Red Hat Enterprise Linux 9

Sécurisation des réseaux

Configuration des réseaux sécurisés et de la communication réseau

Last Updated: 2024-02-15

Red Hat Enterprise Linux 9 Sécurisation des réseaux

Configuration des réseaux sécurisés et de la communication réseau

Notice légale

Résumé

Apprenez les outils et les techniques pour améliorer la sécurité de vos réseaux et réduire les risques de violations de données et d'intrusions.

Table des matières

RENDRE L'OPEN SOURCE PLUS INCLUSIF

Red Hat s'engage à remplacer les termes problématiques dans son code, sa documentation et ses propriétés Web. Nous commençons par ces quatre termes : master, slave, blacklist et whitelist. En raison de l'ampleur de cette entreprise, ces changements seront mis en œuvre progressivement au cours de plusieurs versions à venir. Pour plus de détails, voir le message de notre directeur technique Chris Wright.

FOURNIR UN RETOUR D'INFORMATION SUR LA DOCUMENTATION DE RED HAT

Nous apprécions vos commentaires sur notre documentation. Faites-nous savoir comment nous pouvons l'améliorer.

Soumettre des commentaires sur des passages spécifiques

1. Consultez la documentation au format **Multi-page HTML** et assurez-vous que le bouton **Feedback** apparaît dans le coin supérieur droit après le chargement complet de la page.
2. Utilisez votre curseur pour mettre en évidence la partie du texte que vous souhaitez commenter.
3. Cliquez sur le bouton **Add Feedback** qui apparaît près du texte en surbrillance.
4. Ajoutez vos commentaires et cliquez sur **Submit**.

Soumettre des commentaires via Bugzilla (compte requis)

1. Connectez-vous au site Web de Bugzilla.
2. Sélectionnez la version correcte dans le menu **Version**.
3. Saisissez un titre descriptif dans le champ **Summary**.
4. Saisissez votre suggestion d'amélioration dans le champ **Description**. Incluez des liens vers les parties pertinentes de la documentation.
5. Cliquez sur **Submit Bug**.

CHAPITRE 1. UTILISER DES COMMUNICATIONS SÉCURISÉES ENTRE DEUX SYSTÈMES AVEC OPENSSH

SSH (Secure Shell) est un protocole qui assure des communications sécurisées entre deux systèmes à l'aide d'une architecture client-serveur et permet aux utilisateurs de se connecter à distance aux systèmes hôtes des serveurs. Contrairement à d'autres protocoles de communication à distance, tels que FTP ou Telnet, SSH crypte la session de connexion, ce qui empêche les intrus de collecter des mots de passe non cryptés à partir de la connexion.

Red Hat Enterprise Linux inclut les paquetages de base **OpenSSH**: le paquetage général **openssh**, le paquetage **openssh-server** et le paquetage **openssh-clients**. Notez que les paquets **OpenSSH** nécessitent le paquetage **OpenSSL openssl-libs** , qui installe plusieurs bibliothèques cryptographiques importantes permettant à **OpenSSH** de fournir des communications cryptées.

1.1. SSH ET OPENSSH

SSH (Secure Shell) est un programme permettant de se connecter à une machine distante et d'y exécuter des commandes. Le protocole SSH fournit des communications cryptées et sécurisées entre deux hôtes non fiables sur un réseau non sécurisé. Vous pouvez également transférer des connexions X11 et des ports TCP/IP arbitraires sur le canal sécurisé.

Le protocole SSH atténue les menaces de sécurité, telles que l'interception des communications entre deux systèmes et l'usurpation d'identité d'un hôte particulier, lorsque vous l'utilisez pour l'ouverture d'une session shell à distance ou la copie de fichiers. En effet, le client et le serveur SSH utilisent des signatures numériques pour vérifier leur identité. En outre, toutes les communications entre les systèmes client et serveur sont cryptées.

Une clé d'hôte authentifie les hôtes dans le protocole SSH. Les clés d'hôte sont des clés cryptographiques générées automatiquement lors de la première installation d'OpenSSH ou lors du premier démarrage de l'hôte.

OpenSSH est une implémentation du protocole SSH supporté par Linux, UNIX et d'autres systèmes d'exploitation similaires. Il comprend les fichiers de base nécessaires au client et au serveur OpenSSH. La suite OpenSSH se compose des outils suivants dans l'espace utilisateur :

- **ssh** est un programme de connexion à distance (client SSH).
- **sshd** est un démon SSH OpenSSH.
- **scp** est un programme sécurisé de copie de fichiers à distance.
- **sftp** est un programme de transfert de fichiers sécurisé.
- **ssh-agent** est un agent d'authentification pour la mise en cache des clés privées.
- **ssh-add** ajoute des identités de clés privées à **ssh-agent**.
- **ssh-keygen** génère, gère et convertit les clés d'authentification pour **ssh**.
- **ssh-copy-id** est un script qui ajoute les clés publiques locales au fichier **authorized_keys** d'un serveur SSH distant.
- **ssh-keyscan** rassemble les clés publiques d'hôte SSH.

NOTE

Dans RHEL 9, le protocole de copie sécurisée (SCP) est remplacé par le protocole de transfert de fichiers SSH (SFTP) par défaut. Cela s'explique par le fait que SCP a déjà causé des problèmes de sécurité, par exemple CVE-2020-15778.

Si SFTP n'est pas disponible ou incompatible dans votre scénario, vous pouvez utiliser l'option **-O** pour forcer l'utilisation du protocole SCP/RCP d'origine.

Pour plus d'informations, consultez l'article sur la dépréciation du protocole SCP d'OpenSSH dans Red Hat Enterprise Linux 9 .

Il existe actuellement deux versions de SSH : la version 1 et la version 2, plus récente. La suite OpenSSH de RHEL ne prend en charge que la version 2 de SSH. Elle dispose d'un algorithme d'échange de clés amélioré qui n'est pas vulnérable aux exploits connus dans la version 1.

OpenSSH, l'un des principaux sous-systèmes cryptographiques de RHEL, utilise des politiques cryptographiques à l'échelle du système. Cela garantit que les suites de chiffrement et les algorithmes cryptographiques faibles sont désactivés dans la configuration par défaut. Pour modifier la politique, l'administrateur doit soit utiliser la commande **update-crypto-policies** pour ajuster les paramètres, soit se retirer manuellement des politiques de chiffrement à l'échelle du système.

La suite OpenSSH utilise deux ensembles de fichiers de configuration : un pour les programmes clients (c'est-à-dire **ssh**, **scp**, et **sftp**), et un autre pour le serveur (le démon **sshd**).

Les informations relatives à la configuration SSH de l'ensemble du système sont stockées dans le répertoire **/etc/ssh/**. Les informations de configuration SSH spécifiques à l'utilisateur sont stockées dans le répertoire **~/.ssh/**, dans le répertoire personnel de l'utilisateur. Pour une liste détaillée des fichiers de configuration OpenSSH, voir la section **FILES** dans la page de manuel **sshd(8)**.

Ressources supplémentaires

- Pages de manuel répertoriées à l'aide de la commande **man -k ssh**
- Utilisation de politiques cryptographiques à l'échelle du système

1.2. CONFIGURATION ET DÉMARRAGE D'UN SERVEUR OPENSSH

Utilisez la procédure suivante pour une configuration de base qui peut être nécessaire pour votre environnement et pour démarrer un serveur OpenSSH. Notez qu'après l'installation par défaut de RHEL, le démon **sshd** est déjà lancé et les clés d'hôte du serveur sont automatiquement créées.

Conditions préalables

- Le paquet **openssh-server** est installé.

Procédure

1. Démarrer le démon **sshd** dans la session en cours et le configurer pour qu'il démarre automatiquement au moment du démarrage :

```
# systemctl start sshd
# systemctl enable sshd
```

2. Pour spécifier des adresses différentes des adresses par défaut **0.0.0.0** (IPv4) ou **::** (IPv6) pour

la directive **ListenAddress** dans le fichier de configuration **/etc/ssh/sshd_config** et pour utiliser une configuration réseau dynamique plus lente, ajoutez la dépendance de l'unité cible **network-online.target** au fichier d'unité **sshd.service**. Pour ce faire, créez le fichier **/etc/systemd/system/sshd.service.d/local.conf** avec le contenu suivant :

```
[Unit]
Wants=network-online.target
After=network-online.target
```

3. Vérifiez que les paramètres du serveur OpenSSH dans le fichier de configuration **/etc/ssh/sshd_config** répondent aux exigences de votre scénario.

4. Vous pouvez également modifier le message de bienvenue que votre serveur OpenSSH affiche avant qu'un client ne s'authentifie en éditant le fichier **/etc/issue**, par exemple :

```
Welcome to ssh-server.example.com
Warning: By accessing this server, you agree to the referenced terms and conditions.
```

Assurez-vous que l'option **Banner** n'est pas commentée dans **/etc/ssh/sshd_config** et que sa valeur contient **/etc/issue**:

```
# less /etc/ssh/sshd_config | grep Banner
Banner /etc/issue
```

Notez que pour modifier le message affiché après une connexion réussie, vous devez éditer le fichier **/etc/motd** sur le serveur. Voir la page de manuel **pam_motd** pour plus d'informations.

5. Rechargez la configuration de **systemd** et redémarrez **sshd** pour appliquer les changements :

```
# systemctl daemon-reload
# systemctl restart sshd
```

Vérification

1. Vérifiez que le démon **sshd** est en cours d'exécution :

```
# systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2019-11-18 14:59:58 CET; 6min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
 Main PID: 1149 (sshd)
    Tasks: 1 (limit: 11491)
   Memory: 1.9M
   CGroup: /system.slice/sshd.service
           └─1149 /usr/sbin/sshd -D -oCiphers=aes128-ctr,aes256-ctr,aes128-cbc,aes256-cbc -
oMACs=hmac-sha2-256,>

Nov 18 14:59:58 ssh-server-example.com systemd[1]: Starting OpenSSH server daemon...
Nov 18 14:59:58 ssh-server-example.com sshd[1149]: Server listening on 0.0.0.0 port 22.
Nov 18 14:59:58 ssh-server-example.com sshd[1149]: Server listening on :: port 22.
Nov 18 14:59:58 ssh-server-example.com systemd[1]: Started OpenSSH server daemon.
```

2. Connectez-vous au serveur SSH avec un client SSH.

```
# ssh user@ssh-server-example.com
ECDSA key fingerprint is SHA256:dXbaS0RG/UzlTTku8GtXSz0S1++lPegSy31v3L/FAEc.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ssh-server-example.com' (ECDSA) to the list of known hosts.

user@ssh-server-example.com's password:
```

Ressources supplémentaires

- **sshd(8)** et **sshd_config(5)**.

1.3. CONFIGURATION D'UN SERVEUR OPENSSH POUR L'AUTHENTIFICATION PAR CLÉ

Pour améliorer la sécurité du système, appliquez l'authentification par clé en désactivant l'authentification par mot de passe sur votre serveur OpenSSH.

Conditions préalables

- Le paquet **openssh-server** est installé.
- Le démon **sshd** est en cours d'exécution sur le serveur.

Procédure

1. Ouvrez la configuration de **/etc/ssh/sshd_config** dans un éditeur de texte, par exemple :

```
# vi /etc/ssh/sshd_config
```

2. Remplacer l'option **PasswordAuthentication** par **no**:

```
PasswordAuthentication no
```

   Sur un système autre qu'une nouvelle installation par défaut, vérifiez que **PubkeyAuthentication no** n'a pas été défini et que la directive **KbdInteractiveAuthentication** est définie sur **no**. Si vous êtes connecté à distance, sans utiliser la console ou l'accès hors bande, testez le processus de connexion par clé avant de désactiver l'authentification par mot de passe.

3. Pour utiliser l'authentification par clé avec les répertoires personnels montés sur NFS, activez le booléen SELinux **use_nfs_home_dirs**:

```
# setsebool -P use_nfs_home_dirs 1
```

4. Rechargez le démon **sshd** pour appliquer les modifications :

```
# systemctl reload sshd
```

Ressources supplémentaires

- **sshd(8)**, **sshd_config(5)** et **setsebool(8)**.

1.4. GÉNÉRER DES PAIRES DE CLÉS SSH

Cette procédure permet de générer une paire de clés SSH sur un système local et de copier la clé publique générée sur un serveur OpenSSH. Si le serveur est configuré en conséquence, vous pouvez vous connecter au serveur OpenSSH sans fournir de mot de passe.

IMPORTANT

Si vous effectuez les étapes suivantes en tant que **root**, seul **root** pourra utiliser les clés.

