enter a Digit::   0: Start Interactive Session  1: Remove
Interactive Session  2: Exit
0
Starting an interactive session
Enter directory to store encrypted files:/home/user/vault/
Enter Keystore URL:/home/user/vault/vault.keystore
Enter Keystore password: ...
Enter Keystore password again: ...
Values match
Enter 8 character salt:12345678
Enter iteration count as a number (Eg: 44):25

Enter Keystore Alias:vault
Vault is initialized and ready for use
Handshake with Vault complete
Please enter a Digit::   0: Store a password  1: Check whether password
exists  2: Exit
0
Task:  Store a password
Please enter attribute value: sa
Please enter attribute value again: sa
Values match
Enter Vault Block:DS
Enter Attribute Name:thePass
Secured attribute value has been stored in vault.

Please make note of the following:
********************************************
Vault Block:DS
Attribute Name:thePass
Configuration should be done as follows:
VAULT::DS::thePass::1
********************************************

Please enter a Digit::   0: Store a password  1: Check whether password
exists  2: Exit
2
```

La chaîne qui sera ajoutée au code Java est la dernière valeur de sortie, la ligne commençant par **VAULT**.

Le servlet suivant utilise la chaîne voûtée au lieu d'un mot de passe de texte clair. La version en texte clair est commentée afin que vous puissiez voir la différence.

Exemple 7.3. Servlet qui utilise un mot de passe d'archivage sécurisé.

```
package vaulterror.web;

import java.io.IOException;
import java.io.Writer;
```

```
import javax.annotation.Resource;
import javax.annotation.sql.DataSourceDefinition;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.sql.DataSource;


/*@DataSourceDefinition(
        name = "java:jboss/datasources/LoginDS",
        user = "sa",
        password = "sa",
        className = "org.h2.jdbcx.JdbcDataSource",
        url = "jdbc:h2:tcp://localhost/mem:test"
)*/
@DataSourceDefinition(
        name = "java:jboss/datasources/LoginDS",
        user = "sa",
        password = "VAULT::DS::thePass::1",
        className = "org.h2.jdbcx.JdbcDataSource",
        url = "jdbc:h2:tcp://localhost/mem:test"
)
@WebServlet(name = "MyTestServlet", urlPatterns = { "/my/" },
loadOnStartup = 1)
public class MyTestServlet  extends HttpServlet {

    private static final long serialVersionUID = 1L;


    @Resource(lookup = "java:jboss/datasources/LoginDS")
    private DataSource ds;

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse
resp) throws ServletException, IOException {
        Writer writer = resp.getWriter();
        writer.write((ds != null) + "");
    }
}
```

Votre servlet est maintenant capable de résoudre le la chaîne d'archivage sécurité.

Rapporter un bogue

CHAPITRE 8. WEB, CONNECTEURS HTTP, ET HTTP CLUSTERING

8.1. CONFIGURER UN NŒUD DE WORKER DE MOD_CLUSTER

Résumé

Un worker node en cluster_mode est un serveur d'applications qui participe à l'équilibrage des charges dans un cluster. Les requêtes utilisateurs sont reçues par un serveur web, qui redirige ces requêtes vers un pool de noeuds de workers mod_cluster. Un noeud de workers peut faire partie d'un groupe de serveurs d'un domaine géré, ou d'un serveur autonome. Pour obtenir des informations sur l'équilibrage des charges, voir *Overview of HTTP Connectors* dans le guide *Administration and Configuration Guide*.

Le serveur web d'équilibrage des charges est configuré par l'intermédiaire du sous-système de **`mod_cluster`**. Pour configurer le sous-système **`mod_cluster`**, reportez-vous à *Configure the mod_cluster Subsystem* dans *Administration and Configuration Guide*.

Les noeuds de worker d'un domaine géré partage une configuration identique à travers un groupe de serveurs. Les nœuds de workers exécutant comme serveurs autonomes sont configurés individuellement. Les étapes de configuration, sinon, sont identiques.

Configuration d'un nœud de worker

- Un serveur autonome devra démarrer avec le profil **`standalone-ha`** ou **`standalone-full-ha`**.
- Un groupe de serveurs de domaine géré devra utiliser le profil **`ha`** ou **`full-ha`**, et le groupe de liaisons de sockets **`ha-sockets`** ou **`full-ha-sockets`**. JBoss EAP 6 est fourni avec un groupe de serveurs à clusterisation activée, nommé **`other-server-group`** qui remplit ces prérequis.

NOTE

Quand vous avez des commandes de Management CLI, celles-ci présument que vous utilisez un domaine géré. Si vous utilisez un serveur autonome, supprimez la portion **`/profile=full-ha`** des commandes.

Procédure 8.1. Configurez un nœud de worker

1. **Configurez les interfaces de réseau**
   Les interfaces de réseau ont toutes la valeur **`127.0.0.1`** par défaut. Chaque hôte physique, qui accueille un serveur autonome ou bien un ou plusieurs serveurs au sein d'un groupe de serveurs, a besoin d'interfaces configurées pour utiliser son adresse IP, que les autres serveurs peuvent apercevoir.

   Pour changer l'adresse IP d'un hôte de JBoss EAP 6, vous devrez le fermer et modifier son fichier de configuration directement. C'est parce que l'API de gestion qui actionne la console de gestion et le CLI se fie à une adresse de gestion stable.

   Suivez ces étapes pour changer l'adresse IP sur chaque serveur de votre cluster par votre adresse IP publique de master.

   a. Démarrez le serveur JBoss EAP en utilisant le profil décrit plus haut dans cette section.

b. Lancez la Management CLI, avec la commande ***EAP_HOME*/bin/jboss-cli.sh** dans Linux ou bien la commande ***EAP_HOME*\bin\jboss-cli.bat** dans le serveur Microsoft Windows. Saisissez la commande **connect** pour connecter le contrôleur de domaine sur l'hôte local, ou **connect *IP_ADDRESS*** pour vous connecter à un contrôleur de domaines sur un serveur éloigné.

c. Modifier l'adresse IP externe des interfaces **management**, **public** et **unsecure** en saisissant les commandes suivantes. Veillez bien à remplacer **EXTERNAL_IP_ADDRESS** qui se trouve dans la commande, par l'adresse IP externe de l'hôte.

```
/interface=management:write-attribute(name=inet-
address,value="${jboss.bind.address.management:EXTERNAL_IP_ADDRES
S}"
/interface=public:write-attribute(name=inet-
address,value="${jboss.bind.address.public:EXTERNAL_IP_ADDRESS}"
/interface=unsecure:write-attribute(name=inet-
address,value="${jboss.bind.address.unsecure:EXTERNAL_IP_ADDRESS}
"
:reload
```

Vous devriez voir le résultat suivant pour chaque commande.

```
 "outcome" => "success"
```

d. Pour les hôtes qui participent à un domaine géré, mais qui ne sont pas master, vous devrez modifiez le nom d'hôte du **master** par un nom unique. Ce nom devra être unique (parmi les noms d'esclave) et sera utilisé par l'esclave pour s'authentifier au cluster, donc notez bien le nom que vous utilisez.

i. Démarrer l'hôte esclave JBoss EAP à l'aide de la syntaxe suivante :

```
bin/domain.sh --host-config=HOST_SLAVE_XML_FILE_NAME
```

Par exemple :

```
bin/domain.sh --host-config=host-slave01.xml
```

ii. Lancer l'interface CLI.

iii. Utiliser la syntaxe suivante pour remplacer le nom d'hôte :

```
/host=master:write-
attribute(name="name",value=UNIQUE_HOST_SLAVE_NAME)
```

Par exemple :

```
/host=master:write-attribute(name="name",value="host-slave01")
```

Vous devriez voir apparaître le résultat suivant.

```
 "outcome" => "success"
```

Cela modifie le XML du fichier **host-slave01.xml** comme suit :

```
<host name="host-slave01" xmlns="urn:jboss:domain:1.6">
```

e. Pour les hôtes nouvellement configurés qui ont besoin de rejoindre un domaine géré, cherchez l'élément **`local`** et ajoutez l'attribut **`host`** de l'élement **`remote`** qui pointe en direction du contrôleur de domaine. Cette étape ne s'applique pas à un serveur autonome.

i. Démarrer l'hôte esclave JBoss EAP à l'aide de la syntaxe suivante :

```
bin/domain.sh --host-config=HOST_SLAVE_XML_FILE_NAME
```

Par exemple :

```
bin/domain.sh --host-config=host-slave01.xml
```

ii. Lancer l'interface CLI.

iii. Utilisez la syntaxe suivante en spécifiant le contrôleur de domaine :

```
/host=UNIQUE_HOST_SLAVE_NAME/:write-remote-domain-
controller(host=DOMAIN_CONTROLLER_IP_ADDRESS,port=${jboss.doma
in.master.port:9999},security-realm="ManagementRealm")
```

Par exemple :

```
/host=host-slave01/:write-remote-domain-
controller(host="192.168.1.200",port=${jboss.domain.master.por
t:9999},security-realm="ManagementRealm")
```

Vous devriez voir apparaître le résultat suivant.

```
 "outcome" => "success"
```

Cela modifie le XML du fichier **`host-slave01.xml`** comme suit :

```
<domain-controller>
    <remote host="192.168.1.200"
port="${jboss.domain.master.port:9999}" security-
realm="ManagementRealm"/>
</domain-controller>
```

2. **Configurez l'authentification pour chaque serveur esclave.**
Chaque serveur esclave a besoin d'un nom d'utilisateur et d'un mot de passe créé dans le **`ManagementRealm`** du contrôleur de domaine ou du master autonome. Sur le contrôleur de domaine ou sur le master autonome, exécutez la commande ***`EAP_HOME`*`/bin/add-user.sh`**. Ajoutez un utilisateur avec le même nom d'utilisateur comme esclave au **`ManagementRealm`**. Quand on vous demandera si cet utilisateur doit s'authentifier auprès d'une instance de JBoss AS externe, répondez **`Oui`**. Vous trouverez un exemple de l'entrée et de la sortie de la commande ci-dessous, pour un esclave appelé **`slave1`**, et un mot de passe **`changeme`**.

```
user:bin user$ ./add-user.sh

What type of user do you wish to add?
 a) Management User (mgmt-users.properties)
```

```
 b) Application User (application-users.properties)
(a): a

Enter the details of the new user to add.
Realm (ManagementRealm) :
Username : slave1
Password : changeme
Re-enter Password : changeme
About to add user 'slave1' for realm 'ManagementRealm'
Is this correct yes/no? yes
Added user 'slave1' to file '/home/user/jboss-eap-
6.0/standalone/configuration/mgmt-users.properties'
Added user 'slave1' to file '/home/user/jboss-eap-
6.0/domain/configuration/mgmt-users.properties'
Is this new user going to be used for one AS process to connect to
another AS process e.g. slave domain controller?
yes/no? yes
To represent the user add the following to the server-identities
definition <secret value="Y2hhbmdlbWU=" />
```

3. **Copiez l'élément codé-Base64 `<secret>` à partir de la sortie `add-user.sh`.**
   Si vous prévoyez de spécifier un mot de passe codé Base64 pour l'authentification, copiez l'élément **`<secret>`** à partir de la dernière ligne de la sortie **`add-user.sh`** car vous en aurez besoin à l'étape suivante.

4. **Modifiez le domaine de sécurité de l'hôte esclave pour la nouvelle authentification.**
   Vous pourrez spécifier la valeur secrète d'une des manières suivantes :

   - **Spécifier le mot de passe codé-Base64 dans le fichier de configuration du serveur par le CLI.**

     a. Lancez la Management CLI, avec la commande ***EAP_HOME*`/bin/jboss-cli.sh`** dans Linux ou bien la commande ***EAP_HOME*`\bin\jboss-cli.bat`** dans le serveur Microsoft Windows. Saisissez la commande **`connect`** pour connecter le contrôleur de domaine sur l'hôte local, ou **`connect`** ***IP_ADDRESS*** pour vous connecter à un contrôleur de domaines sur un serveur éloigné.

     b. Spécifiez la valeur secrète en saisissant la commande suivante. Veillez bien à remplacer **`SECRET_VALUE`** par la valeur secrète retournée de la sortie **`add-user.sh`** lors de l'étape précédente.

     ```
     /core-service=management/security-
     realm=ManagementRealm/server-
     identity=secret:add(value="SECRET_VALUE")
     :reload
     ```

     Vous devriez voir le résultat suivant pour chaque commande.

     ```
      "outcome" => "success"
     ```

   - **Configurez l'hôte pour obtenir un mot de passe de l'archivage sécurisé.**

     a. Utilisez le script **`vault.sh`** pour générer un mot de passe masqué. Cela génèrera une chaîne comme la suivante :
     `VAULT::secret::password::ODVmYmJjNGMtZDU2ZC00YmNlLWE4ODMtZjQ1NWNm`

NDU4ZDc1TElORV9CUkVBS3ZhdWx0.

Vous pourrez trouver plus d'informations sur l'archivage sécurisé dans Password Vaults de la section Sensitive Strings de ce guide, ici : Section 7.1, « Système d'archivage sécurisé de mots de passe ».

b. Lancez la Management CLI, avec la commande ***EAP_HOME*/bin/jboss-cli.sh** dans Linux ou bien la commande ***EAP_HOME*\bin\jboss-cli.bat** dans le serveur Microsoft Windows. Saisissez la commande **connect** pour connecter le contrôleur de domaine sur l'hôte local, ou **connect *IP_ADDRESS*** pour vous connecter à un contrôleur de domaines sur un serveur éloigné.

c. Spécifiez la valeur secrète en saisissant la commande suivante. Veillez bien à remplacer **SECRET_VALUE** par le mot de passe masqué généré lors de l'étape précédente.

```
/core-service=management/security-
realm=ManagementRealm/server-
identity=secret:add(value="${VAULT::secret::password::SECRET_V
ALUE}")
:reload
```

Vous devriez voir le résultat suivant pour chaque commande.

```
 "outcome" => "success"
```

NOTE

Quand vous créez un mot de passe dans l'archivage sécurisé, celui-ci devra être spécifié en texte brut, et non pas codé Base64.

- **Spécifiez le mot de passe en tant que propriété système.**
Les exemples suivants utilisent **server.identity.password** comme nom de propriété de système pour le mot de passe.

a. Spécifiez la propriété système pour le mot de passe dans le fichier de configuration du serveur par le CLI.

i. Lancez la Management CLI, avec la commande ***EAP_HOME*/bin/jboss-cli.sh** dans Linux ou bien la commande ***EAP_HOME*\bin\jboss-cli.bat** dans le serveur Microsoft Windows. Saisissez la commande **connect** pour connecter le contrôleur de domaine sur l'hôte local, ou **connect *IP_ADDRESS*** pour vous connecter à un contrôleur de domaines sur un serveur éloigné.

ii. Saisissez la commande suivante pour configurer l'identité secrète pour utiliser la propriété système.

```
/core-service=management/security-
realm=ManagementRealm/server-
identity=secret:add(value="${server.identity.password}")
:reload
```

Vous devriez voir le résultat suivant pour chaque commande.

```
"outcome" => "success"
```

b. Quand vous spécifiez le mot de passe en tant que propriété système, vous pouvez configurer l'hôte d'une des manières suivantes :

- Démarrez le serveur en saisissant le mot de passe en texte brut comme argument de ligne de commande, comme par exemple :

```
-Dserver.identity.password=changeme
```

NOTE

Le mot de passe doit être saisi en texte brut et sera visible par quiconque lance la commande **`ps -ef`**.

- Mettez le mot de passe dans un fichier de propriétés et passez l'URL du fichier de propriétés sous forme d'argument de ligne de commande.

i. Ajoutez la paire clé/valeur à un fichier de propriétés. Par exemple :

```
server.identity.password=changeme
```

ii. Démarrez le serveur par les arguments de ligne de commande

```
--properties=URL_TO_PROPERTIES_FILE
```

.

5. **Redémarrez le serveur.**
L'esclave va maintenant authentifier le master en utilisant son nom d'hôte comme nom d'utilisateur et le string codifié comme mot de passe.

Résultat

Votre serveur autonome, ou les serveurs au sein d'un groupe de serveurs d'un domaine géré, sont désormais configurés en tant que od_cluster Worker Node. Si vous déployez une application en cluster, ses sessions sont répliquées sur tous les nœuds de cluster de basculement, et il peut accepter les demandes provenant d'un serveur Web externe ou d'un équilibreur de charge. Chaque nœud du cluster détecte les autres nœuds à l'aide de découverte automatique, par défaut. Pour configurer la détection automatique et les autres paramètres spécifiques du sous-système **`mod_cluster`**, reportez-vous à *Configure the mod_cluster Subsystem* dans le guide *Administration and Configuration Guide*. Pour configurer le serveur Apache HTTP, reportez-vous à *Use an External Web as the Web Front-end for JBoss EAP 6 Applications* dans *Administration and Configuration Guide*.

Rapporter un bogue

CHAPITRE 9. INSTALLATION DE CORRECTIF

9.1. CORRECTIFS ET MISES À JOUR

Les correctifs de JBoss EAP 6 appliquent des mises à jour qui sont rendues disponibles dans une version 'mineure' de JBoss EAP 6, comme par exemple dans JBoss EAP 6.2. Les correctifs peuvent contenir des mises à jour ponctuelles ou des mises à jour cumulatives.

La mise à niveau est le processus de déplacement vers une nouvelle version majeure (par exemple, de 5.0 à 6.0) ou une nouvelle version mineure (par exemple, de 6.1 à 6.2) et ne peut se faire par un correctif uniquement.

Rapporter un bogue

9.2. MÉCANISMES DE CORRECTION

Les correctifs de JBoss sont distribués sous deux formes : zip (pour tous les produits) et RPM (pour un sous-groupe de produits).

IMPORTANT

Une installation de produits JBoss doit toujours être mise à jour par la méthode: soit Zip, soit RPM. Seuls les correctifs de sécurité et les correctifs cumulatifs sont disponibles via RPM, et les clients qui utilisent une installation RPM ne seront pas en mesure d'effectuer les mises à jour par la méthode Zip.

Les correctifs de JBoss peuvent être une mise à jour asynchrône ou plannifiée :

- Mises à jour asynchrônes : correctifs ponctuels publiés en dehors d'un cycle de mise à jour normal du produit existant. Ces mises à jour peuvent inclure des correctifs de sécurité, ainsi que d'autres correctifs ponctuels fournis par GSS (Red Hat Global Support Services) pour résoudre ces problèmes particuliers.

- Mises à jour prévues : comprennent les correctifs cumulatifs, ainsi que les mises à niveau mineures ou majeures d'un produit existant. Les correctifs cumulatifs incluent toutes les mises à jour déjà développées pour cette version du produit.

La décision de savoir si un patch doit être divulgué ou non dans le cadre d'une mise à jour planifiée ou d'une mise à jour asynchrone dépend de la sévérité du problème que l'on tente de régler. Les problèmes de faible impact sont généralement différés, et sont résolus dans la prochaine version mineure des produits concernés. Les problème dont l'impact est modéré ou supérieur sont généralement traités par ordre d'importance sous forme de mise à jour de produit lors d'une sortie asynchrone et sont résolus par un correctif au problème particulier.

Les mises à jour de sécurité pour les produits de JBoss sont fournis par un erratum (pour les zip et les méthodes RPM). L'erratum comprend une liste de défauts résolus, leurs indices de gravité, les produits concernés, une description textuelle des défauts et une référence de correctifs. Les mises à jour de correction de bogues sont annoncées par un erratum.

IMPORTANT

Il est important de noter qu'une fois qu'un correctif a été appliqué, les jars sont collectés en cours d'exécution à partir du répertoire ***EAP_HOME*/modules/system/layers/base/.overlays/$PATCH_ID/$MODULE**. Les fichiers d'origine sont laissés en ***EAP_HOME*/modules/system/layers/base/.overlays/$PATCH_ID/$MODULE**. Le mécanisme de correction paralyse les fichiers jar d'origine pour des raisons de sécurité. Cela signifie que si vous appliquez un correctif qui met à jour un module, les fichiers jar du module d'origine seront modifiés afin de devenir inutilisables. Si le correctif est renversé, les fichiers d'origine reviennent dans à un état utilisable. Il faut également que la procédure appropriée soit utilisée pour renverser l'action de tout correctif appliqué. Voir Section 9.4.3, « Annulation d'un correctif sous forme zip par le système de gestion des correctifs » pour la procédure de rollback qui convient.

Pour savoir comment Red Hat évalue les erreurs dans JBoss, consultez : Section 9.6, « Évaluation de la gravité et de l'impact des correctifs JBoss Security »

Red Hat maintient une liste de distribution pour notifier les abonnés à propos des défauts liés à la sécurité. Voir Section 9.3, « Abonnez-vous aux listes de diffusion de correctifs (Patch Mailing Lists) »

Rapporter un bogue

9.3. ABONNEZ-VOUS AUX LISTES DE DIFFUSION DE CORRECTIFS (PATCH MAILING LISTS)

Résumé

L'équipe JBoss de Red Hat maintient une liste de diffusion pour les annonces de sécurité pour les produits Red Hat JBoss Middleware. Cette section couvre ce que vous devez faire pour vous abonner à cette liste.

Pré-requis

- Aucun

Procédure 9.1. Abonnez-vous à JBoss Watch List

1. Cliquez sur le lien suivant pour vous rendre sur la page de la liste de diffusion JBoss Watch : JBoss Watch Mailing List.
2. Saisissez votre adresse email dans la section **`Subscribing to Jboss-watch-list`**.
3. [Vous pourrez aussi saisir votre nom et sélectionner un mot de passe. En option, mais recommandé.]
4. Cliquez sur le bouton **`Subscribe`** (s'abonner) pour démarrer le processus d'abonnement.
5. Vous pouvez naviguer les archives de la liste de diffusion en vous rendant à : JBoss Watch Mailing List Archives.

Résultat

Après la confirmation de votre adresse email, vous serez abonné aux communiqués de sécurité sur la liste de diffusion des correctifs de JBoss.

Rapporter un bogue

9.4. INSTALLER DES CORRECTIFS EN ZIP

9.4.1. Le Patch Management System (système de gestion des correctifs)

Le système de gestion de correctifs de JBoss EAP 6 est utilisé pour appliquer les correctifs zip téléchargés dans un serveur JBoss EAP 6 individuel. Il est accessible via la console de gestion ou par le biais de l'interface CLI à l'aide de la commande **`patch`**. Le système de gestion de correctifs ne permet pas de corriger automatiquement les instances du serveur JBoss EAP 6 sur un domaine géré, mais les instances de serveur individuel d'un domaine géré peuvent être corrigées indépendamment.

IMPORTANT

Les instances de serveur JBoss EAP 6 qui ont été installées par la méthode RPM ne peuvent pas être mises à jour par le système de gestion des correctifs. Consultez Section 9.5, « Correctifs d'installation RPM » pour mettre à jour les serveurs JBoss EAP 6 installés via RPM.

NOTE

Le système de gestion des correctifs ne peut être utilisé qu'avec les correctifs produits pour les versions JBoss EAP 6.2 ou versions supérieures. Pour les correctifs des versions JBoss EAP antérieures à 6.2, vous devrez, à la place, consulter la documentation de la version concernée à l'adresse suivante https://access.redhat.com/site/documentation/.

En plus d'appliquer des correctifs, le système de gestion des correctifs peut vous donner des informations de base sur l'état des correctifs installés, et peut aussi vous fournir une façon de retirer immédiatement une application de correctif.

Quand vous appliquez ou annulez un correctif, le système de gestion des correctifs vérifiera les modules et autres fichiers divers qu'il est en train de modifier pour vérifier tout changement de la part de l'utilisateur. Si un changement utilisateur est détecté, et qu'un commutateur de gestion des conflits n'a pas été spécifié, le système de gestion des correctifs va abandonner l'opération et avertir qu'il y a un conflit. L'avertissement comprendra une liste des modules et des autres fichiers qui sont en conflit. Pour terminer, l'opération devra être ré-exécutée avec une option (commutateur) spécifiant comment résoudre le conflit : soit pour conserver les modifications utilisateur, ou pour les substituer.

Le tableau ci-dessous énumère tous les arguments et toutes les options de la commande d'interface CLI **`patch`**.

Tableau 9.1. Arguments et Options de commande `patch`

Argument ou Option	Description
`apply`	Applique un correctif
`--override-all`	S'il y a un conflit, l'opération de correctif va remplacer toutes les modifications utilisateur.

Argument ou Option	Description
--override-modules	Si le conflit résulte de la modification d'un module, cette option remplacera ces modifications par le contenu de l'opération de correctif.
--override=*path*(,*path*)	Pour les divers fichiers spécifiés uniquement, cela remplacera les fichiers modifiés en conflit par les fichiers de l'opération de correctif.
--preserve=*path*(,*path*)	Pour les divers fichiers spécifiés uniquement, cela préservera les fichiers modifiés en conflit.
--host=*HOST_NAME*	Disponible en mode de domaine, indique l'hôte sur lequel l'opération de correction aura lieu.
info	Renvoie l'information sur les correctifs actuellement installés.
history	Renvoie l'information sur l'historique des correctifs.
rollback	Opération de restauration suite à l'application d'un correctif.
--patch-id=*PATCH_ID*	Requis pour la suppression, l'ID du correctif à restaurer.
--reset-configuration=*TRUE*|*FALSE*	Dans le cadre d'une restauration, cela indique si on doit restaurer les fichiers de configuration du serveur.
--rollback-to	Si le correctif à supprimer est un correctif isolé (une seule fois), en utilisant cet argument, vous spécifiez que l'opération de restauration s'appliquera à tous les autres correctifs isolés appliqués au dessus du correctif spécifié.

Rapporter un bogue

9.4.2. Installation des correctifs sous forme zip par le Patch Management System (système de gestion des correctifs)

Résumé

Les correctifs qui sont en format zip peuvent être installés en utilisant le système de gestion de JBoss EAP 6 patch via l'interface de gestion CLI ou la console de gestion.

IMPORTANT

Le systéme de gestion des correctifs a été ajouté dans la version JBoss EAP 6.2. Pour les versions JBoss EAP antérieures à 6.2, le processus de mise en place de correctifs sous la forme zip est différent, et vous devrez, à la place, consulter la documentation de la version concernée à l'adresse suivante https://access.redhat.com/site/documentation/.

Pré-requis

- Accès valide et abonnement au portail clients de Red Hat.
- Un abonnement en cours à un produit JBoss installé en format zip.
- Accédez à l'interface CLI ou à la console de gestion pour mettre à jour le serveur de JBoss EAP 6. Voir la section *Launch the Management CLI* ou *Log in to the Management Console* du guide *Administration and Configuration Guide*.

AVERTISSEMENT

Avant d'installer un correctif, vous devez sauvegarder votre produit JBoss, ainsi que tous les fichiers de configuration personnalisés.

Procédure 9.2. Appliquez un correctif zip à une instance de serveur JBoss EAP 6 par l'interface de gestion CLI

1. Téléchargez le fichier zip de correctif à partir du portail clients à partir de https://access.redhat.com/downloads/
2. À l'aide du CLI, appliquez le correctif par la commande suivante et le chemin qui convient pour le fichier de correction :

```
[standalone@localhost:9999 /] patch apply /path/to/downloaded-patch.zip
```

   L'outil **patch** avertira s'il y a conflit de tentative d'application de correctif. Consultez Section 9.4.1, « Le Patch Management System (système de gestion des correctifs) » pour voir les options d'exécution à ajouter quand vous lancerez la commande **patch** à nouveau pour résoudre les conflits.

3. Démarrez à nouveau le serveur JBoss EAP 6 pour que le correctif puisse prendre effet :

```
[standalone@localhost:9999 /] shutdown --restart=true
```

Procédure 9.3. Appliquez un correctif zip à une instance de serveur JBoss EAP 6 par le biais de la console de gestion

1. Téléchargez le fichier zip de correctif à partir du portail clients à partir de https://access.redhat.com/downloads/

2. Dans la console de gestion :
    - Pour un serveur autonome : cliquez sur l'onglet **Runtime** (exécution) en haut de l'écran, puis cliquez sur **Patch Management** (gestion des correctifs).
    - Pour un domaine géré : cliquez sur l'onglet **Domain** en haut de l'écran, puis cliquez sur le menu déroulant **Host** (hôte), puis cliquez sur **Patch Management** (gestion des correctifs).
3. Cliquez sur **Apply a New Patch** (appliquer un nouveau correctif).
    a. Si vous corrigez un hôte de domaine géré, sélectionnez si vous devez arrêter des serveurs sur l'hôte sur le prochain écran, puis cliquez sur **Next** (suite).
4. Cliquez sur le bouton de navigation **Browse**, puis sélectionnez le correctif téléchargé que vous souhaitez appliquer, puis cliquez sur **Next**.
    a. S'il existe des conflits lors du processus d'application du correctif, un message d'avertissement s'affichera. Cliquez sur **View error details** pour voir le détail des conflits. S'il y a un conflit, vous pouvez annuler l'opération, ou bien, activez la case à cocher **Override all conflicts** (supprimer tous les conflits) et cliquez sur le bouton **Next**. La suppression des conflits se traduira par le contenu du patch prévalant sur toute modification de l'utilisateur.
5. Une fois que le correctif aura été appliqué, sélectionnez s'il faut redémarrer le serveur JBoss EAP 6 dès maintenant pour que le correctif prenne effet, puis cliquez sur **Finish**.

Résultat

L'instance de serveur JBoss EAP 6 est corrigée avec la dernière mise à jour.

Rapporter un bogue

9.4.3. Annulation d'un correctif sous forme zip par le système de gestion des correctifs

Résumé

Le système de gestion de correctifs de JBoss EAP 6 permet d'annuler l'application d'un correctif zip précédemment appliqué, via l'interface CLI ou par la console de gestion.

AVERTISSEMENT

Il n'est pas prévu de pouvoir annuler l'application d'un correctif à l'aide du système de gestion des correctifs, comme une fonctionnalité de désinstallation générale. Il est uniquement destiné à être utilisé immédiatement après l'application d'un correctif qui a eu des conséquences indésirables.

IMPORTANT

Le systéme de gestion des correctifs a été ajouté dans la version JBoss EAP 6.2. Pour les versions JBoss EAP antérieures à 6.2, le processus de suppression de correctifs sous la forme zip est différent, et vous devrez, à la place, consulter la documentation de la version concernée à l'adresse suivante https://access.redhat.com/site/documentation/.

Pré-requis

- Un correctif ayant déjà été appliqué par le système de gestion des correctifs de JBoss EAP 6.
- Accédez à l'interface CLI ou à la console de gestion pour le serveur de JBoss EAP 6. Voir la section *Launch the Management CLI* ou *Log in to the Management Console* du guide *Administration and Configuration Guide*.

AVERTISSEMENT

Quand vous suivez cette procédure, faîtes attention quand vous spécifiez la valeur de l'option **`Reset Configuration`**.

Si défini à **TRUE**, en retirant le correctif, vous restaurerez également les fichiers de configuration du serveur JBoss EAP 6 à leur état d'avant correctif. Tout changement qui aura lieu sur les fichiers de configuration du serveur JBoss EAP 6 à la suite de ce correctif sera perdu.

Si défini à **FALSE**, les fichiers de configuration ne seront pas supprimés. Dans un tel cas, il est possible que le serveur ne démarre pas à nouveau après l'opération de restauration, car il y a pu avoir des altérations de configuration comme des altérations d'espace-nom, qui ne seront plus valides et qui devront être réparées manuellement.

Procédure 9.4. Supprimer un correctif zip à une instance de serveur JBoss EAP 6 par l'interface de gestion CLI

1. Avec l'interface CLI, appliquez la commande **`patch info`** pour trouver l'ID du patch à retirer.
   - Avec les correctifs cumulatifs, l'ID du correctif correspond à la première valeur **`cumulative-patch-id`** qui apparaît dans la sortie **`patch info`**.
   - Les ID de correctifs de bogues ou de sécurité spontanés sont listés comme valeur des premiers **`patches`** (correctifs) qui apparaissent dans la sortie **`patch info`**, avec le correctif spontané listé en premier.
2. Avec l'interface CLI, retirez le correctif avec l'ID de correctif de l'étape précédente.

```
[standalone@localhost:9999 /] patch rollback --patch-id=PATCH_ID --
reset-configuration=TRUE
```

L'outil **`patch`** avertira s'il y a conflit pour la tentative de suppression de correctif. Consultez Section 9.4.1, « Le Patch Management System (système de gestion des correctifs) » pour voir

les options disponibles d'exécution à nouveau de la commande **patch** pour résoudre les conflits.

3. Démarrez à nouveau le serveur JBoss EAP 6 pour que la suppression de correctif puisse prendre effet :

```
[standalone@localhost:9999 /] shutdown --restart=true
```

Procédure 9.5. Supprimer un correctif zip à une instance de serveur JBoss EAP 6 par la console de gestion.

1. Dans la console de gestion :
   - Pour un serveur autonome : cliquez sur l'onglet **Runtime** (exécution) en haut de l'écran, puis cliquez sur **Patch Management** (gestion des correctifs).
   - Pour un domaine géré : cliquez sur l'onglet **Domain** en haut de l'écran, puis sélectionnez l'hôte qui convient sur le menu déroulant **Host** (hôte), puis cliquez sur **Patch Management** (gestion des correctifs).
2. Dans le tableau **Recent Patch History** (historique récent des correctifs), sélectionnez le correctif que vous souhaitez supprimer, puis cliquez sur **Rollback**.
   a. Pour un hôte de domaine géré, sélectionnez si vous devez arrêter des serveurs sur l'hôte, sur le prochain écran, puis cliquez sur **Next** (suite).
3. Sélectionner vos options pour le processus de suppression, puis cliquez sur **Next**.
4. Confirmez vos options et le correctif à supprimer, puis cliquez sur **Next**.
   a. Si l'option **Override all** n'est pas sélectionnée, et qu'il n'y a pas de conflit lors du processus d'application du correctif, un message d'avertissement s'affichera. Cliquez sur **View error details** pour voir le détail des conflits. S'il y a un conflit, vous pouvez annuler l'opération, ou bien, cliquez sur **Choose Options** et essayez l'opération à nouveau avec la case **Override all** sélectionnée. La suppression des conflits se traduira par l'annulation du correctif prévalant sur toute modification de l'utilisateur.
5. Une fois que le correctif aura été supprimé, sélectionnez s'il faut redémarrer le serveur JBoss EAP 6 dès maintenant pour que les changements prennent effet, puis cliquez sur **Finish**.

Résultat

Le correctif, et parfois aussi les fichiers de configuration du serveur sont retirés de l'instance de serveur de JBoss EAP 6.

Rapporter un bogue

9.5. CORRECTIFS D'INSTALLATION RPM

Résumé

Les correctifs JBoss sont distribués de deux façons: ZIP (pour tous les produits) et RPM (pour un sous ensemble de produits). Cette tâche décrit les étapes d'installation des correctifs en format RPM.

Pré-requis

- Un abonnement RHN valide.
- Un abonnement en cours à un produit JBoss installé via un package RPM.

Procédure 9.6. Appliquer un correctif à un produit JBoss via méthode RPM

Les mises à jour de sécurité pour les produits de JBoss sont fournis par un erratum (pour les zip et les méthodes RPM). L'erratum comprend une liste de problèmes résolus, leur indice de gravité, les produits concernés, une description textuelle des problèmes et une référence aux correctifs.

Pour les distributions RPM de produits JBoss, l'errata inclut des références aux packages RPM mis à jour. Le correctif peut être installé par **yum**.

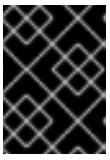
AVERTISSEMENT

Avant d'installer un correctif, vous devrez sauvegarder votre produit JBoss avec tous les fichiers de configuration personnalisés.

1. Recevez vos notifications de correctifs de sécurité, soit en tant qu'abonné de la liste de diffusion JBoss watch, ou bien en parcourant les archives de la liste de diffusion de JBoss watch.
2. Lire l'errata du correctif de sécurité et confirmez qu'elle s'applique bien à un produit JBoss de votre environnement.
3. Si le correctif de sécurité s'applique à un produit JBoss de votre environnement, alors suivez le lien pour télécharger le package RPM mis à jour qui contient l'errata.
4. Utilisation

   ```
   yum update
   ```

   pour installer le correctif.



IMPORTANT

Quand vous mettez une installation RPM à jour, votre produit JBoss sera mis à jour de façon cumulative avec tous les correctifs sortis-RPM.

Résultat

Le produit JBoss est corrigé par la dernière mise à jour en format RPM.

Rapporter un bogue

9.6. ÉVALUATION DE LA GRAVITÉ ET DE L'IMPACT DES CORRECTIFS JBOSS SECURITY

Pour communiquer le risque de chaque faille de sécurité de JBoss, Red Hat utilise une échelle de gravité à quatre niveaux: faible, modéré, important et critique, en plus des scores de base de vulnérabilité CVSS version 2, qui peuvent être utilisés pour identifier l'impact de la faille.

Tableau 9.2. Système d'évaluation de gravité de JBoss Security Patches (correctifs de JBoss Security)

Gravité	Description
Critique	Ce score est donné aux failles pouvant être facilement exploitées par un attaquant non authentifié à distance et conduire à des compromis de système (exécution de code arbitraire) sans intervention de la part de l'utilisateur. Ce sont les types de vulnérabilité pouvant être exploités par des vers informatiques. Il s'agit de failles nécessitant un utilisateur distant authentifié, un utilisateur local ou une configuration peu probable et elles ne sont pas classées comme ayant un impact critique.
Important	Ce score est donné aux failles qui peuvent facilement compromettre la confidentialité, l'intégrité ou la disponibilité des ressources. Ce sont les types de vulnérabilité qui permettent à des utilisateurs locaux d'obtenir des privilèges, qui permettent à des utilisateurs distants non authentifiés d'afficher des ressources qui devraient normalement être protégées par une authentification, qui permettent à des utilisateurs authentifiés distants d'exécuter du code arbitraire, ou d'autoriser des utilisateurs locaux ou distants de causer un déni de service.
Modéré	Ce score est donné aux failles qui risquent d'être plus difficiles à exploiter mais qui peuvent toujours résulter en compromis de confidentialité, d'intégrité, et disponibilité de ressources, sous certaines circonstances. Ce sont les types de vulnérabilité qui auraient pu avoir un impact critique ou un impact important, mais pouvant être moins facilement exploités selon une évaluation technique de la faille, ou qui risquent moins d'affecter les configurations.
Moindre	Ce score est donné à toutes les autres questions qui ont un impact de sécurité. Ce sont les types de vulnérabilité qui semblent liés à des circonstances improbables pour pouvoir être exploitées, ou dans les cas où l'exploitation de la faille puisse avoir des conséquences minimes.

Le composant de l'impact d'un score CVSS v2 repose sur une évaluation combinée des trois impacts potentiels : confidentialité (C), intégrité (I) et disponibilité (A). Chacun d'entre eux peut être évalué comme aucun (N), partiel (P) ou total (C).

Comme le process de serveur JBoss exécute en tant qu'utilisateur non privilégié et est isolé du système d'exploitation hôte, les failles de sécurité de JBoss ne peuvent avoir comme score d'impact que N (non/aucun), ou P (partiel).

Exemple 9.1. Score d'impact CVSS v2

> L'exemple ci-dessous vous montre un score d'impact CVSS v2, quand l'exploitation d'une faille n'a aucun impact sur la confidentialité du système, un impact partiel sur l'intégrité, et un impact total sur la disponibilité du système (c-a-d que le système est rendu indisponible, par exemple, suite à un crash de noyau).
>
> ```
> C:N/I:P/A:C
> ```

Grâce à l'indice de gravité du score et du score CVSS, les organisations peuvent prendre des décisions informées sur le risque que chaque problème pose à leur environnement et à leur programmation de mises à jours respectives.

Pour davantage d'informations sur CVSS2, veuillez consulter : CVSS2 Guide.

Rapporter un bogue

9.7. GÉRER LES MISES À JOUR DE SÉCURITÉ POUR LES DÉPENDANCES GROUPÉES AU SEIN D'APPLICATIONS DÉPLOYÉES DANS JBOSS EAP

Red Hat fournit des correctifs de sécurité pour tous les composants qui font partie de la distribution de JBoss EAP. Cependant, beaucoup d'utilisateurs de JBoss EAP déploient des applications qui regroupent leurs propres dépendances, plutôt que d'utiliser exclusivement des composants fournis dans le cadre de la distribution JBoss EAP. Par exemple, un fichier WAR déployé peut inclure des JAR de dépendance dans le répertoire WEB-INF/lib/. Ces JAR sont hors de portée des correctifs de sécurité fournis par Red Hat. La gestion des mises à jour de sécurité pour les dépendances groupées à l'intérieur des applications déployées sur JBoss EAP est la responsabilité des spécialistes de la maintenance des applications. Les sources de données et outils suivants peuvent contribuer à cet effort et sont fournis sans garantie ni support.

Outils et Sources de données

Listes de diffusion pour le patch JBoss

S'abonner aux listes de diffusion de patch de JBoss vous tiendra informé au sujet des failles de sécurité qui ont été corrigées dans les produits JBoss, et vous permettra de vérifier si vos applications déployées regroupent des versions vulnérables des composants concernés.

Page consultative de sécurité pour les composants regroupés.

De nombreux composants open source ont leur propre page consultative de sécurité. Par exemple, Struts 2 est un composant couramment utilisé pour de nombreux problèmes de sécurité connus, qui n'est pas fourni dans la distribution de JBoss EAP. Le projet Struts 2 tient à jour une page consultative de sécurité en amont, qui doit être surveillée si vos applications déployées sont regroupées avec Struts 2. De nombreux composants fournis commercialement maintiennent également des pages consultatives de sécurité.

Scannez régulièrement votre applications déployées pour les vulnerabilités connues

Il existe plusieurs outils commerciaux disponibles pour ce faire. Il y a également un outil open source appelé Victims, qui est développé par les employés de Red Hat, mais qui est livré sans appui, ni garantie. Victims fournit des plugins pour plusieurs outils de génération et d'intégration, qui analysent automatiquement les applications pour les regroupements connus pour leur vulnérabilité aux dépendances. Il existe des plugins Maven, Ant et Jenkins. Pour plus d'informations sur l'outil Victims, voir https://victi.ms/about.html.

Voir également :

- Section 9.3, « Abonnez-vous aux listes de diffusion de correctifs (Patch Mailing Lists) »

Rapporter un bogue

PARTIE III. DÉVELOPPEMENT D'APPLICATIONS SÉCURISÉES

CHAPITRE 10. APERÇU SÉCURITÉ

10.1. LA SÉCURITÉ DES APPLICATIONS

Pour chaque développeur d'applications, sécuriser vos applications est une tâche importante couvrant des aspects multiples. JBoss EAP 6 vous donne tous les outils dont vous aurez besoin pour rédiger des applications sécurisées, y compris les possibilités suivantes :

- Section 13.2.1, « Authentification »
- Section 13.5.1, « L'autorisation »
- Section 13.7.1, « Security Auditing »
- Section 13.8.1, « Mappage de sécurité »
- Section 10.2, « Sécurité déclarative »
- Section 11.2.2.1, « Permissions de méthodes EJB »
- Section 11.2.3.1, « Les annotations de sécurité EJB »

Voir également Section 13.9, « Utiliser un domaine de sécurité dans votre application ».

Rapporter un bogue

10.2. SÉCURITÉ DÉCLARATIVE

La *sécurité déclarative* est une méthode de séparation des problèmes de sécurité du code d'application, en utilisant le conteneur pour gérer la sécurité. Le conteneur procure un système d'autorisation basée soit sur les permissions de fichiers ou basée utilisateur, groupe ou rôle. Cette approche est normalement supérieure à la sécurité *programmatique*, qui donne à l'application toutes les responsabilités pour la sécurité.

La plate-forme JBoss EAP 6 fournit une sécurité déclarative par les domaines de sécurité.

Rapporter un bogue

10.2.1. Sécurité déclarative Java EE

Le modèle de sécurité Java EE est déclaratif en ce sens que vous décrivez les rôles de sécurité et les autorisations dans un descripteur XML standard aulieu d'encastrer la sécurité dans votre composant d'entreprise. Cela isole la sécurité à partir du code niveau entreprise parce que la sécurité a tendance à être davantage une fonction se rapportant à l'endroit où le composant est déployé plutôt que liée à un aspect inhérent de la logique métier du composant. Par exemple, considérez un guichet automatique bancaire (GAB) devant être utilisé pour accéder à un compte de banque. Les exigences de sécurité, les rôles et les autorisations peuvent varier indépendamment de la façon dont vous accédez au compte en banque, suivant la Banque qui gère le compte, où se trouve l'ATM, et ainsi de suite.

La sécurisation d'une application Java EE repose sur la spécification des besoins de sécurité de l'application via les descripteurs de déploiement Java EE standards. Vous sécuriser l'accès aux EJBs et aux composants web d'une application d'entreprise en utilisant les descripteurs de déploiement **`ejb-jar.xml`** et **`web.xml`**.

Rapporter un bogue

10.2.2. Références de sécurité

Les EJB (Enterprise Java Beans) et les servlets peuvent déclarer un ou plusieurs éléments <security-role-ref>.

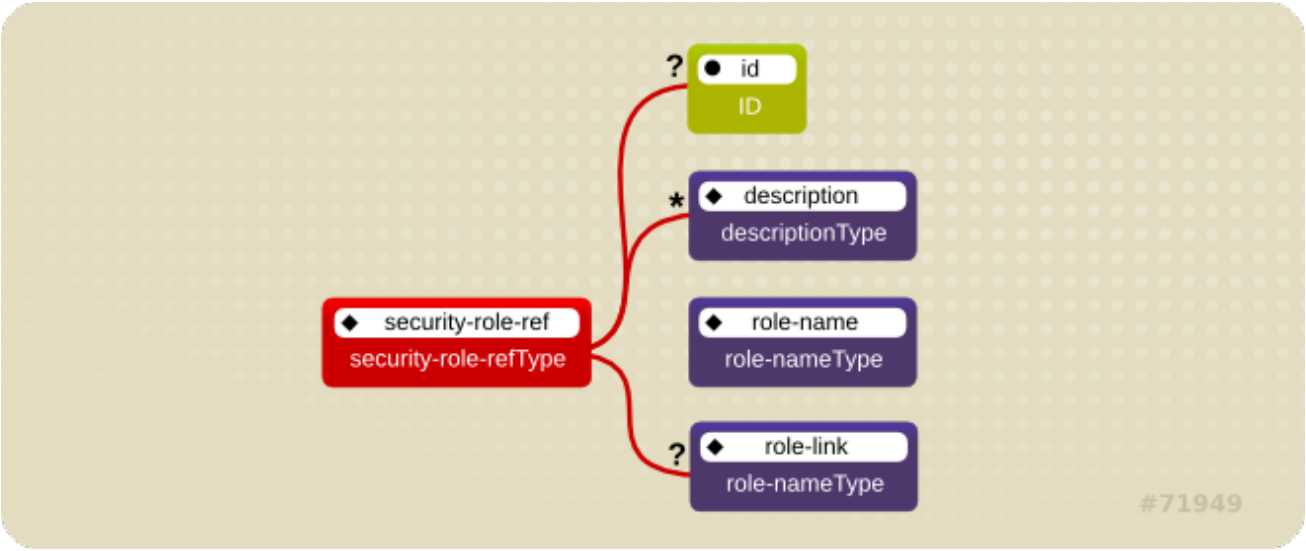
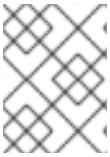


Figure 10.1. Modèle de référence de rôles de sécurité

Cet élément déclare que le composant utilise la valeur de l'attribut **`role-nameType`** de l'élément <role-name> comme argument de la méthode **`isCallerInRole(String)`**. En utilisant la méthode **`isCallerInRole`**, un composant est en mesure de vérifier si l'invocateur est dans un rôle qui a été déclaré avec un élément <security-role-ref> ou <role-name>. La valeur de l'élément <role-name> doit être reliée à un élément <security-role> par l'élément <role-link>. L'utilisation typique de **`isCallerInRole`** est de procéder à un contrôle de sécurité qui ne puisse pas être défini en utilisant les éléments <method-permissions> basé-rôle.

Exemple 10.1. fragment de descripteur ejb-jar.xml

```
<!-- A sample ejb-jar.xml fragment -->
  <ejb-jar>
    <enterprise-beans>
      <session>
        <ejb-name>ASessionBean</ejb-name>
        ...
        <security-role-ref>
          <role-name>TheRoleICheck<role-name>
            <role-link>TheApplicationRole</role-link>
        </security-role-ref>
      </session>
    </enterprise-beans>
  ...
  </ejb-jar>
```



NOTE

Il s'agit d'un exemple uniquement. Dans les déploiements, les éléments de cette section doivent contenir les noms de rôle pertinents au déploiement EJB.

Exemple 10.2. fragment de descripteur web.xml

```
<web-app>
  <servlet>
    <servlet-name>AServlet</servlet-name>
    ...
    <security-role-ref>
      <role-name>TheServletRole</role-name>
      <role-link>TheApplicationRole</role-link>
    </security-role-ref>
  </servlet>
    ...
</web-app>
```

Rapporter un bogue

10.2.3. Identité Sécurité

Un EJB (Enterprise Java Bean) peut spécifier l'identité qu'un autre EJB doit utiliser lorsqu'il appelle des méthodes sur les composants à l'aide de l'élément <security-identity>.

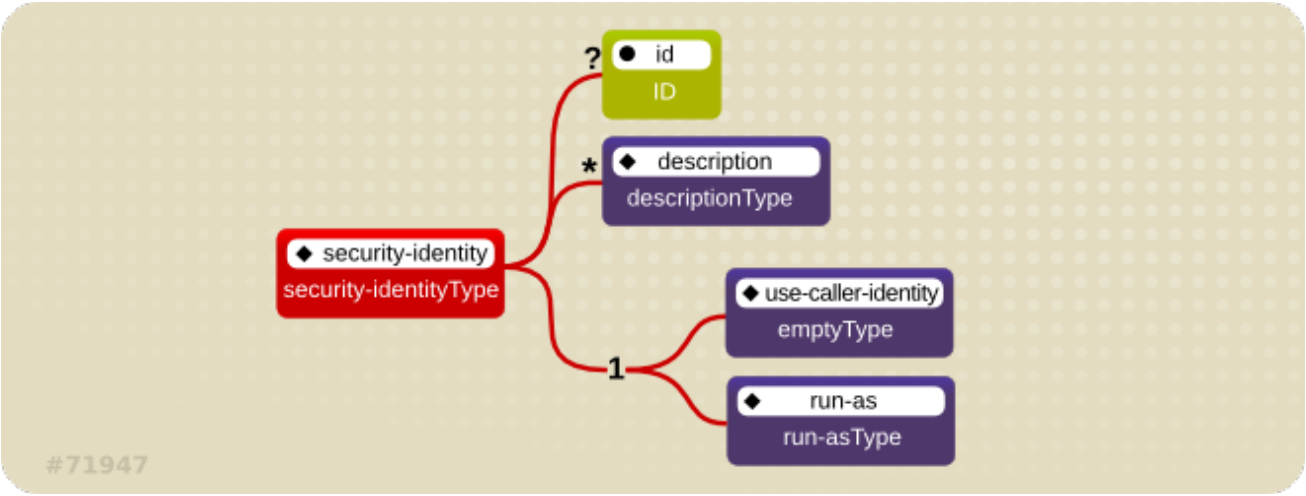


Figure 10.2. Modèle Java EE Security Identity Data Model

L'identité de l'invocation peut être celle de l'appelant en cours ou peut correspondre un rôle spécifique. L'assembleur d'applications utilise l'élément <security-identity> avec un élément enfant de <use-caller-identity>. Cela indique que l'identité de l'appelant en cours doit être propagée comme l'identité de sécurité pour les appels de méthode de l'EJB. La propagation d'identité de l'appelant est la valeur par défaut utilisée en l'absence d'une déclaration d'élément <security-identity> explicite.

Sinon, l'assembleur d'application peut utiliser l'élément enfant <run-as> ou <role-name> pour spécifier qu'un rôle de sécurité spécifique fourni par la valeur de l'élément <role-name> doit servir d'identité de sécurité pour les appels de méthode de l'EJB.

Notez que cela ne change pas l'identité de l'appelant telle qu'elle est perçue par la méthode **`EJBContext.getCallerPrincipal()`**. À la place, les rôles de sécurité de l'appelant sont définis par l'élément <run-as> ou <role-name>.

Un cas d'utilisation de l'élément <run-as> consiste à empêcher les clients externes d'accéder aux beans

EJB internes. Vous pouvez configurer ce comportement en affectant des éléments EJB internes <method-permission> qui limitent l'accès à un rôle jamais attribué à un client externe. Les EJB qui doivent à leur tour utiliser les EJB internes sont ensuite configurés avec un <run-as> ou un <role-name> qui correspond au rôle restreint. Le fragment suivant de descripteur décrit un exemple d'utilisation de l'élément <security-identity>.

```
<ejb-jar>
    <enterprise-beans>
        <session>
            <ejb-name>ASessionBean</ejb-name>
            <!-- ... -->
            <security-identity>
                <use-caller-identity/>
            </security-identity>
        </session>
        <session>
            <ejb-name>RunAsBean</ejb-name>
            <!-- ... -->
            <security-identity>
                <run-as>
                    <description>A private internal role</description>
                    <role-name>InternalRole</role-name>
                </run-as>
            </security-identity>
        </session>
    </enterprise-beans>
    <!-- ... -->
</ejb-jar>
```

Lorsque vous utilisez un <run-as> pour attribuer un rôle spécifique aux appels sortants, un principal nommé **anonymous** est attribué à tous les appels sortants. Si vous voulez un autre principal à associer à l'appel, vous devez associer un <run-as-principal> au bean du fichier **jboss-ejb3.xml**. Le fragment suivant associe un principal nommé **internal** avec **RunAsBean** de l'exemple précédent.

```
<session>
    <ejb-name>RunAsBean</ejb-name>
    <security-identity>
        <run-as-principal>internal</run-as-principal>
    </security-identity>
</session>
```

L'élément <run-as> est également disponible dans les définitions du servlet du fichier**web.xml**. L'exemple suivant montre comment assigner le rôle **InternalRole** à un servlet :

```
  <servlet>
    <servlet-name>AServlet</servlet-name>
    <!-- ... -->
    <run-as>
        <role-name>InternalRole</role-name>
    </run-as>
  </servlet>
```

Les appels de ce servlet sont associés au **principal** «anonymous». L'élément <run-as-principal> se trouve dans le fichier **jboss-web.xml** pour assigner un principal spécifique qui puisse correspondre à un rôle **run-as**. Le fragment suivant montre comment associer un principal nommé **internal** au serveur ci-dessus.