Procédure

1. Pour générer une paire de clés ECDSA pour la version 2 du protocole SSH :

```
$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/joesec/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/joesec/.ssh/id_ecdsa.
Your public key has been saved in /home/joesec/.ssh/id_ecdsa.pub.
The key fingerprint is:
SHA256:Q/x+qms4j7PCQ0qFd09iZEFHA+SqwBKRNaU72oZfaCI
joesec@localhost.example.com
The key's randomart image is:
+---[ECDSA 256]---+
|.oo..o=++        |
|.. o .oo .       |
|. .. o. o        |
|....o.+...       |
|o.oo.o +S .      |
|.=.+.   .o       |
|E.*+.  .  . .    |
|.=..+ +..  o     |
|  .  oo*+o.      |
+----[SHA256]-----+
```

   Vous pouvez également générer une paire de clés RSA en utilisant l'option **-t rsa** avec la commande **ssh-keygen** ou une paire de clés Ed25519 en entrant la commande **ssh-keygen -t ed25519**.

2. Pour copier la clé publique sur une machine distante :

```
$ ssh-copy-id joesec@ssh-server-example.com
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
joesec@ssh-server-example.com's password:
...
Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'joesec@ssh-server-example.com'" and check to make sure that only the key(s) you wanted were added.
```

Si vous n'utilisez pas le programme **ssh-agent** dans votre session, la commande précédente copie la clé publique **~/.ssh/id*.pub** la plus récemment modifiée si elle n'est pas encore installée. Pour spécifier un autre fichier de clé publique ou pour donner la priorité aux clés contenues dans les fichiers par rapport aux clés mises en mémoire par **ssh-agent**, utilisez la commande **ssh-copy-id** avec l'option **-i**.

NOTE

Si vous réinstallez votre système et souhaitez conserver les paires de clés générées précédemment, sauvegardez le répertoire **~/.ssh/**. Après la réinstallation, copiez-le dans votre répertoire personnel. Vous pouvez faire cela pour tous les utilisateurs de votre système, y compris **root**.

Vérification

1. Connectez-vous au serveur OpenSSH sans fournir de mot de passe :

```
$ ssh joesec@ssh-server-example.com
Welcome message.
...
Last login: Mon Nov 18 18:28:42 2019 from ::1
```

Ressources supplémentaires

- **ssh-keygen(1)** et **ssh-copy-id(1)**.

1.5. UTILISATION DE CLÉS SSH STOCKÉES SUR UNE CARTE À PUCE

Red Hat Enterprise Linux vous permet d'utiliser les clés RSA et ECDSA stockées sur une carte à puce sur les clients OpenSSH. Utilisez cette procédure pour activer l'authentification à l'aide d'une carte à puce au lieu d'un mot de passe.

Conditions préalables

- Côté client, le paquet **opensc** est installé et le service **pcscd** est en cours d'exécution.

Procédure

1. Dresser la liste de toutes les clés fournies par le module PKCS #11 d'OpenSC, y compris leurs URI PKCS #11, et enregistrer le résultat dans le fichier *keys.pub*:

```
$ ssh-keygen -D pkcs11: > keys.pub
$ ssh-keygen -D pkcs11:
ssh-rsa AAAAB3NzaC1yc2E...KKZMzcQZzx
pkcs11:id=%02;object=SIGN%20pubkey;token=SSH%20key;manufacturer=piv_II?module-
path=/usr/lib64/pkcs11/opensc-pkcs11.so
ecdsa-sha2-nistp256 AAA...J0hkYnnsM=
pkcs11:id=%01;object=PIV%20AUTH%20pubkey;token=SSH%20key;manufacturer=piv_II?
module-path=/usr/lib64/pkcs11/opensc-pkcs11.so
```

2. Pour permettre l'authentification à l'aide d'une carte à puce sur un serveur distant (*example.com*), transférez la clé publique sur le serveur distant. Utilisez la commande **ssh-copy-id** avec *keys.pub* créé à l'étape précédente :

```
$ ssh-copy-id -f -i keys.pub username@example.com
```

3. Pour vous connecter à *example.com* à l'aide de la clé ECDSA issue de la commande **ssh-keygen -D** à l'étape 1, vous pouvez utiliser uniquement un sous-ensemble de l'URI, qui fait référence de manière unique à votre clé, par exemple :

```
$ ssh -i "pkcs11:id=%01?module-path=/usr/lib64/pkcs11/opensc-pkcs11.so" example.com
Enter PIN for 'SSH key':
[example.com] $
```

4. Vous pouvez utiliser la même chaîne URI dans le fichier **~/.ssh/config** pour rendre la configuration permanente :

```
$ cat ~/.ssh/config
IdentityFile "pkcs11:id=%01?module-path=/usr/lib64/pkcs11/opensc-pkcs11.so"
$ ssh example.com
Enter PIN for 'SSH key':
[example.com] $
```

Comme OpenSSH utilise le wrapper **p11-kit-proxy** et que le module OpenSC PKCS #11 est enregistré dans le kit PKCS#11, vous pouvez simplifier les commandes précédentes :

```
$ ssh -i "pkcs11:id=%01" example.com
Enter PIN for 'SSH key':
[example.com] $
```

Si vous omettez la partie **id=** d'un URI PKCS #11, OpenSSH charge toutes les clés disponibles dans le module proxy. Cela peut réduire la quantité de données à saisir :

```
$ ssh -i pkcs11: example.com
Enter PIN for 'SSH key':
[example.com] $
```

Ressources supplémentaires

- Fedora 28 : Meilleur support des cartes à puce dans OpenSSH
- **p11-kit(8)**, **opensc.conf(5)**, **pcscd(8)**, **ssh(1)**, et **ssh-keygen(1)** pages de manuel

1.6. RENDRE OPENSSH PLUS SÛR

Les conseils suivants vous aideront à renforcer la sécurité lors de l'utilisation d'OpenSSH. Notez que les modifications apportées au fichier de configuration d'OpenSSH **/etc/ssh/sshd_config** nécessitent le rechargement du démon **sshd** pour être prises en compte :

```
# systemctl reload sshd
```

IMPORTANT

La majorité des modifications apportées à la configuration du renforcement de la sécurité réduisent la compatibilité avec les clients qui ne prennent pas en charge les algorithmes ou les suites de chiffrement les plus récents.

Désactivation des protocoles de connexion non sécurisés

- Pour que SSH soit vraiment efficace, il faut empêcher l'utilisation de protocoles de connexion non sécurisés qui sont remplacés par la suite OpenSSH. Sinon, le mot de passe d'un utilisateur peut être protégé par SSH pendant une session et être capturé plus tard lors d'une connexion par Telnet. Pour cette raison, envisagez de désactiver les protocoles non sécurisés, tels que telnet, rsh, rlogin et ftp.

Activation de l'authentification par clé et désactivation de l'authentification par mot de passe

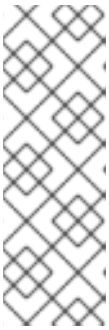
- Le fait de désactiver les mots de passe pour l'authentification et de n'autoriser que les paires de clés réduit la surface d'attaque et peut également faire gagner du temps aux utilisateurs. Sur les clients, générez des paires de clés à l'aide de l'outil **ssh-keygen** et utilisez l'utilitaire **ssh-copy-id** pour copier les clés publiques des clients sur le serveur OpenSSH. Pour désactiver l'authentification par mot de passe sur votre serveur OpenSSH, modifiez **/etc/ssh/sshd_config** et remplacez l'option **PasswordAuthentication** par **no**:

```
PasswordAuthentication no
```

Types de clés

- Bien que la commande **ssh-keygen** génère par défaut une paire de clés RSA, vous pouvez lui demander de générer des clés ECDSA ou Ed25519 en utilisant l'option **-t**. L'ECDSA (Elliptic Curve Digital Signature Algorithm) offre de meilleures performances que le RSA à puissance de clé symétrique équivalente. Il génère également des clés plus courtes. L'algorithme de clé publique Ed25519 est une implémentation des courbes d'Edwards torsadées qui est plus sûre et plus rapide que RSA, DSA et ECDSA.
  OpenSSH crée automatiquement les clés d'hôte RSA, ECDSA et Ed25519 du serveur si elles sont manquantes. Pour configurer la création de clés hôte dans RHEL, utilisez le service instancié **sshd-keygen@.service**. Par exemple, pour désactiver la création automatique du type de clé RSA :

```
# systemctl mask sshd-keygen@rsa.service
```



NOTE

Dans les images où **cloud-init** est activé, les unités **ssh-keygen** sont automatiquement désactivées. En effet, le service **ssh-keygen template** peut interférer avec l'outil **cloud-init** et causer des problèmes avec la génération des clés de l'hôte. Pour éviter ces problèmes, le fichier de configuration drop-in **etc/systemd/system/sshd-keygen@.service.d/disable-sshd-keygen-if-cloud-init-active.conf** désactive les unités **ssh-keygen** si **cloud-init** est en cours d'exécution.

- Pour exclure certains types de clés pour les connexions SSH, commentez les lignes correspondantes dans **/etc/ssh/sshd_config**, et rechargez le service **sshd**. Par exemple, pour n'autoriser que les clés d'hôte Ed25519 :

```
# HostKey /etc/ssh/ssh_host_rsa_key
# HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
```

Port autre que celui par défaut

- Par défaut, le démon **sshd** écoute sur le port TCP 22. La modification du port réduit l'exposition du système aux attaques basées sur l'analyse automatisée du réseau et augmente donc la sécurité par l'obscurité. Vous pouvez spécifier le port en utilisant la directive **Port** dans le fichier de configuration **/etc/ssh/sshd_config**.
  Vous devez également mettre à jour la politique SELinux par défaut afin d'autoriser l'utilisation d'un port autre que celui par défaut. Pour ce faire, utilisez l'outil **semanage** du paquetage **policycoreutils-python-utils**:

  ```
  # semanage port -a -t ssh_port_t -p tcp port_number
  ```

  En outre, mettez à jour la configuration de **firewalld**:

  ```
  # firewall-cmd --add-port port_number/tcp
  # firewall-cmd --runtime-to-permanent
  ```

  Dans les commandes précédentes, remplacez *port_number* par le nouveau numéro de port spécifié à l'aide de la directive **Port**.

Connexion racine

- **PermitRootLogin** est défini par défaut sur **prohibit-password**. Cela permet d'imposer l'utilisation d'une authentification par clé plutôt que par mot de passe pour se connecter en tant que root et de réduire les risques en empêchant les attaques par force brute.

  ATTENTION

  L'activation de la connexion en tant qu'utilisateur root n'est pas une pratique sûre, car l'administrateur ne peut pas vérifier quels utilisateurs exécutent quelles commandes privilégiées. Pour utiliser les commandes administratives, connectez-vous et utilisez plutôt **sudo**.

Utilisation de l'extension X Security

- Le serveur X des clients Red Hat Enterprise Linux ne fournit pas l'extension X Security. Par conséquent, les clients ne peuvent pas demander une autre couche de sécurité lorsqu'ils se connectent à des serveurs SSH non fiables avec le transfert X11. De toute façon, la plupart des applications ne peuvent pas fonctionner avec cette extension activée.
  Par défaut, l'option **ForwardX11Trusted** du fichier **/etc/ssh/ssh_config.d/05-redhat.conf** est définie sur **yes**, et il n'y a pas de différence entre les commandes **ssh -X remote_machine** (hôte non fiable) et **ssh -Y remote_machine** (hôte fiable).

  Si votre scénario ne nécessite pas du tout la fonction de transfert X11, définissez la directive **X11Forwarding** dans le fichier de configuration **/etc/ssh/sshd_config** à **no**.

Restreindre l'accès à des utilisateurs, groupes ou domaines spécifiques

- Les directives **AllowUsers** et **AllowGroups** du fichier de configuration **/etc/ssh/sshd_config** vous permettent d'autoriser uniquement certains utilisateurs, domaines ou groupes à se connecter à votre serveur OpenSSH. Vous pouvez combiner **AllowUsers** et **AllowGroups** pour restreindre l'accès de manière plus précise, par exemple :

  ```
  AllowUsers *@192.168.1.*,*@10.0.0.*,!*@192.168.1.2
  AllowGroups example-group
  ```

  Les lignes de configuration précédentes acceptent les connexions de tous les utilisateurs des

systèmes des sous-réseaux 192.168.1.* et 10.0.0.*, à l'exception du système portant l'adresse 192.168.1.2. Tous les utilisateurs doivent faire partie du groupe **example-group**. Le serveur OpenSSH refuse toutes les autres connexions.

Notez que l'utilisation de listes d'autorisations (directives commençant par Allow) est plus sûre que l'utilisation de listes de blocages (options commençant par Deny) car les listes d'autorisations bloquent également les nouveaux utilisateurs ou groupes non autorisés.