```
<servlet>
  <servlet-name>AServlet</servlet-name>
  <run-as-principal>internal</run-as-principal>
</servlet>
```

Rapporter un bogue

10.2.4. Rôles de sécurité

Le nom de rôle de sécurité référencé soit par l'élément **security-role-ref** ou élément de **security-identity** doit mapper à un des rôles déclarés de l'application. Un assembleur d'applications définit les rôles de sécurité logique en déclarant les éléments **security-role**. La valeur de **role-name** est un nom de rôle d'application logique comme Administrateur, Architecte, ResponsableVentes, etc.

Les spécifications Java EE précisent qu'il est important de garder à l'esprit que les rôles de sécurité du descripteur de déploiement soient utilisés pour définir l'aperçu de la sécurité logique d'une application. Les rôles définis dans les descripteurs de déploiement Java EE ne doivent pas être confondus avec les groupes d'utilisateurs, les utilisateurs, les principaux et d'autres concepts qui existent dans l'environnement opérationnel de l'entreprise cible. Les rôles de descripteur de déploiement sont des constructions d'applications avec des noms de domaine d'application spécifiques. Par exemple, une application bancaire pourrait utiliser des noms de rôles tels que DirecteurBank, Guichetier ou Client.

Dans JBoss EAP, un élément de **security-role** est seulement utilisé pour mapper les valeurs de **security-role-ref/role-name** au rôle logique qui fait référence au rôle du composant. Les rôles assignés à l'utilisateur représentent une fonction dynamique du gestionnaire de sécurité de l'application. JBoss n'a as besoin de la définition des éléments de **security-role** pour déclarer des permissions de méthode. Toutefois, la spécification d'éléments de **security-role** est toujours une pratique conseillée pour assurer la portabilité entre les serveurs d'applications et la maintenance du descripteur de déploiement.

Exemple 10.3. Un fragment de descripteur ejb-jar.xml qui illustre l'utilisation de l'élément security-role.

```
<!-- A sample ejb-jar.xml fragment -->
<ejb-jar>
    <assembly-descriptor>
        <security-role>
            <description>The single application role</description>
            <role-name>TheApplicationRole</role-name>
        </security-role>
    </assembly-descriptor>
</ejb-jar>
```

Exemple 10.4. Un exemple web.xml qui illustre l'utilisation de l'élément security-role.

```
<!-- A sample web.xml fragment -->
<web-app>
  <security-role>
    <description>The single application role</description>
    <role-name>TheApplicationRole</role-name>
  </security-role>
</web-app>
```

Rapporter un bogue

10.2.5. Permissions de méthodes EJB

Un assembleur d'applications peut définir les rôles autorisés à appeler les méthodes d'interfaces d'accueil ou éloignées de l'EJB par le biais de déclarations d'élément method-permission.

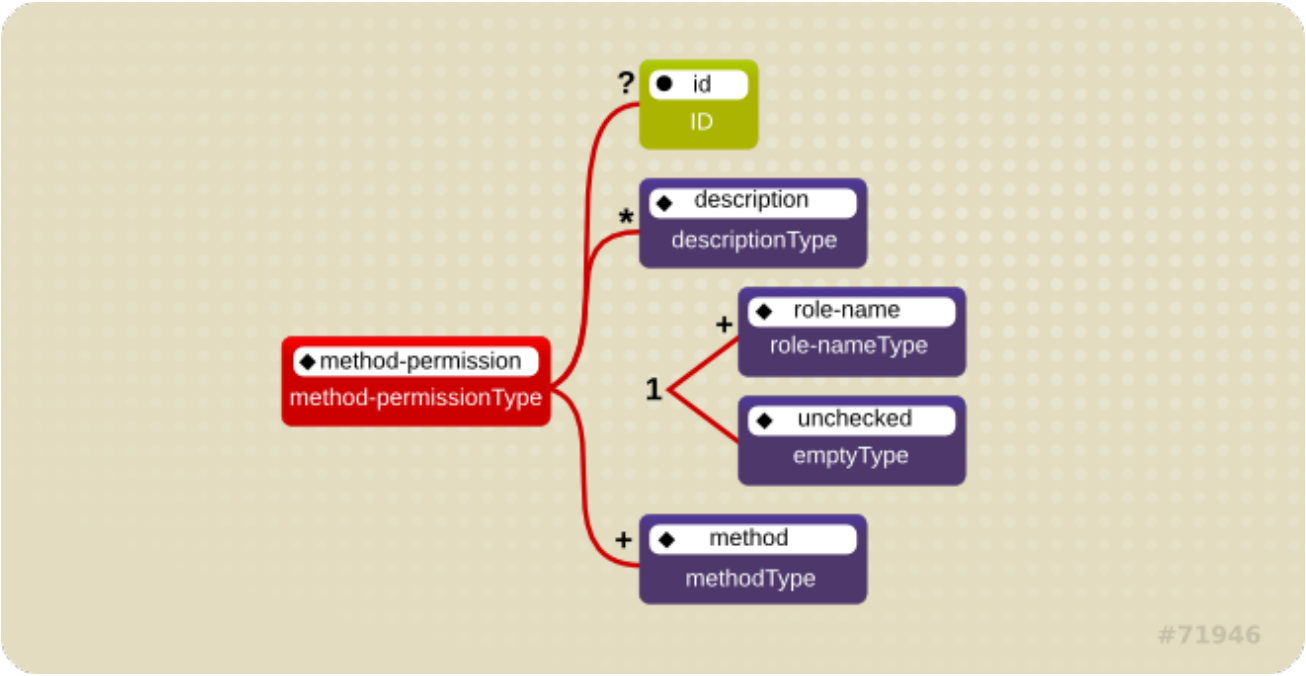


Figure 10.3. Élément de Permission de méthode Java EE

Chaque élément de **`method-permission`** contient un ou plusieurs éléments enfants role-name qui définissent les rôles logiques autorisés à accéder aux méthodes EJB identifiées par les éléments enfants de la méthode. Vous pouvez également spécifier un élément **unchecked** à la place de l'élément **`role-name`** pour déclarer que tout utilisateur authentifié peut accéder aux méthodes identifiées par les éléments enfants de méthode. De plus, vous pouvez déclarer que personne ne devrait avoir accès à une méthode qui comporte l'élément de la **`exclude-list`** (liste d'exclusion). Si un EJB a des méthodes qui n'ont pas été déclarées comme accessibles par un rôle à l'aide d'un élément de **`method-permission`**, la valeur par défaut des méthodes EJB sera exclusion d'utilisation. Cela revient à rendre les méthodes par défaut à **`exclude-list`**.

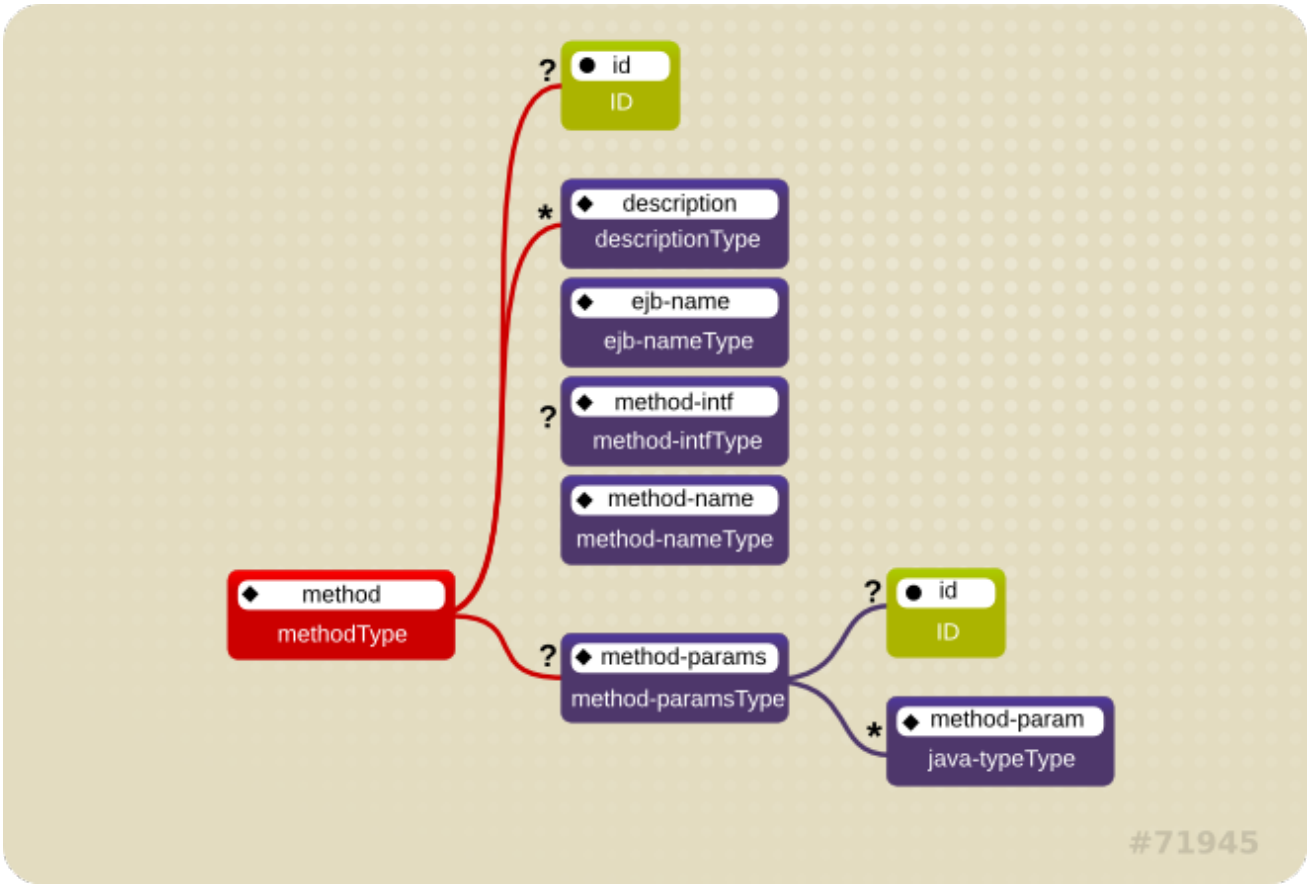


Figure 10.4. Élément de méthode Java EE

Il existe trois styles de déclarations d'éléments de méthodes pris en charge.

Le premier est utilisé pour se référer à toutes les méthodes d'interface de composant et d'accueil du bean d'entreprise nommé :

```
<method>
  <ejb-name>EJBNAME</ejb-name>
  <method-name>*</method-name>
</method>
```

Le second style est utilisé pour faire référence à une méthode particulière de l'interface de composant ou d'accueil du bean enterprise nommé :

```
  <method>
    <ejb-name>EJBNAME</ejb-name>
    <method-name>METHOD</method-name>
  </method>
```

S'il y a plusieurs méthodes possédant le même nom surchargé, ce style se réfèrera à toutes les méthodes de surchargement.

Le troisième style se réfère à une méthode particulière qui fait partie d'un ensemble de méthodes ayant un nom surchargé :

```
<method>
    <ejb-name>EJBNAME</ejb-name>
    <method-name>METHOD</method-name>
    <method-params>
```

```
        <method-param>PARAMETER_1</method-param>
        <!-- ... -->
        <method-param>PARAMETER_N</method-param>
    </method-params>
</method>
```

La méthode doit être définie dans l'interface de domicile ou à distance du bean entreprise spécifié. Les valeurs d'élément method-param correspondent au nom qualifié complet du type de paramètre de méthode correspondante. S'il y a plusieurs méthodes avec la même signature surchargée, la permission s'appliquera à l'ensemble des méthodes surchargées correspondantes.

L'élément optionnel **`method-intf`** peut être utilisé pour différencier des méthodes qui ont le même nom ou signature définis à la fois dans les interfaces d'accueil ou éloignées d'un bean enterprise.

Exemple 10.5, « Un fragment de descripteur ejb-jar.xml qui illustre l'utilisation de l'élément method-permission. » fournit des exemples complets d'utilisation d'élément **`method-permission`**.

Exemple 10.5. Un fragment de descripteur ejb-jar.xml qui illustre l'utilisation de l'élément method-permission.

```
<ejb-jar>
    <assembly-descriptor>
        <method-permission>
            <description>The employee and temp-employee roles may
access any
                method of the EmployeeService bean </description>
            <role-name>employee</role-name>
            <role-name>temp-employee</role-name>
            <method>
                <ejb-name>EmployeeService</ejb-name>
                <method-name>*</method-name>
            </method>
        </method-permission>
        <method-permission>
            <description>The employee role may access the
findByPrimaryKey,
                getEmployeeInfo, and the updateEmployeeInfo(String)
method of
                the AardvarkPayroll bean </description>
            <role-name>employee</role-name>
            <method>
                <ejb-name>AardvarkPayroll</ejb-name>
                <method-name>findByPrimaryKey</method-name>
            </method>
            <method>
                <ejb-name>AardvarkPayroll</ejb-name>
                <method-name>getEmployeeInfo</method-name>
            </method>
            <method>
                <ejb-name>AardvarkPayroll</ejb-name>
                <method-name>updateEmployeeInfo</method-name>
                <method-params>
                    <method-param>java.lang.String</method-param>
                </method-params>
            </method>
        </method-permission>
```

```
        <method-permission>
            <description>The admin role may access any method of the
                EmployeeServiceAdmin bean </description>
            <role-name>admin</role-name>
            <method>
                <ejb-name>EmployeeServiceAdmin</ejb-name>
                <method-name>*</method-name>
            </method>
        </method-permission>
        <method-permission>
            <description>Any authenticated user may access any method
of the
                EmployeeServiceHelp bean</description>
            <unchecked/>
            <method>
                <ejb-name>EmployeeServiceHelp</ejb-name>
                <method-name>*</method-name>
            </method>
        </method-permission>
        <exclude-list>
            <description>No fireTheCTO methods of the EmployeeFiring
bean may be
                used in this deployment</description>
            <method>
                <ejb-name>EmployeeFiring</ejb-name>
                <method-name>fireTheCTO</method-name>
            </method>
        </exclude-list>
    </assembly-descriptor>
</ejb-jar>
```

Rapporter un bogue

10.2.6. Annotations de sécurité de Beans Enterprise

Les Beans Enterprise utilisent des annotations pour transmettre des informations au programme de déploiement sur la sécurité et autres aspects de l'application. Le déployeur peut mettre en place la stratégie de sécurité de bean enterprise qui convient à l'application, si spécifiée dans les annotations, ou le descripteur de déploiement.

Toute valeur de méthode spécifiée explicitement dans le descripteur de déploiement substituent les valeurs de l'annotation. Si une valeur de méthode n'est pas spécifiée dans le descripteur de déploiement, ces valeurs assorties d'annotations seront utilisées. La granularité dominante est sur une base méthode.

Voici quelques annotations de sécurité qui peuvent être utilisées dans un Bean Enterprise :

@DeclareRoles

Déclare chaque rôle de sécurité déclaré dans le code. Pour plus d'informations sur la configuration des rôles, voir *Java EE 6 Tutorial* Specifying Authorized Users by Declaring Security Roles.

@RolesAllowed*, *@PermitAll*, and *@DenyAll

Indique les permissions de méthodes pour les annotations. Pour plus d'informations sur la configuration des permissions de méthodes d'annotations, voir *Java EE 56 Tutorial* Specifying Authorized Users by Declaring Security Roles.

@RunAs

Configure l'identité de sécurité propagée d'un composant. Pour plus d'informations sur la façon de configurer les identités de sécurité propagées par des annotations, voir *Java EE 6 Tutorial* Propagating a Security Identity (Run-As).

Rapporter un bogue

10.2.7. Contraintes de sécurité de contenu web

Dans une application web, la sécurité se définit par les rôles qui ont accès au contenu par un modèle d'url qui identifie le contenu protégé. Ce groupe d'informations est déclaré par l'élément **`web.xml`** security-constraint.

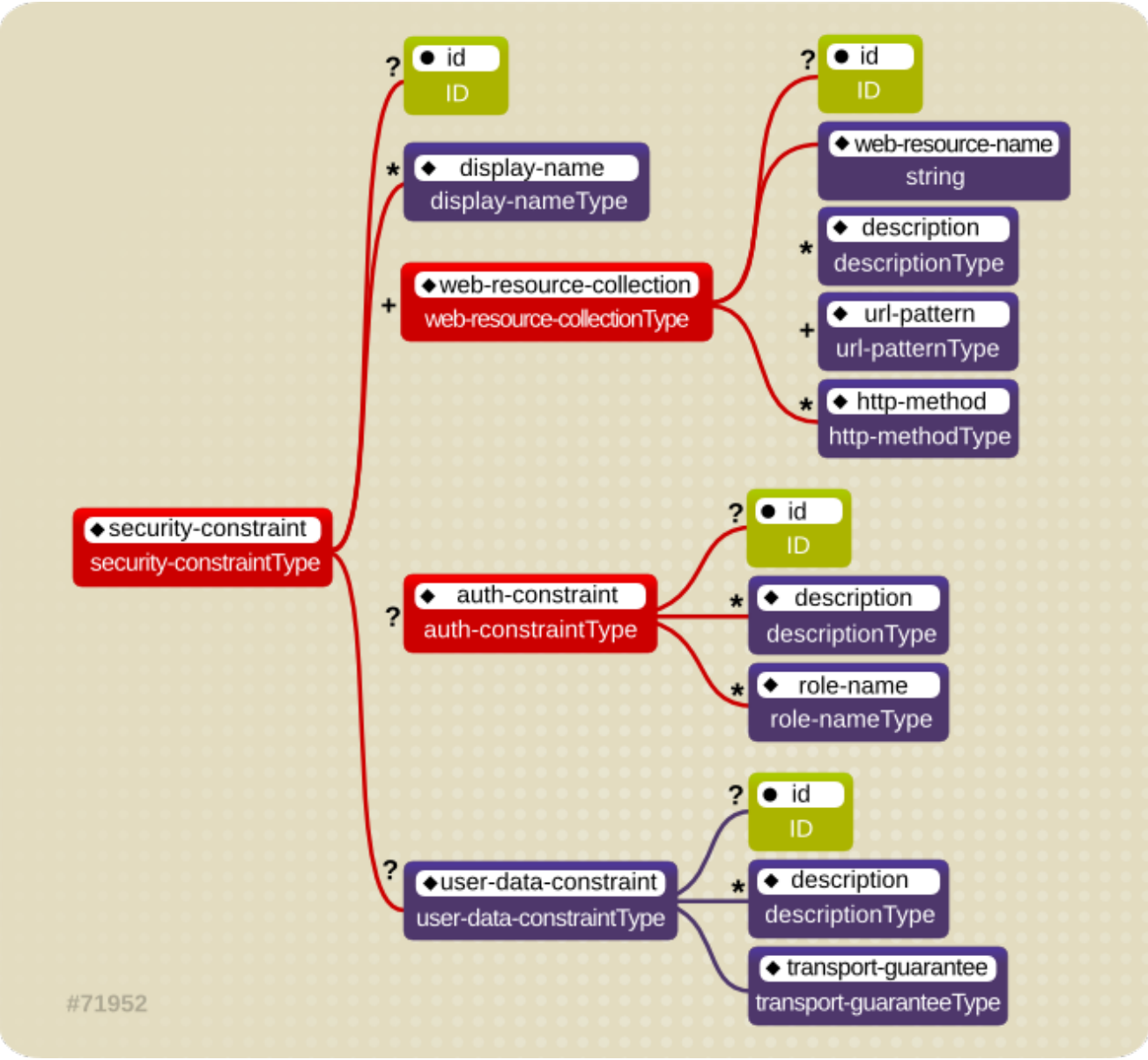


Figure 10.5. Contraintes de sécurité de contenu web

Le contenu à sécuriser est déclaré à l'aide d'un ou plusieurs éléments <web-resource-collection>.

Chaque élément <web-resource-collection> contient une série facultative d'éléments <url-pattern> suivie d'une série facultative d'éléments <http-method>. La valeur de l'élément <url-pattern > spécifie un URL qui doit correspondre à un URL de requête pour que la demande corresponde à une tentative d'accès au contenu sécurisé. La valeur de l'élément <http-method> spécifie un type de requête HTTP à autoriser.

L'élément optionnel <user-data-constraint> spécifie les exigences de la couche de transport du client pour la connexion au serveur. L'exigence peut être pour l'intégrité du contenu (empêchant la falsification dans le processus de communication de données) ou de la confidentialité (empêchant la lecture en transit). L'élément <transport-guarantee> indique le degré de protection de la communication entre le client et le serveur. Ses valeurs sont **`NONE`**, **`INTEGRAL`**, et **`CONFIDENTIAL`**. Une valeur **`NONE`** signifie que l'application n'a pas besoin de garantie de transport. Une valeur **`INTEGRAL`** signifie que l'application requiert que les données envoyées entre le client et le serveur soient envoyées de telle sorte qu'elles ne puissent pas être modifiées en transit. Une valeur **`CONFIDENTIAL`** signifie que l'application requiert que les données soient transmises d'une manière qui empêche les autres entités d'observer le contenu de la transmission. Dans la plupart des cas, la présence de **`INTEGRAL`** ou **`CONFIDENTIAL`** indique qu'il faut utiliser SSL.

L'élément optionnel <login-config> est utilisé pour configurer la méthode d'authentification qui doit être utilisée, le nom de domaine qui doit être utilisé pour l'application et les attributs qui sont nécessaires pour le mécanisme de connexion du formulaire.

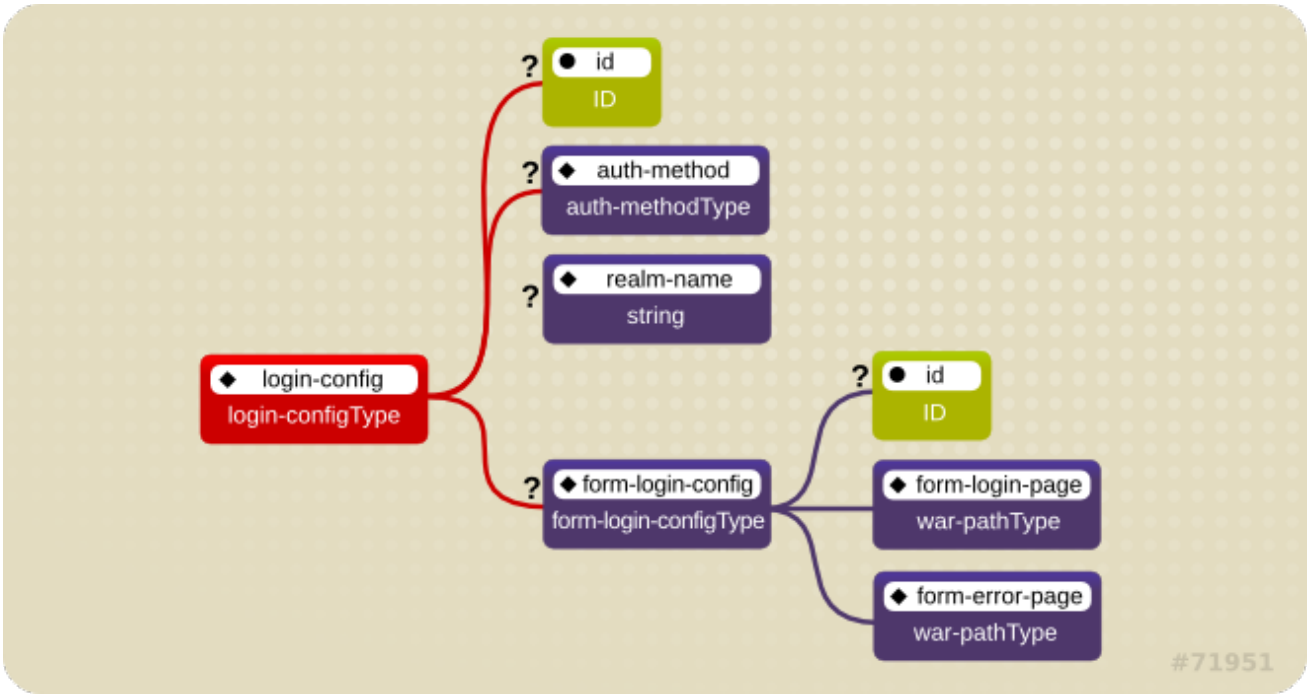


Figure 10.6. Configuration de connexion web

L'élément enfant <auth-method> spécifie le mécanisme d'authentification de l'application web. Comme condition préalable, l'accès à toutes les ressources web qui sont protégées par une contrainte de permission, un utilisateur doit avoir authentifié par le biais du mécanisme configuré. Les valeurs <auth-method> légales sont **`BASIC`**, **`DIGEST`**, **`FORM`**, **`SPNEGO`**, et **`CLIENT-CERT`**. L'élément enfant <realm-name> spécifie le nom de domaine à utiliser en HTTP de base et l'autorisation digest. L'élément enfant <form-login-config> spécifie le log in et les pages d'erreur devant être utilisées pour le log in basé-formulaire. Si la valeur <auth-method> n'est pas **`FORM`**, alors **`form-login-config`** et ses éléments enfants seront ignorés.

L'exemple de configuration suivant indique que n'importe quel URL se trouvant sous le chemin d'accès **`/restriction`** de l'application web requiert un rôle **`AuthorizedUser`**. Il n'y a aucune garantie de transport nécessaire et la méthode d'authentification utilisée pour obtenir l'identité de l'utilisateur est l'authentification HTTP BASIC.

Exemple 10.6. Fragment de descripteur web.xml

```
<web-app>
    <security-constraint>
        <web-resource-collection>
            <web-resource-name>Secure Content</web-resource-name>
            <url-pattern>/restricted/*</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <role-name>AuthorizedUser</role-name>
        </auth-constraint>
        <user-data-constraint>
            <transport-guarantee>NONE</transport-guarantee>
        </user-data-constraint>
    </security-constraint>
    <!-- ... -->
    <login-config>
        <auth-method>BASIC</auth-method>
        <realm-name>The Restricted Zone</realm-name>
    </login-config>
    <!-- ... -->
    <security-role>
        <description>The role required to access restricted content
</description>
        <role-name>AuthorizedUser</role-name>
    </security-role>
</web-app>
```

Rapporter un bogue

10.2.8. Activer l'authentification basée formulaire

L'authentification basée formulaire offre la flexibilité de définir une page JSP/HTML personnalisée de connexion, et une page séparée sur laquelle les utilisateurs sont redirigés si une erreur a lieu lors de la connexion.

L'authentification basée formulaire est définie en incluant **`<auth-method>FORM</auth-method>`** dans l'élément <login-config> du descripteur de déploiement, **`web.xml`**. La connexion et les pages d'erreur sont également définies dans <login-config>, comme suit :

```
<login-config>
  <auth-method>FORM</auth-method>
  <form-login-config>
    <form-login-page>/login.html</form-login-page>
    <form-error-page>/error.html</form-error-page>
  </form-login-config>
</login-config>
```

Lorsqu'une application web avec l'authentification par formulaire est déployée, le conteneur web utilise **`FormAuthenticator`** pour diriger les utilisateurs vers la page appropriée. JBoss EAP gère un pool de session afin que les informations d'authentification n'aient pas besoin d'être présentes pour chaque demande. Lorsque **`FormAuthenticator`** reçoit une demande, il interroge

org.apache.catalina.session.Manager pour savoir s'il y a une session existante. Si aucune session n'existe, une nouvelle session sera créée. **FormAuthenticator** vérifie alors les informations d'identification de la session.

NOTE

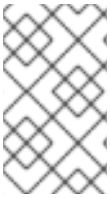
Chaque session est identifiée par un ID de session, une chaîne de 16 octets générée à partir de valeurs aléatoires. Ces valeurs sont extraites de **/dev/urandom** (Linux) par défaut et hachées avec MD5. Les vérifications sont effectuées lors de la création de l'ID de session pour s'assurer que l'ID créé soit unique.

Une fois vérifiée, l'ID de session est attribué dans le cadre d'un cookie et ensuite renvoyé au client. Ce cookie revient dans les requêtes clientes suivantes et sera utilisé pour identifier la session utilisateur.

Le cookie passée au client est une paire nom-valeur avec plusieurs attributs facultatifs. L'attribut d'identifiant est appelé **JSESSIONID**. Sa valeur est une chaîne hexadécimale de l'ID de session. Ce cookie est configuré pour être non persistant. Cela signifie que du côté client, il sera supprimé lorsque le navigateur se fermera. Côté serveur, les sessions expirent après 30 minutes d'inactivité, quand les objets session et leurs informations d'identification seront supprimés.

Si un utilisateur tente d'accéder à une application web protégée par l'authentification formulaire, **FormAuthenticator** met en cache la demande, crée une nouvelle session si nécessaire et redirige l'utilisateur vers la page de connexion définie dans **login-config**. Dans l'exemple de code précédent, la page de connexion est **login.html**. L'utilisateur entre alors son nom d'utilisateur et son mot de passe dans le formulaire HTML fourni. Les nom d'utilisateur et mot de passe sont passés à **FormAuthenticator** par l'intermédiaire de l'action de formulaire **j_security_check**.

Le **FormAuthenticator** authentifie alors le nom d'utilisateur et le mot de passe par rapport au domaine attaché au contexte de l'application web. Dans JBoss Enterprise Application Platform, le domaine est **JBossWebRealm**. Lorsque l'authentification réussit, **FormAuthenticator** récupère la requête enregistrée à partir du cache et redirige l'utilisateur vers sa requête initiale.



NOTE

Le serveur reconnaît les requêtes d'authentification de formulaire que lorsque les URI se terminent par **/j_security_check** et que les paramètres **j_username** et **j_password** existent.

Rapporter un bogue

10.2.9. Activation de la Sécurité déclarative

Les éléments de sécurité Java EE dont on a parlé jusqu'à présent décrivent les exigences de sécurité uniquement du point de vue de l'application. Comme les éléments de sécurité Java EE déclarent des rôles logiques, le déployeur d'applications mappe les rôles du domaine d'application dans l'environnement de déploiement. Les spécifications Java EE omettent ces détails spécifiques au serveur de l'application.

Pour mapper les rôles d'application dans l'environnement de déploiement, vous devez spécifier un gestionnaire de sécurité qui implémente le modèle de sécurité de Java EE à l'aide de descripteurs de déploiement JBoss EAP.

Voir également :

- Chapitre 17, *Sécurité basée-rôle pour les applications*

Rapporter un bogue

CHAPITRE 11. SÉCURITÉ DES APPLICATIONS

11.1. SÉCURITÉ DES BASES DE DONNÉES

11.1.1. Sécurité des bases de données

La notion de sécurité des sources de données se réfère à l'encodage ou au déguisement des mots de passe pour les connexions aux sources de données. Les mots de passe peuvent être stockés au format texte brut dans les fichiers de configuration, mais cela représente un risque de sécurité.

La meilleure solution de sécuriser les bases de données est soit l'utilisation des domaines de sécurité, soit les mots de passe. Vous trouverez un exemple de chacun ci-dessous. Pour plus d'informations, consulter :

- Domaines de sécurité : Section 12.3.3.1, « Les domaines de sécurité ».
- Mots de passe : Section 7.1, « Système d'archivage sécurisé de mots de passe ».

Exemple 11.1. Exemple de domaine de sécurité

```
 <security-domain name="DsRealm" cache-type="default">
  <authentication>
    <login-module code="ConfiguredIdentity" flag="required">
      <module-option name="userName" value="sa"/>
      <module-option name="principal" value="sa"/>
      <module-option name="password" value="sa"/>
    </login-module>
  </authentication>
</security-domain>
```

Le domaine DsRealm est référencé par une source de données comme :

```
<datasources>
  <datasource jndi-name="java:jboss/datasources/securityDs"
    pool-name="securityDs">
    <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-1</connection-url>
      <driver>h2</driver>
      <new-connection-sql>select current_user()</new-connection-sql>
      <security>
        <security-domain>DsRealm</security-domain>
      </security>
    </datasource>
</datasources>
```

Exemple 11.2. Exemple de mots de passe

```
<security>
  <user-name>admin</user-name>

<password>${VAULT::ds_ExampleDS::password::N2NhZDYzOTMtNWE0OS00ZGQ0LWE4M
mEtMWNlMDMyNDdmNmI2TElORV9CUkVBS3ZhdWx0}</password>
</security>
```

Rapporter un bogue

11.2. SÉCURITÉ DES APPLICATIONS EJB

11.2.1. Identité Sécurité

11.2.1.1. L'identité de sécurité EJB

Un EJB peut spécifier une identité à utiliser quand on invoque des méthodes sur d'autres composants. Il s'agit de *security identity* (également connu sous le nom *invocation identity*) de l'EJB.

Par défaut, l'EJB utilise sa propre identité d'appelant. L'identité peut également être définie à un rôle de sécurité spécifique. Les rôles de sécurité spécifiques sont utiles si vous voulez construire un modèle de sécurité segmenté - afin de, par exemple, restreindre l'accès à un ensemble de composants EJB internes uniquement.

Rapporter un bogue

11.2.1.2. Définir l'identité de sécurité d'un EJB

L'identité de sécurité de l'EJB est indiqué dans une balise **<security-identity>** dans la configuration de la sécurité.

Par défaut - si aucune balise **<security-identity>** n'est présente - l'identité de l'appelant de l'EJB sera utilisée.

Exemple 11.3. Définir l'identité de sécurité d'un EJB pour que ce soit la même que celle de l'appelant

Cet exemple définit l'identité de sécurité pour les invocations de méthode faîtes par un EJB de façon à ce qu'elle soit la même identité que celle de l'appelant actuel. Ce comportement correspond au comportement par défaut si vous ne spécifiez pas une déclaration d'élément **<security-identity>**.

```
<ejb-jar>
  <enterprise-beans>
  <session>
  <ejb-name>ASessionBean</ejb-name>
  <!-- ... -->
  <security-identity>
    <use-caller-identity/>
  </security-identity>
  </session>
  <!-- ... -->
  </enterprise-beans>
</ejb-jar>
```

Exemple 11.4. Définir l'idendité de sécurité d'un EJB à un rôle spécifique

Pour définir l'id de sécurité à un rôle spécifique, utiliser **<run-as>** et les balises **<role-name>** dans la balise **<security-identity>**.

```
<ejb-jar>
  <enterprise-beans>
  <session>
  <ejb-name>RunAsBean</ejb-name>
  <!-- ... -->
  <security-identity>
    <run-as>
    <description>A private internal role</description>
    <role-name>InternalRole</role-name>
    </run-as>
  </security-identity>
  </session>
  </enterprise-beans>
  <!-- ... -->
</ejb-jar>
```

Par défaut, quand vous utilisez **<run-as>**, un principal nommé **anonymous** est assigné aux appels sortants. Pour assigner un autre principal, utiliser **<run-as-principal>**.

```
<session>
    <ejb-name>RunAsBean</ejb-name>
    <security-identity>
        <run-as-principal>internal</run-as-principal>
    </security-identity>
</session>
```

NOTE

Vous pouvez utiliser les éléments **<run-as>** et **<run-as-principal>** à l'intérieur d'un élément de servlet.

Voir également :

- Section 11.2.1.1, « L'identité de sécurité EJB »
- Section A.6, « Référence de paramètre de sécurité EJB »

Rapporter un bogue

11.2.2. Permissions de méthodes EJB

11.2.2.1. Permissions de méthodes EJB

Les EJB peuvent réduire l'accès à leurs méthodes pour spécifier des rôles de sécurité spécifiques.

La déclaration d'élément EJB **<method-permisison>** définit les rôles qui peuvent appeler des méthodes de l'interface EJB. Vous pouvez définir des permissions pour les combinaisons suivantes :

- Toutes les méthodes d'interface de composant ou d'accueil de l'EJB nommé

- Une méthode spécifiée d'interface de composant ou d'accueil de l'EJB nommé
- Une méthode spécifiée à l'intérieur d'un ensemble de méthodes avec un nom surchargé

Rapporter un bogue

11.2.2.2. Utilisation des permissions de méthodes EJB

Aperçu

L'élément **<method-permission>** définit les roles logiques qui peuvent accéder aux méthodes EJB définies par les éléments **<method>**. Un certain nombre d'exemples expliquent la syntaxe XML. Plusieurs énoncés de method-permission peuvent être présents, et avoir un effet cumulatif. L'élément **<method-permission>** est un dépendant de l'élément **<assembly-descriptor>** du descripteur **<ejb-jar>**.

La syntaxe XML est une alternative aux annotations pour les permissions de méthode EJB.

Exemple 11.5. Permet aux rôles d'accéder à toutes les méthodes d'un EJB

```
<method-permission>
  <description>The employee and temp-employee roles may access any
method
  of the EmployeeService bean </description>
  <role-name>employee</role-name>
  <role-name>temp-employee</role-name>
  <method>
    <ejb-name>EmployeeService</ejb-name>
    <method-name>*</method-name>
  </method>
</method-permission>
```

Exemple 11.6. Permet aux rôles d'accéder uniquement à des méthodes spécifiques d'un EJB, et de déterminer quels paramètres de méthode peuvent être passés.

```
<method-permission>
  <description>The employee role may access the findByPrimaryKey,
  getEmployeeInfo, and the updateEmployeeInfo(String) method of
  the AcmePayroll bean </description>
  <role-name>employee</role-name>
  <method>
 <ejb-name>AcmePayroll</ejb-name>
 <method-name>findByPrimaryKey</method-name>
  </method>
  <method>
 <ejb-name>AcmePayroll</ejb-name>
 <method-name>getEmployeeInfo</method-name>
  </method>
  <method>
 <ejb-name>AcmePayroll</ejb-name>
 <method-name>updateEmployeeInfo</method-name>
 <method-params>
   <method-param>java.lang.String</method-param>
```

```
 </method-params>
  </method>
</method-permission>
```

Exemple 11.7. Permet à n'importe quel utilisateur authentifié d'accéder aux méthodes des EJB

Utiliser l'élément **<unchecked/>** permet à un utilisateur authentifié d'utiliser les méthodes spécifiées.

```
<method-permission>
  <description>Any authenticated user may access any method of the
  EmployeeServiceHelp bean</description>
  <unchecked/>
  <method>
 <ejb-name>EmployeeServiceHelp</ejb-name>
 <method-name>*</method-name>
  </method>
</method-permission>
```

Exemple 11.8. Exclut totalement certaines méthodes EJB à l'utilisation

```
<exclude-list>
  <description>No fireTheCTO methods of the EmployeeFiring bean may be
  used in this deployment</description>
  <method>
 <ejb-name>EmployeeFiring</ejb-name>
 <method-name>fireTheCTO</method-name>
  </method>
</exclude-list>
```

Exemple 11.9. Un `<assembly-descriptor>` complet contenant plusieurs blocs de `<method-permission>`

```
<ejb-jar>
    <assembly-descriptor>
        <method-permission>
            <description>The employee and temp-employee roles may
access any
                method of the EmployeeService bean </description>
            <role-name>employee</role-name>
            <role-name>temp-employee</role-name>
            <method>
                <ejb-name>EmployeeService</ejb-name>
                <method-name>*</method-name>
            </method>
        </method-permission>
        <method-permission>
            <description>The employee role may access the
findByPrimaryKey,
                getEmployeeInfo, and the updateEmployeeInfo(String)
```

```
method of
                the AcmePayroll bean </description>
            <role-name>employee</role-name>
            <method>
                <ejb-name>AcmePayroll</ejb-name>
                <method-name>findByPrimaryKey</method-name>
            </method>
            <method>
                <ejb-name>AcmePayroll</ejb-name>
                <method-name>getEmployeeInfo</method-name>
            </method>
            <method>
                <ejb-name>AcmePayroll</ejb-name>
                <method-name>updateEmployeeInfo</method-name>
                <method-params>
                    <method-param>java.lang.String</method-param>
                </method-params>
            </method>
        </method-permission>
        <method-permission>
            <description>The admin role may access any method of the
                EmployeeServiceAdmin bean </description>
            <role-name>admin</role-name>
            <method>
                <ejb-name>EmployeeServiceAdmin</ejb-name>
                <method-name>*</method-name>
            </method>
        </method-permission>
        <method-permission>
            <description>Any authenticated user may access any method
of the
                EmployeeServiceHelp bean</description>
            <unchecked/>
            <method>
                <ejb-name>EmployeeServiceHelp</ejb-name>
                <method-name>*</method-name>
            </method>
        </method-permission>
        <exclude-list>
            <description>No fireTheCTO methods of the EmployeeFiring
bean may be
                used in this deployment</description>
            <method>
                <ejb-name>EmployeeFiring</ejb-name>
                <method-name>fireTheCTO</method-name>
            </method>
        </exclude-list>
    </assembly-descriptor>
</ejb-jar>
```

Rapporter un bogue

11.2.3. Annotations de sécurité EJB

11.2.3.1. Les annotations de sécurité EJB

Les annotations EJB **`javax.annotation.security`** sont définies dans JSR250.

Les EJB utilisent des annotations de sécurité pour faire passer des informations de sécurité au déployeur. Celles-ci comprennent :

@DeclareRoles

Déclare quels rôles sont disponibles.

@RunAs

Configure l'identification de sécurité propagée d'un composant.

Rapporter un bogue

11.2.3.2. Utilisation des annotations de sécurité EJB

Aperçu

Vous pouvez utiliser deux descripteurs XML ou des annotations pour contrôler quels rôles de sécurité sont en mesure d'appeler des méthodes dans votre Enterprise JavaBeans (EJB). Pour plus d'informations sur l'utilisation des descripteurs XML, reportez-vous à Section 11.2.2.2, « Utilisation des permissions de méthodes EJB ».

Toute valeur de méthode spécifiée explicitement dans le descripteur de déploiement substituent les valeurs de l'annotation. Si une valeur de méthode n'est pas spécifiée dans le descripteur de déploiement, ces valeurs assorties d'annotations seront utilisées. La granularité dominante est sur une base méthode.

Annotations pour contrôler les permissions de sécurité des EJB

@DeclareRoles

Utiliser @DeclareRoles pour définir dans quels rôles vérifier les permissions. Si aucun @DeclareRoles n'est présent, la liste sera créée automatiquement avec l'annotation @RolesAllowed. Pour plus d'informations sur la configuration des rôles, voir *Java EE 6 Tutorial* Specifying Authorized Users by Declaring Security Roles.

@RolesAllowed, @PermitAll, @DenyAll

Utiliser **`@RolesAllowed`** pour lister les rôles qui peuvent accéder à une méthode ou à des méthodes. Utiliser **`@PermitAll`** ou **`@DenyAll`** pour autoriser ou empêcher à des rôles d'utiliser une méthode ou des méthodes. Pour plus d'informations sur la configuration des permissions de méthode d'annotation, consultez *Java EE 6 Tutorial* Specifying Authorized Users by Declaring Security Roles.

@RunAs

Utiliser **@RunAs** pour spécifier un rôle utilisé par une méthode quand elle lance des appels à partir d'une méthode annotée. Pour plus d'informations sur la façon de configurer les identités de sécurité propagées par des annotations, voir *Java EE 6 Tutorial* Propagating a Security Identity (Run-As).

Exemple 11.10. Exemple d'annotations de sécurité

```
@Stateless
```

```
@RolesAllowed({"admin"})
@SecurityDomain("other")
public class WelcomeEJB implements Welcome {
 @PermitAll
 public String WelcomeEveryone(String msg) {
  return "Welcome to " + msg;
 }
 @RunAs("tempemployee")
 public String GoodBye(String msg) {
     return "Goodbye, " + msg;
 }
 public String GoodbyeAdmin(String msg) {
  return "See you later, " + msg;
 }
}
```

Avec ce code, tous les rôles peuvent accéder à la méthode **`WelcomeEveryone`**. La méthode **`GoodBye`** exécute sous la forme du rôle **`tempemployee`** pour les appels. Seul le rôle **`admin`** peut accéder à la méthode **`GoodbyeAdmin`**, ou à toute autre méthode sans annotation de sécurité.

Rapporter un bogue

11.2.4. Accès à distance aux EJB

11.2.4.1. Remote Method Access

JBoss Remoting est le framework qui fournit un accès à distance aux EJBs, JMX MBeans, et autres services similaires. Fonctionne avec les types de transport suivants, avec ou sans SSL :

Types de services pris en charge

- Socket / Secure Socket
- RMI / RMI sur SSL
- HTTP / HTTPS
- Servlet / Secure Servlet
- Bisocket / Secure Bisocket

JBoss Remoting fournit aussi automatic discovery via Multicast ou JNDI.

Il est utilisé par bon nombre des sous-systèmes au sein de la plateforme JBoss EAP 6 et permet également de concevoir, d'implémenter et de déployer des services pouvant être appelés à distance par les clients sur plusieurs mécanismes de transport différents. Il vous permet également d'accéder aux services existants dans JBoss EAP 6.

Data Marshalling

Le système Remoting d'accès distant fournit également des services de marshalling et unmarshalling de données. Le marshaling de données désigne la capacité de déplacer en toute sécurité des données au-delà des limites de réseau et de la plate-forme, afin qu'un système séparé puisse effectuer des tâches dessus. Le travail est ensuite renvoyé vers le système d'origine et se comporte comme s'il avait eu lieu localement.

Aperçu de l'architecture

Lorsque vous créez une application cliente qui utilise Remoting, vous indiquez à votre application de communiquer avec le serveur en la configurant pour qu'elle utilise un type spécial de localisateur de ressources appelé un **`InvokerLocator`**, qui est une chaîne simple avec un format de type URL. Le serveur écoute les requêtes des ressources distantes sur un **`connecteur`**, qui est configuré comme faisant partie du sous-système **`remoting`**. Le **`connecteur`** transmet la demande à un **`ServerInvocationHandler`** configuré. Chaque **`ServerInvocationHandler`** implémente une méthode **`invoke(InvocationRequest)`** qui sait comment gérer la demande.

Le framework JBoss Remoting contient trois couches qui se miroitent les unes par rapport aux autres côté client et côté serveur.

Couches de framework JBoss Remoting

- L'utilisateur interagit avec la couche externe. Côté client, la couche externe est la classe **`Client`**, qui envoie des requêtes d'invocation. Côté serveur, c'est InvocationHandler, mis en œuvre par l'utilisateur, qui reçoit des demandes d'invocation.
- Le transport est contrôlé par la couche d'invocateur.
- La couche inférieure contient le marshaller et le unmarshaller, qui convertit les formats de données en formats de transmission.

Rapporter un bogue

11.2.4.2. Remoting Callbacks

Lorsqu'un client Remoting (à distance) demande des informations du serveur, il peut y avoir un blocage et il faut alors attendre que le serveur réponde, mais ce n'est pas un comportement idéal. Pour permettre au client d'être à l'écoute des événements asynchrones sur le serveur et pour pouvoir continuer à faire d'autre tâches en attendant que le serveur complète la requête, votre application peut demander au serveur d'envoyer une notification quand il aura fini. C'est ce que l'on appelle un callback. Un client peut s'ajouter comme un écouteur d'événements asynchrones générés au nom d'un autre client, aussi bien. Il y a deux choix différents pour recevoir des callbacks : «pull callbacks» ou «push callbacks». Les clients vérifient les «pull callbacks» de façon synchrone, mais écoutent passivement les «push callbacks».

En substance, un callback fonctionne de cette façon: le serveur envoie une **`InvocationRequest`** au client. Votre code côté serveur fonctionne de la même façon que le rappel soit synchrone ou asynchrone. Seul le client a besoin de connaître la différence. InvocationRequest du serveur envoie un **`responseObject`** au client. Il s'agit de la charge que le client a demandé. C'est peut-être une réponse directe à une demande ou à une notification d'événement.

Votre serveur suit aussi les listeners à l'aide d'un objet **`m_listeners`**. Il contient une liste de tous les listeners qui ont été ajoutés à votre server handler. L'interface du **`ServerInvocationHandler`** inclut des méthodes qui vous permettent de gérer cette liste.

Le client gère les «pull callbacks» et les «push callbacks» de différentes manières. Dans les deux cas, il doit implémenter un callback handler. Un callback handler est une implémentation de l'interface **`org.jboss.remoting.InvokerCallbackHandler`**, qui traite les données de callback. Après l'implémentation du callback handler, soit vous vous ajoutez vous-même, en tant que listener de «pull callback», ou bien, vous installez un serveur de callbacks pour un «push callback».

Pull Callbacks

Pour un «pull callback», votre client s'ajoute à la liste du serveur des listeners à l'aide de la méthode **`Client.addListener()`**. Ensuite, il interroge le serveur périodiquement au sujet de l'exécution synchrone des données de callback. Ce sondage est effectué à l'aide de **`Client.getCallbacks()`**.

Push Callback

Un «push callback» requiert que votre application cliente exécute elle-même son propre InvocationHandler. Pour ce faire, vous devez exécuter un service Remoting sur le client lui-même. Ceci s'appelle un *callback server*. Le callback server accepte les requêtes entrantes de façon asynchrone et les traite pour l'auteur de la demande (dans ce cas, le serveur). Pour inscrire le callback server de votre client avec le serveur principal, passez l'argument **`InvokerLocator`** du callback server comme deuxième argument à la méthode **`addListener`**.

Rapporter un bogue

11.2.4.3. Remoting Server Detection

Les clients et les serveurs d'accès distant peuvent se détecter les uns les autres automatiquement à l'aide de JNDI ou Multicast. Un détecteur Remoting Detector est ajouté aux client et serveur, et un NetworkRegistry est ajoutée au client.

Le détecteur côté serveur scanne périodiquement InvokerRegistry et extrait tous les invocateurs de serveur qu'il a créés. Il utilise ces informations pour publier un message de détection, qui contient le localisateur et les sous-systèmes pris en charge par chaque invocateur de serveur. Il publie ce message via une multidiffusion ou une liaison vers un serveur JNDI.

Côté client, le détecteur reçoit le message de multidiffusion ou interroge périodiquement le serveur JNDI pour récupérer les messages de détection. Si le détecteur remarque qu'un message de détection est pour un serveur d'accès distant nouvellement détecté, il l'inscrit dans NetworkRegistry. Le détecteur met également à jour NetworkRegistry s'il détecte qu'un serveur n'est plus disponible

Rapporter un bogue

11.2.4.4. Configurer le sous-système de JBoss Remoting

Aperçu

JBoss Remoting a trois éléments configurables de niveau supérieur : le pool de worker threads, un ou plusieurs connecteurs et une série de liens URI locaux et distants. Cette rubrique propose une explication pour chaque élément configurable, des exemples de commandes CLI pour savoir comment configurer chaque élément et un exemple XML d'un sous-système entièrement configuré. Cette configuration s'applique uniquement au serveur. La plupart des gens n'auront pas à configurer le sous-système de communication à distance, sauf s'ils utilisent des connecteurs personnalisés pour leurs propres applications. Les applications qui agissent comme des clients Remoting, comme les EJB, nécessitent une configuration distincte pour se connecter à un connecteur spécifique.

NOTE

La configuration du sous-système Remoting n'est pas exposée à la Console de gestion sur web, mais est entièrement configurable de l'interface CLI en ligne de commande. Il n'est pas recommandé de modifier le code XML à la main.

Adaptation des commandes CLI

Les commandes CLI sont formulées pour un domaine géré, lorsque vous configurez le profil **par défaut**. Pour configurer un profil différent, changez-en le nom. Pour un serveur autonome, omettre la section **/profile=default** de la commande.

Configuration en dehors du sous-système Remoting

Il y a un certain nombre d'aspects de configuration qui sont en dehors du sous-système **remoting** :

Network Interface

L'interface de réseau qui est utilisée par le sous-système **remoting** est l'interface **unsecure** définie dans **domain/configuration/domain.xml** ou dans **standalone/configuration/standalone.xml**.

```
<interfaces>
   <interface name="management"/>
   <interface name="public"/>
   <interface name="unsecure"/>
</interfaces>
```

La définition par-hôte de l'interface **unsecure** est définie dans **host.xml** dans le même répertoire que **domain.xml** ou **standalone.xml**. Cette interface est également utilisée par plusieurs autres sous-systèmes. Soyez vigilants quand vous la modifierez.