Modification des politiques cryptographiques à l'échelle du système

- OpenSSH utilise les stratégies cryptographiques du système RHEL, et le niveau de stratégie cryptographique par défaut du système offre des paramètres sûrs pour les modèles de menace actuels. Pour rendre vos paramètres cryptographiques plus stricts, modifiez le niveau de stratégie actuel :

```
# update-crypto-policies --set FUTURE
Setting system policy to FUTURE
```

- Pour ne pas appliquer les politiques de cryptage à l'échelle du système pour votre serveur OpenSSH, décommentez la ligne contenant la variable **CRYPTO_POLICY=** dans le fichier **/etc/sysconfig/sshd**. Après cette modification, les valeurs spécifiées dans les sections **Ciphers**, **MACs**, **KexAlgoritms**, et **GSSAPIKexAlgorithms** du fichier **/etc/ssh/sshd_config** ne sont pas remplacées. Notez que cette tâche nécessite des connaissances approfondies en matière de configuration des options cryptographiques.

- Pour plus d'informations, voir Utilisation de stratégies cryptographiques à l'échelle du système dans le titre Durcissement de la sécurité .

Ressources supplémentaires

- **sshd_config(5)**, **ssh-keygen(1)**, **crypto-policies(7)**, et **update-crypto-policies(8)**.

1.7. CONNEXION À UN SERVEUR DISTANT À L'AIDE D'UN HÔTE DE SAUT SSH

Utilisez cette procédure pour connecter votre système local à un serveur distant par l'intermédiaire d'un serveur intermédiaire, également appelé jump host.

Conditions préalables

- Un hôte de saut accepte les connexions SSH à partir de votre système local.

- Un serveur distant n'accepte les connexions SSH qu'à partir de l'hôte de saut.

Procédure

1. Définissez l'hôte de saut en modifiant le fichier **~/.ssh/config** sur votre système local, par exemple :

```
Host jump-server1
  HostName jump1.example.com
```

- Le paramètre **Host** définit un nom ou un alias pour l'hôte que vous pouvez utiliser dans les commandes **ssh**. La valeur peut correspondre au nom réel de l'hôte, mais peut également être une chaîne quelconque.
- Le paramètre **HostName** définit le nom d'hôte ou l'adresse IP de l'hôte de saut.

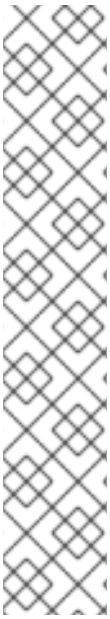
2. Ajoutez la configuration de saut du serveur distant avec la directive **ProxyJump** au fichier **~/.ssh/config** de votre système local, par exemple :

```
Host remote-server
  HostName remote1.example.com
  ProxyJump jump-server1
```

3. Utilisez votre système local pour vous connecter au serveur distant via le serveur de saut :

```
$ ssh remote-server
```

La commande précédente est équivalente à la commande **ssh -J jump-server1 remote-server** si vous omettez les étapes de configuration 1 et 2.



NOTE

Vous pouvez spécifier davantage de serveurs de saut et vous pouvez également éviter d'ajouter des définitions d'hôtes au fichier de configuration lorsque vous fournissez leurs noms d'hôtes complets, par exemple :

```
$ ssh -J jump1.example.com,jump2.example.com,jump3.example.com
remote1.example.com
```

Modifiez la notation du nom d'hôte uniquement dans la commande précédente si les noms d'utilisateur ou les ports SSH sur les serveurs de saut diffèrent des noms et des ports sur le serveur distant, par exemple :

```
$ ssh -J
johndoe@jump1.example.com:75,johndoe@jump2.example.com:75,johndoe@jump3.e
xample.com:75 joesec@remote1.example.com:220
```

Ressources supplémentaires

- **ssh_config(5)** et **ssh(1)**.

1.8. SE CONNECTER À DES MACHINES DISTANTES AVEC DES CLÉS SSH EN UTILISANT SSH-AGENT

Pour éviter de saisir une phrase de passe à chaque fois que vous établissez une connexion SSH, vous pouvez utiliser l'utilitaire **ssh-agent** pour mettre en cache la clé privée SSH. La clé privée et la phrase de passe restent sécurisées.

Conditions préalables

- Vous disposez d'un hôte distant avec un démon SSH en cours d'exécution et accessible via le réseau.

- Vous connaissez l'adresse IP ou le nom d'hôte et les informations d'identification pour vous connecter à l'hôte distant.
- Vous avez généré une paire de clés SSH avec une phrase de passe et transféré la clé publique à la machine distante.

Procédure

1. Facultatif : Vérifiez que vous pouvez utiliser la clé pour vous authentifier auprès de l'hôte distant :
   a. Connectez-vous à l'hôte distant à l'aide de SSH :

   ```
   $ ssh example.user1@198.51.100.1 hostname
   ```

   b. Saisissez la phrase de passe que vous avez définie lors de la création de la clé pour autoriser l'accès à la clé privée.

   ```
   $ ssh example.user1@198.51.100.1 hostname
    host.example.com
   ```

2. Démarrer le site **ssh-agent**.

   ```
   $ eval $(ssh-agent)
   Agent pid 20062
   ```

3. Ajouter la clé à **ssh-agent**.

   ```
   $ ssh-add ~/.ssh/id_rsa
   Enter passphrase for ~/.ssh/id_rsa:
   Identity added: ~/.ssh/id_rsa (example.user0@198.51.100.12)
   ```

Vérification

- Facultatif : Connectez-vous à l'ordinateur hôte à l'aide de SSH.

  ```
  $ ssh example.user1@198.51.100.1

  Last login: Mon Sep 14 12:56:37 2020
  ```

  Notez qu'il n'est pas nécessaire de saisir la phrase d'authentification.

1.9. RESSOURCES SUPPLÉMENTAIRES

- **sshd(8)**, **ssh(1)**, **scp(1)**, **sftp(1)**, **ssh-keygen(1)**, **ssh-copy-id(1)**, **ssh_config(5)**, **sshd_config(5)**, **update-crypto-policies(8)**, et **crypto-policies(7)**.
- Page d'accueil d'OpenSSH
- Configuration de SELinux pour les applications et les services avec des configurations non standard
- Contrôle du trafic réseau à l'aide de firewalld

CHAPITRE 2. CONFIGURATION DE LA COMMUNICATION SÉCURISÉE AVEC LE SITE SSH RÔLES DU SYSTÈME

En tant qu'administrateur, vous pouvez utiliser le rôle de système **sshd** pour configurer les serveurs SSH et le rôle de système **ssh** pour configurer les clients SSH de manière cohérente sur un nombre quelconque de systèmes RHEL en même temps à l'aide du paquetage Ansible Core.

2.1. SSH VARIABLES DE RÔLE DU SYSTÈME DE SERVEUR

Dans un playbook de rôle de système **sshd**, vous pouvez définir les paramètres du fichier de configuration SSH en fonction de vos préférences et de vos limites.

Si vous ne configurez pas ces variables, le rôle de système produit un fichier **sshd_config** qui correspond aux valeurs par défaut de RHEL.

Dans tous les cas, les booléens s'affichent correctement sous la forme **yes** et **no** dans la configuration **sshd**. Vous pouvez définir des éléments de configuration sur plusieurs lignes à l'aide de listes. Par exemple, vous pouvez définir des éléments de configuration sur plusieurs lignes à l'aide de listes :

```
sshd_ListenAddress:
  - 0.0.0.0
  - '::'
```

se traduit par :

```
ListenAddress 0.0.0.0
ListenAddress ::
```

Variables pour le rôle de systèmesshd

sshd_enable

Si la valeur est **False**, le rôle est complètement désactivé. La valeur par défaut est **True**.

sshd_skip_defaults

S'il est défini sur **True**, le rôle de système n'applique pas de valeurs par défaut. Au lieu de cela, vous devez spécifier l'ensemble des valeurs par défaut de la configuration en utilisant les variables **sshd** dict ou **sshd_Key**. La valeur par défaut est **False**.

sshd_manage_service

S'il vaut **False**, le service n'est pas géré, ce qui signifie qu'il n'est pas activé au démarrage et qu'il ne démarre pas ou ne se recharge pas. La valeur par défaut est **True**, sauf en cas d'exécution dans un conteneur ou sous AIX, car le module de service Ansible ne prend pas actuellement en charge **enabled** pour AIX.

sshd_allow_reload

S'il est réglé sur **False**, **sshd** ne se recharge pas après un changement de configuration. Cela peut faciliter le dépannage. Pour appliquer la configuration modifiée, rechargez manuellement **sshd**. La valeur par défaut est la même que celle de **sshd_manage_service**, sauf sous AIX, où **sshd_manage_service** a pour valeur par défaut **False** mais **sshd_allow_reload** a pour valeur par défaut **True**.

sshd_install_service

S'il est défini sur **True**, le rôle installe les fichiers de service pour le service **sshd**. Ces fichiers sont prioritaires sur ceux fournis par le système d'exploitation. Ne définissez pas **True** à moins que vous

ne configuriez une deuxième instance et que vous ne modifiiez également la variable **sshd_service**. La valeur par défaut est **False**.

Le rôle utilise comme modèles les fichiers désignés par les variables suivantes :

```
sshd_service_template_service (default: templates/sshd.service.j2)
sshd_service_template_at_service (default: templates/sshd@.service.j2)
sshd_service_template_socket (default: templates/sshd.socket.j2)
```

sshd_service

Cette variable modifie le nom du service **sshd**, ce qui est utile pour configurer une deuxième instance du service **sshd**.

sshd

Un dict qui contient la configuration. Par exemple :

```
sshd:
  Compression: yes
  ListenAddress:
    - 0.0.0.0
```

sshd_*OptionName*

Vous pouvez définir des options en utilisant des variables simples composées du préfixe **sshd_** et du nom de l'option au lieu d'un dict. Les variables simples remplacent les valeurs du dict **sshd**. Par exemple :

```
sshd_Compression: no
```

sshd_match et **sshd_match_1** à **sshd_match_9**

Une liste de dicts ou juste un dict pour une section Match. Notez que ces variables ne remplacent pas les blocs de correspondance tels qu'ils sont définis dans le dict **sshd**. Toutes les sources seront reflétées dans le fichier de configuration résultant.

Variables secondaires pour le rôle du système sshd

Vous pouvez utiliser ces variables pour remplacer les valeurs par défaut correspondant à chaque plate-forme prise en charge.

sshd_packages

Vous pouvez remplacer la liste par défaut des paquets installés en utilisant cette variable.

sshd_config_owner, sshd_config_group, et **sshd_config_mode**

Vous pouvez définir la propriété et les autorisations du fichier de configuration **openssh** que ce rôle produit à l'aide de ces variables.

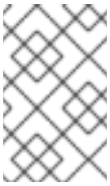
sshd_config_file

Le chemin où ce rôle enregistre la configuration du serveur **openssh** produite.

sshd_config_namespace

La valeur par défaut de cette variable est null, ce qui signifie que le rôle définit l'intégralité du contenu du fichier de configuration, y compris les valeurs par défaut du système. Vous pouvez également utiliser cette variable pour invoquer ce rôle à partir d'autres rôles ou à partir de plusieurs endroits dans un seul playbook sur des systèmes qui ne prennent pas en charge le répertoire drop-in. La variable **sshd_skip_defaults** est ignorée et aucune valeur par défaut du système n'est utilisée dans ce cas.

Lorsque cette variable est définie, le rôle place la configuration que vous spécifiez aux extraits de configuration dans un fichier de configuration existant sous l'espace de noms donné. Si votre scénario nécessite d'appliquer le rôle plusieurs fois, vous devez sélectionner un espace de noms différent pour chaque application.



NOTE

Les limitations du fichier de configuration **openssh** restent d'application. Par exemple, seule la première option spécifiée dans un fichier de configuration est effective pour la plupart des options de configuration.

Techniquement, le rôle place les extraits dans les blocs "Match all", à moins qu'ils ne contiennent d'autres blocs de correspondance, afin de s'assurer qu'ils sont appliqués indépendamment des blocs de correspondance précédents dans le fichier de configuration existant. Cela permet de configurer des options non contradictoires à partir de différentes invocations de rôles.

sshd_binary

Chemin d'accès à l'exécutable **sshd** de **openssh**.

sshd_service

Le nom du service **sshd**. Par défaut, cette variable contient le nom du service **sshd** utilisé par la plate-forme cible. Vous pouvez également l'utiliser pour définir le nom du service personnalisé **sshd** lorsque le rôle utilise la variable **sshd_install_service**.

sshd_verify_hostkeys

La valeur par défaut est **auto**. Si la valeur est **auto**, toutes les clés d'hôte présentes dans le fichier de configuration produit sont répertoriées et tous les chemins d'accès non présents sont générés. De plus, les permissions et les propriétaires de fichiers sont définis sur des valeurs par défaut. Ceci est utile si le rôle est utilisé dans la phase de déploiement pour s'assurer que le service est capable de démarrer à la première tentative. Pour désactiver cette vérification, donnez à cette variable la valeur d'une liste vide **[]**.

sshd_hostkey_owner, sshd_hostkey_group, sshd_hostkey_mode

Utilisez ces variables pour définir la propriété et les autorisations des clés de l'hôte à partir de **sshd_verify_hostkeys**.

sshd_sysconfig

Sur les systèmes basés sur RHEL, cette variable configure des détails supplémentaires du service **sshd**. S'il est défini sur **true**, ce rôle gère également le fichier de configuration **/etc/sysconfig/sshd** sur la base de la configuration suivante. La valeur par défaut est **false**.

sshd_sysconfig_override_crypto_policy

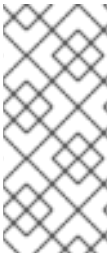
Dans RHEL, lorsqu'elle est définie sur **true**, cette variable remplace la politique cryptographique du système. La valeur par défaut est **false**.

sshd_sysconfig_use_strong_rng

Sur les systèmes basés sur RHEL, cette variable peut forcer **sshd** à réalimenter le générateur de nombres aléatoires **openssl** avec le nombre d'octets donné en argument. La valeur par défaut est **0**, qui désactive cette fonctionnalité. Ne l'activez pas si le système ne dispose pas d'un générateur de nombres aléatoires matériel.

2.2. CONFIGURATION DES SERVEURS OPENSSH À L'AIDE DU RÔLE DE SYSTÈME SSHD

Vous pouvez utiliser le rôle de système **sshd** pour configurer plusieurs serveurs SSH en exécutant un script Ansible.



NOTE

Vous pouvez utiliser le rôle système **sshd** avec d'autres rôles système qui modifient la configuration de SSH et de SSHD, par exemple les rôles système RHEL de gestion des identités. Pour éviter que la configuration ne soit écrasée, assurez-vous que le rôle **sshd** utilise des espaces de noms (RHEL 8 et versions antérieures) ou un répertoire de dépôt (RHEL 9).

Conditions préalables

- Accès et autorisations à un ou plusieurs *managed nodes*, qui sont des systèmes que vous souhaitez configurer avec le rôle de système **sshd**.
- Accès et permissions à un *control node*, qui est un système à partir duquel Red Hat Ansible Core configure d'autres systèmes.
  Sur le nœud de contrôle :
  - Les paquets **ansible-core** et **rhel-system-roles** sont installés.

IMPORTANT

RHEL 8.0-8.5 donne accès à un dépôt Ansible distinct qui contient Ansible Engine 2.9 pour l'automatisation basée sur Ansible. Ansible Engine contient des utilitaires de ligne de commande tels que **ansible**, **ansible-playbook**, des connecteurs tels que **docker** et **podman**, ainsi que de nombreux plugins et modules. Pour plus d'informations sur la manière d'obtenir et d'installer Ansible Engine, consultez l'article de la base de connaissances Comment télécharger et installer Red Hat Ansible Engine .

RHEL 8.6 et 9.0 ont introduit Ansible Core (fourni en tant que paquetage **ansible-core**), qui contient les utilitaires de ligne de commande Ansible, les commandes et un petit ensemble de plugins Ansible intégrés. RHEL fournit ce paquetage par l'intermédiaire du dépôt AppStream, et sa prise en charge est limitée. Pour plus d'informations, consultez l'article de la base de connaissances intitulé Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories (Portée de la prise en charge du package Ansible Core inclus dans les dépôts AppStream RHEL 9 et RHEL 8.6 et versions ultérieures).

- Un fichier d'inventaire qui répertorie les nœuds gérés.

Procédure

1. Copiez l'exemple de playbook pour le rôle de système **sshd**:

```
# cp /usr/share/doc/rhel-system-roles/sshd/example-root-login-playbook.yml path/custom-playbook.yml
```

2. Ouvrez le playbook copié en utilisant un éditeur de texte, par exemple :

```
# vim path/custom-playbook.yml

---
```

```
- hosts: all
  tasks:
  - name: Configure sshd to prevent root and password login except from particular subnet
    include_role:
      name: rhel-system-roles.sshd
    vars:
      sshd:
        # root login and password login is enabled only from a particular subnet
        PermitRootLogin: no
        PasswordAuthentication: no
        Match:
        - Condition: "Address 192.0.2.0/24"
          PermitRootLogin: yes
          PasswordAuthentication: yes
```

Le playbook configure le nœud géré en tant que serveur SSH configuré de telle sorte que :

- et le login de l'utilisateur **root** est désactivé
- et **root** n'est possible qu'à partir du sous-réseau **192.0.2.0/24**

Vous pouvez modifier les variables en fonction de vos préférences. Pour plus de détails, voir Variables de rôle du système du serveur SSH .

3. Facultatif : Vérifier la syntaxe du playbook.

```
# ansible-playbook --syntax-check path/custom-playbook.yml
```

4. Exécutez le playbook sur votre fichier d'inventaire :

```
# ansible-playbook -i inventory_file path/custom-playbook.yml

...

PLAY RECAP
**************************************************

localhost : ok=12 changed=2 unreachable=0 failed=0
skipped=10 rescued=0 ignored=0
```

Vérification

1. Connectez-vous au serveur SSH :

```
$ ssh user1@10.1.1.1
```

Où ?

- ***user1*** est un utilisateur du serveur SSH.
- ***10.1.1.1*** est l'adresse IP du serveur SSH.

2. Vérifiez le contenu du fichier **sshd_config** sur le serveur SSH :

```
$ cat /etc/ssh/sshd_config.d/00-ansible_system_role.conf
```

```
#
# Ansible managed
#
PasswordAuthentication no
PermitRootLogin no
Match Address 192.0.2.0/24
  PasswordAuthentication yes
  PermitRootLogin yes
```

3. Vérifiez que vous pouvez vous connecter au serveur en tant que root depuis le sous-réseau **192.0.2.0/24**:

   a. Déterminez votre adresse IP :

   ```
   $ hostname -I
   192.0.2.1
   ```

   Si l'adresse IP se situe dans la plage **192.0.2.1** - **192.0.2.254**, vous pouvez vous connecter au serveur.

   b. Se connecter au serveur en tant que **root**:

   ```
   $ ssh root@10.1.1.1
   ```

Ressources supplémentaires

- **/usr/share/doc/rhel-system-roles/sshd/README.md** fichier.
- **ansible-playbook(1)** page de manuel.

2.3. SSH VARIABLES DE RÔLE DU SYSTÈME

Dans un playbook **ssh** System Role, vous pouvez définir les paramètres du fichier de configuration SSH du client en fonction de vos préférences et de vos limites.

Si vous ne configurez pas ces variables, le rôle de système produit un fichier global **ssh_config** qui correspond aux valeurs par défaut de RHEL.

Dans tous les cas, les booléens s'affichent correctement sous la forme **yes** ou **no** dans la configuration **ssh**. Vous pouvez définir des éléments de configuration sur plusieurs lignes à l'aide de listes. Par exemple, vous pouvez définir des éléments de configuration sur plusieurs lignes à l'aide de listes :

```
LocalForward:
  - 22 localhost:2222
  - 403 localhost:4003
```

se traduit par :

```
LocalForward 22 localhost:2222
LocalForward 403 localhost:4003
```

NOTE

Les options de configuration sont sensibles à la casse.

Variables pour le rôle de systèmessh

ssh_user

Vous pouvez définir un nom d'utilisateur existant pour lequel le rôle système modifie la configuration spécifique à l'utilisateur. La configuration spécifique à l'utilisateur est sauvegardée dans **~/.ssh/config** de l'utilisateur donné. La valeur par défaut est null, ce qui modifie la configuration globale pour tous les utilisateurs.

ssh_skip_defaults

La valeur par défaut est **auto**. Si la valeur est **auto**, le rôle système écrit le fichier de configuration de l'ensemble du système **/etc/ssh/ssh_config** et conserve les valeurs par défaut RHEL qui y sont définies. La création d'un fichier de configuration d'insertion, par exemple en définissant la variable **ssh_drop_in_name**, désactive automatiquement la variable **ssh_skip_defaults**.

ssh_drop_in_name

Définit le nom du fichier de configuration d'insertion, qui est placé dans le répertoire d'insertion de l'ensemble du système. Ce nom est utilisé dans le modèle **/etc/ssh/ssh_config.d/{ssh_drop_in_name}.conf** pour référencer le fichier de configuration à modifier. Si le système ne prend pas en charge les répertoires de dépôt, la valeur par défaut est null. Si le système prend en charge les répertoires de dépôt, la valeur par défaut est **00-ansible**.

AVERTISSEMENT

Si le système ne prend pas en charge les répertoires d'insertion, la définition de cette option entraînera l'échec de la lecture.

Le format suggéré est **NN-name**, où **NN** est un numéro à deux chiffres utilisé pour classer les fichiers de configuration et **name** est un nom descriptif du contenu ou du propriétaire du fichier.

ssh

Un dict qui contient les options de configuration et leurs valeurs respectives.

ssh_*OptionName*

Vous pouvez définir des options en utilisant des variables simples composées du préfixe **ssh_** et du nom de l'option au lieu d'un dict. Les variables simples remplacent les valeurs du dict **ssh**.

ssh_additional_packages

Ce rôle installe automatiquement les paquets **openssh** et **openssh-clients**, qui sont nécessaires pour les cas d'utilisation les plus courants. Si vous devez installer des paquets supplémentaires, par exemple **openssh-keysign** pour l'authentification basée sur l'hôte, vous pouvez les spécifier dans cette variable.

ssh_config_file

Chemin dans lequel le rôle enregistre le fichier de configuration produit. Valeur par défaut :

- Si le système dispose d'un répertoire de dépôt, la valeur par défaut est définie par le modèle **/etc/ssh/ssh_config.d/{ssh_drop_in_name}.conf**.

- Si le système ne dispose pas d'un répertoire d'accueil, la valeur par défaut est **/etc/ssh/ssh_config**.
- si la variable **ssh_user** est définie, la valeur par défaut est **~/.ssh/config**.

ssh_config_owner, ssh_config_group, ssh_config_mode

Le propriétaire, le groupe et les modes du fichier de configuration créé. Par défaut, le propriétaire du fichier est **root:root**, et le mode est **0644**. Si **ssh_user** est défini, le mode est **0600**, et le propriétaire et le groupe sont dérivés du nom d'utilisateur spécifié dans la variable **ssh_user**.

2.4. CONFIGURATION DES CLIENTS OPENSSH À L'AIDE DU RÔLE DE SYSTÈME SSH

Vous pouvez utiliser le rôle de système **ssh** pour configurer plusieurs clients SSH en exécutant un script Ansible.

NOTE

Vous pouvez utiliser le rôle de système **ssh** avec d'autres rôles de système qui modifient la configuration de SSH et de SSHD, par exemple les rôles de système RHEL de gestion des identités. Pour éviter que la configuration ne soit écrasée, assurez-vous que le rôle **ssh** utilise un répertoire de dépôt (par défaut à partir de RHEL 8).

Conditions préalables

- Accès et autorisations à un ou plusieurs *managed nodes*, qui sont des systèmes que vous souhaitez configurer avec le rôle de système **ssh**.
- Accès et permissions à un *control node*, qui est un système à partir duquel Red Hat Ansible Core configure d'autres systèmes.
  Sur le nœud de contrôle :
  - Les paquets **ansible-core** et **rhel-system-roles** sont installés.

IMPORTANT

RHEL 8.0-8.5 donne accès à un dépôt Ansible distinct qui contient Ansible Engine 2.9 pour l'automatisation basée sur Ansible. Ansible Engine contient des utilitaires de ligne de commande tels que **ansible**, **ansible-playbook**, des connecteurs tels que **docker** et **podman**, ainsi que de nombreux plugins et modules. Pour plus d'informations sur la manière d'obtenir et d'installer Ansible Engine, consultez l'article de la base de connaissances Comment télécharger et installer Red Hat Ansible Engine .

RHEL 8.6 et 9.0 ont introduit Ansible Core (fourni en tant que paquetage **ansible-core**), qui contient les utilitaires de ligne de commande Ansible, les commandes et un petit ensemble de plugins Ansible intégrés. RHEL fournit ce paquetage par l'intermédiaire du dépôt AppStream, et sa prise en charge est limitée. Pour plus d'informations, consultez l'article de la base de connaissances intitulé Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories (Portée de la prise en charge du package Ansible Core inclus dans les dépôts AppStream RHEL 9 et RHEL 8.6 et versions ultérieures).

- Un fichier d'inventaire qui répertorie les nœuds gérés.

Procédure

1. Créez un nouveau fichier ***playbook.yml*** avec le contenu suivant :

```
---
- hosts: all
  tasks:
  - name: "Configure ssh clients"
    include_role:
      name: rhel-system-roles.ssh
    vars:
      ssh_user: root
      ssh:
        Compression: true
        GSSAPIAuthentication: no
        ControlMaster: auto
        ControlPath: ~/.ssh/.cm%C
        Host:
          - Condition: example
            Hostname: example.com
            User: user1
      ssh_ForwardX11: no
```

   Ce playbook configure les préférences du client SSH de l'utilisateur **root** sur les nœuds gérés avec les configurations suivantes :

   - La compression est activée.
   - Le multiplexage ControlMaster est réglé sur **auto**.
   - L'alias ***example*** pour se connecter à l'hôte ***example.com*** est ***user1***.
   - L'alias ***example*** l'alias de l'hôte est créé, ce qui représente une connexion à l'hôte ***example.com*** hôte avec le ***user1*** nom d'utilisateur.
   - Le transfert X11 est désactivé.