```
<interfaces>
   <interface name="management">
      <inet-address
value="${jboss.bind.address.management:127.0.0.1}"/>
   </interface>
   <interface name="public">
      <inet-address value="${jboss.bind.address:127.0.0.1}"/>
   </interface>
   <interface name="unsecure">
      <!-- Used for IIOP sockets in the standard configuration.
         To secure JacORB you need to setup SSL -->
      <inet-address value="${jboss.bind.address.unsecure:127.0.0.1}"/>
   </interface>
</interfaces>
```

socket-binding

La liaison-socket par défaut utilisée par le sous-système **remoting** se lie au port TCP 4777. Reportez-vous à la documentation sur les liaisons de socket et de groupes de liaisons de socket pour plus d'informations si vous avez besoin de procéder à une modification.

Remoting Connector Reference pour EJB

Le sous-système EJB contient une référence vers le connecteur à distance pour les invocations de méthodes à distance. Voici la configuration par défaut :

```
<remote connector-ref="remoting-connector" thread-pool-name="default"/>
```

Configuration du transport sécurisé

Les transports à distance (Remoting) utilisent StartTLS pour avoir une connexion sécurisée (HTTPS, Secure Servlet, etc.) si le client le demande. La même liaison de socket (port réseau) est utilisée pour les connexions sécurisées et non sécurisées, donc aucune configuration côté serveur supplémentaire n'est nécessaire. Le client demande le transport sécurisé ou non sécurisé, tel que le dictent ses besoins. Les composants JBoss EAP 6 qui utilisent Remoting, tels que les fournisseur JMS, EJB et ORB exigent des interfaces sécurisées par défaut.

AVERTISSEMENT

StartTLS fonctionne en activant une connexion sécurisée si le client le demande au lieu de, par défaut, d'une connexion non sécurisée. StartTLS est, par nature, sensible à un exploit de style *Man in the Middle* dans lequel un attaquant intercepte la demande du client et la modifie pour demander une connexion non sécurisée. Les clients doivent avoir reçu des écritures pour échouer correctement s'ils ne reçoivent pas une connexion sécurisée, sauf si une connexion non sécurisée constitue un fall-back approprié.

Worker Thread Pool

Un Worker Thread Pool est un ensemble de threads qui peuvent traiter les tâches qui arrivent par les connecteurs Remoting. Il s'agit d'un seul élément **`<worker-thread-pool>`**, et nécessite un certain nombre d'attributs. Régler ces attributs si vous avez des timeouts de réseau, ou si vous devez limiter l'utilisation de la mémoire. Les conseils varient suivant la situation dans laquelle vous vous trouvez. Contacter Red Hat Global Support Services pour obtenir davantage d'informations.

Tableau 11.1. Attributs de Worker Thread Pool

Attribut	Description	Commande CLI
read-threads	Le nombre de read-threads à créer pour le worker à distance. La valeur par défaut est **`1`**.	**`/profile=default/subsystem=remoting/:write-attribute(name=worker-read-threads,value=1)`**
write-threads	Le nombre de write-threads à créer pour le worker à distance. La valeur par défaut est **`1`**.	**`/profile=default/subsystem=remoting/:write-attribute(name=worker-write-threads,value=1)`**
task-keepalive	Le nombre de millisecondes pour conserver les threads de tâche de workers à distance non-core vivants. La valeur par défaut est **`60`**.	**`/profile=default/subsystem=remoting/:write-attribute(name=worker-task-keepalive,value=60)`**

Attribut	Description	Commande CLI
task-max-threads	Le nombre de maximum de threads pour le Worker Task Thread Pool distant. La valeur par défaut est **60**.	**/profile=default/subsystem=remoting/:write-attribute(name=worker-task-max-threads,value=16)**
task-core-threads	Le nombre de threads principaux pour le Worker Task Thread Pool distant. La valeur par défaut est **4**.	**/profile=default/subsystem=remoting/:write-attribute(name=worker-task-core-threads,value=4)**
task-limit	Le nombre de maximum de tâches de worker distantes. La valeur par défaut est **16384**.	**/profile=default/subsystem=remoting/:write-attribute(name=worker-task-limit,value=16384)**

Connecteur

Le connecteur est le principal élément de configuration de Remoting (d'accès à distance). Les connecteurs multiples sont autorisés. Chacun se compose d'un élément **<connector>** avec plusieurs sous-éléments, ainsi que quelques attributs possibles. Le connecteur par défaut est utilisé par plusieurs sous-systèmes de JBoss EAP 6. Des paramètres spécifiques pour les éléments et les attributs de vos connecteurs personnalisés dépendent de vos applications, donc contactez Red Hat Global Support Services pour plus d'informations.

Tableau 11.2. Attributs de connecteur

Attribut	Description	Commande CLI
socket-binding	Le nom de la liaison de socket à utiliser pour ce connecteur.	**/profile=default/subsystem=remoting/connector=remoting-connector/:write-attribute(name=socket-binding,value=remoting)**
authentication-provider	Le module JASPIC (de l'anglais Java Authentication Service Provider Interface for Containers) à utiliser avec ce connecteur. Le module doit être dans le chemin de classes.	**/profile=default/subsystem=remoting/connector=remoting-connector/:write-attribute(name=authentication-provider,value=myProvider)**

Attribut	Description	Commande CLI
security-realm	En option. Le domaine de la sécurité qui contient les utilisateurs, mots de passe et les rôles de votre application. Un EJB ou une Application Web peut authentifier sur un domaine de sécurité. **`ApplicationRealm`** est disponible dans une installation de JBoss EAP 6 par défaut.	**`/profile=default/subsystem=remoting/connector=remoting-connector/:write-attribute(name=security-realm,value=ApplicationRealm)`**

Tableau 11.3. Éléments de connecteur

Attribut	Description	Commande CLI
sasl	Élément englobant des mécanismes d'authentification SASL (Simple Authentication and Security Layer)	**`N/A`**
propriétés	Contient un ou plusieurs éléments **`<property>`**, contenant chacun un attribut **`name`** et un attribut **`value`** optionnel.	**`/profile=default/subsystem=remoting/connector=remoting-connector/property=myProp/:add(value=myPropValue)`**

Les connexions de sortie

Vous pouvez spécifier trois types différents d'attribut de connexion sortante :

- Connexion sortante vers un URI.
- Connexion sortante locale – se connectant à une ressource locale, comme un socket.
- Connexion sortante à distance – se connectant à une ressource à distance et s'authentifiant par l'intermédiaire d'un domaine de sécurité.

Toutes les connexions sortantes sont enfermées dans un élément **`<outbound-connections>`**. Chacun de ces types de connexion prend un attribut **`outbound-socket-binding-ref`**. La connexion sortante prend un attribut **`uri`**. La connexion sortante prend les attributs facultatifs **`username`** (nom d'utilisateur) et **`security-realm`** (domaine de sécurité) pour l'autorisation.

Tableau 11.4. Éléments de connexion sortante

Attribut	Description	Commande CLI

Attribut	Description	Commande CLI
outbound-connection	Connexion sortante standard	**`/profile=default/subsystem=remoting/outbound-connection=my-connection/:add(uri=http://my-connection)`**
local-outbound-connection	Connexion sortante en schéma implicite local:// URI.	**`/profile=default/subsystem=remoting/local-outbound-connection=my-connection/:add(outbound-socket-binding-ref=remoting2)`**
remote-outbound-connection	Connexions sortantes en schéma remote:// URI, utilisant l'authentification de base/digest avec un domaine de sécurité.	**`/profile=default/subsystem=remoting/remote-outbound-connection=my-connection/:add(outbound-socket-binding-ref=remoting,username=myUser,security-realm=ApplicationRealm)`**

Éléments SASL

Avant de définir des éléments enfants SASL, vous devez créer l'élément SASL initial. Utiliser la commande suivante :

```
/profile=default/subsystem=remoting/connector=remoting-
connector/security=sasl:add
```

Les éléments enfants de l'élément SASL sont décrits dans le tableau ci-dessous.

Attribut	Description	Commande CLI
include-mechanisms	Contient un attribut **`value`**, qui correspond à une liste de mécanismes SASL séparés par des espaces.	`/profile=default/subsystem=remoting/connector=remoting-connector/security=sasl:write-attribute(name=include-mechanisms,value=["DIGEST","PLAIN","GSSAPI"])`

Attribut	Description	Commande CLI
qop	Contient un attribut **`value`**, qui correspond à une liste de valeurs de protection SASL séparées par des espaces, en ordre décroissant de préférence.	`/profile=default/subsystem=remoting/connector=remoting-connector/security=sasl:write-attribute(name=qop,value=["auth"])`
puissance	Contient un attribut **`value`**, qui correspond à une liste de valeurs de puissance cipher SASL séparées par des espaces, en ordre décroissant de préférence.	`/profile=default/subsystem=remoting/connector=remoting-connector/security=sasl:write-attribute(name=strength,value=["medium"])`
reuse-session	Contient un attribut **`value`**, qui correspond à une valeur booléenne. Si sur true, tente de réutiliser les sessions.	`/profile=default/subsystem=remoting/connector=remoting-connector/security=sasl:write-attribute(name=reuse-session,value=false)`
server-auth	Contient un attribut **`value`**, qui correspond à une valeur booléenne. Si sur true, le serveur s'authentifie auprès du client.	`/profile=default/subsystem=remoting/connector=remoting-connector/security=sasl:write-attribute(name=server-auth,value=false)`
politique	Un élément clôturé qui contient zéro ou plusieurs des éléments suivants, prenant chacun une seule **`valeur`**. • forward-secrecy – indique si on a besoin de mécanismes pour implémenter la forward-secrecy (l'infiltration dans une session ne va pas forcément donner des	`/profile=default/subsystem=remoting/connector=remoting-connector/security=sasl/sasl-policy=policy:add` `/profile=default/subsystem=remoting/conn`

Attribut	Description	Commande CLI
	informations sur la façon de s'infiltrer dans des sessions futures) • no-active – indique si les mécanismes sensibles aux attaques hors dictionnaire sont permis. Une valeur **`false`** permet, et **`true`** non. • no-anonymous – indique si les mécanismes qui acceptent la connexion anonyme sont permis. Une valeur de **`false`** le permet, et **`true`** non. • no-active – indique si les mécanismes sensibles aux attaques dictionnaire passives sont permis. Une valeur **`false`** le permet, et **`true`** non. • no-plain-text – indique si les mécanismes sensibles aux simples attaques dictionnaire passives sont permis. Une valeur **`false`** le permet, et **`true`** non. • pass-credentials – indique si les mécanismes qui font passer les authentifications clients sont permis.	ector=remoting-connector/security=sasl/sasl-policy=policy:write-attribute(name=forward-secrecy,value=true) /profile=default/subsystem=remoting/connector=remoting-connector/security=sasl/sasl-policy=policy:write-attribute(name=no-active,value=false) /profile=default/subsystem=remoting/connector=remoting-connector/security=sasl/sasl-policy=policy:write-attribute(name=no-anonymous,value=false) /profile=default/subsystem=remoting/connector=remoting-connector/security=sasl/sasl-policy=policy:write-attribute(name=no-dictionary,value=true) /profile=default/subsystem=remoting/connector=remoting-connector/security=sasl/sasl-policy=policy:write-attribute(name=no-plain-text,value=false) /profile=default/subsystem=remoting/connector=remoting-connector/security=sasl/sasl-policy=policy:write-

Attribut	Description	Commande CLI
		attribute(name=pass-credentials,value=true)
propriétés	Contient un ou plusieurs éléments **<property>**, contenant chacun un attribut **name** et un attribut **value** optionnel.	/profile=default/subsystem=remoting/connector=remoting-connector/security=sasl/property=myprop:add(value=1) /profile=default/subsystem=remoting/connector=remoting-connector/security=sasl/property=myprop2:add(value=2)

Exemple 11.11. Exemples de configurations

Cet exemple montre le sous-système à distance par défaut fourni dans JBoss EAP 6.

```
<subsystem xmlns="urn:jboss:domain:remoting:1.1">
    <connector name="remoting-connector" socket-binding="remoting"
security-realm="ApplicationRealm"/>
</subsystem>
```

Cet exemple contient de nombreuses valeurs hypothétiques, et est présenté de façon à ce que l'on puisse mettre les éléments et les attributs dont on a discutés plus haut en contexte.

```
<subsystem xmlns="urn:jboss:domain:remoting:1.1">
    <worker-thread-pool read-threads="1" task-keepalive="60" task-max-
threads="16" task-core-thread="4" task-limit="16384" write-threads="1"
/>
    <connector name="remoting-connector" socket-binding="remoting"
security-realm="ApplicationRealm">
        <sasl>
            <include-mechanisms value="GSSAPI PLAIN DIGEST-MD5" />
            <qop value="auth" />
            <strength value="medium" />
            <reuse-session value="false" />
            <server-auth value="false" />
            <policy>
                <forward-secrecy value="true" />
                <no-active value="false" />
                <no-anonymous value="false" />
                <no-dictionary value="true" />
                <no-plain-text value="false" />
                <pass-credentials value="true" />
            </policy>
```

```
            <properties>
                <property name="myprop1" value="1" />
                <property name="myprop2" value="2" />
            </properties>
        </sasl>
        <authentication-provider name="myprovider" />
        <properties>
            <property name="myprop3" value="propValue" />
        </properties>
    </connector>
    <outbound-connections>
        <outbound-connection name="my-outbound-connection"
uri="http://myhost:7777/"/>
        <remote-outbound-connection name="my-remote-connection"
outbound-socket-binding-ref="my-remote-socket" username="myUser"
security-realm="ApplicationRealm"/>
        <local-outbound-connection name="myLocalConnection" outbound-
socket-binding-ref="my-outbound-socket"/>
    </outbound-connections>
</subsystem>
```

Aspects de la configuration non encore documentés

- JNDI et Détection automatique multi diffusion

Rapporter un bogue

11.2.4.5. Utilisation des domaines de sécurité avec les clients EJB distants

Une façon d'ajouter la sécurité aux clients qui font appel aux EJB à distance consiste à utiliser des domaines de sécurité. Un domaine de sécurité est une simple base de données de paires nom d'utilisateur/mot de passe et nom d'utilisateur/rôle. La terminologie est également utilisée dans le cadre de conteneurs web, avec un sens légèrement différent.

Pour authentifier un EJB par une paire nom d'utilisateur et mot de passe particulière existant déjà dans un domaine de sécurité, suivre les étapes suivantes :

- Ajoutez un nouveau domaine de sécurité au contrôleur de domaine ou au serveur autonome.
- Ajoutez les paramètres suivants au fichier **`jboss-ejb-client.properties`**, qui est dans le chemin de classes de l'application. Cet exemple suppose que la connexion est appelée **`par défaut`** par les autres paramètres qui se trouvent dans le fichier.

  ```
  remote.connection.default.username=appuser
  remote.connection.default.password=apppassword
  ```

- Créez un connecteur Remoting personnalisé sur le domaine ou sur le serveur autonome qui utilise le nouveau domaine de sécurité.
- Déployez votre EJB dans le groupe de serveur configuré pour utiliser le profil avec le connecteur Remoting personnalisé, ou dans le serveur autonome si vous n'utilisez pas de domaine géré.

Rapporter un bogue

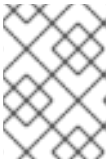
11.2.4.6. Ajout d'un domaine de sécurité

1. **Exécuter l'interface CLI**
   Démarrez par la commande **`jboss-cli.sh`** ou **`jboss-cli.bat`** et connectez-vous au serveur.

2. **Créer le nouveau domaine de sécurité lui-même.**
   Exécutez la commande suivante pour créer un nouveau domaine de sécurité nommé **`MyDomainRealm`** sur un contrôleur de domaine ou sur un serveur autonome.

   ```
   /host=master/core-service=management/security-
   realm=MyDomainRealm:add()
   ```

3. **Créer les références du fichier de propriétés qui stocke les informations sur le nouveau rôle.**
   Exécutez la commande suivante pour créer un pointeur au fichier nommé **`myfile.properties`**, qui contiendra les propriétés attachées au nouveau rôle.

   NOTE

   Le fichier de propriétés nouvellement créées n'est pas géré par les scripts **`add-user.sh`** et **`add-user.bat`** inclus. Il devra être administré en externe.

   ```
   /host=master/core-service=management/security-
   realm=MyDomainRealm/authentication=properties:add(path=myfile.proper
   ties)
   ```

Résultat

Votre nouveau domaine de sécurité sera créé. Lorsque vous ajoutez des utilisateurs et des rôles à ce nouveau domaine, l'information va être stockée dans un fichier séparé des domaines de sécurité par défaut. Vous pouvez gérer ce nouveau fichier à l'aide de vos propres applications ou procédures.

Rapporter un bogue

11.2.4.7. Ajout d'un utilisateur à un domaine de sécurité

1. **Éxécuter la commande `add-user.sh` ou `add-user.bat`.**
   Ouvrez un terminal (shell) et changez de répertoire ***`EAP_HOME`*`/bin/`**. Si vous exécutez Red Hat Enterprise Linux ou un autre système d'exploitation style UNIX, exécutez **`add-user.sh`**. Si vous exécutez sur un serveur Microsoft Windows, exécutez **`add-user.bat`**.

2. **Choisir d'ajouter un utilisateur de gestion ou un utilisateur d'application.**
   Pour cette procédure, saisir **b** pour ajouter un utilisateur d'application.

3. **Choisir le domaine dans lequel l'utilisateur sera ajouté.**
   Par défaut, le seul domaine disponible est **`ApplicationRealm`**. Si vous avez ajouté un domaine personnalisé, vous pouvez saisir son nom à la place.

4. **Saisir le nom d'utilisateur, le mot de passe et les rôles lorsque vous y serez invité.**
   Saisir le nom d'utilisateur, le mot de passe et les rôles lorsque vous y serez invité. Vérifiez votre choix en tapant **`yes`**, ou **no** pour annuler les changements. Les changements sont inscrits dans les fichiers de propriétés du domaine de sécurité.

Rapporter un bogue

11.2.4.8. Accès EJB à distance utilisant le cryptage SSL

Par défaut, le trafic réseau pour les RMI (Remote Method Invocation) des Beans EJB2 et EJB3 n'est pas crypté. Dans le cas où le cryptage est requis, SSL (Secure Sockets Layer) peut être utilisé afin que la connexion entre le client et le serveur soit cryptée. L'utilisation SSL a l'avantage supplémentaire de permettre au trafic réseau de traverser certains pare-feux, selon leur configuration.

Rapporter un bogue

11.3. SÉCURITÉ DES APPLICATIONS JAX-RS

11.3.1. Activez la sécurité basée-rôle pour RESTEasy JAX-RS Web Service

Résumé

RESTEasy supporte les annotations @RolesAllowed, @PermitAll, et @DenyAll sur les méthodes JAX-RS. Cependant, il ne reconnaît pas ces annotations par défaut. Suivre les étapes suivantes pour configurer le fichier **web.xml** et pour activer la sécurité basée-rôle.

AVERTISSEMENT

Ne pas activer la sécurité basée-rôle si l'application utilise les EJB. Le conteneur EJB procurera la fonctionnalité à la place de RESTEasy.

Procédure 11.1. Activez la sécurité basée-rôle pour RESTEasy JAX-RS Web Service

1. Ouvrir le fichier **web.xml** de l'application dans un éditeur de texte.
2. Ajoutez le <context-param> suivant au fichier, dans les balises **web-app**:

```
<context-param>
    <param-name>resteasy.role.based.security</param-name>
    <param-value>true</param-value>
</context-param>
```

3. Déclarez tous les rôles utilisés dans le fichier RESTEasy JAX-RS WAR file, en utilisant les balises de <security-role>:

```
<security-role>
    <role-name>ROLE_NAME</role-name>
</security-role>
<security-role>
    <role-name>ROLE_NAME</role-name>
</security-role>
```

4. Autorisez l'accès à tous les URL gérés par le runtime JAX-RS pour tous les rôles :

```
<security-constraint>
    <web-resource-collection>
 <web-resource-name>Resteasy</web-resource-name>
 <url-pattern>/PATH</url-pattern>
    </web-resource-collection>
    <auth-constraint>
 <role-name>ROLE_NAME</role-name>
 <role-name>ROLE_NAME</role-name>
    </auth-constraint>
</security-constraint>
```

Résultat

La sécurité basée rôle à été activée dans l'application, avec un certain nombre de rôles définis.

Exemple 11.12. Exemple de configuration de sécurité basée rôles

```
<web-app>

    <context-param>
 <param-name>resteasy.role.based.security</param-name>
 <param-value>true</param-value>
    </context-param>

    <servlet-mapping>
 <servlet-name>Resteasy</servlet-name>
 <url-pattern>/*</url-pattern>
    </servlet-mapping>

    <security-constraint>
 <web-resource-collection>
     <web-resource-name>Resteasy</web-resource-name>
     <url-pattern>/security</url-pattern>
 </web-resource-collection>
 <auth-constraint>
     <role-name>admin</role-name>
     <role-name>user</role-name>
 </auth-constraint>
    </security-constraint>

    <security-role>
 <role-name>admin</role-name>
    </security-role>
    <security-role>
 <role-name>user</role-name>
    </security-role>

</web-app>
```

Rapporter un bogue

11.3.2. Sécuriser un service JAX-RS Web par des annotations

Résumé

Cette rubrique couvre les étapes à parcourir pour sécuriser un service JAX-RS Web par les annotations de sécurité supportées.

Procédure 11.2. Sécuriser un service JAX-RS Web par des annotations de sécurité supportées.

1. Activez la sécurité basée-rôle. Pour plus d'informations, voir Section 11.3.1, « Activez la sécurité basée-rôle pour RESTEasy JAX-RS Web Service ».

2. Ajoutez des annotations de sécurité au service JAX-RS Web. RESTEasy supporte les annotations suivantes :

   @RolesAllowed

   Définit les rôles qui peuvent accéder à la méthode. Tous les rôles doivent être définis dans le fichier **`web.xml`**.

   @PermitAll

   Autorise tous les rôles définis dans le fichier **`web.xml`** à accéder à la méthode.

   @DenyAll

   Refuse tout accès à la méthode.

Rapporter un bogue

CHAPITRE 12. LE SOUS-SYSTÈME DE SÉCURITÉ

12.1. LA SÉCURITÉ DU SOUS-SYSTÈME

Le sous-système de **sécurité** fournit l'infrastructure de sécurité pour les applications. Le sous-système utilise un contexte de sécurité associé à la demande en cours pour exposer les fonctionnalités du gestionnaire d'authentification, du gestionnaire d'autorisations, du gestionnaire d'auditing et du gestionnaire de mappage (mises en correspondance) pour le conteneur approprié.

Le sous-système **security** est préconfiguré par défaut, donc les éléments de sécurité ont rarement besoin d'être changés. Le seul élément de sécurité qui ait besoin d'être changé est pour savoir si on doit utiliser *deep-copy-subject-mode*. Dans la plupart des cas, les administrateurs se concentreront sur la configuration des *security domains*.

Mode Deep Copy

Voir Section 12.3.2.1, « Mode de sujet Deep Copy » pour obtenir des détails supplémentaires sur le mode de sujet Deep Copy.

Domaine de sécurité

Un domaine de sécurité est un ensemble de configurations de sécurité déclarative *Java Authentication and Authorization Service (JAAS)* qu'une ou plusieurs applications utilisent pour contrôler l'authentification, l'autorisation, l'auditing et le mapping. Trois domaines de sécurité sont inclus par défaut : **jboss-ejb-policy**, **jboss-web-policy**, et **other**. Vous pouvez créer autant de domaines de sécurité que vous souhaitez pour accommoder les besoins de vos applications. Voir Section 13.9, « Utiliser un domaine de sécurité dans votre application » pour obtenir des détails sur le domaine de sécurité.

Rapporter un bogue

12.2. STRUCTURE DU SOUS-SYSTÈME DE SÉCURITÉ

Le sous-système de sécurité est configuré dans le domaine géré ou le fichier de configuration autonome. La plupart des éléments de configuration peuvent être configurés à l'aide de la Console de gestion via web ou de l'interface CLI sur console. Voici le code XML qui représente un exemple de sous-système de sécurité.

Exemple 12.1. Exemple de configuration de sous-système de sécurité

```
<subsystem xmlns="urn:jboss:domain:security:1.2">
 <security-management>
  ...
 </security-management>
 <security-domains>
        <security-domain name="other" cache-type="default">
            <authentication>
                <login-module code="Remoting" flag="optional">
                    <module-option name="password-stacking"
value="useFirstPass"/>
                </login-module>
                <login-module code="RealmUsersRoles" flag="required">
                    <module-option name="usersProperties"
value="${jboss.domain.config.dir}/application-users.properties"/>
                    <module-option name="rolesProperties"
```

```
value="${jboss.domain.config.dir}/application-roles.properties"/>
                    <module-option name="realm"
value="ApplicationRealm"/>
                    <module-option name="password-stacking"
value="useFirstPass"/>
                </login-module>
            </authentication>
        </security-domain>
        <security-domain name="jboss-web-policy" cache-type="default">
            <authorization>
                <policy-module code="Delegating" flag="required"/>
            </authorization>
        </security-domain>
        <security-domain name="jboss-ejb-policy" cache-type="default">
            <authorization>
                <policy-module code="Delegating" flag="required"/>
            </authorization>
        </security-domain>
    </security-domains>
    <vault>
      ...
    </vault>
</subsystem>
```

Les éléments **<security-management>**, **<subject-factory>** et **<security-properties>** ne sont pas présents dans la configuration par défaut. Les éléments **<subject-factory>** et **<security-properties>** ont été désapprouvés sur JBoss EAP 6.1 et versions ultérieures.

Rapporter un bogue

12.3. CONFIGURER LE SOUS-SYSTÈME DE SÉCURITÉ

12.3.1. Configurer le sous-système de sécurité

Vous pouvez configurer le sous-système de sécurité par l'interface CLI ou par la Console de gestion basée web.

Chaque élément de niveau supérieur du sous-système de sécurité contient des informations sur un aspect différent de la configuration de la sécurité. Voir Section 12.2, « Structure du sous-système de sécurité » pour obtenir un exemple de configuration de sous-système.

<security-management>

Cette section remplace les comportements de haut niveau du sous-système de sécurité. Chaque paramètre est optionnel. Il est rare de modifier ces paramètres sauf pour le mode de sujet Deep Copy.

Option	Description
deep-copy-subject-mode	Indique si l'on doit copier ou lier les tokens de sécurité pour la sécurité des threads.

Option	Description
authentication-manager-class-name	Indique un nom de classe d'implémentation AthentificationManager alternatif à utiliser.
authorization-manager-class-name	Indique un nom de classe d'implémentation AthorizationManager alternatif à utiliser.
audit-manager-class-name	Indique un nom de classe d'implémentation Audit Manager alternatif à utiliser.
identity-trust-manager-class-name	Indique un nom de classe d'implémentation IdentityTrustManager alternatif à utiliser.
mapping-manager-class-name	Indique le nom de classe d'implémentation MappingManager à utiliser.

<subject-factory>

La fabrique de sujets contrôle la création d'instances de sujets. Elle utilise sans doute le gestionnaire d'authentification pour vérifier l'appelant. L'utilisation principale du sujet est la création d'un sujet par les composants JCA. Il est rare que l'on ait besoin de modifier la fabrique de sujets.

<security-domains>

Un élément de conteneur qui contient plusieurs domaines de sécurité. Un domaine de sécurité peut contenir des informations sur des modules d'authentification, d'autorisation, de mappage, et d'auditing, ainsi qu'une autorisation JASPI et une configuration JSSE. Votre application indique ainsi un domaine de sécurité pour gérer ses informations de sécurité.

<security-properties>

Contient des noms et valeurs définis dans la classe java.security.Security

Rapporter un bogue

12.3.2. Gestion de la sécurité

12.3.2.1. Mode de sujet Deep Copy

Si le mode de sujet Deep Copy (*deep copy subject mode*) est désactivé (par défaut), copier une structure de données de sécurité fait référence à la structure d'origine, au lieu de copier toute la structure de données. Ce comportement est plus efficace, mais enclin à la corruption des données si plusieurs threads possédant la même identité effacent le sujet lors d'un vidage ou d'une déconnexion.

Le mode de sujet Deep Copy entraîne la copie totale de la structure des données et de toutes les données associées possibles, quand elles sont marquées « clonables ». C'est plus sûr au niveau thread, mais moins efficace.

Le mode de sujet Deep Copy est configuré dans le cadre du sous-système de sécurité.

Rapporter un bogue

12.3.2.2. Activer le Mode de sujet Deep Copy

Vous pouvez activer le mode de sécurité Deep Copy à partir de la console de gestion basée web ou de l'interface CLI.

Procédure 12.1. Activez le mode de sécurité Deep Copy à partir de la console de gestion

1. **Connectez-vous à la console de management.**
   Pour obtenir plus d'informations, voir la section intitulée *The Management Console* dans le guide *Administration and Configuration Guide* de la plateforme JBoss Enterprise Application 6.x qui se situe sur le portail clients à l'adresse suivante https://access.redhat.com/site/documentation/JBoss_Enterprise_Application_Platform/.

2. **Domaine géré : sélectionnez le profil qui convient.**
   Dans un domaine géré, le sous-système de sécurité est configuré par profil, ou vous pouvez activer ou désactiver le mode de sécurité Deep Copy de manière indépendante dans chaque profil.

   Pour sélectionner un profil, cliquez sur **`Configuration`** en haut et à droite de l'écran, puis sélectionnez un profil à partir du menu déroulant **`Profile`** en haut et à gauche.

3. **Ouvrir le menu de configuration `Security Subsystem`.**
   Étendre l'item de menu **`Security`**, puis cliquez sur le lien **`Security Subsystem`**.

4. **Activez le mode `Deep Copy Subject`.**
   Cliquez sur **`Edit`**. Cochez la case qui se trouve à côté de **`Deep Copy Subjects`** pour activer le mode Copier Sujet (copy subject).

Activez Deep Copy Subject Mode par la Console CLI

Si vous préférez utiliser l'interface CLI pour activer cette option, utilisez une des commandes suivantes.

Exemple 12.2. Domaine géré

```
/profile=full/subsystem=security/:write-attribute(name=deep-copy-
subject-mode,value=TRUE)
```

Exemple 12.3. Serveur autonome

```
/subsystem=security/:write-attribute(name=deep-copy-subject-
mode,value=TRUE)
```

Rapporter un bogue

12.3.3. Domaines de sécurité

12.3.3.1. Les domaines de sécurité

Les domaines de sécurité font partie du sous-système de sécurité JBoss EAP. La configuration de la sécurité est désormais gérée de façon centralisée, ou par le contrôleur de domaines d'un domaine géré ou par le serveur autonome.

Un domaine de sécurité se compose de configurations d'authentification, d'autorisation, de mappage de sécurité et d'audit. Il met en place la sécurité déclarative *Java Authentication and Authorization Service (JAAS)*.

L'authentification est impliquée dans la vérification de l'identité d'un utilisateur. Dans la terminologie de la sécurité, cet utilisateur est appelé un *principal*. Bien que l'authentification et l'autorisation soient différentes, de nombreux modules d'authentification intégrés gèrent également l'autorisation.

Une *authorization* est un process par lequel le serveur détermine si un utilisateur authentifié a la permission d'accéder à des privilèges ou ressources particulières dans le système ou opération.

Security mapping se rapporte à la possibilité d'ajouter, de modifier ou de supprimer des informations d'un principal, rôle ou attribut avant de passer les informations à votre application.

L'Auditing Manager vous permet de configurer les *provider modules* pour contrôler la façon dont les événements de sécurité sont rapportés.

Si vous utilisez des domaines de sécurité, vous pouvez supprimer toutes les configurations de sécurité spécifiques à votre application proprement dite. Cela permet de modifier les paramètres de sécurité de façon centralisée. Un scénario courant qui bénéficie de ce type de structure de configuration est le processus de déplacement des applications entre les environnements de test et de production.

Rapporter un bogue

12.3.3.2. Opérations de CLI liées aux domaines de sécurité

Exemple 12.4. flush-cache

Cette commande CLI supprime les entrées stockées dans le cache d'authentification d'un domaine de sécurité. Une entrée unique peut être vidée à l'aide de l'argument principal avec le nom d'utilisateur comme valeur. Si aucun argument n'est passé à l'opération, toutes les entrées seront vidées. Pour plus de détails sur cette opération, utilisez la commande CLI

```
/subsystem=security/security-domain=other:read-operation-
description(name=flush-cache)
```

Exemple 12.5. list-cached-principals

Cette commande CLI supprime les principaux stockés dans le cache d'authentification d'un domaine de sécurité. Pour plus de détails sur cette opération, utilisez la commande CLI

```
/subsystem=security/security-domain=other:read-operation-
description(name=list-cached-principals)
```

Rapporter un bogue

CHAPITRE 13. AUTHENTIFICATION ET AUTORISATION

13.1. INTÉGRATION KERBEROS ET SPNEGO

13.1.1. Intégration Kerberos et SPNEGO

Kerberos est une méthode d'authentification qui est conçue pour les environnements à réseau ouvert. Fonctionne sur la base d'un ticket et d'un authentificateur pour établir l'identité de l'utilisateur et du serveur. Il aide les deux nœuds à communiquer sur un environnement non sécurisé afin d'établir leur identité entre eux de manière sécurisée.

SPNEGO est une méthode d'authentification utilisée par une application de client pour s'authentifier sur le serveur. Cette technologie est utilisée lorsque l'application cliente et le serveur essaient de communiquer avec l'un l'autre mais ne sont pas sûrs du protocole d'authentification. SPNEGO détermine les mécanismes GSSAPI communs entre l'application cliente et le serveur et lui envoie ensuite toutes les opérations de sécurité.

Intégration Kerberos et SPNEGO

Dans une installation normale, l'utilisateur se connecte à un ordinateur de bureau qui est régi par le domaine Active Directory. L'utilisateur utilise ensuite le navigateur web, Firefox ou Internet Explorer pour accéder à une application web qui utilise la négociation de JBoss Negociation sur JBoss EAP. Le navigateur web transfère l'information de sign on de bureau à l'application web. JBoss EAP utilise les messages GSS d'arrière-plan avec Active Directory ou n'importe quel serveur Kerberos pour valider l'utilisateur. Cela permet à l'utilisateur d"effectuer un SSO sans faille dans l'application web

Rapporter un bogue

13.1.2. Desktop SSO using SPNEGO

Pour configurer un desktop qui utilise SPNEGO, configurer ce qui suit :

- Domaine de sécurité
- Propriétés système
- Application Web

Procédure 13.1. Configurer Desktop SSO utilisant SPNEGO

1. **Configurer Domaine de sécurité**
   Configurer les domaines de sécurité pour qu'il représentent l'identité du serveur et pour sécuriser l'application web.

   Exemple 13.1. Configuration du domaine de sécurité

   ```
   <security-domains>

       <security-domain name="host" cache-type="default">

         <authentication>

           <login-module code="Kerberos" flag="required">
   ```

```
          <module-option name="storeKey" value="true"/>

          <module-option name="useKeyTab" value="true"/>

          <module-option name="principal"
value="host/testserver@MY_REALM"/>

          <module-option name="keyTab"
value="/home/username/service.keytab"/>

          <module-option name="doNotPrompt" value="true"/>

          <module-option name="debug" value="false"/>

        </login-module>

       </authentication>

     </security-domain>



     <security-domain name="SPNEGO" cache-type="default">

       <authentication>

         <login-module code="SPNEGO"  flag="requisite">

           <module-option name="password-stacking"
value="useFirstPass"/>

           <module-option name="serverSecurityDomain"
value="host"/>

         </login-module>


         <!-- Login Module For Roles Search -->

       </security-domain>
```

2. **Configuration des propriétés système**
   Si nécessaire, les propriétés système peuvent être configurées dans le modèle de domaine.

Exemple 13.2. Configurer les propriétés système

```
<system-properties>

      <property name="java.security.krb5.kdc"
value="mykdc.mydomain"/>

      <property name="java.security.krb5.realm" value="MY_REALM"/>

    </system-properties>
```

3. **Configurer Application Web**
   Il n'est pas possible de remplacer les authentificateurs, mais il est possible d'ajouter le **`NegotiationAuthenticator`** comme valve à votre jboss-web.xml pour configurer l'application web.

NOTE

La valve a besoin d'avoir **`security-constraint`** et **`login-config`** définis dans le fichier web.xml car c'est utilisé pour décider quelles sources sont sécurisées. Cependant, l' **`auth-method`** est remplacée par cet authenticateur.

Exemple 13.3. Configurer Application Web

```
<!DOCTYPE jboss-web PUBLIC
 "-//JBoss//DTD Web Application 2.4//EN"
 "http://www.jboss.org/j2ee/dtd/jboss-web_4_0.dtd">

 <jboss-web>

   <security-domain>SPNEGO</security-domain>

   <valve>

     <class-
name>org.jboss.security.negotiation.NegotiationAuthenticator</clas
s-name>

   </valve>

 </jboss-web>
```

L'application web exige aussi qu'une dépendance soit définie dans **`META-INF/MANIFEST.MF`** pour trouver les classes de JBoss Negotiation.

Exemple 13.4. Définir la dépendance dans `META-INF/MANIFEST.MF`

```
   Manifest-Version: 1.0

   Build-Jdk: 1.6.0_24

   Dependencies: org.jboss.security.negotiation
```

Rapporter un bogue

13.1.3. Configurer JBoss Negociation sur un domaine Microsoft Windows.

Cette section décrit comment configurer les comptes requis par JBoss Negociation à utiliser quand JBoss EAP doit exécuter sur un serveur Microsoft Windows, faisant partie du domaine Active Directory.

Dans cette section, le nom d'hôte utilisé pour accéder au serveur dénommé **{hostname}**, **{realm}**, le domaine **{domain}** et le serveur hébergeant l'instance de JBoss EAP est dénommé **{machine_name}**.

Procédure 13.2. Configurer JBoss Negociation sur un domaine Microsoft Windows.

1. **Supprimer les mappages principaux du service existant**
   Dans un réseau Microsoft Windows, certains mappages sont créés automatiquement. Supprimez les mappages créés automatiquement pour mapper l'identité du serveur avec le service principal pour que la négociation se déroule correctement. Le mappage permet au navigateur web de l'ordinateur client de faire confiance au serveur et d'essayer SPNEGO. L'ordinateur client vérifie avec le contrôleur de domaine un mappage de la forme **HTTP {hostname}**

   Voici les étapes nécessaires pour supprimer les mappages existants :

   - Listez les mappages enregistrés dans le domaine de l'ordinateur qui utilise la commande **setspn -L {machine_name}**.
   - Supprimez les mappages existants par les commandes, **setspn -D HTTP/{hostname} {machine_name}** et **setspn -D host/{hostname} {machine_name}**.

2. Créez un compte d'utilisateur hôte.

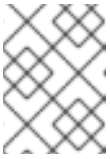
   NOTE

   Veillez à ce que le nom d'utilisateur hôte soit différent de **{machine_name}**.

   Dans le reste de la section, le nom d'utilisateur hôte est **{user_name}**.

3. **Définir le mappage entre {user_name} et {hostname}.**

   - Exécutez la commande suivante pour configurer le mappage pricipal du service, **ktpass -princ HTTP/{hostname}@{realm} -pass * -mapuser {domain}\{user_name}**.
   - Saisir le mot de passe du nom d'utilisateur quand vous y serez invité.

     NOTE

     Réinitialisez le mot de passe de l'utilisateur comme prérequis d'exportation du keytab.

   - Vérifiez le mappage en exécutant la commande suivante, **setspn -L {user_name}**

4. **Exportez le keytab de l'utilisateur dans le serveur sur lequel EAP est installé.**
   Exécutez la commande suivante pour exporter le keytab, **ktab -k service.keytab -a HTTP/{hostname}@{realm}**.

   NOTE

   Cette commande exporte le ticket duHTTP/{hostname} principal dans le fichier keytab **service.keytab**, qui est utilisé pour configurer le domaine de sécurité hôte sur JBoss.

5. Définir le principal dans le domaine de sécurité comme suit :

```
<module-option name="principal">HTTP/{hostname}@{realm}</module-
option>
```

Rapporter un bogue

13.1.4. Authentifiction Kerberos pour IDP PicketLink

Les sections suivantes expliquent comment mettre en place une authentification Kerberos dans IDP PicketLink.

Procédure 13.3. Installez JBoss EAP 6 et configurer Kerberos

1. Télécharger et installer JBoss EAP 6. Voir les instructions dans Installation Guide.
2. Que vous utilisiez Oracle Java ou IBM Java, utilisez JCE illimité. Sans JCE illimité, le serveur JBoss ne peut pas négocier sur un type de mécanisme SPNEGO qui convient (en utilisant 1.3.6.1.5.2.5, qui est **`GSS_IAKERB_MECHANISM`**).
3. Utilisez l'exemple ci-dessous pour configurer JBoss à utiliser la version Java qui vous convienne.

```
export JAVA_HOME=JDK/JRE_directory
```

Procédure 13.4. Tester votre installation Kerberos avec le JBoss Negotiation Toolkit

1. Utiliser le JBoss Negotiation Toolkit se trouvant Github
2. Modifier les fichiers de configuration et utiliser la commande **`mvn clean install`** pour créer le projet.
3. Copier le fichier **`jboss-negotiation-toolkit/target/jboss-negotiation-toolkit.war`** dans **`$JBOSS_HOME/standalone/deployments/`**.
4. Vérifier que les trois sections passent par le JBoss Negotiation Toolkit

Procédure 13.5. Configurer PicketLink IDP

Allez dans `idp.war`

L'exemple fourni utilise les archives **`idp.war`** et **`employee.war`**, qui se trouvent dans le référentiel PicketLink Quickstarts. Modifier les fichiers dans **`idp.war`** comme décrit ci-dessous.

1. Ajouter le module **`org.jboss.security.negotiation`** à **`$JBOSS_HOME/standalone/deployments/idp.war/META-INF/jboss-deployment-structure.xml`** car IDP utilise le module JBoss Negotiation.

```
<jboss-deployment-structure>
   <deployment>
   <!-- Add picketlink module dependency -->
     <dependencies>
        <module name="org.picketlink" />
        <module name="org.jboss.security.negotiation" />
```

```
      </dependencies>
    </deployment>
 </jboss-deployment-structure>
```

2. Ajouter une valve supplémentaire **`org.jboss.security.negotiation.NegotiationAuthenticator`** à SPNEGO pour **`$JBOSS_HOME/standalone/deployments/idp.war/WEB-INF/jboss-web.xml`**.

3. Modifier **`security-domain`** de **`idp`** vers **`SPNEGO`** dans **`$JBOSS_HOME/standalone/deployments/idp.war/WEB-INF/jboss-web.xml`** comme suit :

```
<jboss-web>
   <security-domain>SPNEGO</security-domain>
   <context-root>idp</context-root>
     <valve>
       <class-
name>org.picketlink.identity.federation.bindings.tomcat.idp.IDPWebBr
owserSSOValve</class-name>
       <param>
         <param-name>passUserPrincipalToAttributeManager</param-
name>
         <param-value>true</param-value>
       </param>
     </valve>
     <valve>
       <class-
name>org.jboss.security.negotiation.NegotiationAuthenticator</class-
name>
     </valve>
</jboss-web>
```

4. Ajouter et modifier le security-role ajouté à votre principal par l'installatiion du serveur Kerberos en **`$JBOSS_HOME/standalone/deployments/idp.war/WEB-INF/web.xml`**.

5. Modifier le fichier **`$JBOSS_HOME/standalone/deployments/idp.war/WEB-INF/picketlink.xml`** comme suit :

```
<PicketLink xmlns="urn:picketlink:identity-federation:config:2.1">
  <PicketLinkIDP xmlns="urn:picketlink:identity-
federation:config:2.1">
  <IdentityURL>${idp.url::http://localhost:8080/idp/}</IdentityURL>
    <Trust>
     <Domains>redhat.com,localhost,amazonaws.com</Domains>
    </Trust>
  </PicketLinkIDP>
  <Handlers xmlns="urn:picketlink:identity-
federation:handler:config:2.1">
      <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2Is
suerTrustHandler" />
      <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2Lo
gOutHandler" />
      <Handler
```

```
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2Au
thenticationHandler" />
      <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.RolesGe
nerationHandler" />
  </Handlers>
  <!-- The configuration bellow defines a token timeout and a clock
skew. Both  configurations will be used during the SAML Assertion
creation. This configuration is optional. It is defined only to show
you how to set the token timeout and clock skew configuration. -->
  <PicketLinkSTS xmlns="urn:picketlink:identity-
federation:config:1.0" TokenTimeout="5000" ClockSkew="0">
    <TokenProviders>
    <TokenProvider

ProviderClass="org.picketlink.identity.federation.core.saml.v1.provi
ders.SAML11AssertionTokenProvider"
       TokenType="urn:oasis:names:tc:SAML:1.0:assertion"
       TokenElement="Assertion"
TokenElementNS="urn:oasis:names:tc:SAML:1.0:assertion" />
    <TokenProvider

ProviderClass="org.picketlink.identity.federation.core.saml.v2.provi
ders.SAML20AssertionTokenProvider"
       TokenType="urn:oasis:names:tc:SAML:2.0:assertion"
       TokenElement="Assertion"
TokenElementNS="urn:oasis:names:tc:SAML:2.0:assertion" />
    </TokenProviders>
  </PicketLinkSTS>
</PicketLink>
```

6. Modifier **`IdentityURL`** pour qu'il corresponde au nom d'hôte du serveur sur lequel vous exécutez IDP.

7. Modifier **`Trust`** pour qu'il contienne les noms de domaine autorisés par le fournisseur d'identité.

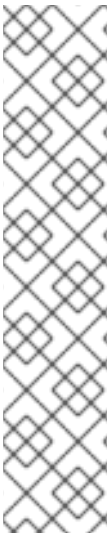
8. Modifiez le **`employee.war`**. Ajoutez et modifiez le security-role ajouté à votre principal par l'installatiion du serveur Kerberos en **`$JBOSS_HOME/standalone/deployments/idp.war/WEB-INF/web.xml`**.

9. Modifiez la configuration **`security domain`** dans le fichier **`$JBOSS_HOME/standalone/configuration/standalone.xml`**. La configuration de mappage des rôles est la même que celle qui existe dans les configurations de domaine de sécurité.

```
<security-domain name="host" cache-type="default">
  <authentication>
    <login-module code="Kerberos" flag="required">
      <module-option name="principal"
value="HTTP/something.com@yourdomain.COM"/>       <module-option
name="storeKey" value="true"/>
      <module-option name="useKeyTab" value="true"/>
      <module-option name="doNotPrompt" value="true"/>
      <module-option name="keyTab" value="/root/keytab"/>
    </login-module>
  </authentication>
```

```
</security-domain>
<security-domain name="SPNEGO" cache-type="default">
   <authentication>
     <login-module code="SPNEGO" flag="required">
        <module-option name="serverSecurityDomain" value="host"/>
     </login-module>
   </authentication>
</security-domain>
<security-domain name="sp" cache-type="default">
   <authentication>
     <login-module

code="org.picketlink.identity.federation.bindings.jboss.auth.SAML2Lo
ginModule"
      flag="required" />
   </authentication>
</security-domain>
```



NOTE

Si vous utilisez IBM JDK, les options de module Kerberos seront différentes. Vous devrez définir la propriété système **`jboss.security.disable.secdomain.option`** à **`true`**. Voir Section 13.2.2, « Configurer l'authentification dans un domaine de sécurité » pour obtenir des détails supplémentaires. Mettre le module de connexion à jour ainsi :

```
<login-module code="Kerberos" flag="required">
 <module-option name="principal"
value="HTTP/something.com@yourdomain.COM"/>
 <module-option name="credsType" value="acceptor"/>
 <module-option name="useKeytab" value="file:///root/keytab"/>
</login-module>
```

Procédure 13.6. Vérifiez la configuration d'authentification Kerberos dans PicketLink IDP

1. Démarrez le serveur JBoss EAP par la commande **`$JBOSS_HOME/bin/standalone.sh`**.
2. Configurez votre système Kerberos. Il y a plusieurs façons de procéder, comme en utilisant l'une des options suivantes :
   - FreeIPA: il y a plusieurs options de configuration. Vous devrez en choisir une qui convient à votre installation.
   - ApacheDS
3. Configurez votre navigateur, comme Firefox, pour qu'il utilise Kerberos. Suivre les instructions fournies ici : https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/5/html/Deployment_Guide/sso-config-firefox.html
4. Vérifiez de pouvoir accéder à **`http://YOUR_DOMAIN:8080/employee`** à partir de Firefox configuré comme mentionné ci-dessus.

Rapporter un bogue

13.1.5. Se connecter par un certificat avec PicketLink IDP

Configurer IDP pour prendre en charge SSL

Vous pouvez configurer IDP PicketLink pour prendre en charge SSL. La procédure suivante est un exemple qui montre comment configurer une application web en tant qu'IDP supportant l'authentification client SSL. Il y a deux façons de configurer l'IDP pour authentifier les utilisateurs :

- Si SSL est utilisé, le serveur demandera au client un certificat et utilisera ce certificat pour authentifier l'utilisateur.
- Si aucun certificat n'est fourni par le client, une authentification basée formulaire sera effectuée.

Rapporter un bogue

13.1.5.1. Configuration SSL JBoss EAP 6.3

La première chose à faire est de créer les certificats - les keystores et les truststores qui seront utilisés pendant toute la procédure de configuration.

Procédure 13.7. Créez le certificat, le keystore et le truststore pour votre serveur.

1. **Créez un certificat pour votre serveur.**
   Utilisez la commande suivante pour créer un certificat pour votre serveur :

   ```
   keytool -genkey -alias server -keyalg RSA -keystore server.keystore
   -storepass change_it -validity 365
   ```

   Le système vous demande des informations supplémentaires. Vous devez fournir les valeurs selon les besoins. Le nom CN du certificat doit être le même que le nom de votre serveur DNS. Par exemple, dans le cas de localhost, vous pouvez utiliser la commande suivante :

   ```
   keytool -genkey -alias server -keystore server.keystore -storepass
   change_it -keypass password -dname
   "CN=localhost,OU=QE,O=example.com,L=Brno,C=CZ"
   ```

2. **Créez un certificat de client**
   Vous utiliserez ce certificat de client pour vous authentifier auprès du serveur quand vous accéderez à une ressource via SSL.

   ```
   keytool -genkey -alias client -keystore client.keystore -storepass
   change_it -validity 365 -keyalg RSA -keysize 2048 -storetype pkcs12
   ```

3. **Créez le truststore**
   Exportez le certificat du client et créer un truststore en important ce certificat :

   ```
   keytool -exportcert -keystore client.keystore -storetype pkcs12 -
   storepass change_it -alias client -keypass change_it -file
   client.keystore
   keytool -import -file client.keystore -alias client -keystore
   client.truststore
   ```

4. **Modifiez l'installation du serveur JBoss EAP 6.3 pour activer SSL**
   Ajoutez le connecteur suivant au sous-système web pour activer SSL :

```
<connector name="https" protocol="HTTP/1.1" scheme="https" socket-
binding="https" enable-lookups="false" secure="true">
   <ssl name="localhost-ssl" key-alias="server" password="change_it"
   certificate-key-file="${jboss.server.config.dir}/server.keystore"
   protocol="TLSv1"
   verify-client="want"
   ca-certificate-
file="${jboss.server.config.dir}/client.truststore"/>
</connector>
```

5. **Relancez le serveur**
   Redémarrez le serveur et vérifiez qu'il répond sur : https://localhost:8443

6. **Acceptation du certificat**
   On vous invitera à approuver le certificat du serveur.

Configurez le certificat de client dans votre navigateur

Avant d'accéder à l'application, vous devez importer le **`client.keystore`** dans votre navigateur. Ce fichier contient le certificat client. Quand vous accédez à l'application, le navigateur vous invite à sélectionner le certificat dont vous avez besoin pour vous authentifier auprès du serveur. Sélectionnez le certificat désiré.

Configuration du domaine de sécurité

Ajoutez le domaine de sécurité à votre installation de serveur. Si vous êtes en mode autonome, vous devrez l'ajouter au **`JBOSS_HOME/standalone/configuration/standalone.xml`** :

```
 <security-domain name="idp" cache-type="default">
   <authentication>
   <login-module code="CertificateRoles" flag="optional">
   <module-option name="password-stacking" value="useFirstPass"/>
   <module-option name="securityDomain" value="idp"/>
   <module-option name="verifier"
value="org.jboss.security.auth.certs.AnyCertVerifier"/>
   </login-module>

   <login-module
code="org.picketlink.identity.federation.bindings.jboss.auth.RegExUserName
LoginModule" flag="optional">
   <module-option name="regex" value="CN=(.*?),"/>
   </login-module>

   <login-module code="UsersRoles" flag="required">
   <module-option name="password-stacking" value="useFirstPass"/>
   <module-option name="usersProperties" value="users.properties"/>
   <module-option name="rolesProperties" value="roles.properties"/>
   </login-module>
   </authentication>

   <jsse keystore-password="change_it" keystore-
url="${jboss.server.config.dir}" truststore-password="change_it"
truststore-url="${jboss.server.config.dir}" client-auth="true"/>
</security-domain>
```

L'exemple de configuration ci-dessus validifie tout certificat déjà fourni. Si aucun certificat n'est produit ou si l'authentification vient à échouer, la procédure revient à nouveau à l'authentification basée utilisateur/mot de passe.

Module de connexion de nom d'utilisation d'expression régulière

Le module de connexion de nom d'utilisation d'expression régulière peut être utilisé à la suite des modules de connexion de certificats pour extraire un nom d'utilisateur, un UID ou un autre champ du nom de principal pour que les rôles puissent être obtenus à partir de LDAP. Le module contient une option nommée **`regex`** qui spécifie l'expression régulière à appliquer au nom du principal. Le résultat est passé au module de connexion suivant.

Dans cet exemple, un nom de principal d'entrée **`UID=007, EMAILADDRESS=something@something, CN=James Bond, O=SpyAgency`** aura pour résultat **`UID=007`**.

Exemple 13.5. Exemple de module de connexion de nom d'utilisation d'expression régulière

```
<login-module
code="org.picketlink.identity.federation.bindings.jboss.auth.RegExUserNa
meLoginModule" flag="required">
  <module-option name="password-stacking" value="useFirstPass"/>
  <module-option name="regex" value="UID=(.*?),"/>
</login-module>
```

Pour obtenir des informations supplémentaires sur les expressions régulières, voir la documentation sur les classes **`java.util.regex.Pattern`** à l'adresse suivante http://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html.

Rapporter un bogue

13.2. AUTHENTIFICATION

13.2.1. Authentification

L'authentification consiste à identifier un sujet et à vérifier l'authenticité de l'identification. Le mécanisme d'authentification le plus commun est une combinaison de nom d'utilisateur et de mot de passe. D'autres mécanismes d'authentification communs utilisent des clés partagées, des smart cards (cartes à puce) ou des empreintes digitales. Le résultat d'une authentification réussie s'appelle un Principal, en termes de sécurité déclarative Java Enterprise Edition.

JBoss EAP utilise un système pouvant se connecter à des modules d'authentification en vue de flexibilité et d'intégration avec les systèmes d'authentification que vous utilisez dans votre organisation. Chaque domaine de sécurité contient un ou plusieurs modules d'authentification configurés. Chaque module comprend des paramètres de configuration supplémentaires pour personnaliser son comportement. Le plus simple consiste à configurer le sous-système d'authentification dans la console de gestion web.

L'authentification n'est pas la même chose que l'autorisation, même si elles sont souvent liées. Bon nombre des modules d'authentification intégrés peuvent également gérer des autorisations.

Rapporter un bogue

13.2.2. Configurer l'authentification dans un domaine de sécurité

Pour configurer les paramètres d'authentification d'un domaine de sécurité, vous connecter dans la console de gestion et suivre la procédure suivante :

Procédure 13.8. Configurez l'authentification dans un domaine de sécurité

1. **Ouvrir l'affichage détaillé du domaine de sécurité.**
   a. Cliquez sur l'étiquette **`Configuration`** qui se trouve en haut de la console de gestion.
   b. Sélectionnez le profil à modifier à partir du menu de sélection **`Profile`** qui se trouve en haut et à gauche de la vue 'Profile'.
   c. Étendre l'item de menu **`Security`**, puis sélectionnez **`Security Domains`**.
   d. Cliquez sur le lien **`View`** pour obtenir le domaine de sécurité que vous voulez modifier.
2. **Naviguez dans la configuration du sous-système d'authentification.**
   Sélectionnez l'étiquette **`Authentification`** en haut de l'affichage si elle n'a pas déjà été sélectionnée.

   La zone de configuration est divisée en deux : **`Login Modules`** et **`Details`**. Le module de connexion est l'unité de base de configuration. Un domaine de sécurité peut inclure plusieurs polices d'autorisation, et chacune peut inclure plusieurs attributs et options.
3. **Ajoutez un module de connexion**
   Cliquez sur le bouton **`Add`** pour ajouter un module d'authentification JAAS. Remplissez les informations pour votre module.

   Le **`Code`** est le nom de classe de votre module. Les **`Flags`** controlent la façon dont le module est lié à d'autres modules de polices d'authentification du même domaine de sécurité.

   Explication des indicateurs

   La spécification Java Enterprise Edition 6 fournit l'explication suivante sur les marqueurs des modules de sécurité. La liste suivante provient de http://docs.oracle.com/javase/6/docs/technotes/guides/security/jaas/JAASRefGuide.html#Appendix Consultez ce document pour obtenir des informations plus détaillées.

Marqueur	Détails
requis	Le LoginModule est requis pour la réussite. En cas de réussite ou d'échec, l'autorisation continuera son chemin dans la liste du LoginModule.
pré-requis	Le LoginModule est requis pour la réussite. En cas de succès, l'authentification continue vers le bas de la liste LoginModule. En cas d'échec, le contrôle est renvoyé immédiatement à l'application (l'authentification ne continue pas son chemin le long de la liste LoginModule).

Marqueur	Détails
suffisant	Le LoginModule n'est pas nécessaire pour aboutir. S'il ne réussit pas, le contrôle retourne immédiatement à l'application (l'authentification ne se déroule pas vers le bas de la liste LoginModule). S'il échoue, l'authentification se poursuit vers le bas de la liste LoginModule.
option	Le LoginModule n'est pas requis pour la réussite. En cas de réussite ou d'échec, l'autorisation continuera son chemin dans la liste du LoginModule.

4. **Modifiez les paramètres d'authentification**
   Une fois que vous aurez ajouté votre module, vous pourrez modifier son **`Code`** ou ses **`Flags`** (indicateurs) en cliquant **`Edit`** dans la section **`Details`** de l'écran. Veillez à ce que l'onglet **`Attributes`** soit bien sélectionné.

5. **Option : ajouter ou supprimer un module**
   Pour ajouter des options à votre module, cliquez sur leurs entrées dans la liste **`Login Modules`**, et sélectionnez l'onglet **`Module Options`** dans la section **`Details`** qui se trouve en bas de la page. Cliquez sur le bouton **`Add`**, et fournir la clé et la valeur de l'option. Utilisez le bouton **`Remove`** pour supprimer l'option.

Résultat

Votre module d'authentification est ajouté au domaine de sécurité, et est rendu disponible immédiatement auprès des applications qui utilisent le domaine de sécurité.

L'option de module `jboss.security.security_domain`

Par défaut, chaque module de connexion défini dans un domaine de sécurité a l'option de module **`jboss.security.security_domain`** ajoutée automatiquement. Cette option cause des problèmes avec les modules de connexion, qui vérifient que seules les options connues soient définies. Le module de connexion Kerberos d'IBM **`com.ibm.security.auth.module.Krb5LoginModule`** est l'un d'entre eux.

Vous pouvez désactiver le comportement d'ajout de cette option de module, en définissant la propriété de système sur **`true`** en démarrant la plate-forme JBoss EAP 6. Ajoutez ce qui suit pour démarrer les paramètres.

```
-Djboss.security.disable.secdomain.option=true
```

Vous pourrez également définir cette propriété par la console de gestion. Dans un serveur autonome, vous pouvez définir les propriétés de système dans la section **`Profile`** de configuration. Dans un domaine géré, vous pouvez définir les propriétés du système pour chaque groupe de serveur.

Rapporter un bogue

13.3. JAVA AUTHENTICATION ET AUTHORIZATION SERVICE (JAAS)

13.3.1. JAAS

JAAS est le service d'autorisation et d'authentification de Java. Il fait partie de la Spec Java EE, et permet à l'authentification et à l'autorisation d'extraire des application des fournisseurs de sécurité.

L'API JAAS 1.0 se compose d'un ensemble de packages Java conçus pour l'autorisation et l'authentification des utilisateurs. L'API implémente une version Java du cadre standard Pluggable Authentication Modules (PAM) et s'étend à l'architecture de contrôle d'accès Java 2 Platform pour approuver l'autorisation basée utilisateur.

L'authentification JAAS est effectuée de manière enfichable. Cela permet de rester indépendant des technologies sous-jacentes d'authentification des applications Java et permet au gestionnaire de sécurité de travailler dans des infrastructures de sécurité différentes. L'intégration dans une infrastructure de sécurité est réalisable sans modifier l'implémentation de gestionnaire de sécurité. Vous devez uniquement modifier la configuration de la pile d'authentification que JAAS utilise.

Voir la documentation Java EE JAAS Documentation pour obtenir des informations supplémentaires sur JAAS.

Le sous-système de sécurité JBoss Enterprise Application Platform 6 est basé sur l'API JAAS.

Rapporter un bogue

13.3.2. Classes JAAS principales

Les classes de base JAAS peuvent être décomposés en trois catégories : communes, d'authentification et d'autorisation. La liste suivante présente uniquement les classes d'authentification et communes, parce que ce sont les classes particulières utilisées pour implémenter la fonctionnalité du sous-système de sécurité EAP couverte dans ce chapitre.

Voici donc les classes communes :

- **`Subject`** (**`javax.security.auth.Subject`**)

Voici les classes d'authentification :

- **`Configuration`** (**`javax.security.auth.login.Configuration`**)
- **`LoginContext`** (**`javax.security.auth.login.LoginContext`**)

Voici les interfaces associées :

- **`Principal`** (**`java.security.Principal`**)
- **`Callback`** (**`javax.security.auth.callback.Callback`**)
- **`CallbackHandler`** (**`javax.security.auth.callback.CallbackHandler`**)
- **`LoginModule`** (**`javax.security.auth.spi.LoginModule`**)

Rapporter un bogue

13.3.3. Classes Subject et Principal

Pour autoriser l'accès aux ressources, les applications doivent tout d'abord authentifier les sources de la requête. Le framework JAAS définit le terme subject pour représenter une source de demande. La classe **`Subject`** est la classe centrale dans JAAS. Un **`Subject`** représente les informations d'une seule entité, comme une personne ou un service. Elle englobe les principaux de l'entité, les informations

d'authentification publiques et privées. L'API JAAS utilise l'interface Java 2 **`java.security.Principal`** pour représenter un principal, qui est essentiellement un nom dactylographié.

Durant le processus d'authentification, un sujet est rempli d'identités associées ou de principaux. Un sujet peut avoir de nombreux principaux. Par exemple, une personne peut avoir un nom principal (John Doe), un numéro de sécurité sociale principal (123-45-6789) et un principal de nom d'utilisateur (johnd), tous aidant à distinguer le sujet d'autres sujets. Pour récupérer les principaux de sécurité associées à un sujet, il existe deux méthodes :

```
public Set getPrincipals() {...}
public Set getPrincipals(Class c) {...}
```

`getPrincipals()` renvoie tous les principaux contenus dans le subject. **`getPrincipals(Class c)`** renvoie uniquement les principaux qui sont des instances de la classe **`c`** ou d'une de ses sous-classes. Un ensemble vide sera renvoyé si le sujet n'a pas de principal correspondant.

Notez que l'interface **`java.security.acl.Group`** est une sous-interface de **`java.security.Principal`**, donc une instance de l'ensemble des principaux peut représenter un groupement logique d'autres principaux ou groupes de principaux.

Rapporter un bogue

13.3.4. Authentification du Subject

L'authentification de Subject exige une connexion JAAS. Pour obtenir un fichier de configuration de connexion JAAS, voir JAAS Login Configuration File dans la documentation Java.

Le processus de connexion consiste aux étapes suivantes :

1. Une application instancie un **`LoginContext`** et passe le nom de la configuration de la connexion et un **`CallbackHandler`** pour remplir les objets **`Callback`**, selon la configuration **`LoginModule`**.

2. Le **`LoginContext`** consulte une **`Configuration`** pour charger tous les **`LoginModules`** inclus dans la configuration de connexion nommée. Si aucune configuration nommée existe, la configuration **`other`** sera utilisée par défaut.

3. L'application invoque la méthode **`LoginContext.login`**.

4. La méthode login appelle chaque **`LoginModule`** téléchargé. Comme chaque **`LoginModule`** tente d'authentifier l'objet, il appelle la méthode handle sur le **`CallbackHandler`** associé pour obtenir les informations requises par le processus d'authentification. L'information requise est passée à la méthode handle sous la forme d'un tableau d'objets **`Callback`**. En cas de réussite, le **`LoginModule`** associe les principaux et les informations d'identification pertinents avec le subject.

5. Le **`LoginContext`** renvoie le statut d'authentification à l'application. Le renvoi par la méthode de connexion indique un succès. L'échec se traduit par une LoginException lancée par la méthode de connexion.

6. Si l'authentification réussit, l'application extrait le Subject authentifié par la méthode **`LoginContext.getSubject`**.

7. Une fois l'étendue de l'authentification de l'objet terminée, tous les principaux et informations connexes associées au subject par la méthode de **`connexion`** s'éliminent par la méthode **`LoginContext.logout`**.

La classe **`LoginContext`** fournit les méthodes de base d'authentification des subjects et offre un moyen de développer une application qui est indépendante de la technologie sous-jacente de l'authentification. Le **`LoginContext`** consulte une **`Configuration`** pour déterminer les services d'authentification configurés pour une application particulière. Les classes **`LoginModule`** représentent les services d'authentification. Par conséquent, vous pouvez brancher des modules de connexion différents dans une application sans modifier l'application. Le code suivant illustre les étapes requises par une application pour authentifier un subject.

```
CallbackHandler handler = new MyHandler();
LoginContext lc = new LoginContext("some-config", handler);

try {
    lc.login();
    Subject subject = lc.getSubject();
} catch(LoginException e) {
    System.out.println("authentication failed");
    e.printStackTrace();
}

// Perform work as authenticated Subject
// ...

// Scope of work complete, logout to remove authentication info
try {
    lc.logout();
} catch(LoginException e) {
    System.out.println("logout failed");
    e.printStackTrace();
}

// A sample MyHandler class
class MyHandler
    implements CallbackHandler
{
    public void handle(Callback[] callbacks) throws
        IOException, UnsupportedCallbackException
    {
        for (int i = 0; i < callbacks.length; i++) {
            if (callbacks[i] instanceof NameCallback) {
                NameCallback nc = (NameCallback)callbacks[i];
                nc.setName(username);
            } else if (callbacks[i] instanceof PasswordCallback) {
                PasswordCallback pc = (PasswordCallback)callbacks[i];
                pc.setPassword(password);
            } else {
                throw new UnsupportedCallbackException(callbacks[i],
                                                       "Unrecognized
Callback");
            }
        }
    }
}
```

Les développeurs intègrent une technologie d'authentification en créant une implémentation de l'interface **`LoginModule`**. Cela permet à un administrateur d'ajouter des technologies d'authentification différentes dans une application. Vous pouvez enchaîner ensemble de multiples **`LoginModule`** afin de permettre à plus d'une technologie d'authentification de participer au processus d'authentification. Par exemple, un **`LoginModule`** peut effectuer une authentification par nom/mot de passe, tandis que l'autre peut créer une interface avec des périphériques matériels tels que des lecteurs de cartes ou des identificateurs biométriques.

Le cycle de vie d'un **`LoginModule`** est motivé par l'objet **`LoginContext`** avec lequel le client crée et publie la méthode login. Le processus se compose de deux phases. Les étapes du processus sont les suivantes :

- Le **`LoginContext`** crée chaque **`LoginModule`** en utilisant son constructeur public no-arg.
- Chaque **`LoginModule`** est initialisé par un appel à sa méthode d'initialisation. L'argument **`Subject`** est garanti d'être non-null. La signature de la méthode d'initialisation est : **`public void initialize(Subject subject, CallbackHandler callbackHandler, Map sharedState, Map options)`**
- La méthode **`login`** est appelée pour démarrer le processus d'authentification. Par exemple, une implémentation de la méthode peut inviter l'utilisateur à un nom d'utilisateur et à un mot de passe, et puis vérifiez les informations sur les données stockées dans un service d'attribution de noms comme NIS ou LDAP. Des implémentations alternatives pourraient créer des interfaces avec les cartes à puce et des dispositifs biométriques, ou simplement extraire les informations de l'utilisateur du système d'exploitation sous-jacent. La validation de l'identité de l'utilisateur par chaque **`LoginModule`** est considérée comme la phase 1 d'authentification JAAS. La signature de la méthode **`login`** est **`boolean login() throws LoginException`**. Une **`LoginException`** indique un échec. Une valeur retour true indique que la méthode a réussi, tandis qu'une valeur false indique que le module de login doit être ignoré.
- Si la méthode d'authentification générale de **`LoginContext`** réussit, **`commit`** sera invoqué sur chaque **`LoginModule`**. Si la phase 1 réussit pour un module **`LoginModule`**, alors la méthode de validation continue en phase 2 et associe les principaux qui conviennent, les informations d'authentification publiques, et/ou les informations d'authentification privées au sujet. Si la phase 1 échoue pour **`LoginModule`**, alors **`commit`** retirera tout état d'authentification stocké auparavant, comme les noms d'utilisateur et les mots de passe. La signature de la méthode **`commit`** est : **`boolean commit() throws LoginException`**. Tout manquement de la phase de validation (commit) se traduit par l'exception **`LoginException`**. Un renvoi true indique que la méthode a réussi, alors qu'un renvoi false indique que le module de connexion doit être ignoré.
- Si l'authentification globale du **`LoginContext`** échoue, la méthode **`abort`** sera appelée sur chaque **`LoginModule`**. La méthode **`abort`** supprime ou détruit tout état d'authentification créé par les méthodes de connexion ou d'initialisation. La signature de la méthode **`abort`** est **`boolean abort() throws LoginException`**. En cas d'impossibilité d'achever la phase **`d'abandon`** est indiquée en jetant une **`LoginException`**. Un retour de true indique que la méthode a réussi, alors qu'un retour de false indique que le module de connexion doit être ignoré.
- Pour supprimer l'état d'authentification après une connexion réussie, l'application appelle **`logout`** sur le **`LoginContext`**. Il en résulte un appel de méthode **`logout`** sur chaque **`LoginModule`**. La méthode **`logout`** supprime les principaux et les informations d'identification associées initialement au sujet au cours de l'opération **`commit`** (validation). Les informations d'identification doivent être détruites après le retrait. La signature de la méthode **`logout`** est :

`boolean logout() throws LoginException`. L'impossibilité d'achever le processus de déconnexion est indiquée par l'exception **`LoginException`**. Un renvoi true indique que la méthode a réussi, alors qu'un renvoi false indique que le module de connexion doit être ignoré.

Lorsqu'un **`LoginModule`** doit communiquer avec l'utilisateur pour obtenir des informations d'authentification, il utilise un objet **`CallbackHandler`**. Les applications implémentent l'interface CallbackHandler et le font passer au **`LoginContext`**, lequel envoie les informations d'authentification directement sur les modules de connexion sous-jacents.

Les modules de connexion utilisent le **`CallbackHandler`** à la fois pour recueillir les entrées d'utilisateurs, comme un mot de passe ou un PIN de smartcard, afin de fournir des informations aux utilisateurs, comme les informations d'état. En permettant à l'application de spécifier les **`CallbackHandler`**, les **`LoginModule`** sous-jacents restent indépendants des différentes façons dont les applications interagissent avec les utilisateurs. Par exemple, une implémentation de **`CallbackHandler`** d'une application GUI peut afficher une fenêtre pour solliciter une action d'utilisateur. En revanche, une implémentation **`CallbackHandler`** pour un environnement non graphique, comme un serveur d'applications, pourrait simplement obtenir des informations d'identification en utilisant un serveur d'application API. L'interface CallbackHandler dispose d'une méthode d'implémentation :

```
void handle(Callback[] callbacks)
    throws java.io.IOException,
           UnsupportedCallbackException;
```

L'interface de **`Callback`** est la dernière classe d'authentification que nous allons regarder. Il s'agit d'une interface de marquage pour laquelle plusieurs implémentations par défaut sont fournies, y compris le **`NameCallback`** et le **`PasswordCallback`** utilisés dans un exemple précédent. Un **`LoginModule`** utilise un **`Callback`** pour demander des informations requises par le mécanisme d'authentification. Les **`LoginModule`** passent un tableau de **`Callback`** directement à la méthode **`CallbackHandler.handle`** au cours de la phase d'ouverture de session d'authentification. Si un **`callbackhandler`** ne comprend pas comment utiliser un objet **`Callback`** passé dans la méthode handle, il lancera une exception **`UnsupportedCallbackException`** pour annuler l'appel d'ouverture de session.

Rapporter un bogue

13.4. JAVA AUTHENTICATION SPI FOR CONTAINERS (JASPI)

13.4.1. Sécurité Java Authentication SPI pour Conteneurs (JASPI)

Java Authentication SPI pour Conteneurs (JASPI or JASPIC) est une interface enfichable pour applications JSR-196 du Java Community Process. Consulter http://www.jcp.org/en/jsr/detail?id=196 pour obtenir des informations sur la spécification.

Rapporter un bogue

13.4.2. Configuration de la sécurité Java Authentication SPI pour conteneurs (JASPI)

Pour s'authentifier auprès d'un fournisseur JASPI, ajouter un élément **`<authentication-jaspi>`** à votre domaine de sécurité. La configuration est similaire à celle d'un module d'authentification standard, mais les éléments de module de login sont inclus dans l'élément **`<login-module-stack>`**. La structure de configuration est la suivante :

Exemple 13.6. Structure de l'élément `authentication-jaspi`

```
<authentication-jaspi>
 <login-module-stack name="...">
   <login-module code="..." flag="...">
     <module-option name="..." value="..."/>
   </login-module>
 </login-module-stack>
 <auth-module code="..." login-module-stack-ref="...">
   <module-option name="..." value="..."/>
 </auth-module>
</authentication-jaspi>
```

Le module de connexion est lui-même configuré de la même façon que le module d'authentification standard.

Comme la console de gestion basée web n'expose pas la configuration des modules d'authentification JASPI, vous devez stopper la plateforme JBoss EAP 6 complètement avant d'ajouter la configuration directement dans le fichier ***EAP_HOME*/domain/configuration/domain.xml** ou dans le fichier ***EAP_HOME*/standalone/configuration/standalone.xml**.

Rapporter un bogue

13.5. AUTORISATION

13.5.1. L'autorisation

L'autorisation est un mécanisme d'octroi ou de refus de permission d'accéder à une ressource basé sur l'identité. Elle est implémentée sous forme de roles de sécurité déclarative, qui peuvent être ajoutés aux principaux.

JBoss EAP utilise un système modulaire pour configurer l'autorisation. Chaque domaine de sécurité peut contenir une ou plusieurs stratégies d'autorisation. Chaque stratégie possède un module de base qui définit son comportement. Il est configuré via les attributs et indicateurs spécifiques. La façon la plus simple consiste à configurer le sous-système de l'autorisation à l'aide de la console de gestion basée web.

L'authentification est différente de l'autorisation, et a lieu, en général, après l'authentification. La plupart des modules d'authentification gèrent également l'autorisation.

Rapporter un bogue

13.5.2. Configurer l'autorisation pour un domaine de sécurité

Pour configurer les paramètres d'un domaine de sécurité, connectez-vous à la console de gestion et suivre la procédure suivante :

Procédure 13.9. Configurez l'autorisation pour un domaine de sécurité

1. **Ouvrir l'affichage détaillé du domaine de sécurité.**
   a. Cliquez sur l'étiquette **Configuration** qui se trouve en haut de la console de gestion.

b. Sélectionnez le profil à modifier à partir du menu déroulant **`Profile`** en haut et à gauche.

c. Étendre l'item de menu **`Security`**, puis sélectionnez **`Security Domains`**.

d. Cliquez sur le lien **`View`** pour obtenir le domaine de sécurité que vous voulez modifier.

2. **Naviguez dans la configuration du sous-système d'autorisation.**
Sélectionnez l'étiquette **`Authorisation`** en haut de l'écran.

   La zone de configuration est divisée en deux: **`Policies`** et **`Details`**. Le module de connexion est l'unité de base de configuration. Un domaine de sécurité peut inclure plusieurs polices d'autorisation, et chacune d'elle peut inclure plusieurs attributs ou options.

3. **Ajoutez une police.**
Cliquez sur **Add** pour ajouter un module de police d'authentification JAAS. Remplissez les informations pour votre module.

   Le **`Code`** est le nom de classe de votre module. Les **`Flags`** controlent la façon dont le module est lié à d'autres modules de polices d'authentification du même domaine de sécurité.

   Explication des indicateurs

   La spécification Java Enterprise Edition 6 fournit l'explication suivante sur les marqueurs des modules de sécurité. La liste suivante provient de http://docs.oracle.com/javase/6/docs/technotes/guides/security/jaas/JAASRefGuide.html#Appendix Consultez ce document pour obtenir des informations plus détaillées.

Marqueur	Détails
requis	Le LoginModule est requis pour la réussite. En cas de réussite ou d'échec, l'autorisation continuera son chemin dans la liste du LoginModule.
pré-requis	Le LoginModule est requis pour la réussite. En cas de succès, l'autorisation continue vers le bas de la liste LoginModule. En cas d'échec, le contrôle est renvoyé immédiatement à l'application (l'autorisation ne continue pas son chemin le long de la liste LoginModule).
suffisant	Le LoginModule n'est pas nécessaire pour aboutir. S'il ne réussit pas, le contrôle retourne immédiatement à l'application (l'autorisation ne se déroule pas vers le bas de la liste LoginModule). S'il échoue, l'authentification se poursuit vers le bas de la liste LoginModule.
option	Le LoginModule n'est pas requis pour la réussite. En cas de réussite ou d'échec, l'autorisation continuera son chemin dans la liste du LoginModule.

4. **Modifier les paramètres d'authorisation**

Une fois que vous aurez ajouté votre module, vous pourrez modifier son **Code** ou ses **Flags** (indicateurs) en cliquant **Edit** dans la section **Details** de l'écran. Veillez à ce que l'onglet **Attributes** soit bien sélectionné.

5. **Option : ajouter ou supprimer un module**
   Pour ajouter des options à votre module, cliquez sur leurs entrées dans la liste **Policies**, et sélectionnez l'onglet **Module Options** dans la section **Details** qui se trouve en bas de la page. Cliquez sur **Add**, et fournir la clé et la valeur de l'option. Utilisez le bouton **Remove** pour supprimer l'option.

Résultat

Votre module de police de sécurité est ajouté au domaine de sécurité, et est rendu disponible immédiatement auprès des applications qui utilisent le domaine de sécurité.

Rapporter un bogue

13.6. JAVA AUTHORIZATION CONTRACT FOR CONTAINERS (JACC)

13.6.1. Java Authorization Contract for Containers (JACC)

Java Authorization Contract for Containers (JACC) est une norme qui définit un contrat entre les conteneurs et les fournisseurs de services d'autorisation, et qui se traduit par l'implémentation de fournisseurs à utiliser par des conteneurs. Il a été défini dans JSR-115, qui se trouve sur le site Web de Java Community Process http://jcp.org/en/jsr/detail?id=115. A fait partie de la spécification Java Enterprise Edition (Java EE) depuis Java EE version 1.3.

JBoss EAP 6 implémente un support pour JACC dans la fonctionnalité de sécurité du sous-système de sécurité.

Rapporter un bogue

13.6.2. Configurer la sécurité JACC (Java Authorization Contract for Containers)

Pour configurer JACC (Java Authorization Contract for Containers), il convient de configurer votre domaine de sécurité avec le module qui convient, puis de modifier votre fichier **jboss-web.xml** pour y inclure les paramètres qu'il faut.

Ajouter JACC Support au domaine de sécurité

Pour ajouter JACC au domaine de sécurité, ajouter la police d'autorisation **JACC** à la pile d'autorisations du domaine de sécurité, avec l'indicateur **requis**. Voici un exemple de domaine de sécurité avec support JACC. Cependant, le domaine de sécurité est configuré dans la console de gestion ou Management CLI, plutôt que directement dans le code XML.

```
<security-domain name="jacc" cache-type="default">
    <authentication>
        <login-module code="UsersRoles" flag="required">
        </login-module>
    </authentication>
    <authorization>
        <policy-module code="JACC" flag="required"/>
    </authorization>
</security-domain>
```

Configurer une application web qui utilise JACC

Le fichier **jboss-web.xml** se trouve dans **WEB-INF/** de votre déploiement, et contient des ajouts ou remplacements de configuration spécifique JBoss pour le conteneur web. Pour utiliser votre domaine de sécurité activé-JACC, vous devrez inclure l'élément **<security-domain>**, et aussi définir l'élément **<use-jboss-authorization>** à **true**. L'application suivante est configurée correctement pour pouvoir utiliser le domaine de sécurité JACC ci-dessus.

```
<jboss-web>
    <security-domain>jacc</security-domain>
    <use-jboss-authorization>true</use-jboss-authorization>
</jboss-web>
```

Configurer une application EJB pour utiliser JACC

La façon de configurer les EJB à utiliser un domaine de sécurité et JACC diffère en fonction des application Web. Pour un EJB, vous pouvez déclarer des *method permissions* sur une méthode ou sur un groupe de méthodes, dans le descripteur **ejb-jar.xml**. Dans l'élément **<ejb-jar>**, chaque élément **<method-permission>** dépendant contient des informations sur les roles JACC. Voir l'exemple de configuration pour plus d'informations. La classe **EJBMethodPermission** fait partie de Java Enterprise Edition 6 API, et est documentée dans http://docs.oracle.com/javaee/6/api/javax/security/jacc/EJBMethodPermission.html.

Exemple 13.7. Exemple de permissions de méthode JACC dans un EJB

```
<ejb-jar>
  <assembly-descriptor>
    <method-permission>
      <description>The employee and temp-employee roles may access any
method of the EmployeeService bean </description>
      <role-name>employee</role-name>
      <role-name>temp-employee</role-name>
      <method>
        <ejb-name>EmployeeService</ejb-name>
        <method-name>*</method-name>
      </method>
    </method-permission>
  </assembly-descriptor>
</ejb-jar>
```

Vous pouvez également contraindre les mécanismes d'authentification et d'autorisation d'un EJB à l'aide d'un domaine de sécurité, comme vous pouvez le faire pour une application web. Les domaines de sécurité sont déclarés dans le descripteur **jboss-ejb3.xml** qui se trouve dans l'élément enfant **<security>**. En plus du domaine de sécurité, vous pouvez également spécifier le*run-as principal*, qui change le principal que l'EJB exécute.

Exemple 13.8. Exemple de déclaration de domaine de sécurité dans un EJB

```
<ejb-jar>
 <assembly-descriptor>
  <security>
    <ejb-name>*</ejb-name>
    <security-domain>myDomain</security-domain>
    <run-as-principal>myPrincipal</run-as-principal>
```

```
    </security>
    </assembly-descriptor>
</ejb-jar>
```

Rapporter un bogue

13.6.3. Autorisation affinée par XACML

13.6.3.1. Autorisation affinée et XACML

Les autorisations affinées répondent à l'évolution des besoins et des multiples variables impliquées dans le processus de décision, devenant ainsi une base d'autorisation pour accéder à un module. Par conséquent, le processus d'autorisation affiné est complexe en soi.

JBoss utilise XACML comme un moyen d'atteindre une autorisation affinée. XACML fournit la solution standard à la nature complexe d'une autorisation affinée. XACML définit un langage de stratégie et une architecture de prise de décisions. L'architecture XACML comprend un PEP (Policy Enforcement Point) qui intercepte toutes les demandes dans un flux de programme normal, puis demande à un PDP (Policy Decision Point) de prendre une décision d'accès basée sur les stratégies associées au PDP. Le PDP évalue la demande XACML créée par le PEP et vérifie dans les stratégies avant de prendre une décision sur les accès suivants :

- PERMIT - Accès approuvé.
- DENY - Accès non autorisé.
- INDETERMINATE - Erreur dans le PDP.
- NOTAPPLICABLE - Il y a des attributs manquants dans la requête, ou bien, il n'y a pas de politique correspondante.

Voici les fonctionnalités du XACML :

- Bibliothèque Oasis XACML v2.0
- Modèle d'objet basé JAXB v2.0
- Intégration ExistDB pour stocker/extraire des Polices et Attributs XACML

Rapporter un bogue

13.6.3.2. Configurer XACML pour autorisation affinée

La procédure suivante montre comment configurer XACML.

Procédure 13.10. Configurer XACML

1. Télécharger la bibliothèque qui correspond à un simple fichier jar.
2. **Créer un ou plusieurs fichiers de stratégies pour XACML**
   - Sous **`WEB-INF/classes`**, créer un répertoire **`policies`** pour sauvegarder toutes vos stratégies.

- Créer un **`policyConfig.xml`** sous le répertoire **`WEB-INF/classes`**.

  Il existe deux types d'ensembles de stratégies que l'on puisse définir :

  - Role Permission Policy Sets (RPS)
  - Permission Policy Sets (PPS)

Exemple 13.9. Role Permission Policy Sets (RPS)

Employé

```
    <PolicySet
xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
    PolicySetId="RPS:employee:role"
    PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-
combining-algorithm:permit-overrides">
    <Target>
    <Subjects>
    <Subject>
    <SubjectMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
    <AttributeValue

DataType="http://www.w3.org/2001/XMLSchema#anyURI">employee</Attri
buteValue>
    <SubjectAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
    DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
    </SubjectMatch>
    </Subject>
    </Subjects>
    </Target>
    <!-- Use permissions associated with the employee role -->
    <PolicySetIdReference>PPS:employee:role</PolicySetIdReference>
    </PolicySet>
```

Gestionnaire

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
PolicySetId="RPS:manager:role"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-
combining-algorithm:permit-overrides">
<Target>
<Subjects>
<Subject>
<SubjectMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
<AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#anyURI">manager</Attrib
uteValue>
<SubjectAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
</SubjectMatch>
```

```
</Subject>
</Subjects>
</Target>
<!-- Use permissions associated with the manager role -->
<PolicySetIdReference>PPS:manager:role</PolicySetIdReference>
</PolicySet>
```

Exemple 13.10. Permission Policy Sets (PPS)

Employé

```
    <PolicySet
xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
        PolicySetId="PPS:employee:role"

PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-
combining-algorithm:permit-overrides">
        <Target />
        <!-- Permissions specifically for the employee role -->
        <Policy
PolicyId="Permissions:specifically:for:the:employee:role"

RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:permit-overrides">
            <Target />
            <!-- Permission to create a purchase order -->
            <Rule RuleId="Permission:to:create:a:purchase:order"
Effect="Permit">
                <Target>
                    <Resources>
                        <Resource>
                            <ResourceMatch

MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                                <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">purchase order
                                </AttributeValue>
                                <ResourceAttributeDesignator

AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string" />
                            </ResourceMatch>
                        </Resource>
                    </Resources>
                    <Actions>
                        <Action>
                            <ActionMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                                <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">create</Attribu
teValue>
                                <ActionAttributeDesignator
AttributeId="urn:action-id"
```

```
DataType="http://www.w3.org/2001/XMLSchema#string" />
                            </ActionMatch>
                        </Action>
                    </Actions>
                </Target>
            </Rule>
        </Policy>
        <!-- HasPrivilegesOfRole Policy for employee role -->
    <Policy
PolicyId="Permission:to:have:employee:role:permissions"
        RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-
combining-algorithm:permit-overrides">
        <Target />
        <!-- Permission to have employee role permissions -->
        <Rule RuleId="Permission:to:have:employee:permissions"
Effect="Permit">
            <Condition>
                <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
                    <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:anyURI-is-in">
                        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#anyURI">employee</Attri
buteValue>
                        <ResourceAttributeDesignator

AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
DataType="http://www.w3.org/2001/XMLSchema#anyURI" />
                    </Apply>
                    <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:anyURI-is-in">
                        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#anyURI">urn:oasis:names
:tc:xacml:2.0:actions:hasPrivilegesOfRole
                        </AttributeValue>
                        <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"

DataType="http://www.w3.org/2001/XMLSchema#anyURI" />
                    </Apply>
                </Apply>
            </Condition>
        </Rule>
    </Policy>
    </PolicySet>
```

Gestionnaire

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
        PolicySetId="PPS:manager:role"

PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-
combining-algorithm:permit-overrides">
        <Target />
        <!-- Permissions specifically for the manager role -->
        <Policy
```

```
PolicyId="Permissions:specifically:for:the:manager:role"

RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:permit-overrides">
            <Target />
            <!-- Permission to sign a purchase order -->
            <Rule RuleId="Permission:to:sign:a:purchase:order"
Effect="Permit">
                <Target>
                    <Resources>
                        <Resource>
                            <ResourceMatch

MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                                <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">purchase order
                                </AttributeValue>
                                <ResourceAttributeDesignator

AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string" />
                            </ResourceMatch>
                        </Resource>
                    </Resources>
                    <Actions>
                        <Action>
                            <ActionMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                                <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">sign</Attribute
Value>
                                <ActionAttributeDesignator
AttributeId="urn:action-id"

DataType="http://www.w3.org/2001/XMLSchema#string" />
                            </ActionMatch>
                        </Action>
                    </Actions>
                </Target>
            </Rule>
        </Policy>
        <!-- HasPrivilegesOfRole Policy for manager role -->
    <Policy PolicyId="Permission:to:have:manager:role:permissions"
        RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-
combining-algorithm:permit-overrides">
        <Target />
        <!-- Permission to have manager role permissions -->
        <Rule RuleId="Permission:to:have:manager:permissions"
Effect="Permit">
            <Condition>
                <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
                    <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:anyURI-is-in">
                        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#anyURI">manager</Attrib
```

```
uteValue>
                        <ResourceAttributeDesignator

AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
DataType="http://www.w3.org/2001/XMLSchema#anyURI" />
                    </Apply>
                    <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:anyURI-is-in">
                        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#anyURI">urn:oasis:names
:tc:xacml:2.0:actions:hasPrivilegesOfRole
                        </AttributeValue>
                        <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"

DataType="http://www.w3.org/2001/XMLSchema#anyURI" />
                    </Apply>
                </Apply>
            </Condition>
        </Rule>
    </Policy>
        <!-- Include permissions associated with employee role --
>

<PolicySetIdReference>PPS:employee:role</PolicySetIdReference>
    </PolicySet>
```

3. **Créer un fichier de configuration pour XACML engine.**
   Un fichier de configuration est créé pour configurer les localisateurs et pour indiquer les répertoires où les stratégies sont sauvegardées.

Exemple 13.11. Fichier de configuration

Fichier de configuration uniquement qui indique le répertoire de fichiers de stratégies.

```
    <ns:jbosspdp xmlns:ns="urn:jboss:xacml:2.0">
      <ns:Policies>
       <ns:PolicySet>
          <ns:Location>test/policies/rbac/</ns:Location>
       </ns:PolicySet>
      </ns:Policies>
      <ns:Locators>
        <ns:Locator
Name="org.jboss.security.xacml.locators.JBossRBACPolicySetLocator"
/>
      </ns:Locators>
    </ns:jbosspdp>
```

Fichier de configuration indiquant un ensemble de stratégies.

```
<ns:jbosspdp xmlns:ns="urn:jboss:xacml:2.0">
  <ns:Policies>
    <ns:PolicySet>
      <ns:Location>test/policies/rbac/employee-PPS-
```

```
policyset.xml</ns:Location>
    </ns:PolicySet>
    <ns:PolicySet>
      <ns:Location>test/policies/rbac/manager-PPS-
policyset.xml</ns:Location>
    </ns:PolicySet>
    <ns:PolicySet>
      <ns:Location>test/policies/rbac/employee-RPS-
policyset.xml</ns:Location>
    </ns:PolicySet>
    <ns:PolicySet>
      <ns:Location>test/policies/rbac/manager-RPS-
policyset.xml</ns:Location>
    </ns:PolicySet>
  </ns:Policies>
  <ns:Locators>
    <ns:Locator
Name="org.jboss.security.xacml.locators.JBossRBACPolicySetLocator"
/>
  </ns:Locators>
</ns:jbosspdp>
```

4. Créer un PDP (Policy Decision Point) et le passer dans un fichier de configuration.

5. Dans le PEP (Policy Enforcement Point), créer une demande XACML basée sur le contexte. Passer la requête XACML au PDP pour obtenir une des décisions d'accès suivantes :

   - Permission
   - Refuser
   - Indéterminé
   - Non Applicable

Exemple 13.12. Décisions d'accès

Condition de permission

```
    <Request
          xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:context:schema:os

             access_control-xacml-2.0-context-schema-os.xsd">
    <Subject>
    <Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
     DataType="http://www.w3.org/2001/XMLSchema#string">
    <AttributeValue>Anne</AttributeValue>
    </Attribute>

    <Attribute
AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
```

```
     DataType="http://www.w3.org/2001/XMLSchema#anyURI">
    <AttributeValue>manager</AttributeValue>
    </Attribute>
    </Subject>

    <Resource>
    <Attribute
AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
    DataType="http://www.w3.org/2001/XMLSchema#anyURI">
    <AttributeValue>manager</AttributeValue>
    </Attribute>
    </Resource>

    <Action>
    <Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
     DataType="http://www.w3.org/2001/XMLSchema#anyURI">

<AttributeValue>urn:oasis:names:tc:xacml:2.0:actions:hasPrivileges
OfRole</AttributeValue>
    </Attribute>
    </Action>
    </Request>
```

Déni de permission

```
    <Request
xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:context:schema:os

             access_control-xacml-2.0-context-schema-os.xsd">
        <Subject>
            <Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"

DataType="http://www.w3.org/2001/XMLSchema#string">
                <AttributeValue>Anne</AttributeValue>
            </Attribute>

            <Attribute
AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"

DataType="http://www.w3.org/2001/XMLSchema#anyURI">
                <AttributeValue>manager</AttributeValue>
            </Attribute>
        </Subject>

        <Resource>
            <Attribute
AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"

DataType="http://www.w3.org/2001/XMLSchema#anyURI">
                <AttributeValue>manager</AttributeValue>
            </Attribute>
```

```
        </Resource>

        <Action>
            <Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"

DataType="http://www.w3.org/2001/XMLSchema#anyURI">
                <AttributeValue>urn:nobody</AttributeValue>
            </Attribute>
        </Action>
    </Request>
```

Rapporter un bogue

13.7. SECURITY AUDITING

13.7.1. Security Auditing

Security Auditing se réfère au déclenchement d'événements, comme écrire un blog en réponse à un événement qui a lieu dans le sous-système de sécurité. Les mécanismes de sécurité sont configurés dans le cadre du domaine de sécurité, avec les informations d'authentification, d'autorisation ou de mappage de sécurité.

Auditing utilise *des modules de fournisseur*. Vous pouvez en utiliser un existant ou bien créer le vôtre.

Rapporter un bogue

13.7.2. Configurer Security Auditing

Pour configurer les paramètres de Security Auditing, connectez-vous à la console de gestion et suivre la procédure suivante :

Procédure 13.11. Configurez Security Auditing pour un domaine de sécurité

1. **Ouvrir l'affichage détaillé du domaine de sécurité.**
   a. Cliquez sur **`Configuration`** qui se trouve en haut de l'écran.
   b. Dans un domaine géré, sélectionnez le profil à modifier à partir du menu déroulant **`Profile`** en haut et à gauche.
   c. Étendre le menu **`Security`**, puis sélectionnez **`Security Domains`**.
   d. Cliquez sur **`View`** pour obtenir le domaine de sécurité que vous voulez modifier.
2. **Naviguez dans la configuration du sous-système Auditing.**
   Sélectionnez l'onglet **`Audit`** qui se trouve en haut et à droite.

   La zone de configuration est divisée en deux : **`Provider Modules`** et **`Details`**. Le module de fournisseur est l'unité de base de configuration. Un domaine de sécurité peut inclure plusieurs polices d'autorisation, et chacune peut inclure plusieurs attributs et options.
3. **Ajoutez un module de fournisseur.**

Cliquez sur **Add**. Remplir la section **Code** avec le nom de classe du module du fournisseur.

4. **Vérifiez que le module fonctionne**
   Le but d'un module d'auditing est d'offrir un moyen de surveiller les événements dans le sous-système de sécurité. Ce monitoring peut être réalisé sous la forme d'écriture dans un fichier de journalisation, des notifications email ou autre mécanisme d'audit mesurable.

   Par exemple, JBoss EAP 6 inclut le module **LogAuditProvider** par défaut. S'il est activé par les étapes décrites ci-dessus, ce module d'audit écrira des notifications de sécurité dans un fichier **audit.log** qui se situe dans le sous-dossier **log** du répertoire ***EAP_HOME***.

   Pour vérifier si les étapes ci-dessus fonctionnent dans le contexte **LogAuditProvider**, procédez à une action qui risque de déclencher une notification, puis vérifiez le fichier de journalisation de l'audit.

   Pour obtenir une liste complète des Modules Security Auditing Provider, voir : Section A.4, « Modules de fournisseurs d'auditing de sécurité inclus »

5. **Option : ajouter, éditer ou supprimer un module**
   Pour ajouter des options à votre module, cliquez sur leurs entrées dans la liste **Modules**, et sélectionnez l'onglet **Module Options** dans la section **Details** qui se trouve en bas de la page. Cliquez sur **Add**, et fournir la clé et la valeur de l'option.

   Pour modifier une option déjà existante, cliquez sur **Remove** pour la supprimer, et cliquez sur **Add** pour l'ajouter à nouveau avec les options qui conviennent.

Résultat

Votre module d'audit de sécurité est ajouté au domaine de sécurité, et est rendu disponible immédiatement auprès des applications qui utilisent le domaine de sécurité.

Rapporter un bogue

13.7.3. Nouvelles propriétés de sécurité

De nouvelles propriétés système ont été ajoutées à la fonctionnalité d'audit de sécurité pour les versions de JBoss EAP 6.2.2 et versions ultérieures. Ces nouvelles propriétés atténuent les problèmes de sécurité autour des enregistrements de texte brut de composants de requêtes de provenance web, en particulier dans les scénarios impliquant des authentifications basée BASIC ou FORM.

Les nouvelles propriétés permettent un meilleur contrôle des composants sur lesquels une requête web est capturée dans les journaux d'audit (paramètres, cookies, en-têtes ou attributs). Ces composants peuvent également être masqués à l'aide des nouvelles propriétés.

Les nouvelles propriétés sont les suivantes :

Tableau 13.1. Nouvelles propriétés de sécurité

Nom	Description	Valeurs possibles	Comportement	Par défaut

Nom	Description	Valeurs possibles	Comportement	Par défaut
`org.jboss.security.web.audit`	Cette propriété contrôle la granularité de l'auditing de sécurité des requêtes web.	**`off`**, **`headers`**, **`cookies`**, **`parameters`**, **`attributes`**	Tout composant (ou groupe de composants séparés par des virgules) spécifié sera audité à partir des requêtes web.	**`headers,parameters`**
`org.jboss.security.web.audit.mask`	Cette propriété peut être utilisée pour indiquer une liste de chaînes qui puissent correspondre à des en-tête, des paramètres, des cookies, et des attributs de requêtes web. Tout élément correspondant aux masques indiqués sera exclus de la journalisation d'audit de sécurité.	Toute chaîne séparée par des virgules utilisant des en-tête de clé, des paramètre, des cookies ou des attributs.	Actuellement, la mise en correspondance des masques est vague, non précise. Ainsi, un masque d'**`authorization`** masquera à la fois l'en-tête nommé *authorization* et le paramètre nommé *custom_authorization*. Il est possible qu'il y ait des masques plus stricts (précis) dans une version à venir.	j_password,authorization

Rapporter un bogue

13.8. MAPPAGE DE SÉCURITÉ

13.8.1. Mappage de sécurité

Le mappage de sécurité vous permet de combiner l'authentification et l'autorisation d'informations après que l'authentification ou l'autorisation aient eu lieu, mais avant que l'information ait été passée à votre application.

Vous pouvez mapper vos principaux (authentification), rôles (autorisation), ou identifiants (attributs qui ne sont ni principaux, ni rôles).

Le mappage de rôles est utilisé pour ajouter, remplacer ou supprimer des rôles du sujet après l'authentification.

Le mappage du principal est utilisé pour modifier un principal après l'authentification.

Le mappage d'attributs est utilisé pour convertir des attributs d'un système externe à utiliser par votre application, et vice versa.

Rapporter un bogue

13.8.2. Configurer le mappage de sécurité dans un domaine de sécurité

Pour configurer les paramètres de Security Mapping, vous connecter à la console de gestion et suivre la procédure suivante :

Procédure 13.12. Configurer le Mappage de sécurité pour un Domaine de sécurité

1. **Ouvrir l'affichage détaillé du domaine de sécurité.**
   a. Cliquez sur l'étiquette **Configuration** qui se trouve en haut de la console de gestion.
   b. Dans un domaine géré, sélectionnez le profil à partir du menu déroulant **Profile** en haut et à gauche.
   c. Étendre l'item de menu **Security**, puis sélectionnez **Security Domains**.
   d. Cliquez sur **View** pour obtenir le domaine de sécurité que vous voulez modifier.
2. **Naviguez dans la configuration du sous-système de mappage.**
   Sélectionnez l'étiquette **Mapping** en haut de l'écran.

   La zone de configuration est divisée en deux : **Modules** et **Details**. Le module de connexion est l'unité de base de configuration. Un domaine de sécurité peut inclure plusieurs polices de mapping, et chacune peut inclure plusieurs attributs et options.
3. **Ajouter un module de mappage de sécurité.**
   Cliquez sur **Ajouter**.

   Remplissez les informations de module. Le **Code** est le nom de classe du module. Le champ **Type** se rapporte au type de mapping effectué par ce module. Les valeurs autorisées sont les suivantes : principal, role, attribut ou identifiant.
4. **Modifier un module de mappage de sécurité**
   Une fois que vous aurez ajouté votre module, vous pourrez modifer son **Code** ou son **Type**.
   a. Sélectionnez l'onglet **Attributes**.
   b. Cliquez sur **Edit** dans le champ **Details** de cet écran.
5. **Option : ajouter, éditer ou supprimer un module**
   Pour ajouter des options à votre module, cliquez sur leurs entrées dans la liste **Modules**, et sélectionnez l'onglet **Module Options** dans la section **Details** qui se trouve en bas de la page. Cliquez sur **Add**, et fournir la clé et la valeur de l'option.

   Pour modifier une option déjà existante, cliquez sur **Remove** pour la supprimer, et cliquez sur add pour l'ajouter à nouveau avec la nouvelle valeur.

   Utilisez **Remove** pour supprimer une option.

Résultat

Votre module de mappage de sécurité est ajouté au domaine de sécurité, et est rendu disponible immédiatement auprès des applications qui utilisent le domaine de sécurité.

Rapporter un bogue

13.9. UTILISER UN DOMAINE DE SÉCURITÉ DANS VOTRE APPLICATION

Aperçu

Pour utiliser un domaine de sécurité dans votre application, vous devez tout d'abord définir le domaine dans le fichier de configuration du serveur, puis vous devez l'activer pour une application dans le descripteur de déploiement de l'application. Ensuite, vous devez ajouter les annotations requises à l'EJB qui les utilise. Cette rubrique décrit les étapes requises pour utiliser un domaine de sécurité dans votre application.

AVERTISSEMENT

Si une application fait partie d'un domaine de sécurité qui utilise un cache d'authentification, les authentifications utilisateur de cette application seront rendues disponibles à d'autres applications dans ce domaine de sécurité.

Procédure 13.13. Configurez votre application pour qu'elle puisse utiliser un domaine de sécurité

1. **Définir le domaine de sécurité**
   Vous devez définir le domaine de sécurité dans le fichier de configuration du serveur, puis l'activer pour une application dans le fichier du descripteur de l'application.

   a. **Configurez le domaine de sécurité dans le fichier de configuration du serveur**
   Le domaine de sécurité est configuré dans le sous-système de **`sécurité`** du fichier de configuration du serveur. Si l'instance de JBoss EAP 6 s'exécute dans un domaine géré, il s'agira du fichier **`domain/configuration/domain.xml`**. Si l'instance de JBoss EAP 6 s'exécute comme un serveur autonome, ce sera le fichier **`standalone/configuration/standalone.xml`**.

   Les domaines de sécurité **`other`**, **`jboss-web-policy`**, et **`jboss-ejb-policy`** sont fournis par défaut dans JBoss EAP 6. L'exemple XML suivant a été copié à partir du sous-système de **`sécurité`** dans le fichier de configuration du serveur.

   L'attribut **`cach-type`** d'un domaine de sécurité spécifie un cache pour pouvoir effectuer des contrôles d'authentification plus rapides. Les valeur autorisées sont les valeur par **`défaut`** en cas de simile map comme cache ou **`infinispan`** pour un cache Infinispan.

```
<subsystem xmlns="urn:jboss:domain:security:1.2">
    <security-domains>
        <security-domain name="other" cache-type="default">
            <authentication>
                <login-module code="Remoting" flag="optional">
                    <module-option name="password-stacking"
value="useFirstPass"/>
                </login-module>
                <login-module code="RealmDirect"
flag="required">
                    <module-option name="password-stacking"
value="useFirstPass"/>
```

```
                </login-module>
            </authentication>
        </security-domain>
        <security-domain name="jboss-web-policy" cache-
type="default">
            <authorization>
                <policy-module code="Delegating"
flag="required"/>
            </authorization>
        </security-domain>
        <security-domain name="jboss-ejb-policy" cache-
type="default">
            <authorization>
                <policy-module code="Delegating"
flag="required"/>
            </authorization>
        </security-domain>
    </security-domains>
</subsystem>
```

Vous pouvez configurer des domaines de sécurité supplémentaires selon les besoins par la console de gestion ou par l'interface CLI.

b. **Activer le domaine de sécurité dans le fichier de descripteur de l'application.**
Le domaine de sécurité est spécifié dans l'élément enfant **`<security-domain>`** de l'élément **`<jboss-web>`** du fichier **`WEB-INF/jboss-web.xml`** de l'application. L'exemple suivant configure un domaine de sécurité nommé **`my-domain`**.

```
<jboss-web>
    <security-domain>my-domain</security-domain>
</jboss-web>
```

Il s'agit d'une des configurations que vous pouvez indiquer dans le descripteur **`WEB-INF/jboss-web.xml`**.

2. **Ajoutez l'annotation requise à l'EJB.**
Vous pouvez configurer la sécurité dans EJB par les annotations **`@SecurityDomain`** et **`@RolesAllowed`**. L'exemple de code EJB suivant limite l'accès au domaine de sécurité **`other`** aux utilisateurs ayant pour rôle **`guest`** (invité).

```
package example.ejb3;

import java.security.Principal;

import javax.annotation.Resource;
import javax.annotation.security.RolesAllowed;
import javax.ejb.SessionContext;
import javax.ejb.Stateless;

import org.jboss.ejb3.annotation.SecurityDomain;

/**
 * Simple secured EJB using EJB security annotations
 * Allow access to "other" security domain by users in a "guest"
role.
```

```
 */
@Stateless
@RolesAllowed({ "guest" })
@SecurityDomain("other")
public class SecuredEJB {

   // Inject the Session Context
   @Resource
   private SessionContext ctx;

   /**
    * Secured EJB method using security annotations
    */
   public String getSecurityInfo() {
      // Session context injected using the resource annotation
      Principal principal = ctx.getCallerPrincipal();
      return principal.toString();
   }
}
```

Pour obtenir des exemples de code supplémentaires, voir **`ejb-security`** Quickstart dans le package JBoss EAP 6 Quickstarts disponible à partir du Portail Clients Red Hat.

Rapporter un bogue

CHAPITRE 14. SINGLE SIGN ON (SSO)

14.1. SSO (SINGLE SIGN ON) POUR LES APPLICATIONS WEB

Aperçu

Single Sign On (SSO) autorise l'authentification à une ressource, en vue d'autoriser implicitement l'accès à d'autres ressources.

SSO clusterisées ou non-clusterisées

SSO Non-clusterisée limite le partage des informations d'autorisation pour les applications d'un même hôte virtuel. En outre, il n'y a aucun mécanisme de résilience en cas de défaillance de l'hôte. Les données d'authentification SSO clusterisées peuvent être partagées entre les applications de plusieurs hôtes virtuels et sont résistantes au basculement. De plus, SSO clusterisée est capable de recevoir des demandes d'un équilibreur de charges.

La façon dont SSO fonctionne

Si une ressource n'est pas protégée, un utilisateur n'a pas besoin de s'authentifier. Si un utilisateur accède à une ressource protégée, l'utilisateur devra s'authentifier.

Si l'authentification réussit, les rôles associés à l'utilisateur seront stockés et utilisés pour l'autorisation de toutes les ressources associées.

Si l'utilisateur se déconnecte d'une application, ou bien si l'application produit la session par programmation, toutes les données d'autorisation persistées seront retirées, et le processus recommencera.

Un timeout de session ne rend pas la session SSO valide si d'autres sessions sont encore valides.

Rapporter un bogue

14.2. SSO (SINGLE SIGN ON) CLUSTERISÉES POUR LES APPLICATIONS WEB

SSO (Single Sign On) est la possibilité pour les utilisateurs de s'authentifier à une application web unique au moyen d'une authentification réussie, afin d'obtenir l'autorisation de plusieurs autres applications. SSO clusterisée stocke les informations d'authentification et d'autorisation dans un cache clusterisé. Cela permet aux applications de serveurs différents de partager l'information et rend également l'information résistante à une défaillance de l'un des hôtes.

Une configuration SSO s'appelle une vanne. Une vanne est connectée à un domaine de sécurité, qui est configuré au niveau du serveur ou du groupe de serveurs. Chaque application qui doit partager les mêmes informations d'authentification mises en cache est configurée pour utiliser la même valve. Cette configuration s'effectue dans le fichier **`jboss-Web.xml`** de l'application.

Voici quelques valves SSO standards prises en charge par le sous-système de JBoss EAP 6 :

- Apache Tomcat ClusteredSingleSignOn
- Apache Tomcat IDPWebBrowserSSOValve
- SSO basée-SPNEGO de PicketLink

Suivant le type particulier de valve, vous aurez sans doute besoin d'ajouter des configurations supplémentaires au domaine de sécurité, pour que votre valve puisse fonctionner normalement.

Rapporter un bogue

14.3. CHOISIR L'IMPLÉMENTATION SSO QUI VOUS CONVIENT

JBoss EAP 6 exécute des applications de JBoss Enterprise Edition (EE), qui peuvent être des applications web, des applications EJB, des services web ou autres. SSO (Single Sign On) permet de propager les informations de contexte et d'identité de sécurité entre ces applications. Il existe plusieurs solutions SSO mais vous devrez choisir en fonction des besons de votre organisation.

Notez qu'il y a une nette différence entre une application web en cluster et une SSO clusterisée. Une application web cluster est une application distribuée à travers les nœuds d'un cluster pour répartir la charge d'accueil de cette appplication. Si elle est indiquée come étant distribuable, toutes les nouvelles sessions et les changements aux sessions existantes seront répliqués à travers les autres membres du cluster. Une application est marquée comme pouvant être distribuée à travers l'ensemble de nœuds du cluster avec la balise <distributable/> dans le descripteur de déploiement **`web.xml`**. Une SSO clusterisée permet la réplication des informations d'identité et de contexte de sécurité, indépendamment du fait de savoir si les applications sont elles-mêmes clusterisées. Bien que ces technologies peuvent être utilisées conjointement, elles repésentent des concepts distincts.

SSO Desktop basé-Kerberos

Si votre organisation utilise déjà une authentification basée Kerberos et un système d'autorisation comme Microsoft Active Directory, vos pourrez utiliser les mêmes systèmes pour vous authentifier de façon transparente à vos applications enterprise en cours d'exécution sur JBoss EAP 6.

SSO d'application web non clusterisée

Si vous exécutez un certain nombre d'applications sur une seule instance et que vous ayiez besoin d'activer une réplication de session SSO pour ces applications, la SSO non clusterisée sera la solution indiquée.

SSO Application Web et clusterisée

Si vous exécutez une ou plusieurs applications dans un cluster et que vous ayiez besoin d'activer une réplication de session SSO pour toutes ces applications, la SSO clusterisée sera la solution indiquée.

Rapporter un bogue

14.4. UTILISATION DE SSO (SINGLE SIGN ON) POUR LES APPLICATIONS WEB

Aperçu

Les fonctionnalités SSO (Single Sign On) sont fournies par les sous-systèmes web et Infinispan. Utilisez cette procédure pour configurer SSO dans les applications web.

Pré-requis

- Vous devez avoir un domaine de sécurité configuré qui gère les authentifications et autorisations.
- Le sous-système **`infinispan`** est présent dans le profil **`full-ha`** d'un domaine géré, ou en utilisant la configuration **`standalone-full-ha.xml`** dans un serveur autonome.

- Le **web cache-conteneur** et le cache-conteneur SSO doivent chacun être présents. Les exemples de fichiers de configurations contiennent déjà le cache-conteneur **web**, et certaines des configurations contiennent déjà le cache-conteneur SSO également. Utilisez les commandes suivantes pour vérifier et activer le cache conteneur SSO. Notez que ces commandes modifient le profil **ha** d'un domaine géré. Vous pouvez modifier les commandes pour utiliser un profil différent, ou supprimer la portion **/profile=ha** de la commande, pour un serveur autonome.

> **Exemple 14.1. Vérifier le cache-conteneur web**
>
> Les profils et les configurations mentionnées ci-dessus incluent le cache-conteneur **web** par défaut. Utilisez la commande suivante pour vérifier sa présence. Si vous utilisez un profil différent, remplacez par son nom à la place de **ha**.
>
> ```
> /profile=ha/subsystem=infinispan/cache-container=web/:read-
> resource(recursive=false,proxies=false,include-
> runtime=false,include-defaults=true)
> ```
>
> Si le résultat affiche un **success**, le sous-système sera présent. Sinon, vous devrez l'ajouter.

> **Exemple 14.2. Ajouter le cache-conteneur web**
>
> Utilisez les trois commandes suivantes pour activer le cache-conteneur **web** à votre configuration. Modifiez le nom du profil selon le cas, ainsi que les autres paramètres. Ici, les paramètres sont ceux utilisés dans une configuration par défaut.
>
> ```
> /profile=ha/subsystem=infinispan/cache-container=web:add(aliases=
> ["standard-session-cache"],default-
> cache="repl",module="org.jboss.as.clustering.web.infinispan")
> ```
>
> ```
> /profile=ha/subsystem=infinispan/cache-
> container=web/transport=TRANSPORT:add(lock-timeout=60000)
> ```
>
> ```
> /profile=ha/subsystem=infinispan/cache-container=web/replicated-
> cache=repl:add(mode="ASYNC",batching=true)
> ```

> **Exemple 14.3. Vérifier le cache-conteneur SS0**
>
> Éxécutez la commande de Management CLI suivante :
>
> ```
> /profile=ha/subsystem=infinispan/cache-container=web/:read-
> resource(recursive=true,proxies=false,include-
> runtime=false,include-defaults=true)
> ```
>
> Cherchez une sortie qui ressemble à ceci : **"sso" => {**
>
> Si vous ne la trouvez pas, le cache-conteneur ne sera pas présent dans votre configuration.

> **Exemple 14.4. Ajoutez le cache-conteneur SS0**

```
/profile=ha/subsystem=infinispan/cache-container=web/replicated-cache=sso:add(mode="SYNC", batching=true)
```

- Le sous-système **web** doit être configuré pour utiliser SSO. La commande suivante active SSO sur le serveur virtuel appelé **default-host** et le domaine de cookies **domaine.com**. Le nom de cache est **sso**, et une nouvelle authentification est désactivée.