   En option, vous pouvez modifier ces variables selon vos préférences. Pour plus de détails, voir **ssh** System Role variables.

2. Facultatif : Vérifier la syntaxe du playbook.

```
# ansible-playbook --syntax-check path/custom-playbook.yml
```

3. Exécutez le playbook sur votre fichier d'inventaire :

```
# ansible-playbook -i inventory_file path/custom-playbook.yml
```

Vérification

- Vérifiez que le nœud géré a la bonne configuration en ouvrant le fichier de configuration SSH dans un éditeur de texte, par exemple :

```
# vi ~root/.ssh/config
```

Après l'application de l'exemple de playbook présenté ci-dessus, le fichier de configuration devrait avoir le contenu suivant :

```
# Ansible managed
Compression yes
ControlMaster auto
ControlPath ~/.ssh/.cm%C
ForwardX11 no
GSSAPIAuthentication no
Host example
  Hostname example.com
  User user1
```

2.5. UTILISATION DU RÔLE DE SYSTÈME SSHD POUR UNE CONFIGURATION NON EXCLUSIVE

Normalement, l'application du rôle de système **sshd** écrase toute la configuration. Cela peut être problématique si vous avez précédemment ajusté la configuration, par exemple avec un rôle de système ou un livre de jeu différent. Pour appliquer le rôle système **sshd** uniquement aux options de configuration sélectionnées tout en conservant les autres options, vous pouvez utiliser la configuration non exclusive.

Dans RHEL 8 et les versions antérieures, vous pouvez appliquer la configuration non exclusive à l'aide d'un extrait de configuration. Pour plus d'informations, voir Utilisation du rôle de système du serveur SSH pour la configuration non exclusive dans la documentation de RHEL 8.

Dans RHEL 9, vous pouvez appliquer la configuration non exclusive en utilisant des fichiers dans un répertoire de dépôt. Le fichier de configuration par défaut est déjà placé dans le répertoire de dépôt en tant que **/etc/ssh/sshd_config.d/00-ansible_system_role.conf**.

Conditions préalables

- Accès et autorisations à un ou plusieurs *managed nodes*, qui sont des systèmes que vous souhaitez configurer avec le rôle de système **sshd**.
- Accès et permissions à un *control node*, qui est un système à partir duquel Red Hat Ansible Core configure d'autres systèmes.
  Sur le nœud de contrôle :
  - Le paquet **ansible-core** est installé.
  - Un fichier d'inventaire qui répertorie les nœuds gérés.
  - Un playbook pour un rôle de système RHEL différent.

Procédure

1. Ajoutez un extrait de configuration avec la variable **sshd_config_file** au playbook :

```
---
- hosts: all
  tasks:
  - name: <Configure sshd to accept some useful environment variables>
    include_role:
```

```
    name: rhel-system-roles.sshd
  vars:
    sshd_config_file: /etc/ssh/sshd_config.d/<42-my-application>.conf
    sshd:
      # Environment variables to accept
      AcceptEnv:
        LANG
        LS_COLORS
        EDITOR
```

Dans la variable **sshd_config_file**, définissez le fichier **.conf** dans lequel le rôle de système **sshd** écrit les options de configuration.

Utilisez un préfixe à deux chiffres, par exemple ***42-*** pour spécifier l'ordre dans lequel les fichiers de configuration seront appliqués.

Lorsque vous appliquez le playbook à l'inventaire, le rôle ajoute les options de configuration suivantes au fichier défini par la variable **sshd_config_file**.

```
# Ansible managed
#
AcceptEnv LANG LS_COLORS EDITOR
```

Vérification

- Facultatif : Vérifier la syntaxe du playbook.

```
# ansible-playbook --syntax-check playbook.yml -i inventory_file
```

Ressources supplémentaires

- **/usr/share/doc/rhel-system-roles/sshd/README.md** fichier.
- **ansible-playbook(1)** page de manuel.

CHAPITRE 3. CRÉER ET GÉRER DES CLÉS ET DES CERTIFICATS TLS

Vous pouvez crypter les communications transmises entre deux systèmes en utilisant le protocole TLS (Transport Layer Security). Cette norme utilise la cryptographie asymétrique avec des clés privées et publiques, des signatures numériques et des certificats.

3.1. CERTIFICATS TLS

TLS (Transport Layer Security) est un protocole qui permet aux applications client-serveur de transmettre des informations en toute sécurité. TLS utilise un système de paires de clés publiques et privées pour crypter les communications transmises entre les clients et les serveurs. TLS est le protocole qui succède à SSL (Secure Sockets Layer).

TLS utilise des certificats X.509 pour lier des identités, telles que des noms d'hôte ou des organisations, à des clés publiques à l'aide de signatures numériques. X.509 est une norme qui définit le format des certificats de clé publique.

L'authentification d'une application sécurisée dépend de l'intégrité de la valeur de la clé publique du certificat de l'application. Si un pirate remplace la clé publique par sa propre clé publique, il peut se faire passer pour la véritable application et accéder aux données sécurisées. Pour éviter ce type d'attaque, tous les certificats doivent être signés par une autorité de certification (AC). Une autorité de certification est un nœud de confiance qui confirme l'intégrité de la valeur de la clé publique d'un certificat.

Une AC signe une clé publique en y ajoutant sa signature numérique et délivre un certificat. Une signature numérique est un message codé avec la clé privée de l'AC. La clé publique de l'autorité de certification est mise à la disposition des applications en distribuant le certificat de l'autorité de certification. Les applications vérifient que les certificats sont valablement signés en décodant la signature numérique de l'AC avec la clé publique de l'AC.

Pour faire signer un certificat par une autorité de certification, vous devez générer une clé publique et l'envoyer à l'autorité de certification pour qu'elle la signe. C'est ce qu'on appelle une demande de signature de certificat (CSR). Une CSR contient également un nom distinctif (DN) pour le certificat. Les informations DN que vous pouvez fournir pour l'un ou l'autre type de certificat peuvent inclure un code pays à deux lettres pour votre pays, le nom complet de votre état ou province, votre ville, le nom de votre organisation, votre adresse électronique, et elles peuvent également être vides. De nombreuses autorités de certification commerciales actuelles préfèrent l'extension Subject Alternative Name et ignorent les DN dans les CSR.

RHEL fournit deux boîtes à outils principales pour travailler avec des certificats TLS : GnuTLS et OpenSSL. Vous pouvez créer, lire, signer et vérifier des certificats à l'aide de l'utilitaire **openssl** du paquetage **openssl**. L'utilitaire **certtool** fourni par le paquetage **gnutls-utils** peut effectuer les mêmes opérations en utilisant une syntaxe différente et surtout un ensemble différent de bibliothèques en arrière-plan.

Ressources supplémentaires

- RFC 5280 : Profil des certificats et des listes de révocation de certificats (CRL) de l'infrastructure à clé publique X.509 de l'Internet
- **openssl(1)**, **x509(1)**, **ca(1)**, **req(1)**, et **certtool(1)** pages de manuel

3.2. CRÉATION D'UNE AUTORITÉ DE CERTIFICATION PRIVÉE À L'AIDE D'OPENSSL

Les autorités de certification (AC) privées sont utiles lorsque votre scénario exige la vérification d'entités au sein de votre réseau interne. Par exemple, utilisez une autorité de certification privée lorsque vous créez une passerelle VPN dont l'authentification est basée sur des certificats signés par une autorité de certification sous votre contrôle ou lorsque vous ne souhaitez pas payer une autorité de certification commerciale. Pour signer des certificats dans de tels cas d'utilisation, l'AC privée utilise un certificat auto-signé.

Conditions préalables

- Vous disposez des privilèges ou autorisations **root** pour entrer dans les commandes administratives avec **sudo**. Les commandes qui requièrent de tels privilèges sont marquées par **#**.

Procédure

1. Générez une clé privée pour votre autorité de certification. Par exemple, la commande suivante crée une clé ECDSA (Elliptic Curve Digital Signature Algorithm) de 256 bits :

   ```
   $ openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:P-256 -out <ca.key>
   ```

   La durée du processus de génération des clés dépend du matériel et de l'entropie de l'hôte, de l'algorithme sélectionné et de la longueur de la clé.

2. Créez un certificat signé à l'aide de la clé privée générée dans la commande précédente :

   ```
   $ openssl req -key <ca.key> -new -x509 -days 3650 -addext
   keyUsage=critical,keyCertSign,cRLSign -subj "/CN=<Example CA>" -out <ca.crt>
   ```

   Le fichier **ca.crt** généré est un certificat d'autorité de certification auto-signé que vous pouvez utiliser pour signer d'autres certificats pendant dix ans. Dans le cas d'une autorité de certification privée, vous pouvez remplacer *<Example CA>* par n'importe quelle chaîne de caractères comme nom commun (CN).

3. Définissez des autorisations sécurisées pour la clé privée de votre autorité de certification, par exemple :

   ```
   # chown <root>:<root> <ca.key>
   # chmod 600 <ca.key>
   ```

Prochaines étapes

- Pour utiliser un certificat d'autorité de certification auto-signé comme ancre de confiance sur les systèmes clients, copiez le certificat d'autorité de certification sur le client et ajoutez-le à la liste de confiance de l'ensemble du système des clients en tant que **root**:

  ```
  # trust anchor <ca.crt>
  ```

  Voir Chapitre 4, *Utilisation de certificats système partagés* pour plus d'informations.

Vérification

1. Créez une demande de signature de certificat (CSR) et utilisez votre autorité de certification pour signer la demande. L'autorité de certification doit réussir à créer un certificat sur la base de la RSC, par exemple :

```
$ openssl x509 -req -in <client-cert.csr> -CA <ca.crt> -CAkey <ca.key> -CAcreateserial -
days 365 -extfile <openssl.cnf> -extensions <client-cert> -out <client-cert.crt>
Signature ok
subject=C = US, O = Example Organization, CN = server.example.com
Getting CA Private Key
```

   Voir Section 3.5, « Utilisation d'une autorité de certification privée pour émettre des certificats pour les CSR avec OpenSSL » pour plus d'informations.

2. Affichez les informations de base sur votre AC auto-signée :

```
$ openssl x509 -in <ca.crt> -text -noout
Certificate:
…
        X509v3 extensions:
            …
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
…
```

3. Vérifier la cohérence de la clé privée :

```
$ openssl pkey -check -in <ca.key>
Key is valid
-----BEGIN PRIVATE KEY-----
MIGHAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQgcagSaTEBn74xZAwO

18wRpXoCVC9vcPki7WlT+gnmCI+hRANCAARb9NxIvkaVjFhOoZbGp/HtIQxbM78E
lwbDP0BI624xBJ8gK68ogSaq2x4SdezFdV1gNeKScDcU+Pj2pELldmdF
-----END PRIVATE KEY-----
```

Ressources supplémentaires

- **openssl(1)**, **ca(1)**, **genpkey(1)**, **x509(1)**, et **req(1)** pages de manuel

3.3. CRÉATION D'UNE CLÉ PRIVÉE ET D'UNE CSR POUR UN CERTIFICAT DE SERVEUR TLS À L'AIDE D'OPENSSL

Vous ne pouvez utiliser les canaux de communication cryptés TLS que si vous disposez d'un certificat TLS valide délivré par une autorité de certification (AC). Pour obtenir le certificat, vous devez d'abord créer une clé privée et une demande de signature de certificat (CSR) pour votre serveur.

Procédure

1. Générez une clé privée sur votre système serveur, par exemple :

```
$ openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:P-256 -out <server-private.key>
```

2. Facultatif : Utilisez un éditeur de texte de votre choix pour préparer un fichier de configuration qui simplifie la création de votre CSR, par exemple :

```
$ vim <example_server.cnf>
[server-cert]
keyUsage = critical, digitalSignature, keyEncipherment, keyAgreement
extendedKeyUsage = serverAuth
subjectAltName = @alt_name

[req]
distinguished_name = dn
prompt = no

[dn]
C = <US>
O = <Example Organization>
CN = <server.example.com>

[alt_name]
DNS.1 = <example.com>
DNS.2 = <server.example.com>
IP.1 = <192.168.0.1>
IP.2 = <::1>
IP.3 = <127.0.0.1>
```

L'option **extendedKeyUsage = serverAuth** limite l'utilisation d'un certificat.

3. Créez une RSC à l'aide de la clé privée que vous avez créée précédemment :

```
$ openssl req -key <server-private.key> -config <example_server.cnf> -new -out <server-cert.csr>
```

Si vous omettez l'option **-config**, l'utilitaire **req** vous demande des informations supplémentaires, par exemple :

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]: <US>
State or Province Name (full name) []: <Washington>
Locality Name (eg, city) [Default City]: <Seattle>
Organization Name (eg, company) [Default Company Ltd]: <Example Organization>
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []: <server.example.com>
Email Address []: <server@example.com>
```

Prochaines étapes

- Soumettez la CSR à l'autorité de certification de votre choix pour signature. Sinon, pour un scénario d'utilisation interne au sein d'un réseau de confiance, utilisez votre autorité de certification privée pour la signature. Voir Section 3.5, « Utilisation d'une autorité de certification privée pour émettre des certificats pour les CSR avec OpenSSL » pour plus d'informations.

Vérification

1. Après avoir obtenu le certificat demandé auprès de l'autorité de certification, vérifiez que les parties lisibles par l'homme du certificat correspondent à vos exigences, par exemple :

```
$ openssl x509 -text -noout -in <server-cert.crt>
Certificate:
…
        Issuer: CN = Example CA
        Validity
            Not Before: Feb  2 20:27:29 2023 GMT
            Not After : Feb  2 20:27:29 2024 GMT
        Subject: C = US, O = Example Organization, CN = server.example.com
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
…
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment, Key Agreement
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 Subject Alternative Name:
                DNS:example.com, DNS:server.example.com, IP Address:192.168.0.1, IP
…
```

Ressources supplémentaires

- **openssl(1)**, **x509(1)**, **genpkey(1)**, **req(1)**, et **config(5)** pages de manuel

3.4. CRÉATION D'UNE CLÉ PRIVÉE ET D'UNE CSR POUR UN CERTIFICAT CLIENT TLS À L'AIDE D'OPENSSL

Vous ne pouvez utiliser les canaux de communication cryptés TLS que si vous disposez d'un certificat TLS valide délivré par une autorité de certification (AC). Pour obtenir le certificat, vous devez d'abord créer une clé privée et une demande de signature de certificat (CSR) pour votre client.

Procédure

1. Générez une clé privée sur votre système client, par exemple :

```
$ openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:P-256 -out <client-private.key>
```

2. Facultatif : Utilisez un éditeur de texte de votre choix pour préparer un fichier de configuration qui simplifie la création de votre CSR, par exemple :

```
$ vim <example_client.cnf>
[client-cert]
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth
subjectAltName = @alt_name

[req]
distinguished_name = dn
prompt = no

[dn]
CN = <client.example.com>

[clnt_alt_name]
email= <client@example.com>
```

L'option **extendedKeyUsage = clientAuth** limite l'utilisation d'un certificat.

3. Créez une RSC à l'aide de la clé privée que vous avez créée précédemment :

```
$ openssl req -key <client-private.key> -config <example_client.cnf> -new -out <client-cert.csr>
```

Si vous omettez l'option **-config**, l'utilitaire **req** vous demande des informations supplémentaires, par exemple :

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
…
Common Name (eg, your name or your server's hostname) []: <client.example.com>
Email Address []: <client@example.com>
```

Prochaines étapes

- Soumettez la CSR à l'autorité de certification de votre choix pour signature. Sinon, pour un scénario d'utilisation interne au sein d'un réseau de confiance, utilisez votre autorité de certification privée pour la signature. Voir Section 3.5, « Utilisation d'une autorité de certification privée pour émettre des certificats pour les CSR avec OpenSSL » pour plus d'informations.

Vérification

1. Vérifiez que les parties du certificat lisibles par l'homme correspondent à vos exigences, par exemple :

```
$ openssl x509 -text -noout -in <client-cert.crt>
Certificate:
…
            X509v3 Extended Key Usage:
                TLS Web Client Authentication
            X509v3 Subject Alternative Name:
                email:client@example.com
…
```

Ressources supplémentaires

- **openssl(1)**, **x509(1)**, **genpkey(1)**, **req(1)**, et **config(5)** pages de manuel

3.5. UTILISATION D'UNE AUTORITÉ DE CERTIFICATION PRIVÉE POUR ÉMETTRE DES CERTIFICATS POUR LES CSR AVEC OPENSSL

Pour permettre aux systèmes d'établir un canal de communication crypté TLS, une autorité de certification (AC) doit leur fournir des certificats valides. Si vous disposez d'une autorité de certification privée, vous pouvez créer les certificats requis en signant les demandes de signature de certificat (CSR) des systèmes.

Conditions préalables

- Vous avez déjà configuré une autorité de certification privée. Voir Section 3.2, « Création d'une autorité de certification privée à l'aide d'OpenSSL » pour plus d'informations.
- Vous avez un fichier contenant un CSR. Vous trouverez un exemple de création de CSR sur Section 3.3, « Création d'une clé privée et d'une CSR pour un certificat de serveur TLS à l'aide d'OpenSSL ».

Procédure

1. Facultatif : Utilisez un éditeur de texte de votre choix pour préparer un fichier de configuration OpenSSL afin d'ajouter des extensions aux certificats, par exemple :

```
$ vim <openssl.cnf>
[server-cert]
extendedKeyUsage = serverAuth

[client-cert]
extendedKeyUsage = clientAuth
```

2. Utilisez l'utilitaire **x509** pour créer un certificat basé sur une CSR, par exemple :

```
$ openssl x509 -req -in <server-cert.csr> -CA <ca.crt> -CAkey <ca.key> -days 365 -extfile
<openssl.cnf> -extensions <server-cert> -out <server-cert.crt>
Signature ok
subject=C = US, O = Example Organization, CN = server.example.com
Getting CA Private Key
```

Ressources supplémentaires

- **openssl(1)**, **ca(1)**, et **x509(1)** pages de manuel

3.6. CRÉATION D'UNE AUTORITÉ DE CERTIFICATION PRIVÉE À L'AIDE DE GNUTLS

Les autorités de certification (AC) privées sont utiles lorsque votre scénario exige la vérification d'entités au sein de votre réseau interne. Par exemple, utilisez une autorité de certification privée lorsque vous créez une passerelle VPN dont l'authentification est basée sur des certificats signés par

une autorité de certification sous votre contrôle ou lorsque vous ne souhaitez pas payer une autorité de certification commerciale. Pour signer des certificats dans de tels cas d'utilisation, l'AC privée utilise un certificat auto-signé.

Conditions préalables

- Vous disposez des privilèges ou autorisations **root** pour entrer dans les commandes administratives avec **sudo**. Les commandes qui requièrent de tels privilèges sont marquées par **#**.
- Vous avez déjà installé GnuTLS sur votre système. Si ce n'est pas le cas, vous pouvez utiliser cette commande :

```
$ dnf install gnutls-utils
```

Procédure

1. Générez une clé privée pour votre autorité de certification. Par exemple, la commande suivante crée une clé ECDSA (Elliptic Curve Digital Signature Algorithm) de 256 bits :

```
$ certtool --generate-privkey --sec-param High --key-type=ecdsa --outfile <ca.key>
```

   La durée du processus de génération des clés dépend du matériel et de l'entropie de l'hôte, de l'algorithme sélectionné et de la longueur de la clé.

2. Créer un fichier modèle pour un certificat.
   a. Créez un fichier avec votre éditeur de texte préféré, par exemple vi :

```
$ vi <ca.cfg>
```

   b. Modifiez le fichier pour y inclure les détails nécessaires à la certification :

```
organization = "Example Inc."
state = "Example"
country = EX
cn = "Example CA"
serial = 007
expiration_days = 365
ca
cert_signing_key
crl_signing_key
```

3. Créez un certificat signé à l'aide de la clé privée générée à l'étape 1 :
   Le fichier *<ca.crt>* généré est un certificat CA auto-signé que vous pouvez utiliser pour signer d'autres certificats pendant un an. Le fichier *<ca.crt>* est la clé publique (certificat). Le fichier chargé *<ca.key>* est la clé privée. Vous devez conserver ce fichier dans un endroit sûr.

```
$ certtool --generate-self-signed --load-privkey <ca.key> --template <ca.cfg> --outfile
<ca.crt>
```

4. Définissez des autorisations sécurisées pour la clé privée de votre autorité de certification, par exemple :

```
# chown <root>:<root> <ca.key>
# chmod 600 <ca.key>
```

Prochaines étapes

- Pour utiliser un certificat d'autorité de certification auto-signé comme ancre de confiance sur les systèmes clients, copiez le certificat d'autorité de certification sur le client et ajoutez-le à la liste de confiance de l'ensemble du système des clients en tant que **root**:

```
# trust anchor <ca.crt>
```

Voir Chapitre 4, *Utilisation de certificats système partagés* pour plus d'informations.

Vérification

1. Affichez les informations de base sur votre AC auto-signée :

```
$ certtool --certificate-info --infile <ca.crt>
Certificate:
…
    X509v3 extensions:
        …
        X509v3 Basic Constraints: critical
            CA:TRUE
        X509v3 Key Usage: critical
            Certificate Sign, CRL Sign
```

2. Créez une demande de signature de certificat (CSR) et utilisez votre autorité de certification pour signer la demande. L'autorité de certification doit réussir à créer un certificat sur la base de la RSC, par exemple :

   a. Générez une clé privée pour votre autorité de certification :

```
$ certtool --generate-privkey --outfile <example-server.key>
```

   b. Créez un fichier avec votre éditeur de texte préféré, par exemple vi :

```
$ vi <example-server.cfg>
```

   c. Modifiez le fichier pour y inclure les détails nécessaires à la certification :

```
signing_key
encryption_key
key_agreement

tls_www_server

country = "US"
organization = "Example Organization"
cn = "server.example.com"

dns_name = "example.com"
dns_name = "server.example.com"
```

```
ip_address = "192.168.0.1"
ip_address = "::1"
ip_address = "127.0.0.1"
```

d. Générer une demande avec la clé privée créée précédemment

```
$ certtool --generate-request --load-privkey <example-server.key> --template <example-server.cfg> --outfile <example-server.crq>
```

e. Générer le certificat et le signer avec la clé privée de l'autorité de certification :

```
$ certtool --generate-certificate --load-request <example-server.crq> --load-privkey <example-server.key> --load-ca-certificate <ca.crt> --load-ca-privkey <ca.key> --outfile <example-server.crt>
```

Ressources supplémentaires

- **certtool(1)** et **trust(1)** pages de manuel

3.7. CRÉATION D'UNE CLÉ PRIVÉE ET D'UNE CSR POUR UN CERTIFICAT DE SERVEUR TLS À L'AIDE DE GNUTLS

Pour obtenir le certificat, vous devez d'abord créer une clé privée et une demande de signature de certificat (CSR) pour votre serveur.

Procédure

1. Générez une clé privée sur votre système serveur, par exemple :

```
$ certtool --generate-privkey --sec-param High --outfile <example-server.key>
```

2. Facultatif : Utilisez un éditeur de texte de votre choix pour préparer un fichier de configuration qui simplifie la création de votre CSR, par exemple :

```
$ vim <example_server.cnf>
signing_key
encryption_key
key_agreement

tls_www_server

country = "US"
organization = "Example Organization"
cn = "server.example.com"

dns_name = "example.com"
dns_name = "server.example.com"
ip_address = "192.168.0.1"
ip_address = "::1"
ip_address = "127.0.0.1"
```

3. Créez une RSC à l'aide de la clé privée que vous avez créée précédemment :

```
$ certtool --generate-request --template <example-server.cfg> --load-privkey <example-server.key> --outfile <example-server.crq>
```

Si vous omettez l'option **--template**, l'utilitaire **certool** vous demande des informations supplémentaires, par exemple :

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Generating a PKCS #10 certificate request...
Country name (2 chars): <US>
State or province name: <Washington>
Locality name: <Seattle>
Organization name: <Example Organization>
Organizational unit name:
Common name: <server.example.com>
```

Prochaines étapes

- Soumettez la CSR à l'autorité de certification de votre choix pour signature. Sinon, pour un scénario d'utilisation interne au sein d'un réseau de confiance, utilisez votre autorité de certification privée pour la signature. Voir Section 3.9, « Utilisation d'une autorité de certification privée pour émettre des certificats pour les CSR avec GnuTLS » pour plus d'informations.

Vérification

1. Après avoir obtenu le certificat demandé auprès de l'autorité de certification, vérifiez que les parties lisibles par l'homme du certificat correspondent à vos exigences, par exemple :

```
$ certtool --certificate-info --infile <example-server.crt>
Certificate:
…
        Issuer: CN = Example CA
        Validity
            Not Before: Feb  2 20:27:29 2023 GMT
            Not After : Feb  2 20:27:29 2024 GMT
        Subject: C = US, O = Example Organization, CN = server.example.com
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
…
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment, Key Agreement
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 Subject Alternative Name:
                DNS:example.com, DNS:server.example.com, IP Address:192.168.0.1, IP
…
```

Ressources supplémentaires

- **certtool(1)** page de manuel

3.8. CRÉATION D'UNE CLÉ PRIVÉE ET D'UNE CSR POUR UN CERTIFICAT CLIENT TLS À L'AIDE DE GNUTLS

Pour obtenir le certificat, vous devez d'abord créer une clé privée et une demande de signature de certificat (CSR) pour votre client.

Procédure

1. Générez une clé privée sur votre système client, par exemple :