```
/profile=ha/subsystem=web/virtual-server=default-host/sso=configuration:add(cache-container="web",cache-name="sso",reauthenticate="false",domain="domain.com")
```

- Chaque application qui partage les informations SSO doit être configurée pour utiliser le même <security-domain> dans son descripteur de déploiement de **jboss-web.xml** et le même Realm dans son fichier de configuration **web.xml**.

Configurer une SSO clusterisée ou non-clusterisée

Configurer **sso** sous le sous-système web dans le profil utilisateur. La version **ClusteredSingleSignOn** est utilisée quand l'attribut **cache-container** est présent, sinon la classe **SingleSignOn** sera utilisée.

Exemple 14.5. Exemple de configuration SSO clusterisée

```
/subsystem=web/virtual-server=default-host/sso=configuration:add(cache-container="web",cache-name="sso",reauthenticate="false",domain="domain.com")
```

Exemple 14.6. Exemple de configuration SSO non clusterisée

```
/subsystem=web/virtual-server=default-host/sso=configuration:add(reauthenticate="false")
```

Rendre une session non valide

Une application peut rendre invalide une session en invoquant la méthode **javax.servlet.http.HttpSession.invalidate()**.

Rapporter un bogue

14.5. KERBEROS

Kerberos est un protocole d'authentification réseau pour les applications client/serveur. Il permet l'authentification sur un réseau non sécurisé en toute sécurité, à l'aide de la clé secrète de cryptographie symétrique.

Kerberos utilise les tokens de sécurité appelés «tickets». Pour utiliser un service sécurisé, vous devez obtenir un ticket auprès du Ticket Granting Service (TGS), qui est un service de délivrance de tickets exécuté sur un serveur sur votre réseau. Après avoir obtenu le ticket, vous pourrez demander un Ticket de Service (ST ou Service Ticket) d'un Service d'authentification (AS), qui est un autre service

s'exécutant sur votre réseau. Utilisez ensuite ST pour vous authentifier auprès du service que vous souhaitez utiliser. Les TGS et AS tous exécutent tous deux à l'intérieur d'un service intégré appelé KDC (Key Distribution Center).

Kerberos est conçu pour être utilisé dans un environnement client-serveur et est rarement utilisé dans les applications Web ou des environnements clients légers. Cependant, de nombreuses organisations utilisent déjà un système Kerberos pour l'authentification desktop et préfèrent réutiliser leur système existant plutôt que d'en créer un second pour leurs applications web. Kerberos est partie intégrante de Microsoft Active Directory et est également utilisé dans de nombreux environnements Red Hat Enterprise Linux.

Rapporter un bogue

14.6. SPNEGO

Simple and Protected GSS_API Negotiation Mechanism (SPNEGO) propose un mécanisme d'expension d'environnement SSO (Single Sign On) pour les applications web.

Lorsqu'une application d'ordinateur client, comme un navigateur web, tente d'accéder à une page de protection sur le serveur web, le serveur répond qu'une autorisation est requise. Ensuite, l'application demande un ticket de service du Centre de distribution des clés Kerberos (KDC). Après l'obtention du ticket, l'application le renvoie dans une requête formatée SPNEGO et l'envoie à l'application web, par l'intermédiaire du navigateur. Le conteneur web exécutant l'application web déployée ouvre la requête et authentifie le billet. Après une authentification réussie, l'accès est accordé.

SPNECO fonctionne avec tous les types de fournisseurs Kerberos, y compris le service Kerberos inclus dans Red Hat Enterprise Linux et le serveur Kerberos, qui fait partie intégrale du Microsoft Active Directory.

Rapporter un bogue

14.7. MICROSOFT ACTIVE DIRECTORY

Microsoft Active Directory est un service de répertoires développé par Microsoft pour authentifier les utilisateurs et les ordinateurs d'un domaine Microsoft Windows. Il est inclus dans Microsoft Windows Server. L'ordinateur dans Microsoft Windows Server est nommé le Contrôleur de domaine. Les serveurs de Red Hat Enterprise Linux exécutant le service Samba peuvent aussi agir comme Contrôleur de domaine dans ce type de réseau.

Active Directory repose sur trois technologies principales dépendantes les unes des autres :

- LDAP (Lightweight Directory Access Protocol) stocke les informations utilisateurs, ordinateurs, mots de passe et autres ressources.
- Kerberos procure une authentification sécurisée sur le réseau.
- DNS (Domain Service Name) fournit des mappages entre les adresses IP, les noms d'hôtes et les ordinateurs ou autres périphériques du réseau.

Rapporter un bogue

14.8. CONFIGURATION DE KERBEROS OU MICROSOFT ACTIVE DIRECTORY DESKTOP SSO POUR LES APPLICATIONS WEB

Introduction

Pour authentifier vos applications web ou EJB à l'aide de l'authentification basée Kerberos existante de votre organisation et de votre infrastructure d'autorisation, comme Microsoft Active Directory, vous pouvez utiliser les fonctionnalités de JBoss Negotiation intégrées dans JBoss EAP 6. Si vous configurez votre application web correctement, une connexion réussie en desktop ou réseau sera suffisante pour vous authentifier en toute transparence dans votre application web, donc aucune invite de connexion supplémentaire ne sera nécessaire.

Différence par rapport aux versions précédentes de la plateforme

Il existe des différences notables entre la plateforme JBoss EAP 6 et les versions précédentes :

- Les domaines de sécurité sont configurés centralement, pour chaque profil d'un domaine géré, ou pour chaque serveur autonome. Ils ne font pas partie du déploiement lui-même. Le domaine de sécurité qu'un déploiement doit utiliser est nommé dans le fichier de déploiement **`jboss-web.xml`** ou **`jboss-ejb3.xml`**.
- Les propriétés de sécurité sont configurées dans le cadre du domaine de sécurité, faisant partie de sa configuration centrale. Elles ne font pas partie du déploiement.
- Vous pouvez ne plus remplacer les authentificateurs dans votre déploiement. Toutefois, vous pouvez ajouter une valve NegotiationAuthenticator à votre descripteur **`jboss-web.xml`** afin d'obtenir le même effet. La valve nécessite toujours les éléments **`<security-constraint>`** et **`<login-config>`** à définir dans le fichier **`web.xml`**. Ils servent à décider quelles ressources sont sécurisées. Cependant, l'auth-method choisie sera substituée par la valve NegotiationAuthenticator du fichier **`jboss-web.xml`**.
- Les attributs **`CODE`** des domaines de sécurité utilisent maintenant un nom simple au lieu d'un nom de classe complet. Le tableau suivant montre les mappages entre les classes utilisées pour JBoss Negotiation, et leurs classes.

Tableau 14.1. Codes de modules de connexion et Noms de classes

Nom simple	Nom de classe	But
Kerberos	com.sun.security.auth.module.Krb5LoginModule com.ibm.security.auth.module.Krb5LoginModule	Module de connexion Kerberos quand on utilise le JDK d'Oracle. Module de connexion Kerberos quand on utilise le JDK d'IBM.
SPNEGO	org.jboss.security.negotiation.spnego.SPNEGOLoginModule	Le mécanisme qui permet aux applications web de s'authentifier à votre serveur d'authentification Kerberos.
AdvancedLdap	org.jboss.security.negotiation.AdvancedLdapLoginModule	Utilisé avec les serveurs LDAP autres que Microsoft Active Directory.
AdvancedAdLdap	org.jboss.security.negotiation.AdvancedADLoginModule	Utilisé avec les serveurs Microsoft Active Directory LDAP.

JBoss Negotiation Toolkit

`JBoss Negotiation Toolkit` est un outil de débogage que vous pouvez télécharger à partir de https://community.jboss.org/servlet/JiveServlet/download/16876-2-34629/jboss-negotiation-toolkit.war. Il est fourni comme un outil supplémentaire pour vous aider à déboguer et tester les mécanismes

d'authentification avant d'introduire votre application en production. C'est un outil non pris en charge, mais considéré comme très utile. Comme SPNEGO, cet outil peut être difficile à configurer pour les applications web.

Procédure 14.1. Installation de l'authentification SSO (Single Sign On) pour les Applications Web ou EJB

1. **Configurer un domaine de sécurité qui représente l'identité de votre serveur. Définir les propriétés système si nécessaire.**
   Le premier domaine de sécurité authentifie le conteneur lui-même au service de répertoires. Il a besoin d'utiliser un module de connexion qui accepte un type de mécanisme de connexion statique, car un utilisateur réel n'est pas impliqué. Cet exemple utilise une entité de sécurité statique et fait référence à un fichier keytab qui contient les informations d'identification.

   Le code XML est donné ici dans un but de clarification, mais vous devez utiliser la console de gestion ou l'interface CLI pour configurer vos domaines de sécurité.

   ```
   <security-domain name="host" cache-type="default">
      <authentication>
         <login-module code="Kerberos" flag="required">
            <module-option name="storeKey" value="true"/>
            <module-option name="useKeyTab" value="true"/>
            <module-option name="principal"
   value="host/testserver@MY_REALM"/>
            <module-option name="keyTab"
   value="/home/username/service.keytab"/>
            <module-option name="doNotPrompt" value="true"/>
            <module-option name="debug" value="false"/>
         </login-module>
      </authentication>
   </security-domain>
   ```

2. **Configurer un second domaine de sécurité pour sécuriser l'application web ou les applications. Définir les propriétés système si nécessaire.**
   Le deuxième domaine de sécurité est utilisé pour authentifier l'utilisateur particulier au serveur d'authentification Kerberos ou SPNEGO. Vous avez besoin d'au moins un module de connexion pour authentifier l'utilisateur et un autre à la recherche de rôles à appliquer à l'utilisateur. Le code XML suivant montre un exemple de domaine de sécurité SPNEGO. Il inclut un module d'autorisation pour mapper les rôles aux utilisateurs individuels. Vous pouvez également utiliser un module de recherche pour les rôles sur le serveur d'authentification lui-même.

   ```
   <security-domain name="SPNEGO" cache-type="default">
      <authentication>
         <!-- Check the username and password -->
         <login-module code="SPNEGO"  flag="requisite">
            <module-option name="password-stacking"
   value="useFirstPass"/>
            <module-option name="serverSecurityDomain" value="host"/>
         </login-module>
         <!-- Search for roles -->
         <login-module code="UsersRoles" flag="required">
            <module-option name="password-stacking"
   value="useFirstPass" />
            <module-option name="usersProperties" value="spnego-
   users.properties" />
   ```

```
        <module-option name="rolesProperties" value="spnego-
roles.properties" />
      </login-module>
   </authentication>
</security-domain>
```

3. **Spécifier la security-constraint et la login-config dans le fichier `web.xml`**
   Le descripteur **`web.xml`** contient des informations sur les contraintes de sécurité et sur la configuration de la connexion. En voici des exemples :

```
<security-constraint>
   <display-name>Security Constraint on Conversation</display-name>
   <web-resource-collection>
      <web-resource-name>examplesWebApp</web-resource-name>
      <url-pattern>/*</url-pattern>
   </web-resource-collection>
   <auth-constraint>
   <role-name>RequiredRole</role-name>
   </auth-constraint>
</security-constraint>

<login-config>
   <auth-method>SPNEGO</auth-method>
   <realm-name>SPNEGO</realm-name>
</login-config>

<security-role>
   <description> role required to log in to the
Application</description>
   <role-name>RequiredRole</role-name>
</security-role>
```

4. **Indiquer le domaine de sécurité et les autres paramètres dans le descripteur `jboss-web.xml`.**
   Spécifiez le nom du domaine de la sécurité côté client (l'autre dans cet exemple) dans le descripteur **`jboss-Web.xml`** de votre déploiement, pour obliger votre application à utiliser ce domaine de sécurité.

   Vous ne pouvez plus remplacer les authentificateurs directement. À la place, vous devez ajouter NegotiationAuthenticator comme valve au descripteur **`jboss-web.xml`** si nécessaire. **`<jacc-star-role-allow>`** vous permet d'utiliser un astérisque (*) pour faire correspondre plusieurs noms de rôles, et est optionnel.

```
<jboss-web>
   <security-domain>SPNEGO</security-domain>
   <valve>
      <class-
name>org.jboss.security.negotiation.NegotiationAuthenticator</class-
name>
   </valve>
   <jacc-star-role-allow>true</jacc-star-role-allow>
</jboss-web>
```

5. **Ajouter une dépendance au `MANIFEST.MF` de votre application, pour trouver l'emplacement des classes Negotiation.**
   L'application web a besoin d'une dépendance sur la classe **`org.jboss.security.negotiation`** à ajouter au manifeste **`META-INF/MANIFEST.MF`** du déploiement pour pouvoir localiser les classes JBoss Negotiation. Ce qui suit vous montre une entrée formatée correctement.

   ```
   Manifest-Version: 1.0
   Build-Jdk: 1.6.0_24
   Dependencies: org.jboss.security.negotiation
   ```

   - Sinon, ajouter une dépendance à votre application en modifiant le fichier **`META-INF/jboss-deployment-structure.xml`** :

     ```
     <?xml version="1.0" encoding="UTF-8"?>
     <jboss-deployment-structure>
       <deployment>
         <dependencies>
      <module name='org.jboss.security.negotiation'/>
         </dependencies>
       </deployment>
     </jboss-deployment-structure>
     ```

Résultat

Votre application web accepte et authentifie les informations d'identification avec Kerberos, Active Directory de Microsoft ou autre service de répertoire compatible SPNEGO. Si l'utilisateur exécute l'application d'un système qui est déjà enregistré dans le service de répertoires, et où les rôles sont déjà appliqués à l'utilisateur, l'application web n'aura pas besoin d'authentification, et les fonctionnalités SSO seront opérationnelles.

Rapporter un bogue

14.9. CONFIGURER SPNEGO AVEC UN RENVOI À L'AUTHENTIFICATION FORM

Suivre la procédure ci-dessous pour configurer un SPNEGO avec un renvoi à l'authentification Form.

Procédure 14.2. Sécurité SPNEGO avec renvoi à l'authentification Form

1. **Configurer SPNEGO**
   Voir la procédure décrite dans Section 14.8, « Configuration de Kerberos ou Microsoft Active Directory Desktop SSO pour les applications web »

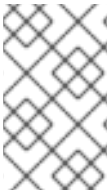
2. **Modifier le fichier `web.xml`**
   Ajouter un élément **`login-config`** à votre application et configurer les pages de connexion et d'erreurs dans le fichier web.xml :

   ```
   <login-config>
       <auth-method>SPNEGO</auth-method>
       <realm-name>SPNEGO</realm-name>
           <form-login-config>
               <form-login-page>/login.jsp</form-login-page>
   ```

```
            <form-error-page>/error.jsp</form-error-page>
        </form-login-config>
    </login-config>
```

3. **Ajouter un contenu web**
   Ajouter des references de **`login.html`** et de **`error.html`** à **`web.xml`**. Ces fichiers sont ajoutés à l'archive d'application web à l'emplacement spécifié dans la configuration **`form-login-config`**. Pour obtenir des informations, voir la section *Enable Form-based Authentication* du *Security Guide* de JBoss EAP 6. Un fichier **`login.html`** typique ressemble à ceci :

```
<html>
    <head>
        <title>Vault Form Authentication</title>
    </head>
    <body>
        <h1>Vault Login Page</h1>
        <p>
        <form method="post" action="j_security_check">
        <table>
            <tr>
                <td>Username</td><td>-</td>
                <td><input type="text" name="j_username"></td>
            </tr>
            <tr>
                <td>Password</td><td>-</td>
                <td><input type="password" name="j_password"></td>
            </tr>
            <tr>
                <td colspan="2"><input type="submit"></td>
            </tr>
        </table>
        </form>
        </p>
        <hr>
    </body>
</html>
```



NOTE

Le renvoi à la logique FORM n'est disponible que dans le cas où il n'y a pas de SPNEGO (ni de NTLM). De ce fait, un formulaire de connexion n'est pas présenté au navigateur si le navigateur envoie un token NTLM.

Rapporter un bogue

CHAPITRE 15. SSO (SINGLE SIGN ON) DANS SAML

15.1. SECURITY TOKEN SERVICE (STS)

Le service de jetons de sécurité (STS) génère et gère des jetons de sécurité. Il ne crée pas de jetons de type particulier. À la place, il définit des interfaces génériques qui autorisent l'ajout de multiples fournisseurs de jetons. De ce fait, il peut être configuré pour gérer plusieurs types de jetons, dans la mesure où un fournisseur de jetons existe pour chaque type de token. Il spécifie également le format de la requête de jeton de sécurité et les messages réponse.

Un message STS indique :

- Type de requête, Issue, Renew, etc.
- Type de jeton
- Durée de vie du jeton donné
- Informations sur le fournisseur de services qui a demandé le jeton
- Informations utilisées pour crypter le jeton créé.

NOTE

Support aux tokens PKCS#11 ajouté à JBoss EAP à partir de la version 6.3.0.

Les domaines de sécurité EAP peuvent accepter les clés PKCS#11 et les définitions de trust stores en adoptant l'attribut ***`provider`***. La valeur spécifiée est passée aux appels **`KeyStore.getInstance("PKCS11")`** pertinents; la clé et le key store sont initialisés.

La configuration de ce nouveau support n'entre pas dans le cadre de la documentation EAP. Les utilisateurs qui souhaitent utiliser cette fonction doivent se familiariser avec l'installation du matériel et des logiciels PKCS #11, ainsi que les entrées correctes nécessaires au fichier de stratégie **`java.security`**. Le document Oracle *Java PKCs #11 Guide de référence* peut être une ressource utile d'information à ce sujet.

Le message de demande de jeton est envoyé dans le corps du message SOAP. Toutes les informations liées à la demande de jeton sont contenues dans l'élément **`RequestSecurityToken`**. L'exemple de requête contient deux autres éléments WS-Trust : **`RequestType`** qui précise que cette requête est une requête d'émission et **`TokenType`** qui spécifie le type du jeton qui sera attribué.

Voici un exemple de message de requête WS-Trust.

Exemple 15.1. Message de requête de jeton de sécurité WS-Trust

```
<S11:Envelope xmlns:S11=".." xmlns:wsu=".." xmlns:wst="..">
   <S11:Header>
      ...
   </S11:Header>
   <S11:Body wsu:Id="body">
      <wst:RequestSecurityToken Context="context">

<wst:TokenType>http://www.tokens.org/SpecialToken</wst:TokenType>
         <wst:RequestType>
```

```
            http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
         </wst:RequestType>
      </wst:RequestSecurityToken>
   </S11:Body>
</S11:Envelope>
```

Voici un exemple de réponse de jeton de sécurité.

Exemple 15.2. Message de réponse de jeton de sécurité

```
    <wst:RequestSecurityTokenResponse Context="context" xmlns:wst=".."
xmlns:wsu="..">
       <wst:TokenType>http://www.tokens.org/SpecialToken</wst:TokenType>
       <wst:RequestedSecurityToken>
          <token:SpecialToken xmlns:token="...">
             ARhjefhE2FEjneovi&@FHfeoveq3
          </token:SpecialToken>
       </wst:RequestedSecurityToken>
       <wst:Lifetime>
          <wsu:Created>...</wsu:Created>
          <wsu:Expires>...</wsu:Expires>
       </wst:Lifetime>
    </wst:RequestSecurityTokenResponse>
```

Dans l'exemple de réponse de jeton de sécurité, l'élément **`TokenType`** spécifie le type du jeton émis, alors que l'élément **`RequestedSecurityToken`** contient le jeton lui-même. Le format du jeton dépend du type du jeton. L'élément de **`Lifetime`** spécifie quand le jeton a été créé et quand il expire.

Traitement de la requête de jeton de sécurité

Voici les étapes de traitement d'une requête de jeton de sécurité :

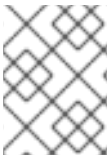
- Un client envoie une requête de jeton de sécurité à **`PicketLinkSTS`**.
- **`PicketLinkSTS`** analyse le message de requête, et génère un modèle d'objet JAXB.
- **`PicketLinkSTS`** lit le fichier de configuration et crée l'objet **`STSConfiguration`**, si nécessaire. Ensuite, il obtient une référence au **`WSTrustRequestHandler`** de la configuration et délègue le traitement de la requête à l'instance du gestionnaire de la demande.
- Le gestionnaire de requêtes utilise **`STSConfiguration`** pour définir les valeurs par défaut si nécessaire (par exemple, lorsque la demande ne spécifie pas une valeur de durée de vie de jeton)
- Le **`WSTrustRequestHandler`** crée un **`WSTrustRequestContext`**, définissant l'objet de la requête **`JAXB`** et le principal de l'appelant reçu de **`PicketLinkSTS`**.
- Le **`WSTrustRequestHandler`** utilise la **`STSConfiguration`** pour obtenir le **`SecurityTokenProvider`** qui doit être utilisé pour traiter la requête selon le type du jeton qui est demandé. Ensuite, il appelle le fournisseur, en passant le **`WSTrustRequestContext`** en tant que paramètre.

- L'instance **`SecurityTokenProvider`** traite la demande de jeton et stocke le jeton à produire dans le contexte de demande.
- Le **`WSTrustRequestHandler`** obtient le jeton du contexte, le crypte si nécessaire, et construit l'objet de réponse WS-Trust contenu dans le jeton de sécurité.
- **`PicketLinkSTS`** dicte la réponse générée par le gestionnaire de requêtes et la renvoie au client.

Rapporter un bogue

15.2. CONFIGURATION DES SECURITY TOKEN SERVICE (STS)

Le Security Token Service (STS) d'EAP définit plusieurs interfaces qui fournissent des points d'extension. Les implémentations peuvent être branchées via la configuration et des valeurs par défaut peuvent être spécifiées pour certaines propriétés par la configuration. Toutes les configurations STS sont spécifiées dans le fichier **`picketlink.xml`**, qui appartient au répertoire **`WEB-INF`** de l'application déployée. Voici les éléments qui peuvent être configurés dans le fichier **`picketlink.xml`**



NOTE

Dans le texte qui suit, un *prestaire de services* fait référence au service Web qui requiert un jeton de sécurité à présenter par ses clients.

- **`PicketLinkSTS`**: élément root. Définit des propriétés qui permettent à l'administrateur STS de définir les valeurs par défaut suivantes :
  - **`STSName`** : chaîne représentant le nom du service de jeton de sécurité. Si non spécifié, la valeur **`PicketLinkSTS`** par défaut sera utilisée.
  - **`TokenTimeout`** : la valeur de durée de vie du jeton en secondes. Si non spécifié, la valeur par défaut 3600 (une heure) sera utilisée.
  - **`EncryptToken`** : valeur booléenne spécifiant si les jetons émis doivent être cryptés ou non. La valeur par défaut est false.
- **`KeyProvider`** : cet élément et tous ses sous-éléments sont utilisés pour configurer le keystore utilisé par PicketLink STS pour signer et crypter les jetons. Les propriétés comme l'emplacement du keystore, son mot de passe et l'alias de signature (clé privée) et mot de passe sont tous configurés dans cette section.
- **`RequestHandler`** : cet élément spécifie le nom qualifié complet de l'implémentation de **`WSTrustRequestHandler`** à utiliser. Si non spécifié, la valeur par défaut **`org.picketlink.identity.federation.core.wstrust.StandardRequestHandler`** sera utilisée.
- **`TokenProvider`**: cette section spécifie les implémentations de **`TokenProvider`** qui doivent être utilisées pour gérer chaque type de token de sécurité. Dans l'exemple, nous avons deux fournisseurs - un qui gère les tokens de type **`SpecialToken`** et un autre qui gère les tokens de type **`SAMLV2.0`**. Le **`WSTrustRequestHandler`** appelle la méthode **`getProviderForTokenType`**(String type) de **`STSConfiguration`** pour obtenir une référence pour le **`TokenProvider`** qui convient.

- **`TokenTimeout`** : ceci est utilisé par le **`WSTrustRequestHandler`** quand aucune durée de vie n'a été spécifiée dans la requête de WS-Trust. Il crée une instance de durée de vie qui a l'heure actuelle comme heure de création et expire après un nombre indiqué de secondes.

- **`ServiceProviders`** : cette section spécifie les types de jetons qui doivent être utilisés pour chaque fournisseur de service (le service Web qui requiert un jeton de sécurité). Lorsqu'une demande WS-Trust ne contient pas le type de jeton, le **`WSTrustRequestHandler`** doit utiliser le point de terminaison du fournisseur de service pour déterminer le type du jeton qui doit être délivré.

- **`EncryptToken`** : ceci est utilisé par le **`WSTrustRequestHandler`** pour décider si le jeton émis doit être crypté ou non. Si true, le certificat de clé publique (PKC) du prestataire de services sera utilisé pour crypter le jeton

Ce qui suit est un exemple de configuration STS.

Exemple 15.3. Configuration STS

```
<PicketLinkSTS xmlns="urn:picketlink:identity-federation:config:1.0"
         STSName="Test STS" TokenTimeout="7200" EncryptToken="true">
  <KeyProvider
ClassName="org.picketlink.identity.federation.bindings.tomcat.KeyStoreKe
yManager">
    <Auth Key="KeyStoreURL" Value="keystore/sts_keystore.jks"/>
    <Auth Key="KeyStorePass" Value="testpass"/>
    <Auth Key="SigningKeyAlias" Value="sts"/>
    <Auth Key="SigningKeyPass" Value="keypass"/>
    <ValidatingAlias Key="http://services.testcorp.org/provider1"
Value="service1"/>
    <ValidatingAlias Key="http://services.testcorp.org/provider2"
Value="service2"/>
 </KeyProvider>
 <RequestHandler>org.picketlink.identity.federation.core.wstrust.Standar
dRequestHandler</RequestHandler>
 <TokenProviders>
    <TokenProvider
ProviderClass="org.picketlink.test.identity.federation.bindings.wstrust.
SpecialTokenProvider"
         TokenType="http://www.tokens.org/SpecialToken"/>
    <TokenProvider
ProviderClass="org.picketlink.identity.federation.api.wstrust.plugins.sa
ml.SAML20TokenProvider"
         TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-
token-profile-1.1#SAMLV2.0"/>
 </TokenProviders>
 <ServiceProviders>
  <ServiceProvider Endpoint="http://services.testcorp.org/provider1"
TokenType="http://www.tokens.org/SpecialToken"
         TruststoreAlias="service1"/>
  <ServiceProvider Endpoint="http://services.testcorp.org/provider2"
TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV2.0"
         TruststoreAlias="service2"/>
 </ServiceProviders>
</PicketLinkSTS>
```

Rapporter un bogue

15.3. MODULES DE CONNEXION DE PICKETLINK STS

Le module de connexion de PicketLink est normalement configuré dans le cadre de l'installation du conteneur JEE pour utiliser un Service de jetons de sécurité (STS) pour l'authentification des utilisateurs. Le STS peut être colocalisé sur le même conteneur que le Module de connexion ou accessible à distance via les appels de Service Web ou une autre technologie. Les modules de connexion de PicketLink prennent en charge les implémentations STS non - PicketLink via des appels standards de WS-Trust.

Types de modules de connexion STS

Voici les différents types de modules de connexion STS.

STSIssuingLoginModule

- Appelle le STS configuré et demande un jeton de sécurité. Dès la réception du jeton **`RequestedSecurityToken`**, l'authentification est considérée comme un succès.
- Un appel à STS exige normalement une authentification. Ce module de connexion utilise les informations d'authentificaition d'une des sources suivantes :
  - Son fichier de propriétés, si l'option de module **`useOptionsCredentials`** est définie à **`true`**.
  - Anciennes informations d'authentification de module de connexion si l'option de module **`password-stacking`** est définie à **`useFirstPass`**.
  - Du **`CallbackHandler`** configuré en donnant un nom et un mot de passe de Callback.
- Après une authentification réussie, le **`SamlCredential`** est inséré dans les informations d'authentification du sujet si une même Assertion n'est pas déjà présente.

STSValidatingLoginModule

- Appelle le STS configuré et valide un token de sécurité disponible.
- Un appel à STS exige normalement une authentification. Ce module de connexion utilise les informations d'authentificaition d'une des sources suivantes :
  - Son fichier de propriétés, si l'option de module **`useOptionsCredentials`** est définie à **`true`**.
  - Anciennes informations d'authentification de module de connexion si l'option de module **`password-stacking`** est définie à **`useFirstPass`**.
  - Du **`CallbackHandler`** configuré en donnant un nom et un mot de passe de Callback.
- Après une authentification réussie, SamlCredential est inséré dans les informations d'authentification du sujet si une même Assertion n'est pas déjà présente.

SAML2STSLoginModule

- Ce module de connexion fournit un **`ObjectCallback`** au **`CallbackHandler`** configuré et attend un objet **`SamlCredential`** retour. L'assertion est validée en rapport au STS configuré.
- Si un ID utilisateur ou un jeton SAML sont partagés, ce module de connexion contournera la validation. S'il sont empilés l'un sur l'autre, connectez-vous au module qui est authentifié.
- Après une authentification réussie, le **`SamlCredential`** est inspecté pour trouver un **`NameID`** et un attribut de rôle à valeur multiples qui est défini respectivement comme ID et rôles de l'utilisateur.

SAML2LoginModule

- Ce module de connexion est utilisé en conjonction avec d'autres composants pour l'authentification SAML et ne réalise aucune authentification.
- **`SPRedirectFormAuthenticator`** utilise ce module de connexion pour l'implémentation PicketLink du profil de redirection SAML v2 HTTP.
- La valve d'authentificateur Tomcat effectue un authentification en redirigeant vers le fournisseur d'identité et en obtenant une assertion SAML.
- Ce module de connexion est utilisé pour faire passer l'ID utilisateur et les rôles au JBoss Security Framework dans le sujet JAAS.

Rapporter un bogue

15.4. CONFIGURER STSISSUINGLOGINMODULE

`STSIssuingLoginModule` utilise un nom d'utilisateur et un mot de passe pour authentifier l'utilisateur par un STS en extrayant un jeton.

Exemple 15.4. Configurer STSIssuingLoginModule

```
<security-domain name="saml-issue-token">
    <authentication>
        <login-module

code="org.picketlink.identity.federation.core.wstrust.auth.STSIssuingLog
inModule" flag="required">          <module-option
name="configFile">./picketlink-sts-client.properties</module-option>
          <module-option
name="endpointURI">http://security_saml/endpoint</module-option>
        </login-module>
    </authentication>
    <mapping>
        <mapping-module

code="org.picketlink.identity.federation.bindings.jboss.auth.mapping.STS
PrincipalMappingProvider"
            type="principal" />
        <mapping-module

code="org.picketlink.identity.federation.bindings.jboss.auth.mapping.STS
GroupMappingProvider"
```

```
            type="role" />
    </mapping>
</security-domain>
```

La plupart des configurations peuvent basculer vers la configuration dans l'exemple ci-dessus en :

- modifiant leur security-domain déclaré
- spécifiant un fournisseur de mappage Principal
- spécifiant un fournisseur de mappage RoleGroup

Le fournisseur de mappage Principal et le fournisseur de mappage Role Group spécifiés résultent en Subject authentifié en mesure d'accepter les autorisations basées rôle et de granularité grossière . Après l'authentification, le jeton de sécurité est rendu disponible et peut être utilisé pour appeler les autres services de Single Sign-On.

Rapporter un bogue

15.5. CONFIGURER STSVALIDATINGLOGINMODULE

STSValidatingLoginModule utilise un TokenCallback pour demander au CallBackHandler un STS en extrayant un jeton.

Exemple 15.5. Configurer STSValidatingLoginModule

```
<security-domain name="saml-validate-token">
    <authentication>
        <login-module

code="org.picketlink.identity.federation.core.wstrust.auth.STSValidating
LoginModule" flag="required">
            <module-option name="configFile">./picketlink-sts-
client.properties</module-option>
            <module-option
name="endpointURI">http://security_saml/endpoint</module-option>
        </login-module>
    </authentication>
    <mapping>
        <mapping-module

code="org.picketlink.identity.federation.bindings.jboss.auth.mapping.STS
PrincipalMappingProvider"
            type="principal" />
        <mapping-module

code="org.picketlink.identity.federation.bindings.jboss.auth.mapping.STS
GroupMappingProvider"
            type="role" />
    </mapping>
</security-domain>
```

La configuration citée dans l'exemple active Single Sign-On pour vos applications et services. Une fois délivré, soit en communiquant directement avec STS ou via un module de connexion d'émission de jeton, un jeton peut être utilisé pour authentifier plusieurs applications et services en utilisant le programme d'installation fourni dans l'exemple. Avec un fournisseur de mappage Principal et un fournisseur de mappage Role Group, on a un sujet authentifié qui permet d'avoir des autorisations à granularité grossière ou basées rôles. Après l'authentification, le jeton de sécurité sera disponible et pourra être utilisé pour appeler d'autres services Single Sign-On.

Rapporter un bogue

15.6. MISE EN POOL DE CLIENTS STS

PicketLink fournit un pool de clients STS sur le serveur. Ainsi, la création de Client STS n'est plus source de goulot d'étranglement.

La mise en pool de clients peut être utilisée par des modules de connexion qui ont besoin d'un client STS pour obtenir des tickets de SAML.

Modules de connexion qui peuvent utiliser une mise en pool de clients STS :

- org.picketlink.identity.federation.core.wstrust.auth.STSIssuingLoginModule
- org.picketlink.identity.federation.core.wstrust.auth.STSValidatingLoginModule
- org.picketlink.trust.jbossws.jaas.JBWSTokenIssuingLoginModule

Le nombre de client dans le pool pour chaque module de connexion par défaut est configuré par l'option de module de connexion **`initialNumberOfClients`**.

Le classe STSClientPoolFactory **`org.picketlink.identity.federation.bindings.stspool.STSClientPoolFactory`** fournit une fonctionnalité de mise en pool de clients aux applications.

Utilisation de STSClientPoolFactory

Les clients STS sont insérés dans des sous-pool en utilisant comme clé configuration. Obtenez une instance de STSClientPool et initialiser un sous-pool sur la base de la configuration, avec le nombre de clients STS initial en option, ou bien basez-vous sur le nombre par défaut.

```
final STSClientPool pool = STSClientPoolFactory.getPoolInstance();
pool.createPool(20, stsClientConfig);
final STSClient client = pool.getClient(stsClientConfig);
```

Quand vous aurez terminé avec un client, vous pouvez le renvoyer au pool de la manière suivante :

```
pool.returnClient();
```

Pour vérifier si un sous-pool existe déjà pour une configuration donnée :

```
if (! pool.configExists(stsClientConfig) {
    pool.createPool(stsClientConfig);
}
```

Lorsque le sous-système Fédération PicketLink est activé, toutes les pools de client créés pour un déploiement sont détruits automatiquement pendant le processus d'annulation du déploiement. Pour détruire un pool manuellement :

```
pool.destroyPool(stsClientConfig);
```

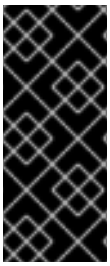
Rapporter un bogue

15.7. NAVIGATEUR WEB SAML BASÉ SSO

15.7.1. SAML Web Browser Basé SSO

Dans JBoss EAP, PicketLink fournit une plateforme d'implémentation de services basés identité fédérés. Inclut les services d'identité centralisés et Single Sign On (SSO) pour les applications.

Le profil SAML a un support pour les liaisons HTTP/POST et les liaisons de redirection HTTP avec des services d'identité centralisés pour activer le web SSO pour vos applications. L'architecture pour le SAML v2 baséeWeb SSO suit l'architecture hub and spoke de gestion des identités. Dans cette architecture, un fournisseur d'identité (IDP) agit comme la source centrale (hub) pour l'identité et les informations de rôle à toutes les applications (fournisseurs de services). Les spokes sont les fournisseurs de services (SP).



IMPORTANT

S'il y a deux ou plusieurs SP pointant vers le même IDP, l'IDP ne distingue pas entre les différents SP. Si vous faites des demandes à différents SP qui pointent vers le même IDP, l'IDP traitera la requête la plus récente d'SP et enverra une assertion SAML sur l'utilisateur authentifié. Pour revenir à une requête SP plus ancienne, vous devrez ré-entrer l'URL de SP dans le navigateur.

Rapporter un bogue

15.7.2. Installation de Web SSO basé SAML v2

Pour Installer Web SSO basé SAML v2, vous devrez configurer ce qui suit :

- Identity Provider: le fournisseur d'identité est l'entité faisant autorité responsable de l'authentification d'un utilisateur final et qui affirme l'identité de cet utilisateur de façon fiable à des partenaires de confiance.
- Service Provider: Le fournisseur de Service s'appuie sur le fournisseur d'identité pour affirmer des informations sur un utilisateur via une information d'identification de l'utilisateur électronique, laissant le prestataire de services gérer le contrôle d'accès et de diffusion sur la base d'un ensemble fiable d'affirmations d'identification utilisateur.

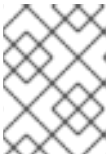
Rapporter un bogue

15.7.3. Configurer le fournisseur d'identités

IDP (Identity Provider) ou fournisseur d'identité est une instance de serveur JBoss EAP.

Procédure 15.1. Configurer le fournisseur d'identités IDP

1. **Configurer la sécurité de l'application web pour l'IDP**
   Configurer une application web comme fournisseur d'identité.



NOTE

L'utilisation de la sécurité d'application web basée FORM est conseillée car elle donne la possibilité de personnaliser une page de connexion.

Ce qui suit est un exemple de configuration **`web.xml`**

Exemple 15.6. Configuration `web.xml` dans IDP

```
<display-name>IDP</display-name>
<description>IDP</description>
<!-- Define a security constraint that gives unlimited access to
images -->
<security-constraint>
  <web-resource-collection>
  <web-resource-name>Images</web-resource-name>
  <url-pattern>/images/*</url-pattern>
</web-resource-collection>
</security-constraint>
<!-- Define a Security Constraint on this Application -->
<security-constraint>
  <web-resource-collection>
  <web-resource-name>IDP</web-resource-name>
  <url-pattern>/*</url-pattern>
</web-resource-collection>
  <auth-constraint>
  <role-name>manager</role-name>
</auth-constraint>
</security-constraint>
<!-- Define the Login Configuration for this Application -->
<login-config>
  <auth-method>FORM</auth-method>
  <realm-name>IDP Application</realm-name>
  <form-login-config>
    <form-login-page>/jsp/login.jsp</form-login-page>
    <form-error-page>/jsp/loginerror.jsp</form-error-page>
  </form-login-config>
</login-config>
<!-- Security roles referenced by this web application -->
<security-role>
 <description>
  The role that is required to log in to the IDP Application
 </description>
 <role-name>manager</role-name>
</security-role>
</web-app>
```

2. **Créer un domaine de sécurité pour IDP**
   Créer un domaine de sécurité avec des mécanismes d'authentification et d'autorisation définis pour IDP. Voir Section 13.9, « Utiliser un domaine de sécurité dans votre application » pour obtenir des informations supplémentaires.

3. **Configurer les Valves IDP**

Créer un fichier **jboss-web.xml** dans le répertoire **WEB-INF** de votre application web IDP pour configurer les valves de l'IDP. Ce qui suit est un exemple du fichier **jboss-web.xml**.

Exemple 15.7. Configuration de fichier jboss-web.xml pour les valves IDP

```
<jboss-web>
  <security-domain>idp</security-domain>
  <context-root>idp</context-root>
  <valve>
    <class-
name>org.picketlink.identity.federation.bindings.tomcat.idp.IDPWeb
BrowserSSOValve</class-name>
  </valve>
</jboss-web>
```

4. **Configurer le fichier de configuration PicketLink (picketlink.xml)**
   Voici un exemple de configuration **picketlink-idfed.xml**. Dans ce fichier de configuration, vous fournissez l'URL ajouté comme émetteur dans les assertions SAML2 sortantes pour les fournisseurs de services et l'IDP.

Exemple 15.8. Configuration picketlink.xml

```
<PicketLink xmlns="urn:picketlink:identity-federation:config:2.1">
  <PicketLinkIDP xmlns="urn:picketlink:identity-
federation:config:2.1">
    <IdentityURL>http://localhost:8080/idp/</IdentityURL>
  </PicketLinkIDP>
  <Handlers xmlns="urn:picketlink:identity-
federation:handler:config:2.1">
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2
IssuerTrustHandler" />
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2
LogOutHandler" />
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2
AuthenticationHandler" />
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.Roles
GenerationHandler" />
  </Handlers>
</PicketLink>
```

Par défaut, **picketlink.xml** se situe dans le répertoire **WEB-INF** de votre application web IDP. Cependant, vous pourrez configurer un chemin personnalisé vers un **picketlink.xml** externe à l'application :

a. **Option : configurer un chemin personnalisé vers picketlink.xml**

Ajouter deux paramètres à l'élément valve du **WEB-INF/jboss-web.xml** de votre application : **configFile** indiquant le chemin vers **picketlink.xml**, et **timerInterval** qui indique l'intervalle qu'il faut pour charger la configuration en millisecondes. Par exemple :

```
<valve>
  <class-name>...</class-name>
    <param>
      <param-name>timerInterval</param-name>
      <param-value>5000</param-value>
    </param>
    <param>
      <param-name>configFile</param-name>
      <param-value>path-to/picketlink.xml</param-value>
    </param>
</valve>
```

5. **Déclare les dépendances sur le module PicketLink (META-INF/MANIFEST.MF, ou jboss-deployment-structure.xml)**
   L'application web exige aussi qu'une dépendance soit définie dans **META-INF/MANIFEST.MF** ou dans **jboss-deployment-structure.xml** pour trouver les classes de PicketLink puissent être localisées.

   Exemple 15.9. Définir la dépendance dans META-INF/MANIFEST.MF

   ```
   Manifest-Version: 1.0
       Build-Jdk: 1.6.0_24
       Dependencies: org.picketlink
   ```

   Exemple 15.10. Définir la dépendance dans META-INF/jboss-deployment-structure.xml

   ```
   <jboss-deployment-structure>
     <deployment>
       <dependencies>
         <module name="org.picketlink" />
       </dependencies>
     </deployment>
   </jboss-deployment-structure>
   ```

Rapporter un bogue

15.7.4. Configuration du fournisseur de services (SP) qui utilise la liaison HTTP/POST

Le fournisseur de services ou Service Provider (SP) peut correspondre à une instance de serveur JBoss EAP.

Procédure 15.2. Configuration du fournisseur de services (SP)

1. **Configurer la sécurité de l'application web pour le SP**

L'application web à configurer en tant que SP doit avoir une sécurité basée FORM activée dans son fichier **web.xml**.

Exemple 15.11. Configuration web.xml dans SP

```
<display-name>SP</display-name>
<description>SP</description>
<!-- Define a security constraint that gives unlimited access to
images -->
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Images</web-resource-name>
    <url-pattern>/images/*</url-pattern>
  </web-resource-collection>
</security-constraint>
<!-- Define a Security Constraint on this Application -->
<security-constraint>
  <web-resource-collection>
    <web-resource-name>SP</web-resource-name>
    <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>manager</role-name>
  </auth-constraint>
</security-constraint>
<!-- Define the Login Configuration for this Application -->
<login-config>
  <auth-method>FORM</auth-method>
  <realm-name>SP Application</realm-name>
  <form-login-config>
    <form-login-page>/jsp/login.jsp</form-login-page>
    <form-error-page>/jsp/loginerror.jsp</form-error-page>
  </form-login-config>
</login-config>
<!-- Security roles referenced by this web application -->
<security-role>
    <description>
   The role that is required to log in to the SP Application
  </description>
  <role-name>manager</role-name>
</security-role>
</web-app>
```

2. **Créer un domaine de sécurité pour SP**
   Créer un domaine de sécurité qui utilise **SAML2LoginModule**. Voici un exemple de configuration :

```
<security-domain name="sp" cache-type="default">
  <authentication>
    <login-module
code="org.picketlink.identity.federation.bindings.jboss.auth.SAML2Lo
ginModule" flag="required"/>
   </authentication>
</security-domain>
```

3. **Configurer la valve SP**
   Pour configurer la valve du SP, créer un fichier **`jboss-web.xml`** dans le répertoire **`WEB-INF`** de votre application web SP.

   Exemple 15.12. Configuration de fichier `jboss-web.xml` pour les valves SP

   ```
   <jboss-web>
     <security-domain>sp</security-domain>
     <context-root>sales-post</context-root>
     <valve>
       <class-
   name>org.picketlink.identity.federation.bindings.tomcat.sp.Service
   ProviderAuthenticator</class-name>
     </valve>
   </jboss-web>
   ```

4. **Configurer le fichier de configuration PicketLink (`picketlink.xml`)**
   Voici un exemple de configuration **`picketlink-idfed.xml`** pour le SP. Dans ce fichier de configuration, vous fournissez l'URL pour le SP et l'IDP, avec les handlers correspondants pour le SP.

   Exemple 15.13. Configuration `picketlink.xml`

   ```
   <PicketLink xmlns="urn:picketlink:identity-federation:config:2.1">
     <PicketLinkSP xmlns="urn:picketlink:identity-
   federation:config:2.1" ServerEnvironment="tomcat"
   BindingType="REDIRECT">
       <IdentityURL>${idp.url::http://localhost:8080/idp/}
   </IdentityURL>
       <ServiceURL>${sales-post.url::http://localhost:8080/sales-
   post/}</ServiceURL>
     </PicketLinkSP>
     <Handlers xmlns="urn:picketlink:identity-
   federation:handler:config:2.1">
       <Handler
   class="org.picketlink.identity.federation.web.handlers.saml2.SAML2
   LogOutHandler" />
       <Handler
   class="org.picketlink.identity.federation.web.handlers.saml2.SAML2
   AuthenticationHandler" />
       <Handler
   class="org.picketlink.identity.federation.web.handlers.saml2.Roles
   GenerationHandler" />
     </Handlers>
   </PicketLink>
   ```

   Par défaut, **`picketlink.xml`** se situe dans le répertoire **`WEB-INF`** de votre application. Cependant, vous pourrez configurer un chemin personnalisé vers un **`picketlink.xml`** externe à l'application :

a. **Option : configurer un chemin personnalisé vers `picketlink.xml`**
Ajouter deux paramètres à l'élément valve du **`WEB-INF/jboss-web.xml`** de votre application : **`configFile`** indiquant le chemin vers **`picketlink.xml`**, et **`timerInterval`** qui indique l'intervalle qu'il faut pour charger la configuration en millisecondes. Par exemple :

```
<valve>
  <class-name>...</class-name>
    <param>
      <param-name>timerInterval</param-name>
      <param-value>5000</param-value>
    </param>
    <param>
      <param-name>configFile</param-name>
      <param-value>path-to/picketlink.xml</param-value>
    </param>
</valve>
```

5. **Déclare les dépendances sur le module PicketLink (`META-INF/MANIFEST.MF`, ou `jboss-deployment-structure.xml`)**
L'application web exige aussi qu'une dépendance soit définie dans **`META-INF/MANIFEST.MF`** ou dans **`jboss-deployment-structure.xml`**. pour trouver les classes de PicketLink puissent être localisées.

Exemple 15.14. Définir la dépendance dans `META-INF/MANIFEST.MF`

```
Manifest-Version: 1.0
    Build-Jdk: 1.6.0_24
    Dependencies: org.picketlink
```

Exemple 15.15. Définir la dépendance dans `META-INF/jboss-deployment-structure.xml`

```
<jboss-deployment-structure>
  <deployment>
    <dependencies>
      <module name="org.picketlink" />
    </dependencies>
  </deployment>
</jboss-deployment-structure>
```

Rapporter un bogue

15.7.5. Installation de Web SSO basé SAML v2 avec HTTP/POST Binding

HTTP/POST binding est la liaison conseillée pour obtenir le navigateur web basé SSO.

Procédure 15.3. Installation de Web SSO basé SAML v2 avec HTTP/POST Binding

1. **Configurer le fournisseur d'identités (IDP).**

Les étapes de configuration d'IDP pour HTTP/POST Binding sont les mêmes que pour HTTP/Redirect Binding. Pour plus d'informations sur la façon de configurer l'IDP, voir Section 15.7.2, « Installation de Web SSO basé SAML v2 »

2. **Configuration du fournisseur de services (SP)**
   Les étapes de configuration de SP pour HTTP/POST Binding sont les mêmes que pour HTTP/Redirect Binding, excepté la variation qui existe dans le fichier **`picketlink.xml`** du SP. Modifiez **`BindingType="REDIRECT"`** en **`BindingType="POST"`**.

   Pour plus d'informations sur la configuration du SP, voir Section 15.7.4, « Configuration du fournisseur de services (SP) qui utilise la liaison HTTP/POST ».

Rapporter un bogue

15.7.6. Configurer un sélecteur de comptes dynamiques (Dynamic Account Chooser) auprès d'un fournisseur de services

Conditions préalables :

- Section 15.7.3, « Configurer le fournisseur d'identités »
- Section 15.7.4, « Configuration du fournisseur de services (SP) qui utilise la liaison HTTP/POST »

Si un SP (Service Provider) est configuré avec plusieurs IDP (Identity Providers), PicketLink peut être configuré pour inviter l'utilisateur à choisir l'IDP à utiliser pour authentifier leurs identitfiants.

Procédure 15.4. Configurer un sélecteur de comptes dynamiques (Dynamic Account Chooser) auprès d'un fournisseur de services

1. Configurer la valve **`jboss-web.xml`** dans le répertoire **`WEB-INF`** de votre application web SP.

   Exemple 15.16. Configuration de fichier `jboss-web.xml` pour les Sélecteurs de comptes SP

```
<jboss-web>
  <security-domain>sp</security-domain>
  <context-root>accountchooser</context-root>
  <valve>
    <class-
name>org.picketlink.identity.federation.bindings.tomcat.sp.Account
ChooserValve</class-name>
  </valve>
  <valve>
    <class-
name>org.picketlink.identity.federation.bindings.tomcat.sp.Service
ProviderAuthenticator</class-name>
  </valve>
</jboss-web>
```

   `AccountChooserValve` a les options de configuration suivantes :

   DomainName

Le nom de domaine à utiliser par le cookie envoyé dans le navigateur de l'utilisateur.

CookieExpiry

La durée d'expiration du cookie en secondes. La valeur par défaut est **`-1`**, ce qui veut dire que le cookie expire quand le navigateur est fermé.

AccountIDPMapProvider

Le nom complet de l'implémentation d'IDP Mapping. La valeur par défaut correspond à un fichier de propriétés **`idpmap.properties`** qui se trouve dans le répertoire **`WEB-INF`** directory of de votre application web SP. Cette implementation doit implémenter **`org.picketlink.identity.federation.bindings.tomcat.sp.AbstractAccountChooserValve.AccountIDPMapProvider`**.

AccountChooserPage

Le nom de la page HTML/JSP qui liste les divers comptes IDP. La valeur par défaut est **`/accountChooser.html`**.

2. Définir le mappage des IDP. Il s'agit par défaut d'un fichier de propriétés **`idpmap.properties`** qui se trouve dans le répertoire **`WEB-INF`** de votre application web SP.

Exemple 15.17. Configuration `idpmap.properties`

```
DomainA=http://localhost:8080/idp1/
DomainB=http://localhost:8080/idp2/
```

3. Créer une page HTML dans votre application web de SP pour que l'utilisateur puisque choisir l'IDP. Par défaut, ce fichier est **`accountChooser.html`**. L'URL de chacun des IDP doit avoir le paramètre **`idp`** qui spécifie le nom de l'IDP répertorié dans **`idpmap.properties`**.

Exemple 15.18. `accountChooser.html` Configuration

```
<html>
  ...
  <a href="?idp=DomainA">DomainA</a>
  <hr/>
  <a href="?idp=DomainB">DomainB</a>
  ...
</html>
```

Rapporter un bogue

15.7.7. Configuration d'un SSO initié par IDP

Conditions préalables :

- Section 15.7.3, « Configurer le fournisseur d'identités »

- Section 15.7.4, « Configuration du fournisseur de services (SP) qui utilise la liaison HTTP/POST »

Dans PicketLink, le SP démarre habituellement le mouvement en envoyant une demande d'authentification à l'IDP, qui, à son tour, envoie une réponse SAML à SP avec une assertion valide. Ce mouvement s'appelle un SSO initié par SP. Mais les specs de SAML 2.0 définit également un autre flux, appelé initié par IDP ou une réponse SSO non sollicitée. Dans ce scénario, le SP n'initie pas le mouvement d'authentification et reçoit une réponse SAML de l'IDP. Le mouvement commence du côté IDP et une fois authentifié, l'utilisateur peut choisir un SP spécifique dans une liste pour être ensuite redirigé vers son URL.

Procédure

1. L'utilisateur accède à l'IDP.
2. L'IDP qui s'aperçoit qu'il n'y a aucune requête, ni aucune réponse SAML assume qu'il s'agit d'un premier scénario IDP utilisant SAML.
3. L'IDP challenge l'utilisateur de s'authentifier.
4. Une fois authentifié, l'IDP montre la section hébergée où l'utlisateur obtient une page qu'il relie à toutes les applications SP.
5. L'utilisateur sélectionne une application SP.
6. L'IDP redirige l'utilisateur vers le fournisseur de services par l'intermédiaire d'une assertion SAML dans le paramètre de requête, réponse SAML.
7. Le SP vérifie l'assertion SAML et fournit l'accès.

Configuration

Aucune configuration spéciale n'est requise la prise en charge des Réponses non sollicitées, donc vous pouvez configurer votre IDP et vos SP comme d'habitude. Pour obtenir plus d'informations sur la façon de configurer IDP et SP, consulter :

- Section 15.7.3, « Configurer le fournisseur d'identités »
- Section 15.7.4, « Configuration du fournisseur de services (SP) qui utilise la liaison HTTP/POST »

Comment utiliser

Une fois que l'utilisateur est authentifié, l'IDP montrera une page contenant les liens menant à toutes les applications du fournisseur. Un lien devrait ressembler à ce qui suit :

```
<a href="http://localhost:8080/idp?
SAML_VERSION=2.0&TARGET=http://localhost:8080/sales-post/">Sales</a>
```

Notez que le lien ci-dessus redirige l'utilisateur vers l'IDP en passant le paramètre de requête CIBLE, dont la valeur correspond à l'URL de l'application SP cible. Une fois que l'utilisateur clique sur le lien ci-dessus, l'IDP extrait le paramètre CIBLE de la requête, génère une réponse SAML v2.0 et redirige l'utilisateur vers l'URL cible. Lorsque l'utilisateur accède au SP, il est automatiquement authentifié.

Vous pouvez utiliser un paramètre de requête SAML_VERSION pour spécifier la version SAML qui doit être utilisée par l'IDP pour créer une réponse SAML. Le paramètre de SAM_VERSION peut avoir pour option 2.0 ou 1.1.

Rapporter un bogue

15.8. CONFIGURATION DU PROFIL SAML GLOBAL LOGOUT

Une déconnexion Global Logout initiée au niveau d'un service fournisseur déconnecte l'utilisateur du fournisseur d'identité (PDI) et de tous les fournisseurs de services.

NOTE

Pour qu'un Global Logout fonctionne correctement, veillez à un maximum de cinq Fournisseurs de service par IDP.

Procédure 15.5. Configuration du Global Logout

1. **Configurer picketlink-handlers.xml**
   Ajouter le **`SAML2LogOutHandler`** dans le picketlink-handlers.xml.

2. **Configurer la page web du fournisseur de services**
   Ajouter **`GLO=true`** au lien qui se trouve en bas de votre page web de fournisseur de service.

   Exemple 15.19. Créer un lien vers Global Logout

   ```
   <a href="?GLO=true">Click to Globally LogOut</a>
   ```

3. **Créer une page `logout.jsp`**
   Dans le cadre du processus de déconnexion, PicketLink redirigera l'utilisateur vers une page **`logout.jsp`** qui se trouve dans le répertoire root de votre application de Service Provider. Veillez à ce que cette page soit bien créée.

Rapporter un bogue

CHAPITRE 16. MODULES DE CONNEXION

Rapporter un bogue

16.1. UTILISATION DES MODULES

JBoss EAP 6 comprend plusieurs modules de connexion groupés adaptés aux besoins de gestion de la plupart des utilisateurs. JBoss EAP 6 peut lire les informations utilisateur d'une base de données relationnelle, d'un serveur LDAP ou de fichiers plats. En plus de ces modules de connexion de base, JBoss EAP 6 fournit des autres modules de connexion qui fournissent des informations utilisateur de besoins très personnalisés.

On peut trouver des modules de connexion supplémentaires et leurs options dans l'annexe A.1.

Rapporter un bogue

16.1.1. Empilage des mots de passe

Plusieurs modules de connexion peuvent être rasemblés dans une pile, avec chaque module fournissant les deux informations suivantes pendant l'authentification : vérification des identifiants et allocation de rôles. Fonctionne pour de nombreux cas d'utilisation, mais parfois la vérification des identifiants et l'allocation des rôles sont répartis entre plusieurs banques de gestion d'utilisateurs.

Section 16.1.4, « Module de connexion Ldap » décrit comment combiner LDAP et une base de données relationnelle, permettant à un utilisateur d'être authentifié par l'un ou l'autre système. Considérons le cas où les utilisateurs sont gérés dans un serveur LDAP central, mais où les rôles propres à l'application sont stockés dans la base de données relationnelle de l'application. L'option de module d'empilage de mots de passe capture cette relation.

Pour utiliser la fonction d'empilage de mots de passe, chaque module de connexion devra avoir l'attribut <module-option> **`password-stacking`** défini à **`useFirstPass`**. Si un module précédemment configuré pour l'empilage de mots de passe a authentifié l'utilisateur, tous les autres modules d'empilage tiendront compte de l'utilisateur authentifié et ne feront que de fournir des rôles pour l'étape de l'autorisation.

Quand l'option **`password-stacking`** est définie à **`useFirstPass`**, ce module commence par chercher le nom d'utilisateur et le mot de passe en commun sous les noms de propriété javax.security.auth.login.name et javax.security.auth.login.password respectivement dans la mappe d'états partagés du module de connexion.

S'il les trouve, ces propriétés seront utilisées comme nom et mot de passe de principal. S'il ne les trouve pas, le nom et le mot de passe du principal seront définis par le module de connexion et seront stockés sous les noms de propriété javax.security.auth.login.name et javax.security.auth.login.password respectivement

NOTE

Quand on utilise la fonction d'empilage de mots de passe, il faut définir tous les modules requis. Cela garantit qu'aucun module ne sera oublié, et pourra contribuer à des rôle utiles au processus d'autorisation.

> **Exemple 16.1. Exemple d'empilage de mots de passe**
>
> Ce CLI de gestion vous montre comment utliser l'empilage des mots de passe.