```
$ certtool --generate-privkey --sec-param High --outfile <example-client.key>
```

2. Facultatif : Utilisez un éditeur de texte de votre choix pour préparer un fichier de configuration qui simplifie la création de votre CSR, par exemple :

```
$ vim <example_client.cnf>
signing_key
encryption_key

tls_www_client

cn = "client.example.com"
email = "client@example.com"
```

3. Créez une RSC à l'aide de la clé privée que vous avez créée précédemment :

```
$ certtool --generate-request --template <example-client.cfg> --load-privkey <example-client.key> --outfile <example-client.crq>
```

Si vous omettez l'option **--template**, l'utilitaire **certtool** vous demande des informations supplémentaires, par exemple :

```
Generating a PKCS #10 certificate request...
Country name (2 chars): <US>
State or province name: <Washington>
Locality name: <Seattle>
Organization name: <Example Organization>
Organizational unit name:
Common name: <server.example.com>
```

Prochaines étapes

- Soumettez la CSR à l'autorité de certification de votre choix pour signature. Sinon, pour un scénario d'utilisation interne au sein d'un réseau de confiance, utilisez votre autorité de certification privée pour la signature. Voir Section 3.9, « Utilisation d'une autorité de certification privée pour émettre des certificats pour les CSR avec GnuTLS » pour plus d'informations.

Vérification

1. Vérifiez que les parties du certificat lisibles par l'homme correspondent à vos exigences, par exemple :

```
$ certtool --certificate-info --infile <example-client.crt>
Certificate:
…
        X509v3 Extended Key Usage:
            TLS Web Client Authentication
        X509v3 Subject Alternative Name:
            email:client@example.com
…
```

Ressources supplémentaires

- **certtool(1)** page de manuel

3.9. UTILISATION D'UNE AUTORITÉ DE CERTIFICATION PRIVÉE POUR ÉMETTRE DES CERTIFICATS POUR LES CSR AVEC GNUTLS

Pour permettre aux systèmes d'établir un canal de communication crypté TLS, une autorité de certification (AC) doit leur fournir des certificats valides. Si vous disposez d'une autorité de certification privée, vous pouvez créer les certificats requis en signant les demandes de signature de certificat (CSR) des systèmes.

Conditions préalables

- Vous avez déjà configuré une autorité de certification privée. Voir Section 3.6, « Création d'une autorité de certification privée à l'aide de GnuTLS » pour plus d'informations.
- Vous avez un fichier contenant un CSR. Vous trouverez un exemple de création de CSR sur Section 3.7, « Création d'une clé privée et d'une CSR pour un certificat de serveur TLS à l'aide de GnuTLS ».

Procédure

1. Facultatif : Utilisez un éditeur de texte de votre choix pour préparer un fichier de configuration GnuTLS afin d'ajouter des extensions aux certificats, par exemple :

```
$ vi <server-extensions.cfg>
honor_crq_extensions
ocsp_uri = "http://ocsp.example.com"
```

2. Utilisez l'utilitaire **certtool** pour créer un certificat basé sur une CSR, par exemple :

```
$ certtool --generate-certificate --load-request <example-server.crq> --load-ca-privkey
<ca.key> --load-ca-certificate <ca.crt> --template <server-extensions.cfg> --outfile
<example-server.crt>
```

Ressources supplémentaires

- **certtool(1)** page de manuel

CHAPITRE 4. UTILISATION DE CERTIFICATS SYSTÈME PARTAGÉS

Le stockage partagé des certificats système permet à NSS, GnuTLS, OpenSSL et Java de partager une source par défaut pour récupérer les ancres des certificats système et les informations de la liste de blocage. Par défaut, le magasin de confiance contient la liste des autorités de certification de Mozilla, y compris la confiance positive et négative. Le système permet de mettre à jour la liste principale des autorités de certification de Mozilla ou de choisir une autre liste de certificats.

4.1. LA BASE DE DONNÉES DE CONFIANCE À L'ÉCHELLE DU SYSTÈME

Dans RHEL, le magasin de confiance consolidé à l'échelle du système est situé dans les répertoires **/etc/pki/ca-trust/** et **/usr/share/pki/ca-trust-source/**. Les paramètres de confiance contenus dans **/usr/share/pki/ca-trust-source/** sont traités avec une priorité inférieure à celle des paramètres contenus dans **/etc/pki/ca-trust/**.

Les fichiers de certificats sont traités en fonction du sous-répertoire dans lequel ils sont installés :

- Les ancres de confiance appartiennent à
  - **/usr/share/pki/ca-trust-source/anchors/** ou
  - **/etc/pki/ca-trust/source/anchors/**.
- Les certificats douteux sont stockés dans le dossier
  - **/usr/share/pki/ca-trust-source/blocklist/** ou
  - **/etc/pki/ca-trust/source/blocklist/**.
- Les certificats dans le format de fichier BEGIN TRUSTED étendu se trouvent dans le dossier
  - **/usr/share/pki/ca-trust-source/** ou
  - **/etc/pki/ca-trust/source/**.

NOTE

Dans un système cryptographique hiérarchique, un point d'ancrage de confiance est une entité faisant autorité que d'autres parties considèrent comme digne de confiance. Dans l'architecture X.509, un certificat racine est une ancre de confiance à partir de laquelle une chaîne de confiance est dérivée. Pour permettre la validation de la chaîne, la partie qui fait confiance doit d'abord avoir accès à l'ancre de confiance.

Ressources supplémentaires

- **update-ca-trust(8)** et **trust(1)** pages de manuel

4.2. AJOUT DE NOUVEAUX CERTIFICATS

Pour reconnaître les applications de votre système avec une nouvelle source de confiance, ajoutez le certificat correspondant au magasin de l'ensemble du système et utilisez la commande **update-ca-trust**.

Conditions préalables

- Le paquetage **ca-certificates** est présent sur le système.

Procédure

1. Pour ajouter un certificat au format simple PEM ou DER à la liste des autorités de certification approuvées par le système, copiez le fichier de certificat dans le répertoire **/usr/share/pki/ca-trust-source/anchors/** ou **/etc/pki/ca-trust/source/anchors/**, par exemple :

```
# cp ~/certificate-trust-examples/Cert-trust-test-ca.pem /usr/share/pki/ca-trust-source/anchors/
```

2. Pour mettre à jour la configuration du magasin de confiance à l'échelle du système, utilisez la commande **update-ca-trust**:

```
# update-ca-trust
```

NOTE

Même si le navigateur Firefox peut utiliser un certificat ajouté sans exécution préalable de la commande **update-ca-trust**, il convient d'exécuter la commande **update-ca-trust** après chaque changement d'autorité de certification. Notez également que les navigateurs, tels que Firefox, Chromium et GNOME Web, mettent en cache des fichiers et que vous devrez peut-être vider le cache de votre navigateur ou le redémarrer pour charger la configuration actuelle des certificats du système.

Ressources supplémentaires

- **update-ca-trust(8)** et **trust(1)** pages de manuel

4.3. GESTION DES CERTIFICATS DU SYSTÈME DE CONFIANCE

La commande **trust** offre un moyen pratique de gérer les certificats dans le magasin de confiance partagé à l'échelle du système.

- Pour lister, extraire, ajouter, supprimer ou modifier les ancres de confiance, utilisez la commande **trust**. Pour consulter l'aide intégrée de cette commande, entrez-la sans arguments ou avec la directive **--help**:

```
$ trust
usage: trust command <args>...

Common trust commands are:
  list            List trust or certificates
  extract         Extract certificates and trust
  extract-compat  Extract trust compatibility bundles
  anchor          Add, remove, change trust anchors
  dump            Dump trust objects in internal format

See 'trust <command> --help' for more information
```

- Pour dresser la liste de tous les certificats et ancres de confiance du système, utilisez la commande **trust list**:

```
$ trust list
pkcs11:id=%d2%87%b4%e3%df%37%27%93%55%f6%56%ea%81%e5%36%cc%8c%1e%3
f%bd;type=cert
    type: certificate
    label: ACCVRAIZ1
    trust: anchor
    category: authority

pkcs11:id=%a6%b3%e1%2b%2b%49%b6%d7%73%a1%aa%94%f5%01%e7%73%65%4c%
ac%50;type=cert
    type: certificate
    label: ACEDICOM Root
    trust: anchor
    category: authority
...
```

- Pour enregistrer une ancre de confiance dans le magasin de confiance du système, utilisez la sous-commande **trust anchor** et indiquez un chemin d'accès à un certificat. Remplacez *<path.to/certificate.crt>* par le chemin d'accès à votre certificat et son nom de fichier :

```
# trust anchor <path.to/certificate.crt>
```

- Pour supprimer un certificat, utilisez soit un chemin d'accès à un certificat, soit l'identifiant d'un certificat :

```
# trust anchor --remove <path.to/certificate.crt>
# trust anchor --remove "pkcs11:id=<%AA%BB%CC%DD%EE>;type=cert"
```

Ressources supplémentaires

- Toutes les sous-commandes des commandes **trust** offrent une aide intégrée détaillée, par exemple :