```
/subsystem=security/security-
domain=pwdStack/authentication=classic/login-module=Ldap:add( \
  code=Ldap, \
  flag=required, \
  module-options=[ \
    ("password-stacking"=>"useFirstPass"), \
    ... Ldap login module configuration
  ])
/subsystem=security/security-
domain=pwdStack/authentication=classic/login-module=Database:add( \
  code=Database, \
  flag=required, \
  module-options=[ \
    ("password-stacking"=>"useFirstPass"), \
    ... Database login module configuration
  ])
```

Rapporter un bogue

16.1.2. Hachage de mots de passe

La plupart des module de connexion doivent faire correspondre un mot de passe fourni par un client à un mot de passe mis dans un système de gestion utilisateur. Ces modules fonctionnent normalement avec des mots de passe en texte brut, mais peuvent être configurés pour prendre en charge les mots de passe hachés pour que les mots de passe en texte brut ne puissent pas être stockés côté serveur.

IMPORTANT

La version certifiée de Red Hat JBoss Enterprise Application Platform Common Criteria ne prend en charge que SHA-256 pour le hachage des mots de passe.

Exemple 16.2. Hachage de mots de passe

Ce qui suit est une configuration de module de connexion qui assigne aux utilisateurs authentifiés le nom de principal **nobody** et qui contient des hachages codé based64, SHA-256, des mots de passe dans un fichier **`usersb64.properties`**. Le fichier **`usersb64.properties`** fait partie du chemin de classe de déploiement.

```
/subsystem=security/security-domain=testUsersRoles:add
/subsystem=security/security-
domain=testUsersRoles/authentication=classic:add
/subsystem=security/security-
domain=testUsersRoles/authentication=classic/login-
module=UsersRoles:add( \
  code=UsersRoles, \
  flag=required, \
  module-options=[ \
    ("usersProperties"=>"usersb64.properties"), \
    ("rolesProperties"=>"test-users-roles.properties"), \
    ("unauthenticatedIdentity"=>"nobody"), \
```

```
      ("hashAlgorithm"=>"SHA-256"), \
      ("hashEncoding"=>"base64") \
    ])
```

hashAlgorithm

Nom de l'algorithme **`java.security.MessageDigest`** à utiliser pour hacher le mot de passe. Il n'y a aucune valeur par défaut, donc cette option doit être spécifiée pour permettre le hachage. Les valeurs typiques sont le **`SHA-256`**, **`SHA-1`** et **`MD5`**.

hashEncoding

String qui spécifie un des trois types d'encodage : **`base64`**, **`hex`** ou **`rfc2617`**. La valeur par défaut est **`base64`**.

hashCharset

Groupe de caractères utilisé pour convertir le mot de passe en texte clair en tableau d'octets. L'encodage par défaut de la plateforme correspond à la valeur par défaut.

hashUserPassword

Indique que l'algorithme de hachage doit s'appliquer au mot de passe que l'utilisateur soumet. Le hachage de mot de passe est comparée à la valeur du module de connexion, ce qui est censé être un hachage du mot de passe. La valeur par défaut est **`true`**.

hashStorePassword

Indique que l'algorithme de hachage doit s'appliquer au mot de passe stocké sur le serveur. Il est utilisé pour l'authentification digest, lorsque l'utilisateur soumet un hachage du mot de passe utilisateur ainsi qu'un jeton spécifique à la demande du serveur pour comparaison. L'algorithme de hachage (pour digest, il s'agit de **`rfc2617`**) sert à calculer un hachage du côté serveur, qui doit correspondre à la valeur de hachage envoyée par le client.

Si vous devez générer des mots de passe en code, la classe **`org.jboss.security.auth.spi.Util`** fournit une méthode d'assistance statique qui servira au hachage de mot de passe à l'aide du codage spécifié. L'exemple suivant génère un mot de passe MD5 haché codé en base64.

```
String hashedPassword = Util.createPasswordHash("SHA-256",
 Util.BASE64_ENCODING, null, null, "password");
```

OpenSSL fournit un autre moyen de générer rapidement des mots de passe hachés à la ligne de commande. L'exemple suivant génère également un hachage de mot de passe SHA-256, encodé en base 64. Ici, le mot de passe en texte brut - **`password`** - est acheminé dans la fonction digest OpenSSL, puis dans une autre fonction OpenSSL pour se convertir au format codé en base 64.

```
echo -n password | openssl dgst -sha256 -binary | openssl base64
```

Dans les deux cas, la version hachée du mot de passe est la même : **`XohImNooBHFR0OVvjcYpJ3NgPQ1qq73WKhHvch0VQtg=`**. Cette valeur devra être stockée dans le fichier de propriété des utilisateurs, dans le domaine de sécurité - **`usersb64.properties`** - comme dans l'exemple ci-dessus.

Rapporter un bogue

16.1.3. Identité non authentifée

Toutes les demandes ne sont pas toutes reçues dans un format authentifié. **`unauthenticatedIdentity`** est une option de configuration de module de login qui attribue une identité spécifique (invité, par exemple) aux demandes qui sont faites sans aucune information d'authentification associée. Ceci peut être utilisé pour permettre à des servlets non protégés d'appeler des méthodes ne nécessitant pas un rôle spécifique sur EJB. Un tel principal n'a aucun rôle associé et ne peut donc accéder qu'à des EJB non sécurisés ou à des méthodes EJB associées à la contrainte de permission non vérifiée.

- **unauthenticatedIdentity**: définit le nom de principal qui doit être assigné aux requêtes ne contenant pas d'informations d'authentification.

Rapporter un bogue

16.1.4. Module de connexion Ldap

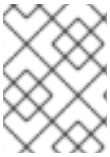
Le module de connexion de **`Ldap`** est une implémentation du **`LoginModule`** qui authentifie sur le serveur LDAP (Lightweight Directory Access Protocol). Utiliser le module de connexion **`Ldap`** si votre nom d'utilisateur et vos identifiants sont stockés dans une serveur LDAP accessible en utilisant un fournisseur LDAP JNDI (Java Naming and Directory Interface).

NOTE

Si vous souhaitez utiliser LDAP pour l'authentification SPNEGO ou éviter certaines phases s'authentification quand vous utilisez un serveur LDAP. considerez utiliser le module de connexion **`AdvancedLdap`** avec le module de connexion SPNEGO ou bien le login de connexion **`AdvancedLdap`** seul.

Nom distinctif (DN)

Avec le protocole LDAP (Lightweight Directory Access Protocol), le nom unique identifie de manière unique un objet dans un répertoire. Chaque nom distinctif doit avoir un nom et un emplacement uniques distincts de tout autre objet, ce qui est réalisé à l'aide d'un certain nombre de paires attribut / valeur (AVPs). Les AVP définissent les informations telles que des noms communs, unité d'organisation, entre autres. La combinaison de ces résultats de valeurs résultent en une chaîne unique, requise par le protocole LDAP.



NOTE

Ce module de connexion prend également en charge l'identité authentifiée et les piles de mots de passe.

Les informations de connectivité LDAP sont fournies sous forme d'options de configuration envoyées par l'intermédiaire de l'objet d'environnement utilisé pour créer le contexte initial JNDI. Les propriétés standard JNDI LDAP utilisées incluent :

java.naming.factory.initial

Nom de classe d'implémentation **`InitialContextFactory`**. A comme valeur par défaut celle de l'implémentaion du fournisseur Sun LDAP **`com.sun.jndi.ldap.LdapCtxFactory`**.

java.naming.provider.url

URL LDAP pour le serveur LDAP.

java.naming.security.authentication

Niveau de protocole de sécurité à utiliser. Les valeurs suivantes sont disponibles **none**, **`simple`**, et **`strong`**. Si la propriété n'est pas définie, le comportement sera dicté par le fournisseur de services.

java.naming.security.protocol

Le protocole de transport à utiliser pour un accès sécurisé. Définir cette option de configuration au type de fournisseur de service (comme par exemple, SSL). Si la propriété n'est pas définie, le comportement sera déterminé par le fournisseur de services.

java.naming.security.principal

Indique l'identité du principal permettant d'authentifier l'appelant vers le service. Il est construit à partir des autres propriétés décrites ci-dessous.

java.naming.security.credentials

Indique les identifiants du principal permettant d'authentifier l'appelant vers le service. Les identifiants peuvent prendre la forme d'un mot de passe haché, un mot de passe en texte clair, une clé ou un certificat. Si la propriété n'est pas définie, le comportement sera déterminé par le fournisseur de services.

Pour obtenir des informations sur les options de configuration du module de connexion ldap, voir Section A.1, « Modules d'authentification inclus ».

NOTE

Dans certains schémas de répertoire (par exemple, Microsoft Active Directory), les attributs de role de l'objet utilisateur sont stockés en tant que DN aux objets de rôle au lieu des noms simples. Pour les implémentations qui utilisent ce type de schéma, roleAttributeIsDN doit avoir la valeur **`true`**

L'authentification utilisateur est effectuée en se connectant au serveur LDAP, sur la base des options de configuration du module de connexion. La connexion au serveur LDAP se fait en créant un **`InitialLdapContext`** avec un environnement composé de propriétés JNDI LDAP décrites précédemment dans cette section.

Le Context.SECURITY_PRINCIPAL est défini au nom distinctif de l'utilisateur qui a été obtenu par le gestionnaire de rappel en combinaison au préfixe principalDNPrefix et au suffixe principalDNSuffix, et la propriété Context.SECURITY_CREDENTIALS est définie au mot de passe String respectif.

Une fois que l'authentification a réussi (quand l'instance **`InitialLdapContext`** est créée), les rôles de l'utilisateur sont vérifiés par le biais d'une recherche sur l'emplacement de **`rolesCtxDN`** avec des attributs de recherche définis aux valeurs roleAttributeName et uidAttributeName. Les noms de rôle sont obtenus par invocation de la méthode **`toString`** sur les attributs de rôle sur le groupe de résultat de la recherche.

Exemple 16.3. Domaine de sécurité du module de connexion LDAP

Cet exemple de CLI montre comment utiliser les paramètres dans une configuration d'authentification de domaine de sécurité.

```
/subsystem=security/security-domain=testLDAP:add(cache-type=default)
/subsystem=security/security-domain=testLDAP/authentication=classic:add
/subsystem=security/security-
domain=testLDAP/authentication=classic/login-module=Ldap:add( \
  code=Ldap, \
  flag=required, \
  module-options=[ \
    ("java.naming.factory.initial"=>"com.sun.jndi.ldap.LdapCtxFactory"),
\
    ("java.naming.provider.url"=>"ldap://ldaphost.jboss.org:1389/"), \
    ("java.naming.security.authentication"=>"simple"), \
    ("principalDNPrefix"=>"uid="), \
    ("principalDNSuffix"=>",ou=People,dc=jboss,dc=org"), \
    ("rolesCtxDN"=>"ou=Roles,dc=jboss,dc=org"), \
    ("uidAttributeID"=>"member"), \
    ("matchOnUserDN"=>true), \
    ("roleAttributeID"=>"cn"), \
    ("roleAttributeIsDN"=>false) \
  ])
```

Les options ***java.naming.factory.initial***, ***java.naming.factory.url*** et ***java.naming.security*** dans la configuration du domaine de sécurité testLDAP indique les conditions suivantes :

- L'implémentation du fournisseur JNDI LDAP sera utilisée
- Le serveur LDAP se situe sur l'hôte **ldaphost.jboss.org** sur le port 1389
- La méthode simple d'authentification LDAP sera utilisée pour se connecter au serveur LDAP.

Le module de connexion tente de se connecter au serveur LDAP à l'aide d'un nom distinct (DN) qui représente l'utilisateur qu'il tente d'authentifier. Ce DN est construit à partir de l'ancien principalDNPrefix, du nom d'utilisateur et du principalDNSuffix comme décrit ci-dessus. Dans Exemple 16.4, « Exemple de fichier LDIF », le nom d'utilisateur **jsmith** mappe au **uid=jsmith,ou=People,dc=jboss,dc=org**.

NOTE

L'exemple suppose que le serveur LDAP authentifie les utilisateurs à l'aide de l'attribut **userPassword** de l'entrée d'utilisateur (**theduke** dans cet exemple). La plupart des serveurs LDAP fonctionnent de cette manière, mais si votre serveur LDAP gère l'authentification différemment, vous devez vous assurer que LDAP soit configuré selon vos besoins d'environnement de production.

Une fois l'authentification réussie, les rôles sur lequel l'autorisation reposera sont récupérés en effectuant une recherche de sous-arborescence de l'entrée **rolesCtxDN** dont le uidAttributeID correspond à l'utilisateur. Si matchOnUserDN a la valeur true, la recherche s'appuiera sur le nom distinctif complet de l'utilisateur. Sinon, la recherche s'appuiera sur le nom d'utilisateur saisi. Dans cet exemple, la recherche est sous **ou=Roles,dc=jboss,dc=org** pour toutes les entrées qui ont un attribut de **membre** égal à **uid=jsmith,ou=People,dc=jboss,dc=org**. La recherche permettait de situer **cn=JBossAdmin** sous la rubrique rôles.

La recherche indique l'attribut spécifié dans l'option roleAttributeID. Dans cet exemple, l'attribut est **cn**. La valeur renvoyée est **JBossAdmin**, donc l'utilisateur **jsmith** reçoit le rôle **JBossAdmin**.

Un serveur LDAP local fournit souvent des services d'identité et d'authentification, mais il est incapable d'utiliser les services d'autorisation. C'est parce que les rôles d'application ne se mappent pas toujours bien sur les groupes LDAP, et les administrateurs LDAP sont souvent réticents à permettre à des données spécifiques à l'application externe d'aller dans les serveurs LDAP centralisés. Le module d'authentification LDAP est souvent associé à un autre module de connexion, tel que le module de connexion de base de données, qui peut fournir des rôles plus adaptés à l'application en cours de développement.

Fichier LDAP LDIF (Data Interchange Format) représentant la structure de répertoire sur laquelle les données opèrent, voir Exemple 16.4, « Exemple de fichier LDIF ».

LDAP LDIF (Data Interchange Format)

Format en texte brut utilisé pour représenter le contenu du répertoire LDAP et mettre à jour les requêtes. Le contenu du répertoire est représenté comme un enregistrement pour chaque requête d'objet ou de mise à jour. Le contenu consiste à ajouter, modifier, supprimer et renommer des requêtes.

Exemple 16.4. Exemple de fichier LDIF

```
dn: dc=jboss,dc=org
objectclass: top
objectclass: dcObject
objectclass: organization
dc: jboss
o: JBoss

dn: ou=People,dc=jboss,dc=org
objectclass: top
objectclass: organizationalUnit
ou: People

dn: uid=jsmith,ou=People,dc=jboss,dc=org
objectclass: top
objectclass: uidObject
objectclass: person
uid: jsmith
cn: John
sn: Smith
userPassword: theduke

dn: ou=Roles,dc=jboss,dc=org
objectclass: top
objectclass: organizationalUnit
ou: Roles

dn: cn=JBossAdmin,ou=Roles,dc=jboss,dc=org
objectclass: top
objectclass: groupOfNames
cn: JBossAdmin
member: uid=jsmith,ou=People,dc=jboss,dc=org
description: the JBossAdmin group
```

Rapporter un bogue

16.1.5. Module de connexion LdapExtended

Nom distinctif (DN)

Avec le protocole LDAP (Lightweight Directory Access Protocol), le nom unique identifie de manière unique un objet dans un répertoire. Chaque nom distinctif doit avoir un nom et un emplacement uniques distincts de tout autre objet, ce qui est réalisé à l'aide d'un certain nombre de paires attribut / valeur (AVPs). Les AVP définissent les informations telles que des noms communs, unité d'organisation, entre autres. La combinaison de ces résultats de valeurs résultent en une chaîne unique, requise par le protocole LDAP.

Le LdapExtended (**`org.jboss.security.auth.spi.LdapExtLoginModule)`** recherche l'utilisateur à relier, ainsi que les rôles associés, en vue d'effectuer l'authentification. La requête récursive des rôles suit les DN pour naviguer à travers une structure de rôles hiérarchiques.

Les options de LoginModules incluent toutes les options prises en charge par les supports de fournisseur JNDI LDAP choisis. Exemples de noms de propriétés standards :

- Context.INITIAL_CONTEXT_FACTORY = "java.naming.factory.initial"
- Context.SECURITY_PROTOCOL = "java.naming.security.protocol"
- Context.PROVIDER_URL = "java.naming.provider.url"
- Context.SECURITY_AUTHENTICATION = "java.naming.security.authentication"
- Context.REFERRAL = "java.naming.referral"

La logique d'implémentation de modules de connexion suit les étapes suivantes :

1. La liaison initiale de serveur LDAP est authentfiée par l'intermédiaire des propriétés bindDN et bindCredential. Le bindDN est un utilisateur ayant les permissions de chercher les utilisateurs et les rôles à la fois dans les arborescences baseCtxDN et rolesCtxDN. Le DN d'utilisateur ou user DN pour s'authentifier est trouvé grâce au filtre spécifié dans la propriété baseFilter.

2. Le DN d'utilisateur userDN qui en résulte est authentifié par la liaison au serveur LDAP en utilisant le userDN comme environnement de InitialLdapContext. Context.SECURITY_PRINCIPAL. La propriété Context.SECURITY_CREDENTIALS est définie avec le mot de passe String obtenu par le gestionnaire de rappel.

3. Si cela réussit, les rôles d'utilisateur associés seront interrogés par l'intermédiaire des options rolesCtxDN, roleAttributeID, roleAttributeIsDN, roleNameAttributeID, et roleFilter.

Pour obtenir des informations sur les options de module de connexion LdapExtended, voir Section A.1, « Modules d'authentification inclus ».

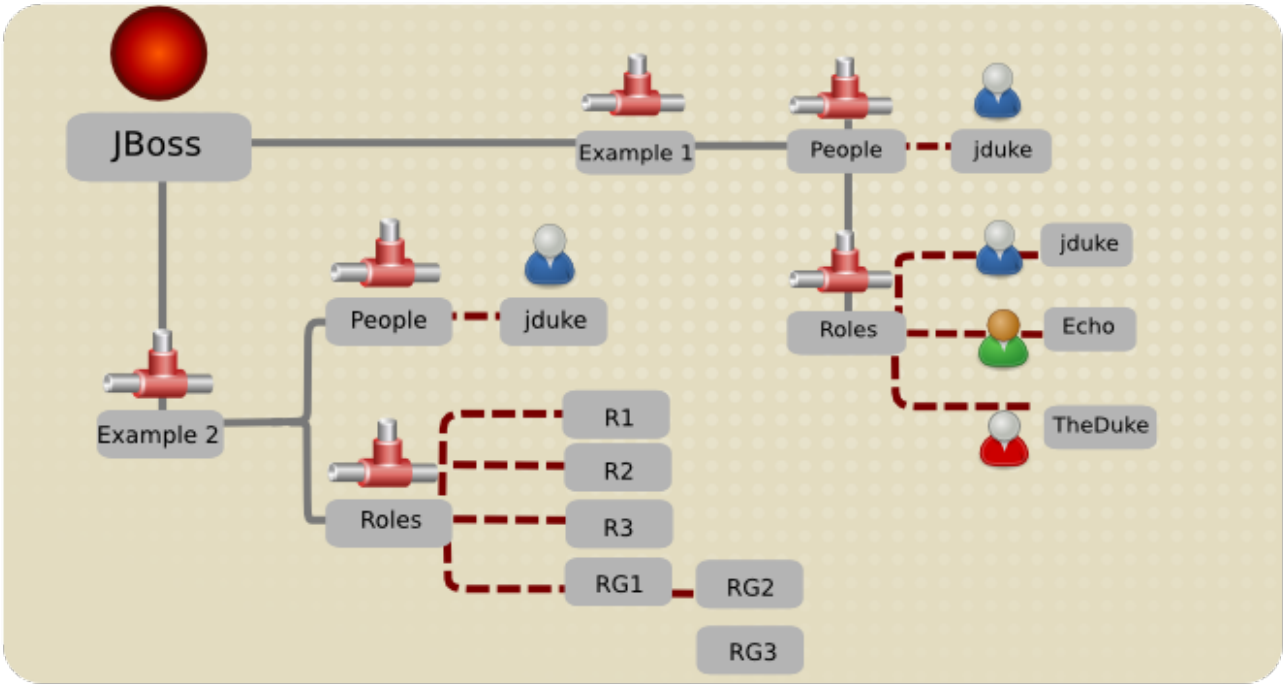


Figure 16.1. Exemple de structure LDAP

Exemple 16.5. Exemple 2 de configuration LDAP

```
version: 1
dn: o=example2,dc=jboss,dc=org
objectClass: top
objectClass: organization
o: example2

dn: ou=People,o=example2,dc=jboss,dc=org
objectClass: top
objectClass: organizationalUnit
ou: People

dn: uid=jduke,ou=People,o=example2,dc=jboss,dc=org
objectClass: top
objectClass: uidObject
objectClass: person
objectClass: inetOrgPerson
cn: Java Duke
employeeNumber: judke-123
sn: Duke
uid: jduke
userPassword:: dGhlZHVrZQ==

dn: uid=jduke2,ou=People,o=example2,dc=jboss,dc=org
objectClass: top
objectClass: uidObject
objectClass: person
objectClass: inetOrgPerson
cn: Java Duke2
employeeNumber: judke2-123
sn: Duke2
uid: jduke2
```

```
userPassword:: dGhlZHVrZTI=

dn: ou=Roles,o=example2,dc=jboss,dc=org
objectClass: top
objectClass: organizationalUnit
ou: Roles

dn: uid=jduke,ou=Roles,o=example2,dc=jboss,dc=org
objectClass: top
objectClass: groupUserEx
memberOf: cn=Echo,ou=Roles,o=example2,dc=jboss,dc=org
memberOf: cn=TheDuke,ou=Roles,o=example2,dc=jboss,dc=org
uid: jduke

dn: uid=jduke2,ou=Roles,o=example2,dc=jboss,dc=org
objectClass: top
objectClass: groupUserEx
memberOf: cn=Echo2,ou=Roles,o=example2,dc=jboss,dc=org
memberOf: cn=TheDuke2,ou=Roles,o=example2,dc=jboss,dc=org
uid: jduke2

dn: cn=Echo,ou=Roles,o=example2,dc=jboss,dc=org
objectClass: top
objectClass: groupOfNames
cn: Echo
description: the echo role
member: uid=jduke,ou=People,dc=jboss,dc=org

dn: cn=TheDuke,ou=Roles,o=example2,dc=jboss,dc=org
objectClass: groupOfNames
objectClass: top
cn: TheDuke
description: the duke role
member: uid=jduke,ou=People,o=example2,dc=jboss,dc=org

dn: cn=Echo2,ou=Roles,o=example2,dc=jboss,dc=org
objectClass: top
objectClass: groupOfNames
cn: Echo2
description: the Echo2 role
member: uid=jduke2,ou=People,dc=jboss,dc=org

dn: cn=TheDuke2,ou=Roles,o=example2,dc=jboss,dc=org
objectClass: groupOfNames
objectClass: top
cn: TheDuke2
description: the duke2 role
member: uid=jduke2,ou=People,o=example2,dc=jboss,dc=org

dn: cn=JBossAdmin,ou=Roles,o=example2,dc=jboss,dc=org
objectClass: top
objectClass: groupOfNames
cn: JBossAdmin
description: the JBossAdmin group
member: uid=jduke,ou=People,dc=jboss,dc=org
```

La configuration de module de cet exemple de structure LDAP est décrite dans la commande CLI suivante :

```
/subsystem=security/security-
domain=testLdapExample2/authentication=classic/login-
module=LdapExtended:add( \
  code=LdapExtended, \
  flag=required, \
  module-options=[ \
    ("java.naming.factory.initial"=>"com.sun.jndi.ldap.LdapCtxFactory"),
\
    ("java.naming.provider.url"=>"ldap://ldaphost.jboss.org"), \
    ("java.naming.security.authentication"=>"simple"), \
    ("bindDN"=>"cn=Root,dc=jboss,dc=org"), \
    ("bindCredential"=>"secret1"), \
    ("baseCtxDN"=>"ou=People,o=example2,dc=jboss,dc=org"), \
    ("baseFilter"=>"(uid={0})"), \
    ("rolesCtxDN"=>"ou=Roles,o=example2,dc=jboss,dc=org"), \
    ("roleFilter"=>"(uid={0})"), \
    ("roleAttributeIsDN"=>"true"), \
    ("roleAttributeID"=>"memberOf"), \
    ("roleNameAttributeID"=>"cn") \
  ])
```

Exemple 16.6. Exemple 3 de configuration LDAP

```
dn: o=example3,dc=jboss,dc=org
objectclass: top
objectclass: organization
o: example3

dn: ou=People,o=example3,dc=jboss,dc=org
objectclass: top
objectclass: organizationalUnit
ou: People

dn: uid=jduke,ou=People,o=example3,dc=jboss,dc=org
objectclass: top
objectclass: uidObject
objectclass: person
objectClass: inetOrgPerson
uid: jduke
employeeNumber: judke-123
cn: Java Duke
sn: Duke
userPassword: theduke

dn: ou=Roles,o=example3,dc=jboss,dc=org
objectClass: top
objectClass: organizationalUnit
ou: Roles
```

```
dn: uid=jduke,ou=Roles,o=example3,dc=jboss,dc=org
objectClass: top
objectClass: groupUserEx
memberOf: cn=Echo,ou=Roles,o=example3,dc=jboss,dc=org
memberOf: cn=TheDuke,ou=Roles,o=example3,dc=jboss,dc=org
uid: jduke

dn: cn=Echo,ou=Roles,o=example3,dc=jboss,dc=org
objectClass: top
objectClass: groupOfNames
cn: Echo
description: the JBossAdmin group
member: uid=jduke,ou=People,o=example3,dc=jboss,dc=org

dn: cn=TheDuke,ou=Roles,o=example3,dc=jboss,dc=org
objectClass: groupOfNames
objectClass: top
cn: TheDuke
member: uid=jduke,ou=People,o=example3,dc=jboss,dc=org
```

La configuration de module de cet exemple de structure LDAP est décrite dans la commande CLI suivante :

```
/subsystem=security/security-
domain=testLdapExample3/authentication=classic/login-
module=LdapExtended:add( \
  code=LdapExtended, \
  flag=required, \
  module-options=[ \
    ("java.naming.factory.initial"=>"com.sun.jndi.ldap.LdapCtxFactory"), \
    ("java.naming.provider.url"=>"ldap://ldaphost.jboss.org"), \
    ("java.naming.security.authentication"=>"simple"), \
    ("bindDN"=>"cn=Root,dc=jboss,dc=org"), \
    ("bindCredential"=>"secret1"), \
    ("baseCtxDN"=>"ou=People,o=example3,dc=jboss,dc=org"), \
    ("baseFilter"=>"(cn={0})"), \
    ("rolesCtxDN"=>"ou=Roles,o=example3,dc=jboss,dc=org"), \
    ("roleFilter"=>"(member={1})"), \
    ("roleAttributeID"=>"cn") \
  ])
```

Exemple 16.7. Exemple 4 de configuration LDAP

```
dn: o=example4,dc=jboss,dc=org
objectclass: top
objectclass: organization
o: example4

dn: ou=People,o=example4,dc=jboss,dc=org
objectclass: top
```

```
objectclass: organizationalUnit
ou: People

dn: uid=jduke,ou=People,o=example4,dc=jboss,dc=org
objectClass: top
objectClass: uidObject
objectClass: person
objectClass: inetOrgPerson
cn: Java Duke
employeeNumber: jduke-123
sn: Duke
uid: jduke
userPassword:: dGhlZHVrZQ==

dn: ou=Roles,o=example4,dc=jboss,dc=org
objectClass: top
objectClass: organizationalUnit
ou: Roles

dn: cn=RG1,ou=Roles,o=example4,dc=jboss,dc=org
objectClass: groupOfNames
objectClass: top
cn: RG1
member: cn=empty

dn: cn=RG2,cn=RG1,ou=Roles,o=example4,dc=jboss,dc=org
objectClass: groupOfNames
objectClass: top
cn: RG2
member: cn=RG1,ou=Roles,o=example4,dc=jboss,dc=org
member: uid=jduke,ou=People,o=example4,dc=jboss,dc=org

dn: cn=RG3,cn=RG1,ou=Roles,o=example4,dc=jboss,dc=org
objectClass: groupOfNames
objectClass: top
cn: RG3
member: cn=RG1,ou=Roles,o=example4,dc=jboss,dc=org

dn: cn=R1,ou=Roles,o=example4,dc=jboss,dc=org
objectClass: groupOfNames
objectClass: top
cn: R1
member: cn=RG2,cn=RG1,ou=Roles,o=example4,dc=jboss,dc=org

dn: cn=R2,ou=Roles,o=example4,dc=jboss,dc=org
objectClass: groupOfNames
objectClass: top
cn: R2
member: cn=RG2,cn=RG1,ou=Roles,o=example4,dc=jboss,dc=org

dn: cn=R3,ou=Roles,o=example4,dc=jboss,dc=org
objectClass: groupOfNames
objectClass: top
cn: R3
member: cn=RG2,cn=RG1,ou=Roles,o=example4,dc=jboss,dc=org
member: cn=RG3,cn=RG1,ou=Roles,o=example4,dc=jboss,dc=org
```

```
dn: cn=R4,ou=Roles,o=example4,dc=jboss,dc=org
objectClass: groupOfNames
objectClass: top
cn: R4
member: cn=RG3,cn=RG1,ou=Roles,o=example4,dc=jboss,dc=org

dn: cn=R5,ou=Roles,o=example4,dc=jboss,dc=org
objectClass: groupOfNames
objectClass: top
cn: R5
member: cn=RG3,cn=RG1,ou=Roles,o=example4,dc=jboss,dc=org
member: uid=jduke,ou=People,o=example4,dc=jboss,dc=org
```

La configuration de module de cet exemple de structure LDAP est décrite dans l'extrait de code en exemple.

```
/subsystem=security/security-
domain=testLdapExample4/authentication=classic/login-
module=LdapExtended:add( \
  code=LdapExtended, \
  flag=required, \
  module-options=[ \
    ("java.naming.factory.initial"=>"com.sun.jndi.ldap.LdapCtxFactory"),
\
    ("java.naming.provider.url"=>"ldap://ldaphost.jboss.org"), \
    ("java.naming.security.authentication"=>"simple"), \
    ("bindDN"=>"cn=Root,dc=jboss,dc=org"), \
    ("bindCredential"=>"secret1"), \
    ("baseCtxDN"=>"ou=People,o=example4,dc=jboss,dc=org"), \
    ("baseFilter"=>"(cn={0})"), \
    ("rolesCtxDN"=>"ou=Roles,o=example4,dc=jboss,dc=org"), \
    ("roleFilter"=>"(member={1})"), \
    ("roleRecursion"=>"1"), \
    ("roleAttributeID"=>"memberOf") \
  ])
```

Exemple 16.8. Configuration Active Directory par défaut

L'exemple ci-dessous montre la configuration d'Active Directory utilisée par défaut.

Certaines configurations d'Active Directory nécessitent sans doute une recherche Port 3268 au lieu d'une recherche Port 389 dans le Global Catalog. C'est d'autant plus probable quand l'Active Directory comprend plusieurs domaines.

```
/subsystem=security/security-
domain=AD_Default/authentication=classic/login-module=LdapExtended:add(
\
  code=LdapExtended, \
  flag=required, \
  module-options=[ \
```

```
    ("java.naming.provider.url"=>"ldap://ldaphost.jboss.org"), \
    ("bindDN"=>"JBOSS\searchuser"), \
    ("bindCredential"=>"password"), \
    ("baseCtxDN"=>"CN=Users,DC=jboss,DC=org"), \
    ("baseFilter"=>"(sAMAccountName={0})"), \
    ("rolesCtxDN"=>"CN=Users,DC=jboss,DC=org"), \
    ("roleFilter"=>"(sAMAccountName={0})"), \
    ("roleAttributeID"=>"memberOf"), \
    ("roleAttributeIsDN"=>"true"), \
    ("roleNameAttributeID"=>"cn"), \
    ("searchScope"=>"ONELEVEL_SCOPE"), \
    ("allowEmptyPasswords"=>"false") \
  ])
```

Exemple 16.9. Configuration Active Directory de rôles récursifs

L'exemple ci-dessous effectue une recherche résursive de rôles dans l'Active Directory. La différence clé entre cet exemple et l'exemple d'Active Directory par défaut est que la recherche de rôle a été rempléacée par une recherche d'attribut de membre par le DN utilisateur. Le module de connexion utilise alors le DN du rôle pour trouver les groupes auquels le groupe appartient.

```
/subsystem=security/security-
domain=AD_Recursive/authentication=classic/login-
module=LdapExtended:add( \
  code=LdapExtended, \
  flag=required, \
  module-options=[ \
    ("java.naming.provider.url"=>"ldap://ldaphost.jboss.org"), \
    ("java.naming.referral"=>"follow"), \
    ("bindDN"=>"JBOSS\searchuser"), \
    ("bindCredential"=>"password"), \
    ("baseCtxDN"=>"CN=Users,DC=jboss,DC=org"), \
    ("baseFilter"=>"(sAMAccountName={0})"), \
    ("rolesCtxDN"=>"CN=Users,DC=jboss,DC=org"), \
    ("roleFilter"=>"(member={1})"), \
    ("roleAttributeID"=>"cn"), \
    ("roleAttributeIsDN"=>"false"), \
    ("roleRecursion"=>"2"), \
    ("searchScope"=>"ONELEVEL_SCOPE"), \
    ("allowEmptyPasswords"=>"false") \
  ])
```

Rapporter un bogue

16.1.6. Module de connexion UserRoles

Le module de connexion **`UsersRoles`** est un module de connexion simple qui prend en charge plusieurs utilisateurs et des rôles d'utilisateur chargés à partir de fichiers de propriétés Java. Le nom de fichier de mappage de nom-d'utilisateur-à-mot-de-passe par défaut est **`users.properties`** et le nom

du fichier de mappage de nom-d'utilisateur-à-rôles par défaut est **roles.properties**.

Pour obtenir des informations sur les options de modules de connexion UsersRoles, voir Section A.1, « Modules d'authentification inclus » .

Ce module de connexion prend en charge l'empilage des mots de passe, le hachage des mots de passe et l'identité authentifiée.

Les fichiers de propriétés sont chargés lors de l'initialisation en utilisant le chargeur de classes de contexte de thread de méthode initialize. Cela signifie que ces fichiers peuvent être placés sur le chemin de classe du déploiement Java EE (par exemple, dans le dossier **WEB-INF/classes** dans l'archive **WAR**), ou dans n'importe quel répertoire sur le chemin de classse du serveur. L'objectif principal de ce module de connexion est de tester facilement les paramètres de sécurité d'utilisateurs et de rôles multiples à l'aide de fichiers déployés avec l'application.

Exemple 16.10. Module de connexion UserRoles

```
/subsystem=security/security-domain=ejb3-
sampleapp/authentication=classic/login-module=UsersRoles:add( \
  code=UsersRoles, \
  flag=required, \
  module-options=[ \
    ("usersProperties"=>"ejb3-sampleapp-users.properties"), \
    ("rolesProperties"=>"ejb3-sampleapp-roles.properties") \
  ])
```

Dans Exemple 16.10, « Module de connexion UserRoles », le fichier **ejb3-sampleapp-users.properties** utilise un format **username=password** pour chaque entrée d'utilisateur sur ligne séparée :

```
username1=password1
username2=password2
...
```

Le fichier **ejb3-sampleapp-roles.properties** référencé dans Exemple 16.10, « Module de connexion UserRoles » utilise le modèle **username=role1,role2,** avec une valeur de nom de groupe en option. Par exemple :

```
username1=role1,role2,...
username1.RoleGroup1=role3,role4,...
username2=role1,role3,...
```

Le modèle de nom de propriété user name.XXX qui se trouve dans **ejb3-sampleapp-roles.properties** est utilisé pour assigner les rôles de nom d'utilisateur à un groupe de rôles de groupe nommé particulier où la portion **XXX** du nom de propriété est le nom de groupe. La forme user name=... est une abbréviation de user name.Roles=..., où le nom de groupe **Roles** correspond au nom standard auquel **JBossAuthorizationManager** s'attend pour contenir les rôles qui définissent les permissions utilisateur.

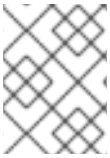
Ce qui suit corrrespond à une définition équivalente pour le nom d'utilisateur **jduke** :

```
jduke=TheDuke,AnimatedCharacter
jduke.Roles=TheDuke,AnimatedCharacter
```

Rapporter un bogue

16.1.7. Module de connexion Database

Le module de connexion **Database** est un JDBC (Java Database Connectivity-based) qui supporte l'authentification et le mappage des rôles. Utiliser ce module de connexion si vous avez vos nom d'utilisateur, mot de passe et information de rôle stockés dans une case de données relationnelle.



NOTE

Ce module prend en charge l'empilage de mots de passe, le hachage des mots de passe et l'identité non authentifiée.

Le module de connexion **Database** est basé sur deux tableaux logiques :

```
Table Principals(PrincipalID text, Password text)
Table Roles(PrincipalID text, Role text, RoleGroup text)
```

Le tableau **Principals** fait correspondre l'utilisateur **PrincipalID** au nom de passe valide et le tableau **Roles** fait correspondre l'utilisateur **PrincipalID** à ses groupes de rôles. Les rôles utilisés pour les permissions utilisateur doivent se trouver dans des lignes ayant pour valeur de colonne **RoleGroup** correspondant à **Roles**.

Les tableaux sont logiques, car vous pouvez spécifier la requête SQL qui utilise le module de connexion. La seule exigence est que **java.sql.ResultSet** ait la même structure logique que les tableaux **Principals** et **Roles** décrits précédemment. Les noms des tables et des colonnes ne sont pas pertinents car les résultats sont accessibles selon l'index de colonne.

Pour éclaircir cette notion, imaginez une base de données composée de deux tableaux **Principals** et **Roles**, déjà déclarés. Les énoncés suivants remplissent les tableaux avec les données suivantes :

- **PrincipalID java** avec un mot de passe (**Password**) correspondant à **echoman** dans le tableau **Principals**
- **PrincipalID java** ayant un rôle nommé **Echo** dans **RolesRoleGroup** dans le tableau **Roles**
- **PrincipalID java** ayant un rôle nommé **caller_java** dans **CallerPrincipalRoleGroup** dans le tableau **Roles**

```
INSERT INTO Principals VALUES('java', 'echoman')
INSERT INTO Roles VALUES('java', 'Echo', 'Roles')
INSERT INTO Roles VALUES('java', 'caller_java', 'CallerPrincipal')
```

Pour obtenir des informations sur les options de module de connexion de la base de données, voir Section A.1, « Modules d'authentification inclus ».

Voici un exemple de configuration de **Database login module** :

```
CREATE TABLE Users(username VARCHAR(64) PRIMARY KEY, passwd VARCHAR(64))
CREATE TABLE UserRoles(username VARCHAR(64), role VARCHAR(32))
```

Configuration de module de connexion correspondant dans un domaine de sécurité :

```
/subsystem=security/security-domain=testDB/authentication=classic/login-
module=Database:add( \
  code=Database, \
  flag=required, \
  module-options=[ \
    ("dsJndiName"=>"java:/MyDatabaseDS"), \
    ("principalsQuery"=>"select passwd from Users where username=?"), \
    ("rolesQuery"=>"select role, 'Roles' from UserRoles where username=?")
\
  ])
```

Rapporter un bogue

16.1.8. Module de connexion Certificate

Le module de connexion **`Certificate`** authentifie les utilisateurs sur la base de certificats X509. L'authentification en web tier **`CLIENT-CERT`** est un cas d'utilisation typique de ce module de connexion.

Ce module de connexion ne s'occupe que de l'authentification : vous devez le combiner avec un autre module capable d'obtenir des rôles d'authorisation pour pouvoir définir l'accès à un site web sécurisé ou à un composant EJB. Les deux sous-classes de ce module de connexion **`CertRolesLoginModule`** et **`DatabaseCertLoginModule`** étendent le comportement pour pouvoir obtenir les rôles d'autorisation soit du fichier de propriétés, soit de la base de données.

Pour obtenir des détails sur les options de module de connexion **`Certificate`**, voir Section A.1, « Modules d'authentification inclus ».

Le module de connexion **`Certificate`** requiert un **`KeyStore`** pour effectuer la validation utilisateur. Elle s'obtient à partir de la configuration JSSE d'un domaine de sécurité lié, comme le montre le fragment de configuration ci-dessous :

```
/subsystem=security/security-domain=trust-domain:add
/subsystem=security/security-domain=trust-domain/jsse=classic:add( \
  truststore={ \
    password=>pass1234, \
    url=>/home/jbosseap/trusted-clients.jks \
  })

/subsystem=security/security-domain=testCert:add
/subsystem=security/security-domain=testCert/authentication=classic:add
/subsystem=security/security-domain=testCert/authentication=classic/login-
module=Certificate:add( \
  code=Certificate, \
  flag=required, \
  module-options=[ \
    ("securityDomain"=>"trust-domain"), \
  ])
```

Procédure 16.1. Sécuriser les applications web par des certificats et des autorisations basées rôle

Cette procédure décrit comment sécuriser une application web, comme **`user-app.war`**, en utilisant les

certificats client et les autorisations basées rôle. Dans cet exemple, le module de connexion **`CertificateRoles`** est utilisé pour l'authentification et l'autorisation. Les **`trusted-clients.keystore`** et **`app-roles.properties`** requiérent une entrée qui mappe le principal associé au certificat du client.

Par défaut, le principal est créé en utilisant le nom distinct du certificat du client, comme le DN spécifié dans Exemple 16.11, « Exemple de certificat ».

1. **Déclarer les ressources et les rôles**
   Modifier **`web.xml`** pour déclarer les ressources à sécuriser avec les rôles autorisés et le domaine de sécurité à utiliser pour l'authentification et l'autorisation.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
        version="3.0">

        <security-constraint>
                <web-resource-collection>
                        <web-resource-name>Protect App</web-
resource-name>
                        <url-pattern>/*</url-pattern>
                </web-resource-collection>
                <auth-constraint>
                        <role-name>Admin</role-name>
                </auth-constraint>
        </security-constraint>

        <login-config>
                <auth-method>CLIENT-CERT</auth-method>
                <realm-name>Secured area</realm-name>
        </login-config>

        <security-role>
                <role-name>Admin</role-name>
        </security-role>
</web-app>
```

2. **Spécifier un domaine de sécurité**
   Dans le fichier **`jboss-web.xml`**, indiquer le domaine de sécurité requis.

```
<jboss-web>
  <security-domain>app-sec-domain</security-domain>
</jboss-web>
```

3. **Configurer le module de sécurité**
   Définir la configuration du module de connexion pour le domaine **`app-sec-domain`** que vous venez d'indiquer par le biais du CLI.

```
[
/subsystem=security/security-domain=trust-domain:add
/subsystem=security/security-domain=trust-domain/jsse=classic:add( \
  truststore={ \
    password=>pass1234, \
    url=>/home/jbosseap/trusted-clients.jks \
  })

/subsystem=security/security-domain=app-sec-domain:add
/subsystem=security/security-domain=app-sec-
domain/authentication=classic:add
/subsystem=security/security-domain=app-sec-
domain/authentication=classic/login-module=CertificateRoles:add( \
  code=CertificateRoles, \
  flag=required, \
  module-options=[ \
    ("securityDomain"=>"trust-domain"), \
    ("rolesProperties"=>"app-roles.properties") \
  ])
```

Exemple 16.11. Exemple de certificat

```
[conf]$ keytool -printcert -file valid-client-cert.crt
Owner: CN=valid-client, OU=Security QE, OU=JBoss, O=Red Hat, C=CZ
Issuer: CN=EAP Certification Authority, OU=Security QE, OU=JBoss, O=Red
Hat, C=CZ
Serial number: 2
Valid from: Mon Mar 24 18:21:55 CET 2014 until: Tue Mar 24 18:21:55 CET
2015
Certificate fingerprints:
         MD5:  0C:54:AE:6E:29:ED:E4:EF:46:B5:14:30:F2:E0:2A:CB
         SHA1:
D6:FB:19:E7:11:28:6C:DE:01:F2:92:2F:22:EF:BB:5D:BF:73:25:3D
         SHA256:
CD:B7:B1:72:A3:02:42:55:A3:1C:30:E1:A6:F0:20:B0:2C:0F:23:4F:7A:8E:2F:2D:
FA:AF:55:3E:A7:9B:2B:F4
         Signature algorithm name: SHA1withRSA
         Version: 3
```

Le **`trusted-clients.keystore`** exige que le certificat Exemple 16.11, « Exemple de certificat » soit stocké avec un alias **`CN=valid-client, OU=Security QE, OU=JBoss, O=Red Hat, C=CZ`**. Les **`app-roles.properties`** doivent avoir les mêmes saisies. Comme le DN contient des caractères qui sont normalement considérés comme des délimiteurs, vous devrez utiliser ('\') pour les carctères problématiques comme illustré ci-dessous.

```
# A sample app-roles.properties file
CN\=valid-client,\ OU\=Security\ QE,\ OU\=JBoss,\ O\=Red\ Hat,\ C\=CZ
```

Rapporter un bogue

16.1.9. Module de connexion d'identité

Le module de connexion **`Identity`** est un simple module de connexion qui associe un nom d'utilisateur codé en dur à tout subject authentifié dans le module. Il crée une instance **`SimplePrincipal`** qui utilise le nom spécifié par l'option **`principal`**.

NOTE

Ce module prend également en charge l'empilage des mots de passe.

Ce module de connexion est utile pour procurer une identité fixe à un service, et dans les environnements de dévelopement, quand vous souhaiez tester la sécurité associée à un principal donné et à ses rôles associés.

Pour obtenir des informations sur les options de module de connexion d'identité, voir Section A.1, « Modules d'authentification inclus ».

Vous trouverez ci-dessous un exemple de configuration de domaine de sécurité. Il authentifie tous les utilisateurs comme principal nommé **`jduke`** et assigne les noms de rôle **`TheDuke`**, et **`AnimatedCharacter`**:.

```
/subsystem=security/security-domain=testIdentity:add
/subsystem=security/security-
domain=testIdentity/authentication=classic:add
/subsystem=security/security-
domain=testIdentity/authentication=classic/login-module=Identity:add( \
  code=Identity, \
  flag=required, \
  module-options=[ \
    ("principal"=>"jduke"), \
    ("roles"=>"TheDuke,AnimatedCharacter") \
  ])
```

Rapporter un bogue

16.1.10. Module de connexion RunAs

Le module de connexion **`RunAs`** est un module d'assitance qui pousse le rôle **`run as`** dans la pile pour le durée de la phase de connexion de l'authentification, puis qui produit le rôle **`run as`** de la pile lors de la phase de validation ou de suppression.

Le but de ce module de connexion est de fournir un rôle pour les autres modules de connexion qui doivent accéder aux ressources sécurisées afin d'effectuer leur authentification, par exemple un module de connexion qui accède à un EJB sécurisé. Le module de connexion **`RunAs`** doit être configuré avant que les modules de connexion qui ont besoin d'une rôle **`run as`** soient mis en place.

Pour obtenir des informations sur les options de module de connexion RunAs voir Section A.1, « Modules d'authentification inclus ».

Rapporter un bogue

16.1.10.1. Création de RunAsIdentity

Pour que JBoss EAP 6 puisse sécuriser un accès aux méthodes EJB, l'identité de l'utilisateur ne doit pas être connue au moment où l"appel de méthode est effectué.

Une identité d'utilisateur sur le serveur est représentée soit par une instance **`javax.security.auth.Subject`** ou par une instance **`org.jboss.security.RunAsIdentity`**. Ces deux classes stockent un ou deux principaux qui représentent l'identité et la liste des rôles que possède l'identité. Dans le cas de la classe **`javax.security.auth.Subject`**, une liste d'informations d'identification est également stockée.

Dans la section <assembly-descriptor> du descripteur de déploiement **`ejb-jar.xml`**, vous spécifiez un ou plusieurs rôles qu'un utilisateur doit avoir pour accéder aux différentes méthodes EJB. Une comparaison de ces listes révèle si l'utilisateur possède un des rôles nécessaires pour accéder à la méthode EJB.

Exemple 16.12. Création de org.jboss.security.RunAsIdentity

Dans le fichier **`ejb-jar.xml`**, vous spécifiez un élément <security-identity> avec un rôle <run-as> défini comme un enfant de l'élément <session>.

```
<session>
   ...
   <security-identity>
      <run-as>
         <role-name>Admin</role-name>
      </run-as>
   </security-identity>
   ...
</session>
```

Cette déclaration signifie que le rôle RunAsIdentity **`Admin`** doit être créé.

Pour nommer un principal pour le rôle **`Admin`**, vous devez définir un élément **`<run-as-principal>`** dans le fichier **`jboss-ejb3.xml`**.

```
<jboss:ejb-jar
        xmlns="http://java.sun.com/xml/ns/javaee"
        xmlns:jboss="http://www.jboss.com/xml/ns/javaee"
        xmlns:s="urn:security:1.1"
        version="3.1" impl-version="2.0">
    <assembly-descriptor>
        <s:security>
            <ejb-name>WhoAmIBean</ejb-name>
            <s:run-as-principal>John</s:run-as-principal>
        </s:security>
    </assembly-descriptor>
</jboss:ejb-jar>
```

L'élément **`<security-identity>`** qui se trouve à la fois dans **`ejb-jar.xml`** et dans l'élément **`<security>`** des fichiers **`jboss-ejb3.xml`** sont traités en cours de déploiement. Le nom de rôle **`<run-as>`** et le nom **`<run-as-principal>`** sont alors stockés dans la classe **`org.jboss.metadata.ejb.spec.SecurityIdentityMetaData`**.

Exemple 16.13. Assigner plusieurs rôles à une RunAsIdentity

Vous pouvez assigner plusieurs rôles à RunAsIdentity en mappant les rôles aux pricipaux dans le groupe d'éléments **<assembly-descriptor>** du descripteur de déploiement **jboss-ejb3.xml**.

```
<jboss:ejb-jar xmlns:sr="urn:security-role"
        ...>
        <assembly-descriptor>
            ...
                <sr:security-role>
                        <sr:role-name>Support</sr:role-name>
                        <sr:principal-name>John</sr:principal-name>
                        <sr:principal-name>Jill</sr:principal-name>
                        <sr:principal-name>Tony</sr:principal-name>
                </sr:security-role>
        </assembly-descriptor>
</jboss:ejb-jar>
```

Dans Exemple 16.12, « Création de org.jboss.security.RunAsIdentity », le **<run-as-principal>** de **John** a été créé. La configuration dans cet exemple étend le rôle **Admin** en ajoutant le rôle **Support**. Le nouveau rôle contient des principaux supplémentaires, y compris le principal **John** de départ.

L'élément **<security-role>** qui se trouve à la fois dans les fichiers **ejb-jar.xml** et **jboss-ejb3.xml** sont traités en cours de déploiement. Les données de **<role-name>** et de **<principal-name>** sont alors stockées dans la classe **org.jboss.metadata.ejb.spec.SecurityIdentityMetaData**.

Rapporter un bogue

16.1.11. Module de connexion Client

Le module de connexion de **Client** (**org.jboss.security.ClientLoginModule**) est une implémentation du **LoginModule** utilisable par les clients JBoss quand ils souhaitenet établir l'identité et les identifiants d'un appelant. Cela crée un nouveau **SecurityContext**, lui assigne un principal et des informations d'identification, et assigne le **SecurityContext** au **ThreadLocal**.

Le module de connexion **client** est le seul mécanisme pris en charge pour qu'un client puisse établir qui est l'appelant du thread en cours. Les applications clientes autonomes et les environnements de serveurs (agissant en tant que clients JBoss EJB où l'environnement de sécurité n'a pas été configuré pour utiliser le sous-système de sécurité EAP de façon transparente) doivent utiliser le module de connexion **Client**.

Notez que ce module de connexion n'effectue aucune authentification. Il copie uniquement les informations de connexion fournies dans la couche de serveur d'invocation EJB pour une authentification ultérieure au serveur. Si vous avez besoin exécuter l'authentification côté client des utilisateurs, vous devez configurer un autre module de connexion en plus du module de connexion **Client**.

Pour obtenir des informations sur les options de module de connexion Client voir Section A.1, « Modules d'authentification inclus ».

Rapporter un bogue

16.1.12. Module de connexion SPNEGO

Le module de connexion **SPNEGO** (**org.jboss.security.negotiation.spnego.SPNEGOLoginModule**) est une implémentation de **LoginModule** qui établit l'identité de l'appelant et les informations d'identification avec un KDC. Le module implémente SPNEGO (mécanisme de négociation GSSAPI simple et protégé) et fait partie du projet JBoss Negotiation. Cette authentification peut être utilisée dans la configuration chaînée avec le module de connexion **AdvancedLdap** pour permettre la coopération avec un serveur LDAP.

Pour obtenir des informations sur les options de module de connexion SPNEGO, voir Section A.1, « Modules d'authentification inclus ».

Le module JBoss Negociation n'est pas inclus comme une dépendance standard pour les applications déployées. Pour utiliser les modules de connexion **SPNEGO** ou **AdvancedLdap** dans votre projet, vous devez ajouter manuellement la dépendance en éditant le fichier de descripteur de déploiement **META-INF/jboss-déploiement-structure.xml**.

Exemple 16.14. Ajouter le module JBoss Negociation en tant que dépendance.

```
<jboss-deployment-structure>
  <deployment>
    <dependencies>
      <module name="org.jboss.security.negotiation" />
    </dependencies>
  </deployment>
</jboss-deployment-structure>
```

Rapporter un bogue

16.1.13. Module de connexion RoleMapping

Le module de connexion **RoleMapping** supporte les rôles de mappage qui résultent en fin de processus d'authentification, à un ou plusiurs rôles déclaratifs. Ainsi, si le processus d'authentification a déterminé que l'utilisateur 'A' a les rôles "ldapAdmin" et "testAdmin"et que le rôle déclaratif défini dans le fichier **web.xml** ou le fichier **ejb-jar.xml** est **admin**, alors ce module de connexion mappera les rôles **admin** à l'utilisateur **A**.

Pour obtenir des informations sur les options de module de connexion **RoleMapping**, voir Section A.1, « Modules d'authentification inclus ».

Le module de connexion **RoleMapping** doit être défini comme un module optionnel de configuration de module de connexion car il modifie le mappage des rôles déjà mappés.

Exemple 16.15. Définir les rôles mappés

```
/subsystem=security/security-domain=test-domain-2/:add
/subsystem=security/security-domain=test-domain-
2/authentication=classic:add
/subsystem=security/security-domain=test-domain-
2/authentication=classic/login-module=test-2-lm/:add(\
flag=required,\
```

```
code=UsersRoles,\
module-options=[("usersProperties"=>"users.properties"),
("rolesProperties"=>"roles.properties")]\
)
/subsystem=security/security-domain=test-domain-
2/authentication=classic/login-module=test2-map/:add(\
flag=optional,\
code=RoleMapping,\
module-options=[("rolesProperties"=>"rolesMapping-roles.properties")]\
)
```

Autre exemple aboutissant au même résultat, mais utilisant le module de mappage. Il s'agit de la méthode de mappage de rôles recommandée :

Exemple 16.16. Méthode recommandée pour définir les rôles mappés

```
/subsystem=security/security-domain=test-domain-2/:add
/subsystem=security/security-domain=test-domain-
2/authentication=classic:add
/subsystem=security/security-domain=test-domain-
2/authentication=classic/login-module=test-2-lm/:add(\
flag=required,\
code=UsersRoles,\
module-options=[("usersProperties"=>"users.properties"),
("rolesProperties"=>"roles.properties")]\
)
/subsystem=security/security-domain=test-domain-
2/mapping=classic/mapping-module=test2-map/:add(\
code=PropertiesRoles,type=role,\
module-options=[("rolesProperties"=>"rolesMapping-roles.properties")]\
)
```

Exemple 16.17. Fichier de propriétés utilisé par un RoleMappingLoginModule

```
ldapAdmin=admin, testAdmin
```

Si le sujet authentifié contient le rôle **`ldapAdmin`**, alors les rôles **`admin`** et **`testAdmin`** seront ajoutés ou se substitueront au sujet authentifié en fonction de la valeur de propriété replaceRole.

Rapporter un bogue

16.1.14. Option du module bindCredential

L'option de module **`bindCredential`** est utilisée pour stocker les identifiants pour le DN et peut être utilisée par plusieurs modules de mappage et de connexion. Il existe plusieurs méthodes d'obtention du mot de passse.

Texte brut dans une commande CLI de gestion.

The password for the **bindCredential** module may be provided in plaintext, in a management CLI command. For example: **("bindCredential"=>"secret1")**. For security reasons, the password should be encrypted using the JBoss EAP vault mechanism.

Utiliser une commande externe.

Pour obtenir le mot de passe à partir de la sortie d'une commande externe, utilisez le format **{EXT}...** où **...** est la commande externe. La première ligne de la sortie de commande est utilisée comme mot de passe.

Pour améliorer la performance, la variante **{EXTC[:expiration_in_millis]}** met en cache le mot de passe pour un nombre de millisecondes spécifié. Par défaut, le mot de passe mis en cache n'expire pas. Si la valeur **0** (zero) est spécifiée, les indentifiants mis en cache n'expirent pas.

Ka variante **EXTC** n'est prise en charge que par le module de connexion **LdapExtended**.

Exemple 16.18. Obtenir un mot de passe à partir d'une commande externe

```
{EXT}cat /mysecretpasswordfile
```

Exemple 16.19. Obtenir un mot de passe à partir d'un fichier externe et le mettre en cache pendant 500 millisecondes

```
{EXTC:500}cat /mysecretpasswordfile
```

Rapporter un bogue

16.2. MODULES PERSONNALISÉS

Si les modules de connexion fournis avec l'infrastructure de sécurité EAP ne fonctionnent pas avec votre environnement de sécurité, vous pouvez écrire votre propre implémentation de module de connexion personnalisée. **AuthenticationManager** nécessite un modèle d'utilisation particulière des groupes de principaux **Subject**. Vous devez comprendre les fonctionnalités de stockage des informations de la classe Subject JAAS et l'usage attendu de ces fonctionnalités pour écrire un module de connexion qui fonctionne avec l' **AuthenticationManager**.

Cette section examine ce besoin et introduit deux extraits d'implémentations de **LoginModule** qui peuvent vous aider à mettre en place des modules de connexion personnalisés.

Vous pouvez obtenir des informations de sécurité associées à un **Subject** en utilisant les méthodes suivantes :

```
java.util.Set getPrincipals()
java.util.Set getPrincipals(java.lang.Class c)
java.util.Set getPrivateCredentials()
java.util.Set getPrivateCredentials(java.lang.Class c)
java.util.Set getPublicCredentials()
java.util.Set getPublicCredentials(java.lang.Class c)
```

Pour les identités et les rôles **`Subject`** EAP a sélectionné le choix le plus logique : les groupes de principaux obtenus via **`getPrincipals()`** et **`getPrincipals(java.lang.Class)`**. Le modèle d'utilisation est le suivant :

- Les éléments d'identité de l'utilisateur (nom d'utilisateur, numéro de sécurité sociale, ID de l'employé, par exemple;) sont stockés sous forme d'objets **`java.security.Principal`** dans les groupements **`SubjectPrincipals`** L'application **`Principal`** qui représente l'identité de l'utilisateur doit fonder des comparaisons et l'équivalence sur le nom du principal. Une implémentation appropriée est disponible sous forme de classe **`org.jboss.security.SimplePrincipal`**. D'autres instances **`Principal`** peuvent être ajoutées aux groupements **`SubjectPrincipals`** selon les besoins.

- Les rôles utisateur assignés sont également stockés dans le groupe **`Principals`**, et sont groupés dans des ensembles de rôles nommés utilisant les instances **`java.security.acl.Group`**. L'interface **`Group`** définit une collection de **`Principal`** et /ou **`Group`** et est une sous-interface de **`java.security.Principal`**.

- Le nombre d'ensembles de rôles qui vous souhaitez peut être assigné à un **`Subject`**.

- Le framework de sécurité EAP utilise deux ensembles de rôles bien connus ayant pour noms **`Roles`** et **`CallerPrincipal`**.

  - Le groupe **`Roles`** est une collection de **`Principal`** de rôles nommés tels qu'ils sont connus dans le domaine d'application sous lequel le **`Subject`** a été authentifié. L'ensemble de rôles est utilisé par des méthodes comme **`EJBContext.isCallerInRole(String)`**, que les EJB peuvent utiliser pour voir si l'appelant en cours appartient au rôle de domaine de l'application nommée. La logique d'intercepteur de sécurité qui s'occupe des vérifications des permissions de méthode utilise également cet ensemble de rôles.

  - Le **`CallerPrincipal`** **`Group`** consiste en une seule identité **`Principal`** assignée à l'utilisateur dans le domaine de sécurité. La méthode **`EJBContext.getCallerPrincipal()`** utilise le **`CallerPrincipal`** pour permettre au domaine d'application de procéder au mappage de l'identité de l'environnement de l'opération à une identité d'utilisateur qui convienne à l'application. Si un **`Subject`** ne possède pas de **`CallerPrincipal`** **`Group`**, l'identité de l'application sera la même que l'identité de l'environnement opérationnel.

Rapporter un bogue

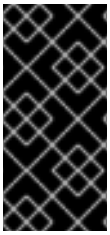
16.2.1. Support des modèles d'utilisation de Subject

Pour simplifier une implémentation correcte des modèles d'utilisation de **`Subject`** décrite dans Section 16.2, « Modules personnalisés », EAP inclut des modules de connexion qui mettent dans le **`Subject`** authentifié un modèle qui oblige une bonne utilisation du **`Subject`**.

AbstractServerLoginModule

La classe la plus standard parmi les deux est **`org.jboss.security.auth.spi.AbstractServerLoginModule`**.

Procure une implémentation de l'interface **`javax.security.auth.spi.LoginModule`** et procure des méthodes abstraties pour les tâches clé spécifiques à une infrastructure de sécurité d'environnement d'opération. Les informations clé de la classe sont expliqués dans Exemple 16.20, « Fragment de classe AbstractServerLoginModule ». La documentation JavaDoc détaille les responsabilités des sous-classes.



IMPORTANT

La variable de l'instance **`loginOk`** est pivotale. Elle doit être définie à **`true`** si la connexion réussit, ou à **`false`** par des sous-classes qui remplacent la méthode de connexion. Si cette variable n'est pas correcte, la méthode de validation ne mettra pas le sujet à jour correctement.

La vérification les résultats de la phase de connexion permet aux modules de connexion d'êtres enchaînés ensemble par des indicateurs de contrôle. Ces indicateurs de contrôle ne nécessitent pas le succès des modules de connexion au cours du processus d'authentification.

Exemple 16.20. Fragment de classe AbstractServerLoginModule

```
package org.jboss.security.auth.spi;
/**
 *  This class implements the common functionality required for a JAAS
 *  server-side LoginModule and implements the PicketBox standard
 *  Subject usage pattern of storing identities and roles. Subclass
 *  this module to create your own custom LoginModule and override the
 *  login(), getRoleSets(), and getIdentity() methods.
 */
public abstract class AbstractServerLoginModule
    implements javax.security.auth.spi.LoginModule
{
    protected Subject subject;
    protected CallbackHandler callbackHandler;
    protected Map sharedState;
    protected Map options;
    protected Logger log;

    /** Flag indicating if the shared credential should be used */
    protected boolean useFirstPass;
    /**
     * Flag indicating if the login phase succeeded. Subclasses that
     * override the login method must set this to true on successful
     * completion of login
     */
    protected boolean loginOk;

    // ...
    /**
     * Initialize the login module. This stores the subject,
     * callbackHandler and sharedState and options for the login
     * session. Subclasses should override if they need to process
     * their own options. A call to super.initialize(...)  must be
     * made in the case of an override.
     *
     * <p>
     * The options are checked for the  <em>password-stacking</em>
parameter.
     * If this is set to "useFirstPass", the login identity will be
taken from the
     * <code>javax.security.auth.login.name</code> value of the
sharedState map,
     * and the proof of identity from the
```

```
     * <code>javax.security.auth.login.password</code> value of the
sharedState map.
     *
     * @param subject the Subject to update after a successful login.
     * @param callbackHandler the CallbackHandler that will be used to
obtain the
     * the user identity and credentials.
     * @param sharedState a Map shared between all configured login
module instances
     * @param options the parameters passed to the login module.
     */
    public void initialize(Subject subject,
                           CallbackHandler callbackHandler,
                           Map sharedState,
                           Map options)
    {
        // ...
    }


    /**
     *  Looks for javax.security.auth.login.name and
     *  javax.security.auth.login.password values in the sharedState
     *  map if the useFirstPass option was true and returns true if
     *  they exist. If they do not or are null this method returns
     *  false.
     *  Note that subclasses that override the login method
     *  must set the loginOk var to true if the login succeeds in
     *  order for the commit phase to populate the Subject. This
     *  implementation sets loginOk to true if the login() method
     *  returns true, otherwise, it sets loginOk to false.
     */
    public boolean login()
        throws LoginException
    {
        // ...
    }

    /**
     *  Overridden by subclasses to return the Principal that
     *  corresponds to the user primary identity.
     */
    abstract protected Principal getIdentity();

    /**
     *  Overridden by subclasses to return the Groups that correspond
     *  to the role sets assigned to the user. Subclasses should
     *  create at least a Group named "Roles" that contains the roles
     *  assigned to the user.  A second common group is
     *  "CallerPrincipal," which provides the application identity of
     *  the user rather than the security domain identity.
     *
     *  @return Group[] containing the sets of roles
     */
    abstract protected Group[] getRoleSets() throws LoginException;
}
```

UsernamePasswordLoginModule

Le second module de connexion basé abstract pour les modules de connexion personnalisés est le module **`org.jboss.security.auth.spi.UsernamePasswordLoginModule`**.

Ce module de connexion simplifie encore plus l'implémentation du module de connexion personnalisé par un nom d'utilisateur basé sur une chaîne comme identité d'utilisateur et un mot de passe **`char[]`** comme informations d'authentification. Il prend également en charge le mappage des utilisateurs anonymes (indiqué par un nom d'utilisateur et un mot de passe null) à un principal sans rôle. Les informations clé de la classe sont mises en évidence dans le fragment de classe suivant. Les commentaires JavaDoc détaillent les responsabilités des sous-classes.

Exemple 16.21. Fragment de classe UsernamePasswordLoginModule

```
package org.jboss.security.auth.spi;

/**
 *  An abstract subclass of AbstractServerLoginModule that imposes a
 *  an identity == String username, credentials == String password
 *  view on the login process. Subclasses override the
 *  getUsersPassword() and getUsersRoles() methods to return the
 *  expected password and roles for the user.
 */
public abstract class UsernamePasswordLoginModule
    extends AbstractServerLoginModule
{
    /** The login identity */
    private Principal identity;
    /** The proof of login identity */
    private char[] credential;
    /** The principal to use when a null username and password are seen
*/
    private Principal unauthenticatedIdentity;

    /**
     * The message digest algorithm used to hash passwords. If null then
     * plain passwords will be used. */
    private String hashAlgorithm = null;

    /**
     *  The name of the charset/encoding to use when converting the
     * password String to a byte array. Default is the platform's
     * default encoding.
     */
     private String hashCharset = null;

    /** The string encoding format to use. Defaults to base64. */
    private String hashEncoding = null;

    // ...

    /**
     *  Override the superclass method to look for an
     *  unauthenticatedIdentity property. This method first invokes
     *  the super version.
```

```
     *
     *  @param options,
     *  @option unauthenticatedIdentity: the name of the principal to
     *  assign and authenticate when a null username and password are
     *  seen.
     */
    public void initialize(Subject subject,
                           CallbackHandler callbackHandler,
                           Map sharedState,
                           Map options)
    {
        super.initialize(subject, callbackHandler, sharedState,
                         options);
        // Check for unauthenticatedIdentity option.
        Object option = options.get("unauthenticatedIdentity");
        String name = (String) option;
        if (name != null) {
            unauthenticatedIdentity = new SimplePrincipal(name);
        }
    }

    // ...

    /**
     *  A hook that allows subclasses to change the validation of the
     *  input password against the expected password. This version
     *  checks that neither inputPassword or expectedPassword are null
     *  and that inputPassword.equals(expectedPassword) is true;
     *
     *  @return true if the inputPassword is valid, false otherwise.
     */
    protected boolean validatePassword(String inputPassword,
                                       String expectedPassword)
    {
        if (inputPassword == null || expectedPassword == null) {
            return false;
        }
        return inputPassword.equals(expectedPassword);
    }

    /**
     *  Get the expected password for the current username available
     * via the getUsername() method. This is called from within the
     * login() method after the CallbackHandler has returned the
     * username and candidate password.
     *
     * @return the valid password String
     */
    abstract protected String getUsersPassword()
        throws LoginException;
}
```

Sous-classement des modules de connexion

Le choix de sous-classement du **`AbstractServerLoginModule`** versus

UsernamePasswordLoginModule est basé sur le fait de savoir si des informations d'identification et un nom d'utilisateur basés chaîne sont utilisables avec la technologie d'authentification pour laquelle vous écrivez le module de connexion. Si la sémantique basée chaîne est valide, alors ce sera la sous classe **UsernamePasswordLoginModule**, sinon, ce sera la sous-classe **AbstractServerLoginModule**.

Étapes de sous-classement

Les étapes que votre module de connexion personnalisée doit s'exécuter dépendent de la classe de module de base de connexion que vous choisissez. Lorsque vous écrivez un module de connexion personnalisé qui s'intègre à votre infrastructure de sécurité, vous devriez commencer en sous-classant **AbstractServerLoginModule** ou **UsernamePasswordLoginModule** pour faire en sorte que votre module de connexion fournisse les informations de **Principal** authentifié dans la forme prévue par le gestionnaire de sécurité EAP.

Lors du sous-classement de **AbstractServerLoginModule**, vous devrez remplacer ce qui suit :

- **void initialize(Subject, CallbackHandler, Map, Map)**: si vous avez des options personnalisées pour le traitement.
- **boolean login()** : utilisé pour effectuer l'authentification. Veillez bien à définir l'instance **loginOk** sur true si la connexion réussit, sur false si elle échoue.
- **Principal getIdentity()** : pour retourner l'objet **Principal** pour l'utilisateur authentifié dans l'étape **log()**.
- **Group[] getRoleSets()** : utilisé pour retourner au moins un **Group** nommé **Roles** qui contienne les rôles assignés au **Principal** authentifié durant le **login()**. Un second **Group** commun se nomme **CallerPrincipal** et il procure l'identité de l'application de l'utilisateur au lieu de l'identité du domaine de sécurité.

Quand vous sous-classez **UsernamePasswordLoginModule**, vous devrez remplacer ce qui suit :

- **void initialize(Subject, CallbackHandler, Map, Map)**: si vous avez des options personnalisées pour le traitement.
- **Group[] getRoleSets()** : utilisé pour retourner au moins un **Group** nommé **Roles** qui contienne les rôles assignés au **Principal** authentifié durant le **login()**. Un second **Group** commun se nomme **CallerPrincipal** et il procure l'identité de l'application de l'utilisateur au lieu de l'identité du domaine de sécurité.
- **String getUsersPassword()** : pour retourner un mot de passe pour le nom d'utilisateur en cours, via la méthode **getUsername()**. La méthode **getUsersPassword()** est appelée à partir de **login()** une fois que le **callbackhandler** a retourné un nom d'utlisateur et un mot de passe de candidat.

Rapporter un bogue

16.2.2. Exemple de LoginModule personnalisé

Les informations suivantes vont vous aider à créer un exemple de module de connexion personnalisé qui étend le **UsernamePasswordLoginModule** pour obtenir un mot de passe utilisateur et des noms de rôle à partir d'une recherche JNDI.

À la fin de cette section, vous aurez créé un module de connexion de contexte JNDI personnalisé qui pourra retourner un mot de passe si vous effectuez une recherche sur le contexte à l'aide d'un nom de la

forme **password/<username>** (avec **<username>** comme utilisateur en cours d'authentification). De même, une recherche de la forme **roles/<username>** retourne les rôles d'utilisateur demandés. Vous trouverez dans Exemple 16.22, « Module de connexion personnalisé JndiUserAndPassLoginModule » le code source du module de connexion personnalisé **JndiUserAndPassLoginModule**.

Notez que comme cela étend le **UsernamePasswordLoginModule**, de JBoss, le **JndiUserAndPassLoginModule** obtient le mot de passe en provenance du store JNDI. Le **JndiUserAndPassLoginModule** n'interfère pas avec les opérations JAAS LoginModule.

Exemple 16.22. Module de connexion personnalisé JndiUserAndPassLoginModule

```
package org.jboss.book.security.ex2;

import java.security.acl.Group;
import java.util.Map;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.security.auth.Subject;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.login.LoginException;
import org.jboss.logging.Logger;
import org.jboss.security.SimpleGroup;
import org.jboss.security.SimplePrincipal;
import org.jboss.security.auth.spi.UsernamePasswordLoginModule;
/**
 * An example custom login module that obtains passwords and roles for a
user from a JNDI lookup.
 *
 * @author Scott.Stark@jboss.org
 */
public class JndiUserAndPassLoginModule extends
UsernamePasswordLoginModule {
  /** The JNDI name to the context that handles the password/username
lookup */
  private String userPathPrefix;
  /** The JNDI name to the context that handles the roles/username
lookup */
  private String rolesPathPrefix;
  private static Logger log =
Logger.getLogger(JndiUserAndPassLoginModule.class);
  /**
   * Override to obtain the userPathPrefix and rolesPathPrefix options.
   */
  @Override
  public void initialize(Subject subject, CallbackHandler
callbackHandler, Map sharedState, Map options) {
    super.initialize(subject, callbackHandler, sharedState, options);
    userPathPrefix = (String) options.get("userPathPrefix");
    rolesPathPrefix = (String) options.get("rolesPathPrefix");
  }
  /**
   * Get the roles the current user belongs to by querying the
rolesPathPrefix + '/' + super.getUsername() JNDI location.
   */
  @Override
  protected Group[] getRoleSets() throws LoginException {
```

```
    try {
      InitialContext ctx = new InitialContext();
      String rolesPath = rolesPathPrefix + '/' + super.getUsername();
      String[] roles = (String[]) ctx.lookup(rolesPath);
      Group[] groups = { new SimpleGroup("Roles") };
      log.info("Getting roles for user=" + super.getUsername());
      for (int r = 0; r < roles.length; r++) {
        SimplePrincipal role = new SimplePrincipal(roles[r]);
        log.info("Found role=" + roles[r]);
        groups[0].addMember(role);
      }
      return groups;
    } catch (NamingException e) {
      log.error("Failed to obtain groups for user=" +
super.getUsername(), e);
      throw new LoginException(e.toString(true));
    }
  }
  /**
   * Get the password of the current user by querying the userPathPrefix
+ '/' + super.getUsername() JNDI location.
   */
  @Override
  protected String getUsersPassword() throws LoginException {
    try {
      InitialContext ctx = new InitialContext();
      String userPath = userPathPrefix + '/' + super.getUsername();
      log.info("Getting password for user=" + super.getUsername());
      String passwd = (String) ctx.lookup(userPath);
      log.info("Found password=" + passwd);
      return passwd;
    } catch (NamingException e) {
      log.error("Failed to obtain password for user=" +
super.getUsername(), e);
      throw new LoginException(e.toString(true));
    }
  }
}
```

Exemple 16.23. Definition d'un domaine de sécurité security-ex2 avec le module de connexion personnalisé nouvellement créé.

```
/subsystem=security/security-domain=security-ex2/:add
/subsystem=security/security-domain=security-
ex2/authentication=classic:add
/subsystem=security/security-domain=security-
ex2/authentication=classic/login-module=ex2/:add(\
flag=required,\
code=org.jboss.book.security.ex2.JndiUserAndPassLoginModule,\
module-options=[("userPathPrefix"=>"/security/store/password"),\
("rolesPathPrefix"=>"/security/store/roles")]\
)
```

–

Le choix d'utilisation du module de connexion personnalisé **`JndiUserAndPassLoginModule`** pour l'authentification côté serveur de l'utilisateur est déterminé par la configuration de la connexion dans l'exemple de domaine de sécurité. Le descripteur JAR EJB **`META-INF/jboss-ejb3.xml`** définit le domaine de sécurité. Dans une application web, il fait partie du fichier **`WEB-INF/jboss-Web.xml`**.

Exemple 16.24. `jboss-ejb3.xml` Example

```
<?xml version="1.0"?>
<jboss:ejb-jar xmlns:jboss="http://www.jboss.com/xml/ns/javaee"
xmlns="http://java.sun.com/xml/ns/javaee" xmlns:s="urn:security"
version="3.1" impl-version="2.0">
  <assembly-descriptor>
    <s:security>
      <ejb-name>*</ejb-name>
      <s:security-domain>security-ex2</s:security-domain>
    </s:security>
  </assembly-descriptor>
</jboss:ejb-jar>
```

Exemple 16.25. `jboss-web.xml` example

```
<?xml version="1.0"?>
<jboss-web>
    <security-domain>security-ex2</security-domain>
</jboss-web>
```

Rapporter un bogue

CHAPITRE 17. SÉCURITÉ BASÉE-RÔLE POUR LES APPLICATIONS

17.1. JAVA AUTHENTICATION ET AUTHORIZATION SERVICE (JAAS)

Java Authentication and Authorization Service (JAAS) est une API de sécurité qui consiste en un ensemble de packages Java conçus pour l'autorisation et l'authentification des utilisateurs. L'API est une implémentation Java du framework standard PAM (de l'anglais Pluggable Authentication Modules). Il étend l'architecture de contrôle d'accès de Java Enterprise Edition pour approuver l'autorisation basée utilisateur.

Dans JBoss EAP 6, JAAS ne fournit qu'une sécurité déclarative basée rôle. Pour plus d'informations sur le sécurité déclarative, voir Section 10.2, « Sécurité déclarative ».

JAAS est indépendant de toute technologie d'authentification sous-jacente, comme Kerberos ou LDAP. Vous pouvez modifier votre structure de sécurité sous-jacente sans modifier votre application. Vous devez uniquement modifier la configuration JAAS.

Rapporter un bogue

17.2. JAVA AUTHENTICATION ET AUTHORIZATION SERVICE (JAAS)

L'architecture de sécurité de JBoss EAP 6 comprend le sous-système de configuration de sécurité, des configurations de sécurité propres aux applications qui sont incluses dans plusieurs fichiers de configuration de l'application.

Domaine, Groupes de serveurs et Configuration spécifique au serveur

Les groupes de serveurs (dans un domaine géré) et les serveurs (dans un serveur autonome) comprennent une configuration des domaines de la sécurité. Un domaine de sécurité comprend des informations sur une combinaison d'authentification, d'autorisation, de mappage et de modules, avec détails de configuration. Une application spécifie quel domaine de sécurité est exigé, par son nom, dans son fichier **`jboss-web.xml`**.

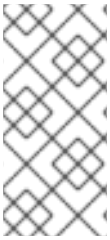
Configuration spécifique à une application

Une configuration spécifique à une application a lieu dans un ou plusieurs fichiers.

Tableau 17.1. Fichiers de configuration spécifique à une application

Fichier	Description
ejb-jar.xml	Le descripteur de déploiement pour une application Enterprise JavaBeans (EJB), situé dans le répertoire **`META-INF`** de l'archive. Utiliser le fichier **`ejb-jar.xml`** pour préciser les rôles et les mapper aux principaux, au niveau de l'application. Vous pouvez également limiter certaines méthodes et classes spécifiques à certains rôles. Également utilisé pour les autres configurations EJB spécifiques non liées à la sécurité.

Fichier	Description
web.xml	Le descripteur de déploiement pour une application web en Java Enterprise Edition (EE). Utilisez le fichier **`web.xml`** pour déclarer les contraintes de transport et ressources, comme limiter les types de demandes HTTP autorisées. Vous pouvez également configurer l'authentification basée-web dans ce fichier. Utilisé également pour d'autres configurations spécifiques à l'application non liées à la sécurité. Le domaine de sécurité que l'application utilise pour l'authentification et l'autorisation est défini dans **`jboss-web.xml`**.
jboss-ejb3.xml	Contient des extensions au descripteur **`ejb-jar.xml`** spécifiques à JBoss.
jboss-web.xml	Contient des extensions au descripteur **`web.xml`** spécifiques à JBoss.



NOTE

Les fichiers **`ejb-jar.xml`** et **`web.xml`** sont définis dans la spécification Java Enterprise Edition (Java EE). Le fichier **`jboss-ejb3.xml`** fournit des extensions spécifiques à JBoss pour le fichier **`ejb-jar.xml`**, et le fichier **`jboss-web.xml`** fournit des extensions spécifiques à JBoss pour le fichier **`web.xml`**.

Rapporter un bogue

17.3. UTILISER UN DOMAINE DE SÉCURITÉ DANS VOTRE APPLICATION

Aperçu

Pour utiliser un domaine de sécurité dans votre application, vous devez tout d'abord définir le domaine dans le fichier de configuration du serveur, puis vous devez l'activer pour une application dans le descripteur de déploiement de l'application. Ensuite, vous devez ajouter les annotations requises à l'EJB qui les utilise. Cette rubrique décrit les étapes requises pour utiliser un domaine de sécurité dans votre application.

AVERTISSEMENT

Si une application fait partie d'un domaine de sécurité qui utilise un cache d'authentification, les authentifications utilisateur de cette application seront rendues disponibles à d'autres applications dans ce domaine de sécurité.

Procédure 17.1. Configurez votre application pour qu'elle puisse utiliser un domaine de sécurité

1. **Définir le domaine de sécurité**
   Vous devez définir le domaine de sécurité dans le fichier de configuration du serveur, puis l'activer pour une application dans le fichier du descripteur de l'application.

   a. **Configurez le domaine de sécurité dans le fichier de configuration du serveur**
   Le domaine de sécurité est configuré dans le sous-système de **`sécurité`** du fichier de configuration du serveur. Si l'instance de JBoss EAP 6 s'exécute dans un domaine géré, il s'agira du fichier **`domain/configuration/domain.xml`**. Si l'instance de JBoss EAP 6 s'exécute comme un serveur autonome, ce sera le fichier **`standalone/configuration/standalone.xml`**.

   Les domaines de sécurité **`other`**, **`jboss-web-policy`**, et **`jboss-ejb-policy`** sont fournis par défaut dans JBoss EAP 6. L'exemple XML suivant a été copié à partir du sous-système de **`sécurité`** dans le fichier de configuration du serveur.

   L'attribut **`cach-type`** d'un domaine de sécurité spécifie un cache pour pouvoir effectuer des contrôles d'authentification plus rapides. Les valeur autorisées sont les valeur par **`défaut`** en cas de simile map comme cache ou **`infinispan`** pour un cache Infinispan.

```
<subsystem xmlns="urn:jboss:domain:security:1.2">
    <security-domains>
        <security-domain name="other" cache-type="default">
            <authentication>
                <login-module code="Remoting" flag="optional">
                    <module-option name="password-stacking"
value="useFirstPass"/>
                </login-module>
                <login-module code="RealmDirect"
flag="required">
                    <module-option name="password-stacking"
value="useFirstPass"/>
                </login-module>
            </authentication>
        </security-domain>
        <security-domain name="jboss-web-policy" cache-
type="default">
            <authorization>
                <policy-module code="Delegating"
flag="required"/>
            </authorization>
        </security-domain>
        <security-domain name="jboss-ejb-policy" cache-
type="default">
            <authorization>
                <policy-module code="Delegating"
flag="required"/>
            </authorization>
        </security-domain>
    </security-domains>
</subsystem>
```

   Vous pouvez configurer des domaines de sécurité supplémentaires selon les besoins par la console de gestion ou par l'interface CLI.

   b. **Activer le domaine de sécurité dans le fichier de descripteur de l'application.**

Le domaine de sécurité est spécifié dans l'élément enfant **<security-domain>** de l'élément **<jboss-web>** du fichier **WEB-INF/jboss-web.xml** de l'application. L'exemple suivant configure un domaine de sécurité nommé **my-domain**.

```
<jboss-web>
    <security-domain>my-domain</security-domain>
</jboss-web>
```

Il s'agit d'une des configurations que vous pouvez indiquer dans le descripteur **WEB-INF/jboss-web.xml**.

2. **Ajoutez l'annotation requise à l'EJB.**
Vous pouvez configurer la sécurité dans EJB par les annotations **@SecurityDomain** et **@RolesAllowed**. L'exemple de code EJB suivant limite l'accès au domaine de sécurité **other** aux utilisateurs ayant pour rôle **guest** (invité).

```
package example.ejb3;

import java.security.Principal;

import javax.annotation.Resource;
import javax.annotation.security.RolesAllowed;
import javax.ejb.SessionContext;
import javax.ejb.Stateless;

import org.jboss.ejb3.annotation.SecurityDomain;

/**
 * Simple secured EJB using EJB security annotations
 * Allow access to "other" security domain by users in a "guest"
role.
 */
@Stateless
@RolesAllowed({ "guest" })
@SecurityDomain("other")
public class SecuredEJB {

   // Inject the Session Context
   @Resource
   private SessionContext ctx;

   /**
    * Secured EJB method using security annotations
    */
   public String getSecurityInfo() {
      // Session context injected using the resource annotation
      Principal principal = ctx.getCallerPrincipal();
      return principal.toString();
   }
}
```

Pour obtenir des exemples de code supplémentaires, voir **ejb-security** Quickstart dans le package JBoss EAP 6 Quickstarts disponible à partir du Portail Clients Red Hat.

Rapporter un bogue

17.4. UTILISATION DE LA SÉCURITÉ BASÉE-RÔLE DANS LES SERVLETS

Pour ajouter la sécurité à un servlet, vous devez mapper chaque servlet à un type d'URL et créer des contraintes de sécurité sur les types d'URL qui doivent être sécurisés. Les contraintes de sécurité limitent l'accès des URL aux rôles. L'authentification et l'autorisation sont gérées par le domaine de sécurité spécifié dans **`jboss-web.xml`** du WAR.

Pré-requis

Avant d'utiliser la sécurité basée-rôles dans une servlet, le domaine de sécurité utilisé pour authentifier et autoriser l'accès doit être configuré sur la plateforme JBoss EAP 6.

Procédure 17.2. Ajout de la sécurité basée-rôle dans les servlets

1. **Ajout de mappages entre les types d'URL et les servlets.**
   Utiliser les éléments **`<servlet-mapping>`** du fichier **`web.xml`** pour mapper les servlets individuels à des types d'URL. L'exemple suivant mappe le serveur nommé **`DisplayOpResult`** au type d'URL **`/DisplayOpResult`**.

   ```
   <servlet-mapping>
       <servlet-name>DisplayOpResult</servlet-name>
       <url-pattern>/DisplayOpResult</url-pattern>
   </servlet-mapping>
   ```

2. **Ajout des contraintes de sécurité aux types d'URL.**
   Pour mapper le type d'URL avec une contrainte de sécurité, utilisez un **`<security-constraint>`**. L'exemple suivant limite l'accès d'un type d'URL **`/DisplayOpResult`** afin qu'il soit accessible aux principaux ayant pour rôle **`eap_admin`**. Le rôle doit être présent dans le domaine de sécurité.

   ```
   <security-constraint>
    <display-name>Restrict access to role eap_admin</display-name>
    <web-resource-collection>
     <web-resource-name>Restrict access to role eap_admin</web-
   resource-name>
     <url-pattern>/DisplayOpResult/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
     <role-name>eap_admin</role-name>
    </auth-constraint>
   </security-constraint>

   <security-role>
     <role-name>eap_admin</role-name>
   </security-role>


   <login-config>
       <auth-method>BASIC</auth-method>
   </login-config>
   ```

Vous aurez besoin d'indiquer la méthode d'authentification, qui peut être une des mthodes suivantes : **`BASIC, FORM, DIGEST, CLIENT-CERT, SPNEGO.`**. Cet exemple utilise l'authentification **`BASIC`**.

3. **Indiquez le domaine de sécurité et le fichier `jboss-web.xml` du WAR.**
   Ajoutez le domaine de sécurité au fichier **`jboss-Web.xml`** du WAR afin de connecter les servlets au domaine de la sécurité configuré sachant comment authentifier et autoriser les principaux selon les contraintes de sécurité. L'exemple suivant utilise le domaine de sécurité appelé **`acme_domain`**.

```
<jboss-web>
 ...
 <security-domain>acme_domain</security-domain>
 ...
</jboss-web>
```

Exemple 17.1. Exemple `web.xml` avec la sécurité basée rôle configurée.

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
         version="3.0">

<display-name>Use Role-Based Security In Servlets</display-name>

<welcome-file-list>
  <welcome-file>/index.jsp</welcome-file>
</welcome-file-list>

<servlet-mapping>
    <servlet-name>DisplayOpResult</servlet-name>
    <url-pattern>/DisplayOpResult</url-pattern>
</servlet-mapping>

<security-constraint>
  <display-name>Restrict access to role eap_admin</display-name>
    <web-resource-collection>
      <web-resource-name>Restrict access to role eap_admin</web-
resource-name>
      <url-pattern>/DisplayOpResult/*</url-pattern>
      </web-resource-collection>
      <auth-constraint>
        <role-name>eap_admin</role-name>
      </auth-constraint>
    </security-constraint>

    <security-role>
      <role-name>eap_admin</role-name>
    </security-role>

    <login-config>
        <auth-method>BASIC</auth-method>
```

```
    </login-config>

</web-app>
```

Rapporter un bogue

17.5. UTILISATION D'UNE AUTHENTIFICATION SYSTÈME DE TIERCE PARTIE POUR VOTRE APPLICATION

Vous pouvez intégrer des systèmes de sécurité de tierce partie avec JBoss EAP 6. Ces types de systèmes sont habituellement basés sur des tokens. Le système externe effectue l'authentification et passe un token à l'application web via les en-têtes de demande. Ceci est souvent dénommé *authentification de périmètre*. Pour configurer l'authentification de périmètre dans votre application, ajoutez une valve d'authentification personnalisée. Si vous avez une valve de fournisseur tiers, veillez à ce qu'elle soit sur votre chemin de classe et suivez les exemples ci-dessous, ainsi que la documentation pour votre module d'authentification tiers.

NOTE

L'emplacement pour la configuration des valves a changé dans JBoss EAP 6. Il n'y a plus de descripteur de déploiement **`context.xml`**. Les valves sont configurées directement dans le descripteur **`jboss-web.xml`** à la place. Le fichier **`context.xml`** peut maintenant être ignoré.

Exemple 17.2. Valve d'authentification de base

```
<jboss-web>
  <valve>
    <class-
name>org.jboss.security.negotiation.NegotiationAuthenticator</class-
name>
  </valve>
</jboss-web>
```

Cette valve est utilisée pour les SSO basés-Kerberos. Montre également les modèles les plus simples d'indication d'authentificateurs de tierce partie pour votre application web.

Exemple 17.3. Personnaliser une valve avec des attributs d'en-tête

```
<jboss-web>
  <valve>
    <class-
name>org.jboss.web.tomcat.security.GenericHeaderAuthenticator</class-
name>
    <param>
      <param-name>httpHeaderForSSOAuth</param-name>
      <param-value>sm_ssoid,ct-remote-user,HTTP_OBLIX_UID</param-value>
    </param>
    <param>
      <param-name>sessionCookieForSSOAuth</param-name>
```

```
      <param-value>SMSESSION,CTSESSION,ObSSOCookie</param-value>
    </param>
  </valve>
</jboss-web>
```

Cet exemple montre comment définir des attributs personnalisés sur votre valve. L'authentificateur vérifie la présence de l'en-tête ID et de la clé de session, puis les passe au framework JAAS qui actionne la couche de sécurité, comme le nom d'utilisateur et le mot de passe. Vous avez besoin d'un module de connexion JAAS personnalisé qui puisse traiter le nom d'utilisateur et le mot de passe tout en remplissant le sujet par les rôles qui conviennent. Si aucune valeur d'en-tête ne correspond aux valeurs configurées, les sémantiques de d'authentification basée sur les formulaires réguliers s'appliqueront.

Rédaction d'un authentificateur personnalisé

Rédiger vous-même votre authentificateur est en dehors de la portée de ce document. Cependant, le code Java suivant est fourni comme exemple.

Exemple 17.4. GenericHeaderAuthenticator.java

```
/*
 * JBoss, Home of Professional Open Source.
 * Copyright 2006, Red Hat Middleware LLC, and individual contributors
 * as indicated by the @author tags. See the copyright.txt file in the
 * distribution for a full listing of individual contributors.
 *
 * This is free software; you can redistribute it and/or modify it
 * under the terms of the GNU Lesser General Public License as
 * published by the Free Software Foundation; either version 2.1 of
 * the License, or (at your option) any later version.
 *
 * This software is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with this software; if not, write to the Free
 * Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
 * 02110-1301 USA, or see the FSF site: http://www.fsf.org.
 */

package org.jboss.web.tomcat.security;

import java.io.IOException;
import java.security.Principal;
import java.util.StringTokenizer;

import javax.management.JMException;
import javax.management.ObjectName;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.catalina.Realm;
```

```
import org.apache.catalina.Session;
import org.apache.catalina.authenticator.Constants;
import org.apache.catalina.connector.Request;
import org.apache.catalina.connector.Response;
import org.apache.catalina.deploy.LoginConfig;
import org.jboss.logging.Logger;

import org.jboss.as.web.security.ExtendedFormAuthenticator;

/**
 * JBAS-2283: Provide custom header based authentication support
 *
 * Header Authenticator that deals with userid from the request header
Requires
 * two attributes configured on the Tomcat Service - one for the http
header
 * denoting the authenticated identity and the other is the SESSION
cookie
 *
 * @author <a href="mailto:Anil.Saldhana@jboss.org">Anil Saldhana</a>
 * @author <a href="mailto:sguilhen@redhat.com">Stefan Guilhen</a>
 * @version $Revision$
 * @since Sep 11, 2006
 */
public class GenericHeaderAuthenticator extends
ExtendedFormAuthenticator {
  protected static Logger log = Logger
      .getLogger(GenericHeaderAuthenticator.class);

  protected boolean trace = log.isTraceEnabled();

  // JBAS-4804: GenericHeaderAuthenticator injection of ssoid and
  // sessioncookie name.
  private String httpHeaderForSSOAuth = null;

  private String sessionCookieForSSOAuth = null;

  /**
   * <p>
   * Obtain the value of the <code>httpHeaderForSSOAuth</code>
attribute. This
   * attribute is used to indicate the request header ids that have to
be
   * checked in order to retrieve the SSO identity set by a third party
   * security system.
   * </p>
   *
   * @return a <code>String</code> containing the value of the
   *         <code>httpHeaderForSSOAuth</code> attribute.
   */
  public String getHttpHeaderForSSOAuth() {
    return httpHeaderForSSOAuth;
  }

  /**
   * <p>
```

```
    * Set the value of the <code>httpHeaderForSSOAuth</code> attribute.
This
    * attribute is used to indicate the request header ids that have to
be
    * checked in order to retrieve the SSO identity set by a third party
    * security system.
    * </p>
    *
    * @param httpHeaderForSSOAuth
    *            a <code>String</code> containing the value of the
    *            <code>httpHeaderForSSOAuth</code> attribute.
    */
  public void setHttpHeaderForSSOAuth(String httpHeaderForSSOAuth) {
    this.httpHeaderForSSOAuth = httpHeaderForSSOAuth;
  }

  /**
    * <p>
    * Obtain the value of the <code>sessionCookieForSSOAuth</code>
attribute.
    * This attribute is used to indicate the names of the SSO cookies
that may
    * be present in the request object.
    * </p>
    *
    * @return a <code>String</code> containing the names (separated by a
    *         <code>','</code>) of the SSO cookies that may have been
set by a
    *         third party security system in the request.
    */
  public String getSessionCookieForSSOAuth() {
    return sessionCookieForSSOAuth;
  }

  /**
    * <p>
    * Set the value of the <code>sessionCookieForSSOAuth</code>
attribute. This
    * attribute is used to indicate the names of the SSO cookies that may
be
    * present in the request object.
    * </p>
    *
    * @param sessionCookieForSSOAuth
    *            a <code>String</code> containing the names (separated
by a
    *            <code>','</code>) of the SSO cookies that may have been
set by
    *            a third party security system in the request.
    */
  public void setSessionCookieForSSOAuth(String sessionCookieForSSOAuth)
{
    this.sessionCookieForSSOAuth = sessionCookieForSSOAuth;
  }

  /**
```

```
     * <p>
     * Creates an instance of <code>GenericHeaderAuthenticator</code>.
     * </p>
     */
    public GenericHeaderAuthenticator() {
      super();
    }

    public boolean authenticate(Request request, HttpServletResponse
response,
        LoginConfig config) throws IOException {
      log.trace("Authenticating user");

      Principal principal = request.getUserPrincipal();
      if (principal != null) {
        if (trace)
          log.trace("Already authenticated '" + principal.getName() +
"'");
        return true;
      }

      Realm realm = context.getRealm();
      Session session = request.getSessionInternal(true);

      String username = getUserId(request);
      String password = getSessionCookie(request);

      // Check if there is sso id as well as sessionkey
      if (username == null || password == null) {
        log.trace("Username is null or password(sessionkey) is
null:fallback to form auth");
        return super.authenticate(request, response, config);
      }
      principal = realm.authenticate(username, password);

      if (principal == null) {
        forwardToErrorPage(request, response, config);
        return false;
      }

      session.setNote(Constants.SESS_USERNAME_NOTE, username);
      session.setNote(Constants.SESS_PASSWORD_NOTE, password);
      request.setUserPrincipal(principal);

      register(request, response, principal, HttpServletRequest.FORM_AUTH,
          username, password);
      return true;
    }

    /**
     * Get the username from the request header
     *
     * @param request
     * @return
     */
    protected String getUserId(Request request) {
```

```
    String ssoid = null;
    // We can have a comma-separated ids
    String ids = "";
    try {
      ids = this.getIdentityHeaderId();
    } catch (JMException e) {
      if (trace)
        log.trace("getUserId exception", e);
    }
    if (ids == null || ids.length() == 0)
      throw new IllegalStateException(
          "Http headers configuration in tomcat service missing");

    StringTokenizer st = new StringTokenizer(ids, ",");
    while (st.hasMoreTokens()) {
      ssoid = request.getHeader(st.nextToken());
      if (ssoid != null)
        break;
    }
    if (trace)
      log.trace("SSOID-" + ssoid);
    return ssoid;
  }

  /**
   * Obtain the session cookie from the request
   *
   * @param request
   * @return
   */
  protected String getSessionCookie(Request request) {
    Cookie[] cookies = request.getCookies();
    log.trace("Cookies:" + cookies);
    int numCookies = cookies != null ? cookies.length : 0;

    // We can have comma-separated ids
    String ids = "";
    try {
      ids = this.getSessionCookieId();
      log.trace("Session Cookie Ids=" + ids);
    } catch (JMException e) {
      if (trace)
        log.trace("checkSessionCookie exception", e);
    }
    if (ids == null || ids.length() == 0)
      throw new IllegalStateException(
          "Session cookies configuration in tomcat service missing");

    StringTokenizer st = new StringTokenizer(ids, ",");
    while (st.hasMoreTokens()) {
      String cookieToken = st.nextToken();
      String val = getCookieValue(cookies, numCookies, cookieToken);
      if (val != null)
        return val;
    }
    if (trace)
```

```
      log.trace("Session Cookie not found");
    return null;
  }

  /**
   * Get the configured header identity id in the tomcat service
   *
   * @return
   * @throws JMException
   */
  protected String getIdentityHeaderId() throws JMException {
    if (this.httpHeaderForSSOAuth != null)
      return this.httpHeaderForSSOAuth;
    return (String) mserver.getAttribute(new ObjectName(
        "jboss.web:service=WebServer"), "HttpHeaderForSSOAuth");
  }

  /**
   * Get the configured session cookie id in the tomcat service
   *
   * @return
   * @throws JMException
   */
  protected String getSessionCookieId() throws JMException {
    if (this.sessionCookieForSSOAuth != null)
      return this.sessionCookieForSSOAuth;
    return (String) mserver.getAttribute(new ObjectName(
        "jboss.web:service=WebServer"), "SessionCookieForSSOAuth");
  }

  /**
   * Get the value of a cookie if the name matches the token
   *
   * @param cookies
   *            array of cookies
   * @param numCookies
   *            number of cookies in the array
   * @param token
   *            Key
   * @return value of cookie
   */
  protected String getCookieValue(Cookie[] cookies, int numCookies,
      String token) {
    for (int i = 0; i < numCookies; i++) {
      Cookie cookie = cookies[i];
      log.trace("Matching cookieToken:" + token + " with cookie name="
          + cookie.getName());
      if (token.equals(cookie.getName())) {
        if (trace)
          log.trace("Cookie-" + token + " value=" + cookie.getValue());
        return cookie.getValue();
      }
    }
    return null;
  }
}
```

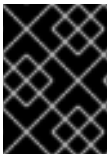
Rapporter un bogue

CHAPITRE 18. MIGRATION

18.1. CONFIGURER LES CHANGEMENTS DE SÉCURITÉ D'APPLICATIONS

Configurer la sécurité pour l'authentification de base

Dans les versions précédentes de JBoss EAP, les fichiers de propriétés qui se trouvent dans le répertoire ***EAP_HOME*/server/*SERVER_NAME*/conf/** étaient sur un chemin de classe et on pouvait les trouver facilement avec **UsersRolesLoginModule**. Dans JBoss EAP 6, la structure du répertoire a changé. Les fichiers de propriétés doivent être empaquetés dans l'application pour les rendre disponibles par le chemin de classe.



IMPORTANT

Vous devez interrompre le serveur avant de modifier le fichier de configuration du serveur pour que votre changement puisse être persisté au redémarrage du serveur.

Pour configurer la sécurité de l'authentification de base, ajouter un nouveau domaine de sécurité sous **security-domains** au **standalone/configuration/standalone.xml** ou au fichier de configuration du serveur **domain/configuration/domain.xml**:

```
<security-domain name="example">
    <authentication>
        <login-module code="UsersRoles" flag="required">
            <module-option name="usersProperties"
                    value="${jboss.server.config.dir}/example-
users.properties"/>
            <module-option name="rolesProperties"
                    value="${jboss.server.config.dir}/example-
roles.properties"/>
        </login-module>
    </authentication>
</security-domain>
```

Si l'instance de JBoss EAP 6 exécute en tant que serveur autonome, **${jboss.server.config.dir}** fait référence au répertoire ***EAP_HOME*/standalone/configuration/**. Si l'instance exécute en domaine géré, **${jboss.server.config.dir}** fait référence au répertoire ***EAP_HOME*/domain/configuration/**.

Modifier les noms de domaine de sécurité

Dans JBoss EAP 6, les domaines de sécurité n'utilisent plus le préfixe **java:/jaas/** dans leurs noms.

- Pour les applications web, vous devez supprimer ce préfixe des configurations du domaine de sécurité dans le fichier **jboss-web.xml**.
- Dans les applications Enterprise, vous devez supprimer ce préfixe des configurations du domaine de sécurité dans le fichier **jboss-ejb3.xml**. Ce fichier remplace le fichier **jboss.xml** de JBoss EAP 6.

Rapporter un bogue

ANNEXE A. RÉFÉRENCE

A.1. MODULES D'AUTHENTIFICATION INCLUS

Les modules d'authentification suivants sont inclus dans JBoss EAP 6. Certains d'entre eux gèrent l'autorisation ainsi que l'authentification. Ceux-ci incluent généralement le mot **`Role`** dans le nom de **`Code`**

Quand vous configurez ces modules, utiliser la valeur **`Code`** ou le nom complet (package) pour vous référer au module.

Modules d'authentification

Tableau A.1. `RealmDirect`

Code	**`RealmDirect`**
Class	**`org.jboss.as.security.RealmDirectLoginModule`**
Description	Une implémentation du module de connexion en interface directe avec le domaine de la sécurité. Ce module de connexion permet à toutes les interactions avec le backing store d'être déléguées au domaine, supprimant ainsi la nécessité de définitions en duplicata ou synchronisées. Utilisé pour les appels d'accès distant et l'interface de gestion.

Tableau A.2. Options du module `RealmDirect`

Option	Type	Par défaut	Description
`realm`	chaîne	**`ApplicationRealm`**	Nom de domaine souhaité.

Tableau A.3. `Client`

Code	**`Client`**
Class	**`org.jboss.security.ClientLoginModule`**
Description	Ce module de connexion est conçu pour établir l'identité de l'appelant et les informations d'identification lorsque JBoss EAP agit en tant que client. Il ne doit jamais servir sous forme de domaine de sécurité pour l'authentification serveur.

Tableau A.4. Options de module `Client`

Option	Type	Par défaut	Description
`multi-threaded`	**`true`** ou **`false`**	**`false`**	Définir la valeur sur true si chaque thread possède son propre stockage d'informations d'identification et principal. La valeur false pour indiquer que tous les threads de la machine virtuelle partagent la même identité et informations d'identification.
`password-stacking`	**`useFirstPass`** ou **`false`**	**`false`**	Définir la valeur sur **`useFirstPass`** pour indiquer que ce module de connexion doive rechercher les informations stockées dans le **`LoginContext`** à utiliser comme identité. Cette option peut être utilisée lorsque vous empilez les autres modules de connexion avec celui-ci.

Option	Type	Par défaut	Description
`restore-login-identity`	**`true`** ou **`false`**	**`false`**	Définir sur true si l'identité et les informations d'identification du début le la méthode **`login()`** doivent être restaurées une fois que la méthode **`logout()`** est invoquée.

Tableau A.5. `Remoting`

Code	**`Remoting`**
Class	**`org.jboss.as.security.remoting.RemotingLoginModule`**
Description	Le module de connexion est utilisé pour vérifier si la demande actuellement authentifiée est une demande reçue via une connexion d'accès distant et, si c'est le cas, l'identité qui aura été créée pendant le processus d'authentification à accès distant sera utilisée et associée à la demande en cours. Si la demande n'est pas parvenue via une connexion d'accès distant, ce module ne fera rien et permettra à la connexion JAAS de continuer vers le module suivant.

Tableau A.6. Options de Modules à accès distant

Option	Type	Par défaut	Description
`password-stacking`	**`useFirstPass`** ou **`false`**	**`false`**	La valeur de **`useFirstPass`** indique si ce module de connexion doit tout d'abord rechercher les informations stockées dans le **`LoginContext`** à utiliser comme identité. Cette option peut être utilisée lorsque vous empilez les autres modules de connexion avec celui-ci.
`principalClass`	Un nom de classe complet.	aucun	Une classe d'implémentation de **`Principal`** contenant un constructeur qui prend les arguments du String comme nom de principal.

Option	Type	Par défaut	Description
`unauthenticatedIdentity`	Un nom de principal.	aucun	Définit le nom principal devant être affecté aux demandes qui ne contiennent aucune information d'authentification. Cela peut permettre à des servlets non protégés d'appeler des méthodes sur EJB ne nécessitant pas un rôle spécifique. Un tel principal ne possédera aucun rôle associé et ne pourra accéder qu'aux méthodes EJB ou EJB non sécurisées associées à la contrainte de **`unchecked permission`**.

Tableau A.7. `Certificate`

Code	**`Certificate`**
Class	**`org.jboss.security.auth.spi.BaseCertLoginModule`**
Description	Ce module de connexion est conçu pour authentifier les utilisateurs basés sur des **`X509 Certificates`**. Un cas d'utilisation pour ceci est l'authentification **`CLIENT-CERT`** d'une application web.

Tableau A.8. Options de module `Certificate`

Option	Type	Par défaut	Description
`securityDomain`	chaîne	aucun	Nom du domaine de sécurité qui possède la configuration JSSE du truststore qui contient les certificats approuvés.
`verifier`	class	aucun	Le nom de classe du **`org.jboss.security.auth.certs.X509CertificateVerifier`** à utiliser pour vérifier le certificat de connexion.

Tableau A.9. `CertificateRoles`

Code	**`CertificateRoles`**
Class	**`org.jboss.security.auth.spi.CertRolesLoginModule`**
Description	Ce module de connexion étend le module de connexion de certificat pour ajouter des fonctions de mappage de rôle à partir d'un fichier de propriétés. Il prend les mêmes options que le module de connexion du certificat et ajoute les options suivantes.

Tableau A.10. Options de module `CertificateRoles`

Option	Type	Par défaut	Description
`rolesProperties`	chaîne	**`roles.properties`**	Le nom de la ressource ou du fichier contenant les rôles à attribuer à chaque utilisateur. Le fichier de propriétés de rôle doit être dans sous format **`username=role1,role2`**, où le nom d'utilisateur est le nom unique du certificat, sans signe égal **=** ou espace blanc. L'exemple suivant est dans le bon format : `CN\=unit-tests-client,\ OU\=Red\ Hat\ Inc.,\ O\=Red\ Hat\ Inc.,\ ST\=North\ Carolina,\ C\=US`
`defaultRolesProperties`	chaîne	**`defaultRoles.properties`**	Nom de la ressource ou du fichier où se replier si le fichier **`rolesProperties`** est introuvable.
`roleGroupSeparator`	Un seul caractère.	**`.`** (un seul point)	Le caractère à utiliser comme séparateur de groupe rôle dans le fichier de **`rolesProperties`**

Tableau A.11. `Database`

Code	**`Database`**
Class	**`org.jboss.security.auth.spi.DatabaseServerLoginModule`**
Description	Un module de connexion basé-JDBC supportant le mappage de rôle et l'authentification. Basé sur deux tables logiques, avec les définitions suivantes. • **`Principals: PrincipalID (text), Password (text)`** • **`Roles: PrincipalID (text), Role (text), RoleGroup (text)`**

Tableau A.12. `Database` Module Options

Option	Type	Par défaut	Description
`digestCallback`	Un nom de classe complet	aucun	Le nom de classe de l'implémentation **`DigestCallback`** qui inclut le contenu pre/post digest comme les valeurs salt pour hacher le mot de passe donné. Utilisé uniquement si l'**`hashStoreAlgorithm`** est spécifié.
`dsJndiName`	Ressource JNDI	aucun	Le nom de la ressource JNDI qui store les informations d'authentification. Cette option est requise.
`hashAlgorithm`	String	Utiliser des mots de passe en texte brut	L'algorithme digest du message utilisé pour hâcher les mots de passe. Les algorithmes pris en charge dépendent du fournisseur de sécurité Java, mais **`MD5`**, **`SHA-1`**, et **`SHA-256`** sont pris en charge.
`hashCharset`	String	Le codage par défaut de la plateforme	Le nom du charset (jeu de caractères)/codification à utiliser pour convertir la chaîne de mot de passe en tableau d'octets. Inclut tous les noms pris en charge par Java.
`hashEncoding`	String	Base64	Indique le format de code de chaîne à utiliser.
`ignorePasswordCase`	booléen	false	Signe qui indique si la comparaison de mot de passe doit ignorer le cas.
`inputValidator`	Un nom de classe complet	aucun	L'instance de l'implémentation de InputValidator utilisée pour valider le nom d'utilisateur et le mot de passe fournis par le client.

Option	Type	Par défaut	Description
principalsQuery	énoncé SQL préparé	**select Password from Principals where PrincipalID=?**	La requête SQL préparée pour obtenir les informations sur le principal.
rolesQuery	énoncé SQL préparé	aucun	Enoncé SQL préparé en vue d'obtenir les informations concernant les rôles. Il doit être équivalent à **select Role, RoleGroup from Roles where PrincipalID=?**, avec Role comme nom de rôle et la valeur de la colonne RoleGroup doit toujours être **Roles** avec un **R** majuscule ou **CallerPrincipal**.
storeDigestCallback	Un nom de classe complet	aucun	Le nom de classe de l'implémentation **DigestCallback** qui inclut le contenu pre/post digest comme les valeurs salt pour hacher le mot de passe du store/attendu. Utilisé uniquement si **hashStorePassword** ou **hashUserPassword** est sur **true** et si **hashAlgorithm** est spécifié.
suspendResume	booléen	true	Indique si une transaction JTA existante doit être suspendue pendant les opérations de base de données.
throwValidatorError	booléen	false	Indique si les erreurs de validation soivent être exposées ou non aux clients
transactionManagerJndiName	Ressource JNDI	java:/TransactionManager	Le nom JNDI de la source du gestionnaire de transactions utilisé par le module de connexion.

Tableau A.13. `DatabaseCertificate`

Code	**DatabaseCertificate**
Class	**org.jboss.security.auth.spi.DatabaseCertLoginModule**
Description	Ce module de connexion étend le module de connexion de certificat pour ajouter des fonctions de mappage de rôle d'une table de bases de données. Il possède les mêmes options mais aussi ces options supplémentaires :

Tableau A.14. `DatabaseCertificate` Module Options

Option	Type	Par défaut	Description
`dsJndiName`	Ressource JNDI		Le nom de la ressource JNDI qui store les informations d'authentification. Cette option est requise.
`rolesQuery`	énoncé SQL préparé	**`select Role,RoleGroup from Roles where PrincipalID=?`**	Enoncé SQL préparé en vue d'exécuter pour mapper les rôles. Doit être équivalent à **`select Role, RoleGroup from Roles where PrincipalID=?`**, avec Role comme nom de rôle et la valeur de la colonne RoleGroup doit toujours être **`Roles`** avec un **`R`** majuscule ou **`CallerPrincipal`**.
`suspendResume`	**`true`** ou **`false`**	**`true`**	Indique si une transaction JTA existante doit être suspendue pendant les opérations de base de données.

Tableau A.15. `Identity`

Code	**`Identity`**
Class	**`org.jboss.security.auth.spi.IdentityLoginModule`**
Description	Associe le principal spécifié dans les options de module avec n'importe quel sujet authentifié dans le module. Le type de classe de principal utilisée est **`org.jboss.security.SimplePrincipal`**. Si aucune option de principal n'est spécifiée, on utilisera un principal ayant pour nom **`guest`**.

Tableau A.16. Options de module `Identity`

Option	Type	Par défaut	Description
`principal`	String	**`guest`**	Le nom à utiliser pour le principal.

Option	Type	Par défaut	Description
`roles`	liste de strings séparée par des virgules	aucun	Une liste de rôles séparés par des virgules qui seront assignés au sujet.

Tableau A.17. `Ldap`

Code	**`Ldap`**
Class	**`org.jboss.security.auth.spi.LdapLoginModule`**
Description	Authentifie sur un serveur LDAP, lorsque le nom d'utilisateur et le mot de passe sont stockés dans un serveur LDAP qui est accessible à l'aide d'un fournisseur LDAP JNDI. Bon nombre des options ne sont pas requises, car elles sont déterminées par le fournisseur LDAP ou l'environnement.

Tableau A.18. Options de module `Ldap`

Option	Type	Par défaut	Description
`java.naming.factory.initial`	class name	**`com.sun.jndi.ldap.LdapCtxFactory`**	Nom de classe de l'implémentation **`InitialContextFactory`**.
`java.naming.provider.url`	**`ldap://`** URL	aucun	URL pour le serveur LDAP
`java.naming.security.authentication`	**`none`**, **`simple`**, ou le nom d'un mécanisme SASL	**`simple`**	Le niveau de sécurité à utiliser pour la liaison avec le serveur LDAP.
`java.naming.security.protocol`	protocole de transport	Si non spécifié, déterminé par le fournisseur.	Le protocole de transport à utiliser pour l'accès sécurisé, comme SSL.
`java.naming.security.principal`	chaîne	aucun	Le nom du principal permettant d'authentifier l'appelant vers le service. Il est construit à partir des autres propriétés décrites ci-dessous.

Option	Type	Par défaut	Description
`java.naming.security.credentials`	Type d'informations d'identification	aucun	Le type d'information d'identification utilisée par le schéma d'authentification. Exemples : mot de passe haché, mot de passe de texte clair, clé ou certificat. Si cette propriété n'est pas spécifiée, le comportement est déterminé par le fournisseur de service.
`principalDNPrefix`	chaîne		Préfixe ajouté au nom d'utilisateur pour former le DN de l'utilisateur. Vous pouvez demander à l'utilisateur un nom d'utilisateur et créer le nom de domaine DN complet en utilisant **`principalDNPrefix`** et **`principalDNSuffix`**.
`principalDNSuffix`	chaîne		Suffixe ajouté au nom d'utilisateur pour former le DN de l'utilisateur. Vous pouvez demander à l'utilisateur un nom d'utilisateur et créer le nom de domaine DN complet en utilisant **`principalDNPrefix`** et **`principalDNSuffix`**.
`useObjectCredential`	**`true`** ou **`false`**	false	Si les informations d'identification doivent être obtenues sous forme d'objet opaque à l'aide du type de Callback **`org.jboss.security.auth.callback.ObjectCallback`** plutôt que comme mot de passe **`char[]`** utilisant un JAAS PasswordCallback. Cela permet de passer les informations d'identification **`char[]`** au serveur LDAP.

Option	Type	Par défaut	Description
rolesCtxDN	DN complet	aucun	Le DN complet pour le contexte à chercher pour les rôles d'utilisateur.
userRolesCtxDNAttributeName	attribut	aucun	L'attribut dans l'objet utilisateur qui contient le nom unique DN pour que le contexte recherche des rôles d'utilisateur. Cela diffère de **rolesCtxDN** car le contexte de recherche de rôles d'utilisateur peut être unique pour chaque utilisateur.
roleAttributeID	attribut	**roles**	Nom de l'attribut qui contient les rôles d'utilisateur.
roleAttributeIsDN	**true** ou **false**	**false**	Indique si le **roleAttributeID** contient le nom de domaine complet d'un objet de rôle. Si false, le nom de rôle est tiré de la valeur de l'attribut **roleNameAttributeId** du nom de contexte. Certains schémas de répertoire, tel que Active Directory, requièrent que cet attribut soit défini à **true**.
roleNameAttributeID	attribut	**name**	Nom de l'attribut du contexte de **roleCtxDN** qui contient le nom de rôle. Si la propriété **roleAttributeIsDN** est définie sur **true**, cette propriété sera utilisée pour rechercher l'attribut de nom de l'objet rôle.
uidAttributeID	attribut	**uid**	Nom de l'attribut qui contient le **UserRolesAttributeDN** correspondant à l'ID d'utilisateur. Utilisé pour localiser les rôles d'utilisateur.

Option	Type	Par défaut	Description
matchOnUserDN	**true** ou **false**	**false**	Si la recherche de rôles d'utilisateur doit correspondre au DN de l'utilisateur entièrement distinct ou au nom d'utilisateur uniquement. Si **true**, l'userDN complet sera utilisé comme valeur de correspondance. Si **false**, seul le nom d'utilisateur sera utilisé comme valeur de correspondance pour l'attribut **UserRolesAttributeDN**.
allownEmptyPasswords	**true** ou **false**	**false**	Indique si on doit autoriser les mots de passe vides. La plupart des serveurs LDAP traitent les mots de passe vides comme des tentatives de connexion anonymes. Pour rejeter les mots de passe vides, définir à **false**.

Tableau A.19. LdapExtended

Code	**LdapExtended**
Class	**org.jboss.security.auth.spi.LdapExtLoginModule**

Description	Une autre implémentation de module de connexion LDAP qui utilise les recherches pour localiser l'utilisateur de liaisons et rôles associés. La requête de rôles suit récursivement les noms de domaines DN pour naviguer dans une structure de rôles hiérarchique. Il utilise les mêmes options **`java.naming`** que le module Ldap, et utilise les options suivantes à la place des autres options du module Ldap. L'authentification a lieu en 2 étapes : 1. Une première liaison au serveur LDAP est faite par les options **`bindCredential`**. **`bindDN`** est l'utilisateur ayant la possibilité de rechercher les arborescences baseCtxDN et **`rolesCtxDN`** pour l'utilisateur et les rôles. Le DN pour authentifier l'utilisateur est interrogé en utilisant le filtre spécifié par l'attribut **`baseFilter`**. 2. Le DN de l'utilisateur qui en résulte est authentifié par la liaison au serveur LDAP en utilisant le DN de l'utilisateur comme environnement de **`InitialLdapContext`** **`Context.SECURITY_PRINCIPAL`**. La propriété **`Context.SECURITY_CREDENTIALS`** est définie avec le mot de passe String obtenu par le gestionnaire de rappel.

Tableau A.20. Options de module `LdapExtended`

Option	Type	Par défaut	Description
`baseCtxDN`	DN complet	aucun	Le DN fixe de contexte de niveau supérieur pour commencer la recherche utilisateur.
`bindCredential`	string, parfois crypté	aucun	Voir Section 16.1.14, « Option du module bindCredential » pour plus de détails.
`bindDN`	DN complet	aucun	Le DN utilisé pour la liaison au serveur LDAP pour les requêtes d'utilisateurs et de rôles. Ce nom unique a besoin d'autorisations de lecture et de recherche pour les valeurs **`baseCtxDN`** et **`rolesCtxDN`**.

Option	Type	Par défaut	Description
`baseFilter`	String de filtre LDAP	aucun	Un filtre de recherche permettant de localiser le contexte de l'utilisateur à authentifier. L'entrée username ou **`userDN`** obtenue à partir du rappel de module de connexion est substitué dans le filtre à chaque fois qu'une expression **`{0}`** est utilisée. Un exemple de filtre de recherche est **`(uid={0})`**.
`rolesCtxDN`	DN complet	aucun	Le DN fixe du contexte pour rechercher des rôles d'utilisateur. Ce n'est pas le DN où les rôles se trouvent, mais le DN où les objets contenant les rôles d'utilisateur se trouvent. Par exemple, dans un serveur Active Directory de Microsoft, c'est le DN où le compte d'utilisateur se trouve.
`roleFilter`	String de filtre LDAP	aucun	Un filtre de recherche permettant de localiser les rôles associés à l'utilisateur authentifié. L'entrée user ou **`userDN`** obtenue à partir du rappel de module de connexion est substitué dans le filtre à chaque fois qu'une expression **`{0}`** est utilisée. L'utilisateurDN **`userDN`** authentifié sera substitué dans le filtre à chaque fois qu'un **`{1}`** est utilisé. Un exemple de filtre de recherche qui correspond au nom d'utilisateur d'entrée serait **`(member={0})`**. Une alternative correspondant au **`userDN`** authentifié serait **`(member={1})`**.

Option	Type	Par défaut	Description
roleAttributeIsDN	**true** ou **false**	**false**	Indique si le **roleAttributeID** contient le nom de domaine complet d'un objet de rôle. Si false, le nom de rôle est tiré de la valeur de l'attribut **roleNameAttributeId** du nom de contexte. Certains schémas de répertoire, tel que Active Directory, requièrent que cet attribut soit défini à **true**.
defaultRole	Un nom de rôle.	aucun	Un rôle inclus pour tous les utilisateurs authentifiés
parseRoleNameFromDN	**true** ou **false**	**false**	Un indicateur qui signale si le DN retourné par une requête contient le roleNameAttributeID. Si la valeur est **true**, le DN contient le roleNameATtributeID. Si la valeur est **false**, le DN ne contient pas le roleNameAttributeID. Cet indicateur peut améliorer les performances des requêtes LDAP.
parseUsername	**true** ou **false**	**false**	Un indicateur qui signale si le DN doit être vérifié niveau nom d'utilisateur. Si la valeur est **true**, le DN est vérifié niveau nom d'utilisateur. Si la valeur est **false**, le DN n'est pas vérifié au niveau nom d'utilisateur. Cet option peut à la fois être utilisée pour usernameBeginString et usernameEndString.
usernameBeginString	chaîne	aucun	Définit le string qui doit être supprimé au début du DN pour révéler le nom d'utilisateur. Cette option est utilisée avec **usernameEndString**.

Option	Type	Par défaut	Description
usernameEndString	chaîne	aucun	Définit le string qui doit être supprimé à la fin du DN pour révéler le nom d'utilisateur. Cette option est utilisée avec **userBeginString**.
roleNameAttributeID	attribut	**name**	Nom de l'attribut du contexte de **roleCtxDN** qui contient le nom de rôle. Si la propriété **roleAttributeIsDN** est définie sur **true**, cette propriété sera utilisée pour rechercher l'attribut de nom de l'objet rôle.
distinguishedNameAttribute	attribut	**distinguishedName**	Le nom de l'attribut dans l'entrée de l'utilisateur qui contient le nom unique de l'utilisateur. Ceci peut être nécessaire si le DN de l'utilisateur lui-même contient des caractères spéciaux (barre oblique inverse par exemple) qui empêchent le mappage de l'utilisateur. Si l'attribut n'existe pas, le DN de l'entrée sera utilisé.
roleRecursion	entier relatif	**0**	Le nombre de niveaux de récursivité de la recherche de rôle dans un contexte de correspondance donné. Désactiver la récursivité en attribuant le paramètre **0**.
searchTimeLimit	entier relatif	**10000** (10 seconds)	Timeout en millisecondes pour les recherches utilisateur/rôle.
searchScope	Un parmi : **OBJECT_SCOPE, ONELEVEL_SCOPE, SUBTREE_SCOPE**	**SUBTREE_SCOPE**	Étendue à utiliser.

Option	Type	Par défaut	Description
allownEmptyPasswords	**true** ou **false**	**false**	Indique si on doit autoriser les mots de passe vides. La plupart des serveurs LDAP traitent les mots de passe vides comme des tentatives de connexion anonymes. Pour rejeter les mots de passe vides, définir à false.
referralUserAttributeIDToCheck	attribut	aucun	Si vous n'utilisez pas de renvois, cette option peut être ignorée. Lorsque vous utilisez des références, cette option indique le nom d'attribut qui contient les utilisateurs définis pour un certain rôle (par exemple, **member**), si l'objet rôle est à l'intérieur du référentiel. Les utilisateurs sont vérifiés par le contenu de cet attribut de nom. Si cette option n'est pas définie, le contrôle échouera à chaque fois, alors les objets role ne peuvent pas être stockés dans un arbre de référence.

Tableau A.21. RoleMapping

Code	**RoleMapping**
Class	**org.jboss.security.auth.spi.RoleMappingLoginModule**
Description	Mappe un rôle qui est le résultat final du processus d'authentification de manière déclarative. Ce module doit être marqué comme étant **optionnel** quand vous y ajoutez le domaine de sécurité.

Tableau A.22. Options de module RoleMapping

Option	Type	Par défaut	Description

Option	Type	Par défaut	Description
rolesProperties	Le chemin d'accès complet et le nom d'une ressource ou d'un fichier de propriétés	**none**	Nom de la ressource ou du fichier de propriétés qui mappe les rôles aux rôles de remplacement. Le format est **original_role=role1,role2,role3**
replaceRole	**true** ou **false**	**false**	Indique si on doit ajouter les rôles en cours, ou les remplacer par les rôles mappés. Sont remplacés si définis sur **true**.

NOTE

L'option de module **rolesProperties** est requise pour le RoleMapping.

Tableau A.23. RunAs

Code	**RunAs**
Class	**org.jboss.security.auth.spi.RunAsLoginModule**
Description	Un module d'assistance qui pousse un rôle **run as** dans la pile pour la durée de la phase d'authentification de la connexion, et qui extrait le rôle **run as** de la pile soit dans la phase commit ou abort. Ce module de connexion fournit un rôle pour les autres modules de connexion qui doivent accéder aux ressources sécurisées afin d'effectuer leur authentification, par exemple un module de connexion qui accède à un EJB sécurisé. **RunAsLoginModule** doit être configuré avant que les modules de connexion qui ont besoin d'un rôle **run as** soient mis en place.

Tableau A.24. Options RunAs

Option	Type	Par défaut	Description
roleName	nom de rôle	**nobody**	Le nom de rôle à utiliser comme rôle **run as** pendant la phase de connexion.

Option	Type	Par défaut	Description
principalName	nom de principal	**nobody**	Le nom de principal à utiliser comme principal **run as** pendant la phase de connexion. S'il n'est pas spécifié, on utilisera la valeur par défaut **nobody**.
principalClass	Un nom de classe complet.	aucun	Une classe d'implémentation de **Principal** contenant un constructeur qui prend les arguments du String comme nom de principal.

Tableau A.25. Simple

Code	**Simple**
Class	**org.jboss.security.auth.spi.SimpleServerLoginModule**
Description	Module d'installation rapide de sécurité pour les tests. Implémente ce simple algorithme : • Si le mot de passe est null, authentifie l'utilisateur et assigne une identité **guest** et un rôle **guest**. • Sinon, quand le mot de passe est égal à l'utilisateur, assigne une identité égale au nom d'utilisateur et aux deux rôles **admin** et **guest**. • Sinon, l'authentification échoue.

Simple Module Options

Le module **Simple** n'a pas d'options.

Tableau A.26. ConfiguredIdentity

Code	**ConfiguredIdentity**
Class	**org.picketbox.datasource.security.ConfiguredIdentityLoginModule**
Description	Associe le principal spécifié dans les options du module avec n'importe quel sujet authentifié dans le module. Le type de classe du Principal classe est **org.jboss.security.SimplePrincipal**.

Tableau A.27. `ConfiguredIdentity` Module Options

Option	Type	Par défaut	Description
`username`	chaîne	aucun	Le nom d'utilisateur pour l'authentification.
`password`	string encodé	""	Le mot de passe à utiliser pour l'authentification. Pour crypter le mot de passe, utilisez le module directement dans la ligne de commande. `java org.picketbox.datasource.security.SecureIdentityLoginModule` *`password_to_encrypt`* Coller le résultat de cette commande dans le champ d'option de module. La valeur par défaut correspond à une chaîne vide.
`principal`	Nom d'un principal	**`none`**	Le principal qui sera associé avec n'importe quel sujet authentifié dans le module.

Tableau A.28. `SecureIdentity`

Code	**`SecureIdentity`**
Class	**`org.picketbox.datasource.security.SecureIdentityLoginModule`**
Description	Ce module est fourni à des fins d'héritage. Il permet de crypter un mot de passe et ensuite d'utiliser le mot de passe crypté avec un principal statique. Si votre application utilise **`SecureIdentity`**, envisager plutôt d'utiliser un mécanisme d'archivage sécurisé de mot de passe.

Tableau A.29. Options de module `SecureIdentity`

Option	Type	Par défaut	Description
username	chaîne	aucun	Le nom d'utilisateur pour l'authentification.
password	string encodé	""	Le mot de passe à utiliser pour l'authentification. Pour crypter le mot de passe, utilisez le module directement dans la ligne de commande. `java org.picketbox.datasource.security.SecureIdentityLoginModule` *`password_to_encrypt`* Coller le résultat de cette commande dans le champ d'option de module. La valeur par défaut correspond à une chaîne vide.
managedConnectionFactoryName	Ressource JCA	aucun	Le nom de la fabrique de connexions JCA à utiliser pour votre source de données.

Tableau A.30. `PropertiesUsers`

Code	**PropertiesUsers**
Class	**org.jboss.security.auth.spi.PropertiesUsersLoginModule**
Description	Utilise un fichier de propriétés pour stocker les noms d'utilisateur et mots de passe pour l'authentification. Aucune autorisation (correspondance de rôle) n'est fournie. Ce module convient seulement pour les tests.

Tableau A.31. `SimpleUsers`

Code	**SimpleUsers**
Class	**org.jboss.security.auth.spi.SimpleUsersLoginModule**

Description	Ce module de connexion stocke le nom d'utilisateur et le mot de passe en texte clair avec ***module-option***. le ***name*** de ***module-option*** et les attributs de ***value*** se réfèrent à un nom d'utlisateur et à un mot de passe. Il est inclus pour les essais uniquement et n'est pas approprié pour un environnement de production.

Tableau A.32. `LdapUsers`

Code	**`LdapUsers`**
Class	**`org.jboss.security.auth.spi.LdapUsersLoginModule`**
Description	Le module **`LdapUsers`** est remplacé par les modules **`ExtendedLDAP`** et **`AdvancedLdap`**.

Tableau A.33. `Kerberos`

Code	**`Kerberos`**
Class	**`com.sun.security.auth.module.Krb5LoginModule`**
Description	Effectue l'authentification de connexion Kerberos avec GSSAPI. Ce module fait partie de la structure de sécurité de l'API fourni par Sun Microsystems. Vous trouverez des informations à ce sujet dans http://docs.oracle.com/javase/7/docs/jre/api/security/jaas/spec/com/sun/security/auth/module/Krb5LoginModule.html Ce module devra être mis en correspondance avec un autre module qui gère les mappages d'authentification et de rôles.

Tableau A.34. Options de module `Kerberos`

Option	Type	Par défaut	Description
`storekey`	**`true`** ou **`false`**	false	Indique si on doit ajouter **`KerberosKey`** dans les informations d'authentification privées du sujet.
`doNotPrompt`	**`true`** ou **`false`**	false	Si défini sur **`true`**, l'utilisateur n'aura pas besoin de mot de passe.

Option	Type	Par défaut	Description
useTicketCache	Valeur booléenne de **true** ou **false** .	false	Si défini à **true**, le TGT sera obtenu à partir du cache du ticket. Si défini sur **false**, le cache du ticket ne sera pas utilisé.
ticketcache	Un fichier ou une ressource qui représente un cache de ticket Kerberos.	La valeur par défaut dépend du système d'exploitation que vous utilisez. • Red Hat Enterprise Linux / Solaris: **/tmp/krb5cc_*uid***, utilisant la valeur UID du système d'exploitation. • Microsoft Windows Server: utilise l'API LSA (Local Security Authority) pour trouver le ticketcache.	Emplacement du ticketcache.
useKeyTab	**true** ou **false**	false	Indique si on doit obtenir la clé du principal à partir d'un keytab.
keytab	Un fichier ou une ressource représentant un onglet de clé Kerberos.	L'emplacement du fichier de configuration Kerberos du système d'exploitation, ou **/home/*user*/krb5.keytab**	Emplacement du fichier keytab.
principal	chaîne	aucun	Le nom du principal. Cela peut être un simple nom d'utilisateur ou un nom de service tel que **host/testserver.acme.com**. Utiliser cela au lieu d'obtenir le principal d'un fichier keytab, ou lorsque le fichier keytab contient plus d'un principal.

Option	Type	Par défaut	Description
useFirstPass	**true** ou **false**	false	Indique si on doit extraire le nom d'utilisateur et le mot de passe de l'état partagé du module à l'aide de **javax.security.auth.login.name** et de **javax.security.auth.login.password** comme clés. Si l'authentification échoue, il n'y aura pas de nouvelle tentative.
tryFirstPass	**true** ou **false**	false	Identique à **useFirstPass**, mais si l'authentification échoue, le module utilise le **CallbackHandler** pour récupérer un nouveau nom d'utilisateur et mot de passe. Si la seconde authentification échoue, l'échec sera signalé à l'application appelante.
storePass	**true** ou **false**	false	Indique si on doit stocker le nom d'utilisateur et le mot de passe de l'état partagé du module. N'a pas lieu si les clés existent déjà dans l'état partagé, ou si l'authentification a échoué.
clearPass	**true** ou **false**	false	Définir à **true** pour supprimer le nom de l'utilisateur et le mot de passe de l'état partagé une fois que les deux phases d'authentification sont terminées.

Tableau A.35. SPNEGO

Code	**SPNEGO**
Class	**org.jboss.security.negotiation.spnego.SPNEGOLoginModule**

Description	Effectue l'authentification de connexion SPNEGO vers un serveur Microsoft Active Directory ou autre environnement qui supporte SPNEGO. SPNEGO peut également transporter les informations d'identification de Kerberos. Ce module a besoin de fonctionner en parallèle à un autre module qui gère l'authentification et le mappage des rôles.

Tableau A.36. Options de module `SPNEGO`

Option	Type	Par défaut	Description
`serverSecurityDomain`	**`string`**	**`null`**.	Définit le domaine utilisé pour extraire l'identité du service de serveur par le biais du module de connexion kerberos. Cette propriété doit être définie.
`removeRealmFromPrincipal`	**`boolean`**	**`false`**	Indique si le domaine Kerberos doit être retiré du principal avant de continuer.
`usernamePasswordDomain`	**`string`**	**`null`**	Indique un autre domaine de sécurité de la configuration qui doit être utilisée comme journalisation d'échec quand Kerberos échoue.

Tableau A.37. `AdvancedLdap`

Code	**`AdvancedLdap`**
Class	**`org.jboss.security.negotiation.AdvancedLdapLoginModule`**
Description	Module qui fournit davantage de fonctionnalité, comme SASL et l'utilisation d'un domaine de sécurité JAAS.

Tableau A.38. Options de module `AdvancedLdap`

Option	Type	Par défaut	Description
`bindAuthentication`	chaîne	aucun	Le type d'authentification SASL à utiliser pour se lier au serveur de répertoires.

Option	Type	Par défaut	Description
java.naming.provider.url	**string**	aucun	L'URI du serveur de répertoires.
baseCtxDN	DN complet	aucun	Le nom distinctif à utiliser comme base pour les recherches.
baseFilter	Chaîne représentant un filtre de recherche LDAP	aucun	Le filtre à utiliser pour réduire les résultats des recherches.
roleAttributeID	Valeur de chaîne représentant un attribut LDAP	aucun	L'attribut LDAP qui contient les noms des rôles d'autorisation.
roleAttributeIsDN	**true** ou **false**	**false**	Indique si l'attribut de rôle est un Nom Distinctif (DN).
roleNameAttributeID	Chaîne représentant un attribut LDAP	aucun	L'attribut contenu dans **RoleAttributeId** qui contient lui-même l'attribut de rôle.
recurseRoles	**true** ou **false**	**false**	Indique si on doit chercher récursivement des rôles dans **RoleAttributeId**.
referralUserAttributeIDToCheck	attribut	aucun	Si vous n'utilisez pas de renvois, cette option peut être ignorée. Lorsque vous utilisez des références, cette option indique le nom d'attribut qui contient les utilisateurs définis pour un certain rôle (par exemple, **member**), si l'objet rôle est à l'intérieur du référentiel. Les utilisateurs sont vérifiés par le contenu de cet attribut de nom. Si cette option n'est pas définie, le contrôle échouera à chaque fois, alors les objets role ne peuvent pas être stockés dans un arbre de référence.

Tableau A.39. AdvancedADLdap

Code	**AdvancedADLdap**
Class	**org.jboss.security.negotiation.AdvancedADLoginModule**
Description	Ce module étend le module de connexion **AdvancedLdap**, et ajoute des paramètres supplémentaires utiles au répertoire Microsoft Active Directory.

Tableau A.40. UsersRoles

Code	**UsersRoles**
Class	**org.jboss.security.auth.spi.UsersRolesLoginModul**
Description	Un module de connexion supportant des utilisateurs multiples et des rôles utilisateur stockés dans deux fichiers de propriétés différents.

Tableau A.41. Options de module UsersRoles

Option	Type	Par défaut	Description
usersProperties	Chemin d'accès à un fichier ou à une ressource.	**users.properties**	Fichier ou ressource qui contient les mappages d'utilisateur aux mots de passe. Le format du fichier est ***username=password***
rolesProperties	Chemin d'accès à un fichier ou à une ressource.	**roles.properties**	Fichier ou ressource qui contient les mappages d'utilisateur aux rôles. Le format du fichier est **username=role1,role2,role3**
password-stacking	**useFirstPass** ou **false**	**false**	La valeur de **useFirstPass** indique si ce module de connexion doit tout d'abord rechercher les informations stockées dans le **LoginContext** à utiliser comme identité. Cette option peut être utilisée lorsque vous empilez les autres modules de connexion avec celui-ci.

Option	Type	Par défaut	Description
hashAlgorithm	Chaîne représentant un algorithme de hachage de mot de passe.	**none**	Le nom de l'algorithme **java.security.MessageDigest** à utiliser pour hacher le mot de passe. Il n'y a pas de valeur par défaut pour cette option, donc vous devez la définir explicitement pour permettre le hachage. Quand **hashAlgorithm** est spécifié, le mot de passe en texte clair obtenu de **CallbackHandler** sera haché avant d'être passé à **UsernamePasswordLoginModule.validatePassword** comme argument **inputPassword**. Le mot de passe stocké dans le fichier **users.properties** doit être haché de façon similaire.
hashEncoding	**base64** ou **hex**	**base64**	Le format du string du mot de passe haché, si hashAlgorithm est défini également.
hashCharset	chaîne	Le codage par défaut défini dans l'environnement runtime du conteneur.	Le codage utilisé pour convertir le mot de passe de texte en clair en un tableau d'octets.
unauthenticatedIdentity	nom de principal	aucun	Définit le nom principal affecté aux demandes qui ne contiennent aucune information d'authentification. Cela peut permettre à des servlets non protégés d'appeler des méthodes sur EJB ne nécessitant pas un rôle spécifique. Un tel principal ne possédera aucun rôle associé et ne pourra accéder qu'aux méthodes EJB ou EJB non sécurisées associées à la contrainte de **permission non contrôlée**.

Modules d'authentification personnalisés

Les modules d'authentification sont des implémentations de `javax.security.auth.spi.LoginModule`. Reportez-vous à la documentation de l'API pour plus d'informations sur la création d'un module d'authentification personnalisé.

Rapporter un bogue

A.2. MODULES D'AUTORISATION INCLUS

Les modules suivants procurent des services d'autorisation.

Code	Class
DenyAll	org.jboss.security.authorization.modules.AllDenyAuthorizationModule
PermitAll	org.jboss.security.authorization.modules.AllPermitAuthorizationModule
Delegating	org.jboss.security.authorization.modules.DelegatingAuthorizationModule
Web	org.jboss.security.authorization.modules.web.WebAuthorizationModule
JACC	org.jboss.security.authorization.modules.JACCAuthorizationModule
XACML	org.jboss.security.authorization.modules.XACMLAuthorizationModule

AllDenyAuthorizationModule

Module d'autorisation simple qui refuse toujours une demande d'autorisation. Aucune option de configuration disponible.

AllPermitAuthorizationModule

Module d'autorisation simple qui accepte toujours une demande d'autorisation. Aucune option de configuration disponible.

DelegatingAuthorizationModule

Module d'autorisation par défaut qui délègue le processus de décision aux délégués configurés.

WebAuthorizationModule

Module d'autorisation web par défaut appartemnant à la logique d'autorisation tomcat par défaut (permit all/tout permis).

JACCAuthorizationModule

Ce module applique les sémantiques JACC en utilisant deux délégués (WebJACCPolicyModuleDelegate pour les requêtes d'autorisation de conteneur web et EJBJACCPolicyModuleDelegate pour les requêtes de conteneurs EJB). Aucune option de configuration n'est disponible.

XACMLAuthorizationModule

Ce module applique l'autorisation XACML grâce à deux délégués pour le les conteneurs EJB et web (WebXACMLPolicyModuleDelegate et EJBXACMLPolicyModuleDelegate). Ce module crée un objet PDP issu des politiques enregistrées et évalue la requête web ou EJB en fonction.

AbstractAuthorizationModule

Module d'autorisation de base remplacé et fournissant une fonction de délégation à d'autres modules d'autorisation.

Rapporter un bogue

A.3. MODULES DE SÉCURITÉ INCLUS

Les rôles de mappage de sécurité suivants sont fournis dans JBoss EAP 6.

Code	Class
PropertiesRoles	**`org.jboss.security.mapping.providers.role.PropertiesRolesMappingProvider`**
SimpleRoles	**`org.jboss.security.mapping.providers.role.SimpleRolesMappingProvider`**
DeploymentRoles	**`org.jboss.security.mapping.providers.DeploymentRolesMappingProvider`**
DatabaseRoles	**`org.jboss.security.mapping.providers.role.DatabaseRolesMappingProvider`**
LdapRoles	**`org.jboss.security.mapping.providers.role.LdapRolesMappingProvider`**
LdapAttributes	**`org.jboss.security.mapping.providers.attribute.LdapAttributeMappingProvider`**

DeploymentRolesMappingProvider

Un module de mappage de rôles qui prend en considération un principal pour des mappages de rôles qui peuvent être effectués dans les descripteurs de déploiement **`jboss-web.xml`** et **`jboss-app.xml`**.

Exemple A.1. Exemple

```
<jboss-web>
...
  <security-role>
      <role-name>Support</role-name>
```

```
        <principal-name>Mark</principal-name>
        <principal-name>Tom</principal-name>
    </security-role>
...
</jboss-web>
```

org.jboss.security.mapping.providers.DeploymentRoleToRolesMappingProvider

Un module de mappage de rôles à rôle qui prend en considération un principal pour des mappages de rôles qui peuvent être effectués dans les descripteurs de déploiement **`jboss-web.xml`** et **`jboss-app.xml`**. Dans ce cas, le nom du principal désigne un rôle qui mappe d'autres rôles.

Exemple A.2. Exemple

```
    <jboss-web>
...
      <security-role>
        <role-name>Employee</role-name>
        <principal-name>Support</principal-name>
        <principal-name>Sales</principal-name>
      </security-role>
...
    </jboss-web>
```

Ce qui signifie que chaque principal ayant pour rôle Support ou Sales aura également le rôle Employee assigné.

org.jboss.security.mapping.providers.OptionsRoleMappingProvider

Le fournisseur de mappage de rôles qui récupère les rôles à partir des options, puis qui les ajoute aux Groupes passés. Prend le mappage de style de propriétés du nom de rôle (clé) dans une liste de rôles (valeurs) séparés par des virgules.

org.jboss.security.mapping.providers.principal.SimplePrincipalMappingProvider

Un fournisseur de mappage de principal qui prend un SimplePrincipal et qui le convertit en SimplePrincipal avec un nom de principal différent.

DatabaseRolesMappingProvider

Un MappingProvider qui lit les rôles à partir d'une base de données.

Options :

- **`dsJndiName`** : nom JNDI de la source de données utilisée pour mapper les rôles vers l'utilisateur.
- **`rolesQuery`** : cette option devrait correspondre à un énoncé préparé équivalent à "select RoleName from Roles where User=?" ? est remplacé par le nom du principal en cours.
- **`suspendResume`**: Booléen - Suspend, puis termine la transaction associée au thread en cours tout en faisant une recherche de rôles.
- **`transactionManagerJndiName`**: nom JNDI du Gestionnaire de transactions (la valeur par défaut est java:/TransactionManager)

LdapRolesMappingProvider

Un fournisseur de mappage qui assigne les rôles à un utilisateur à l'aide d'un serveur LDAP pour chercher les rôles.

Options :

- **`bindDN`**: le DN utilisé pour faire la liaison avec le serveur LDAP pour les recherches de rôles et l'utilisateur. Ce DN a besoin de permissions lecture et recherche dans les valeurs de baseCtxDN et de rolesCtxDN.
- **`bindCredential`**: le mot de passe de bindDN. Peut être encodé si le jaasSecurityDomain est spécifié.
- **`rolesCtxDN`**: le DN fixe du contexte pour rechercher des rôles d'utilisateur. Ce n'est pas le DN où les rôles se trouvent, mais le DN où les objets contenant les rôles d'utilisateur se trouvent. Par exemple, dans un serveur Active Directory de Microsoft, c'est le DN où le compte d'utilisateur se trouve.
- **`roleAttributeID`**: l'attribut LDAP qui contient les noms des rôles d'autorisation.
- **`roleAttributeIsDN`** : indique si le **`roleAttributeID`** contient ou non le nom de domaine complet d'un objet de rôle. Si false, le nom de rôle est tiré de la valeur de l'attribut **`roleNameAttributeId`** du nom de contexte. Certains schémas de répertoire, tel que Active Directory, requièrent que cet attribut soit défini à **`true`**.
- **`roleNameAttributeID`** : nom de l'attribut qui se trouve dans le contexte **`roleCtxDN`** contenant le nom de rôle. Si la propriété **`roleAttributeIsDN`** est définie à **`true`**, cette propriété sera utilisée pour trouver l'attribut du nom de l'objet du rôle.
- **`parseRoleNameFromDN`**: un indicateur qui signale si le DN retourné par une requête contient le roleNameAttributeID. Si la valeur est définie à **`true`**, le DN est vérifié pour le roleNameATtributeID. Si la valeur est **`false`**, le DN n'est pas coché pour le roleNameAttributeID. Cet indicateur peut améliorer les performances des requêtes LDAP.
- **`roleFilter`** : un filtre de recherche permettant de localiser les rôles associés à l'utilisateur authentifié. L'entrée user ou **`userDN`** obtenue à partir du rappel de module de connexion est substitué dans le filtre à chaque fois qu'une expression **`{0}`** est utilisée. L'utilisateurDN **`userDN`** authentifié sera substitué dans le filtre à chaque fois qu'un **`{1}`** est utilisé. Un exemple de filtre de recherche qui correspond au nom d'utilisateur d'entrée serait **`(member={0})`**. Voici une alternative correspondant au **`userDN`** authentifié : **`(member={1})`**
- **`roleRecursion`** : le nombre de niveaux de récursivité de la recherche de rôle dans un contexte de correspondance donné. Désactiver la récursivité en attribuant le paramètre **`0`**.
- **`searchTimeLimit`**: la valeur d'expiration en millisecondes pour les recherches utilisateur/rôle. La valeur par défaut est 10000 (10 secondes).
- **`searchScope`**: l'étendue de recherche à utiliser.

PropertiesRolesMappingProvider

Un MappingProvider qui lit des rôles à partir d'un fichier de propriétés dans le format suivant : username=role1,role2,...

Options :

- **`rolesProperties`**: nom de fichier formaté de propriétés. L'expansion des variables de JBoss peuvent être utilisées sous la forme **`${jboss.variable}`**.

SimpleRolesMappingProvider

Simple MappingProvider qui lit des rôles à partir de la mappe d'options. Le nom d'attribut de l'option est le nom du principal à qui assigner des rôles et la valeur de l'attribut correspond aux noms de rôles séparés par des virgules à assigner au principal.

Exemple A.3. Exemple

```
<module-option name="JavaDuke" value="JBossAdmin,Admin"/>
<module-option name="joe" value="Users"/>
```

org.jboss.security.mapping.providers.attribute.DefaultAttributeMappingProvider

Vérifie le module et cherche le nom de principal dans le contexte de mappage pour créer une adresse email d'attribut à partir d'une option de module nommée principalName + ".email" et la donner au principal donné.

LdapAttributeMappingProvider

Attributs de mappage de LDAP vers le sujet. Les options incluent les options que votre fournisseur LDAP JNDI prend en charge.

Exemple A.4. Exemples de noms de propriété standard :

```
Context.INITIAL_CONTEXT_FACTORY = "java.naming.factory.initial"
Context.SECURITY_PROTOCOL = "java.naming.security.protocol"
Context.PROVIDER_URL = "java.naming.provider.url"
Context.SECURITY_AUTHENTICATION = "java.naming.security.authentication"
```

Options :

- **`bindDN`**: le DN utilisé pour faire la liaison avec le serveur LDAP pour les recherches de rôles et l'utilisateur. Ce DN a besoin de permissions lecture et recherche dans les valeurs de baseCtxDN et de rolesCtxDN.
- **`bindCredential`**: le mot de passe de bindDN. Peut être encodé si le jaasSecurityDomain est spécifié.
- **`baseCtxDN`** : le DN fixe de contexte pour commencer la recherche utilisateur
- **`baseFilter`**: un filtre de recherche permettant de localiser le contexte de l'utilisateur à authentifier. L'entrée username ou **`userDN`** obtenue à partir du rappel de module de connexion est substituée dans le filtre à chaque fois qu'une expression **`{0}`** est utilisée. Ce comportement de substitution vient de la méthode __**`DirContext.search(Name, String, Object[], SearchControls cons)`**__. **`(uid={0})`** est un exemple commun de filtre de recherche.
- **`searchTimeLimit`**: la valeur d'expiration en millisecondes pour les recherches utilisateur/rôle. La valeur par défaut est 10000 (10 secondes).

- **`attributeList`**: une liste d'attributs séparée par des virgules pour l'utilisateur. Exemple : mail,cn,sn,employeeType,employeeNumber.
- **`jaasSecurityDomain`** : Le JaasSecurityDomain à utiliser pour décrypter le **`java.naming.security.principal`**. La forme cryptée du mot de passe retourné par la méthode **`JaasSecurityDomain#encrypt64(byte[])`**. Vous pouvez également utiliser la classe **`org.jboss.security.plugins.PBEUtils`** pour générer la forme cryptée.

Rapporter un bogue

A.4. MODULES DE FOURNISSEURS D'AUDITING DE SÉCURITÉ INCLUS

JBoss EAP 6 contient un fournisseur d'auditing de sécurité.

Code	Class
LogAuditProvider	org.jboss.security.audit.providers.LogAuditProvider

Rapporter un bogue

A.5. RÉFÉRENCE DE CONFIGURATION JBOSS-WEB.XML

Introduction

Le fichier **`jboss-web.xml`** se trouve dans **`WEB-INF`**. Il contient des informations de configuration sur des fonctionnalités que le conteneur JBoss Web ajoute au Servlet 3.0. Les paramètres de configuration spécifiques à la spécification de Servlet 3.0 se trouvent dans **`web.xml`**, dans le même répertoire.

L'élément qui se trouve tout en haut dans le fichier **`jboss-web.xml`** est l'élément **`<jboss-web>`**.

Mappage des ressources globales aux prérequis WAR

La plupart des paramètres de configuration font correspondre les prérequis définis dans le fichier **`web.xml`** de l'application à des ressources locales. Pour plus d'explications sur les paramètres de configuration de **`web.xml`** consulter http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html.

Ainsi, si **`web.xml`** requiert **`jdbc/MyDataSource`**, alors **`jboss-web.xml`** fera sans doute correspondre la source de données globale **`java:/DefaultDS`** pour remplir ce besoin. Le WAR utilise la source de données globale pour faire correspondre **`jdbc/MyDataSource`**.

Tableau A.42. Attributs de haut niveau communs

Attribut	Description
env-entry	Un mappage à **`env-entry`** requis par **`web.xml`**.
ejb-ref	Un mappage à **`ejb-ref`** requis par **`web.xml`**.

Attribut	Description
ejb-local-ref	Un mappage à **`ejb-local-ref`** requis par **`web.xml`**.
service-ref	Un mappage à **`service-ref`** requis par **`web.xml`**.
resource-ref	Un mappage à **`resource-ref`** requis par **`web.xml`**.
resource-env-ref	Un mappage à **`resource-env-ref`** requis par **`web.xml`**.
message-destination-ref	Un mappage à **`message-destination-ref`** requis par **`web.xml`**.
persistence-context-ref	Un mappage à **`persistence-context-ref`** requis par **`web.xml`**.
persistence-unit-ref	Un mappage à **`persistence-unit-ref`** requis par **`web.xml`**.
post-construct	Un mappage à **`post-context`** requis par **`web.xml`**.
pre-destroy	Un mappage à **`pre-destroy`** requis par **`web.xml`**.
data-source	Un mappage à **`data-source`** requis par **`web.xml`**.
context-root	Le contexte root de l'application. La valeur par défaut est le nom du déploiement sans le suffixe **`.war`**.
virtual-host	Le nom de l'hôte virtuel HTTP à partir duquel l'application accepte les requêtes. Se réfère au contenu de l'en-tête de l'**`Host`** HTTP.
annotation	Décrit une annotation utilisée par l'application. Voir <annotation> pour plus d'information.
listener	Décrit un listener utilisée par l'application. Voir <listener> pour plus d'information.
session-config	Cet élément remplit la même fonction que l'élément **`<session-config>`** du fichier **`web.xml`** et est inclus dans un but de compatibilité seulement.
valve	Décrit une valve utilisée par l'application. Voir <valve> pour plus d'information.

Attribut	Description
overlay	Le nom d'une couche supplémentaire à ajouter à l'application.
security-domain	Le nom du domaine de sécurité utilisé par l'application. Le domaine de sécurité lui-même est configuré à partir de la console de gestion basée web ou l'interface CLI.
security-role	Cet élément remplit la même fonction que l'élément **<session-role>** du fichier **web.xml** et est inclus dans un but de compatibilité seulement.
use-jboss-authorization	Si cet élément est présent et contient la valeur non sensible à la casse "true", la pile d'autorisation web JBoss sera utilisée. S'il n'est pas présent ou contient une valeur non "true", alors seuls les mécanismes d'autorisation indiqués dans les spécifications Java Enterprise Edition seront utilisés. Cet élément est nouveau dans JBoss EAP 6.
disable-audit	Définir cet élément de booléen à **false** pour activer et à **true** pour désactiver l'auditing web. L'auditing de sécurité web ne fait pas partie de la spécification Java EE. Cet élément est nouveau dans JBoss EAP 6.
disable-cross-context	Si sur **false**, l'application sera en mesure d'appeler un autre contexte d'application. La valeur par défaut est **true**.

Les éléments suivants ont chacun des éléments enfants.

<annotation>

Décrit une annotation utilisée par l'application. Le tableau suivant liste les éléments enfants d'une **<annotation>**.

Tableau A.43. Éléments de configuration d'une annotation

Attribut	Description
class-name	Nom de la classe de l'annotation
servlet-security	L'élément, comme **@ServletSecurity**, qui représente la sécurité du servlet.
run-as	L'élément, comme **@RunAs**, qui représente l'information run-as

Attribut	Description
multipart-config	L'élément, comme `@MultiPart`, qui représente l'information multi-config.

<listener>

Décrit un listener. Le tableau suivant liste les éléments enfants d'un `<listener>`.

Tableau A.44. Éléments de configuration d'un listener

Attribut	Description
class-name	Nom de la classe du listener
listener-type	Liste les éléments de `condition` qui indiquent quelle sorte de listener ajouter au contexte de l'application. Les choix valides sont les suivants : **CONTAINER** Ajoute un ContainerListener au contexte. **LIFECYCLE** Ajoute un LifecycleListener au contexte. **SERVLET_INSTANCE** Ajoute un InstanceListener au contexte **SERVLET_CONTAINER** Ajoute un WrapperListener au contexte. **SERVLET_LIFECYCLE** Ajoute un WrapperLifecycle au contexte.
module	Le nom du module qui contient la classe du listener.
param	Un paramètre. Contient deux éléments enfants, `<param-name>` et `<param-value>`.

<valve>

Décrit une valve de l'application. Resssemble au <listener>, comprend des élément de param, nom de classe, et module.

Rapporter un bogue

A.6. RÉFÉRENCE DE PARAMÈTRE DE SÉCURITÉ EJB

Tableau A.45. Éléments de paramètres de sécurité EJB

Élément	Description
<security-identity>	Contient des éléments enfants relatifs à l'identité de sécurité d'un EJB.
<use-caller-identity />	Indique que l'EJB utilise la même identité de sécurité que l'appelant.
<run-as>	Contient un élément **<role-name>**.
<run-as-principal>	Si présent, indique le principal assigné aux appels sortants. Si non présent, les appels sortants sont assignés à un principal nommé **anonymous**.
<role-name>	Spécifie le role d'exécution de l'EJB.
<description>	Décrit le role nommé dans **<role-name>** .

Exemple A.5. Exemples d'identité de sécurité

Cet exemple décrit chaque balise décrite dans Tableau A.45, « Éléments de paramètres de sécurité EJB ». Peuvent également être mis dans un **<session>**.

```
<ejb-jar>
    <enterprise-beans>
        <session>
            <ejb-name>ASessionBean</ejb-name>
            <security-identity>
                <use-caller-identity/>
            </security-identity>
        </session>
        <session>
            <ejb-name>RunAsBean</ejb-name>
            <security-identity>
                <run-as>
                    <description>A private internal role</description>
                    <role-name>InternalRole</role-name>
                </run-as>
            </security-identity>
        </session>
    <session>
    <ejb-name>RunAsBean</ejb-name>
    <security-identity>
    <run-as-principal>internal</run-as-principal>
    </security-identity>
    </session>
    </enterprise-beans>
</ejb-jar>
```

Rapporter un bogue

ANNEXE B. HISTORIQUE DE RÉVISION

Version 6.3.0-42.1 **Mon Feb 16 2015** **Corina Roe**

Title Revision

Version 6.3.0-42 **Wednesday, July 30 2014** **Russell Dickenson**

Red Hat JBoss Enterprise Application Platform 6.3.0.GA