```
$ trust list --help
usage: trust list --filter=<what>

  --filter=<what>     filter of what to export
                      ca-anchors         certificate anchors
...
  --purpose=<usage>   limit to certificates usable for the purpose
                      server-auth        for authenticating servers
...
```

Ressources supplémentaires

- **update-ca-trust(8)** et **trust(1)** pages de manuel

CHAPITRE 5. PLANIFICATION ET MISE EN ŒUVRE DE TLS

TLS (Transport Layer Security) est un protocole cryptographique utilisé pour sécuriser les communications réseau. Lorsque l'on renforce les paramètres de sécurité du système en configurant les protocoles d'échange de clés, les méthodes d'authentification et les algorithmes de chiffrement préférés, il faut garder à l'esprit que plus l'éventail des clients pris en charge est large, plus la sécurité est faible. Inversement, des paramètres de sécurité stricts entraînent une compatibilité limitée avec les clients, ce qui peut avoir pour effet de bloquer l'accès au système pour certains utilisateurs. Veillez à cibler la configuration disponible la plus stricte et à ne l'assouplir que lorsque cela est nécessaire pour des raisons de compatibilité.

5.1. PROTOCOLES SSL ET TLS

Le protocole Secure Sockets Layer (SSL) a été développé à l'origine par Netscape Corporation pour fournir un mécanisme de communication sécurisée sur Internet. Par la suite, le protocole a été adopté par l'Internet Engineering Task Force (IETF) et rebaptisé Transport Layer Security (TLS).

Le protocole TLS se situe entre une couche de protocole d'application et une couche de transport fiable, telle que TCP/IP. Il est indépendant du protocole d'application et peut donc être superposé à de nombreux protocoles différents, par exemple : HTTP, FTP, SMTP, etc : HTTP, FTP, SMTP, etc.

Version du protocole	Recommandation d'utilisation
SSL v2	Ne pas utiliser. Présente de sérieuses failles de sécurité. Retiré des bibliothèques cryptographiques de base depuis RHEL 7.
SSL v3	Ne pas utiliser. Présente de sérieuses failles de sécurité. Retiré des bibliothèques cryptographiques principales depuis RHEL 8.
TLS 1.0	Son utilisation n'est pas recommandée. Présente des problèmes connus qui ne peuvent être atténués de manière à garantir l'interopérabilité, et ne prend pas en charge les suites de chiffrement modernes. Dans RHEL 9, désactivé dans toutes les politiques cryptographiques.
TLS 1.1	À utiliser à des fins d'interopérabilité, le cas échéant. Ne prend pas en charge les suites de chiffrement modernes. Dans RHEL 9, désactivé dans toutes les politiques cryptographiques.
TLS 1.2	Prend en charge les suites de chiffrement modernes AEAD. Cette version est activée dans toutes les politiques cryptographiques du système, mais des parties optionnelles de ce protocole contiennent des vulnérabilités et TLS 1.2 autorise également des algorithmes obsolètes.
TLS 1.3	Version recommandée. TLS 1.3 supprime les options problématiques connues, fournit une confidentialité supplémentaire en chiffrant une plus grande partie de la négociation et peut être plus rapide grâce à l'utilisation d'algorithmes cryptographiques modernes plus efficaces. TLS 1.3 est également activé dans toutes les stratégies cryptographiques du système.

Ressources supplémentaires

- IETF : Protocole de sécurité de la couche transport (TLS) version 1.3 .

5.2. CONSIDÉRATIONS DE SÉCURITÉ POUR TLS DANS RHEL 9

Dans RHEL 9, la configuration de TLS est effectuée à l'aide du mécanisme de politiques cryptographiques à l'échelle du système. Les versions de TLS inférieures à 1.2 ne sont plus prises en charge. **DEFAULT** les stratégies cryptographiques à l'échelle du système, **FUTURE** et **LEGACY** n'autorisent que TLS 1.2 et 1.3. Pour plus d'informations, voir Utilisation des stratégies cryptographiques à l'échelle du système.

Les paramètres par défaut fournis par les bibliothèques incluses dans RHEL 9 sont suffisamment sûrs pour la plupart des déploiements. Les implémentations TLS utilisent des algorithmes sécurisés dans la mesure du possible, tout en n'empêchant pas les connexions depuis ou vers les clients ou serveurs existants. Appliquez des paramètres renforcés dans les environnements soumis à des exigences de sécurité strictes, où les clients ou serveurs existants qui ne prennent pas en charge les algorithmes ou protocoles sécurisés ne sont pas censés se connecter ou ne sont pas autorisés à le faire.

La façon la plus simple de renforcer votre configuration TLS est de passer le niveau de politique cryptographique du système à **FUTURE** à l'aide de la commande **update-crypto-policies --set FUTURE**.

AVERTISSEMENT

Les algorithmes désactivés pour la politique cryptographique **LEGACY** ne sont pas conformes à la vision de Red Hat de la sécurité de RHEL 9, et leurs propriétés de sécurité ne sont pas fiables. Envisagez d'abandonner l'utilisation de ces algorithmes au lieu de les réactiver. Si vous décidez de les réactiver, par exemple pour des raisons d'interopérabilité avec du matériel ancien, considérez-les comme non sécurisés et appliquez des mesures de protection supplémentaires, telles que l'isolation de leurs interactions réseau sur des segments de réseau distincts. Ne les utilisez pas sur des réseaux publics.

Si vous décidez de ne pas suivre les politiques cryptographiques du système RHEL ou de créer des politiques cryptographiques personnalisées adaptées à votre configuration, utilisez les recommandations suivantes pour les protocoles, les suites de chiffrement et les longueurs de clé préférés dans votre configuration personnalisée :

5.2.1. Protocoles

La dernière version de TLS offre le meilleur mécanisme de sécurité. TLS 1.2 est désormais la version minimale, même lorsque l'on utilise la politique cryptographique **LEGACY**. Il est possible de réactiver les versions antérieures du protocole en renonçant aux politiques cryptographiques ou en fournissant une politique personnalisée, mais la configuration qui en résultera ne sera pas prise en charge.

Notez que même si RHEL 9 prend en charge la version 1.3 de TLS, toutes les fonctionnalités de ce protocole ne sont pas entièrement prises en charge par les composants de RHEL 9. Par exemple, la fonctionnalité 0-RTT (Zero Round Trip Time), qui réduit la latence de la connexion, n'est pas encore totalement prise en charge par le serveur web Apache.

5.2.2. Suites de chiffrement

Les suites de chiffrement modernes et plus sûres doivent être préférées aux suites anciennes et peu sûres. Désactivez toujours l'utilisation des suites de chiffrement eNULL et aNULL, qui n'offrent aucun chiffrement ni aucune authentification. Dans la mesure du possible, les suites de chiffrement basées sur RC4 ou HMAC-MD5, qui présentent de sérieuses lacunes, doivent également être désactivées. Il en va de même pour les suites de chiffrement dites d'exportation, qui ont été rendues intentionnellement plus faibles et sont donc faciles à casser.

Bien qu'elles ne soient pas immédiatement non sécurisées, les suites de chiffrement qui offrent moins de 128 bits de sécurité ne devraient pas être envisagées en raison de leur courte durée de vie. Les algorithmes qui utilisent 128 bits de sécurité ou plus peuvent être considérés comme inviolables pendant au moins plusieurs années et sont donc fortement recommandés. Il convient de noter que si les chiffrements 3DES annoncent l'utilisation de 168 bits, ils offrent en réalité une sécurité de 112 bits.

Préférez toujours les suites de chiffrement qui prennent en charge le secret de transmission (parfait) (PFS), qui garantit la confidentialité des données chiffrées même en cas de compromission de la clé du serveur. Cela exclut l'échange de clés rapide RSA, mais permet d'utiliser ECDHE et DHE. Parmi les deux, ECDHE est le plus rapide et donc le meilleur choix.

Vous devriez également préférer les chiffrements AEAD, tels que AES-GCM, aux chiffrements en mode CBC, car ils ne sont pas vulnérables aux attaques de l'oracle de remplissage. En outre, dans de nombreux cas, AES-GCM est plus rapide qu'AES en mode CBC, en particulier lorsque le matériel dispose d'accélérateurs cryptographiques pour AES.

Notez également que lorsque vous utilisez l'échange de clés ECDHE avec des certificats ECDSA, la transaction est encore plus rapide qu'un échange de clés RSA pur. Pour prendre en charge les anciens clients, vous pouvez installer deux paires de certificats et de clés sur un serveur : l'une avec des clés ECDSA (pour les nouveaux clients) et l'autre avec des clés RSA (pour les anciens clients).

5.2.3. Longueur de la clé publique

Lorsque vous utilisez des clés RSA, préférez toujours des longueurs de clés d'au moins 3072 bits signées par au moins SHA-256, ce qui est suffisamment important pour assurer une sécurité réelle de 128 bits.

AVERTISSEMENT

La sécurité de votre système est aussi forte que le maillon le plus faible de la chaîne. Par exemple, un cryptogramme puissant ne garantit pas à lui seul une bonne sécurité. Les clés et les certificats sont tout aussi importants, de même que les fonctions de hachage et les clés utilisées par l'autorité de certification (AC) pour signer vos clés.

5.3. DURCISSEMENT DE LA CONFIGURATION TLS DANS LES APPLICATIONS

Dans RHEL, les règles cryptographiques applicables à l'ensemble du système constituent un moyen pratique de s'assurer que les applications utilisant des bibliothèques cryptographiques n'autorisent pas les protocoles, les algorithmes de chiffrement ou les algorithmes non sécurisés connus.

Si vous souhaitez renforcer votre configuration TLS avec vos paramètres cryptographiques personnalisés, vous pouvez utiliser les options de configuration cryptographique décrites dans cette section et remplacer les stratégies cryptographiques du système par le minimum requis.

Quelle que soit la configuration que vous choisissez d'utiliser, veillez toujours à ce que votre application serveur applique *server-side cipher order*, de sorte que la suite de chiffrement à utiliser soit déterminée par l'ordre que vous avez configuré.

5.3.1. Configuration du serveur HTTP Apache pour l'utilisation de TLS

Le site **Apache HTTP Server** peut utiliser les bibliothèques **OpenSSL** et **NSS** pour ses besoins en TLS. RHEL 9 fournit la fonctionnalité **mod_ssl** par le biais de paquets éponymes :

```
# dnf install mod_ssl
```

Le paquet **mod_ssl** installe le fichier de configuration **/etc/httpd/conf.d/ssl.conf**, qui peut être utilisé pour modifier les paramètres relatifs à TLS du paquet **Apache HTTP Server**.

Installez le paquetage **httpd-manual** pour obtenir la documentation complète de **Apache HTTP Server**, y compris la configuration TLS. Les directives disponibles dans le fichier de configuration **/etc/httpd/conf.d/ssl.conf** sont décrites en détail dans le fichier **/usr/share/httpd/manual/mod/mod_ssl.html**. Des exemples de différents paramètres sont décrits dans le fichier **/usr/share/httpd/manual/ssl/ssl_howto.html**.

Lorsque vous modifiez les paramètres du fichier de configuration **/etc/httpd/conf.d/ssl.conf**, veillez à tenir compte au minimum des trois directives suivantes :

SSLProtocol

Utilisez cette directive pour spécifier la version de TLS ou SSL que vous souhaitez autoriser.

SSLCipherSuite

Utilisez cette directive pour spécifier votre suite de chiffrement préférée ou pour désactiver ceux que vous souhaitez interdire.

SSLHonorCipherOrder

Décommentez et mettez cette directive à **on** pour vous assurer que les clients qui se connectent respectent l'ordre des algorithmes de chiffrement que vous avez spécifié.

Par exemple, pour utiliser uniquement les protocoles TLS 1.2 et 1.3 :

```
SSLProtocol             all -SSLv3 -TLSv1 -TLSv1.1
```

Pour plus d'informations, voir le chapitre Configurer le cryptage TLS sur un serveur HTTP Apache dans le document Déployer des serveurs web et des proxys inversés.

5.3.2. Configurer le serveur HTTP et proxy Nginx pour utiliser TLS

Pour activer la prise en charge de TLS 1.3 dans **Nginx**, ajoutez la valeur **TLSv1.3** à l'option **ssl_protocols** dans la section **server** du fichier de configuration **/etc/nginx/nginx.conf**:

```
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    ....
    ssl_protocols TLSv1.2 TLSv1.3;
```

```
    ssl_ciphers
    ....
}
```

Voir le chapitre Ajouter le cryptage TLS à un serveur web Nginx dans le document Déployer des serveurs web et des proxys inversés pour plus d'informations.

5.3.3. Configurer le serveur de messagerie Dovecot pour utiliser TLS

Pour configurer votre installation du serveur de messagerie **Dovecot** afin d'utiliser TLS, modifiez le fichier de configuration **/etc/dovecot/conf.d/10-ssl.conf**. Vous trouverez une explication de certaines des directives de configuration de base disponibles dans ce fichier dans le fichier **/usr/share/doc/dovecot/wiki/SSL.DovecotConfiguration.txt**, qui est installé avec l'installation standard de **Dovecot**.

Lorsque vous modifiez les paramètres du fichier de configuration **/etc/dovecot/conf.d/10-ssl.conf**, veillez à tenir compte au minimum des trois directives suivantes :

ssl_protocols

Utilisez cette directive pour spécifier la version de TLS ou SSL que vous souhaitez autoriser ou désactiver.

ssl_cipher_list

Utilisez cette directive pour spécifier vos suites de chiffrement préférées ou désactiver celles que vous souhaitez interdire.

ssl_prefer_server_ciphers

Décommentez et mettez cette directive à **yes** pour vous assurer que les clients qui se connectent respectent l'ordre des algorithmes de chiffrement que vous avez spécifié.

Par exemple, la ligne suivante dans **/etc/dovecot/conf.d/10-ssl.conf** n'autorise que TLS 1.1 et les versions ultérieures :

```
ssl_protocols = !SSLv2 !SSLv3 !TLSv1
```

Ressources supplémentaires

- Déploiement de serveurs web et de serveurs mandataires (reverse proxies)
- **config(5)** et **ciphers(1)**.
- Recommandations pour l'utilisation sécurisée de la sécurité de la couche transport (TLS) et de la sécurité de la couche transport du datagramme (DTLS).
- Générateur de configuration SSL de Mozilla .
- Test du serveur SSL.

CHAPITRE 6. CONFIGURATION D'UN VPN AVEC IPSEC

Dans RHEL 9, un réseau privé virtuel (VPN) peut être configuré à l'aide du protocole **IPsec**, qui est pris en charge par l'application **Libreswan**.

6.1. LIBRESWAN COMME IMPLÉMENTATION D'UN VPN IPSEC

Dans RHEL, un réseau privé virtuel (VPN) peut être configuré en utilisant le protocole IPsec, qui est pris en charge par l'application Libreswan. Libreswan est une continuation de l'application Openswan, et de nombreux exemples de la documentation Openswan sont interchangeables avec Libreswan.

Le protocole IPsec pour un VPN est configuré à l'aide du protocole IKE (Internet Key Exchange). Les termes IPsec et IKE sont utilisés de manière interchangeable. Un VPN IPsec est également appelé VPN IKE, VPN IKEv2, VPN XAUTH, VPN Cisco ou VPN IKE/IPsec. Une variante d'un VPN IPsec qui utilise également le Layer 2 Tunneling Protocol (L2TP) est généralement appelée VPN L2TP/IPsec, qui nécessite le paquetage **xl2tpd** fourni par le référentiel **optional**.

Libreswan est une implémentation IKE en espace utilisateur à code source ouvert. IKE v1 et v2 sont implémentés en tant que démon au niveau de l'utilisateur. Le protocole IKE est également crypté. Le protocole IPsec est implémenté par le noyau Linux, et Libreswan configure le noyau pour ajouter et supprimer des configurations de tunnel VPN.

Le protocole IKE utilise les ports UDP 500 et 4500. Le protocole IPsec se compose de deux protocoles :

- Encapsulated Security Payload (ESP), qui porte le numéro de protocole 50.
- L'en-tête authentifié (AH), qui porte le numéro de protocole 51.

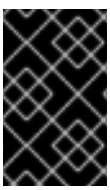
L'utilisation du protocole AH n'est pas recommandée. Il est recommandé aux utilisateurs du protocole AH de migrer vers le protocole ESP à chiffrement nul.

Le protocole IPsec propose deux modes de fonctionnement :

- Mode tunnel (par défaut)
- Mode de transport.

Vous pouvez configurer le noyau avec IPsec sans IKE. C'est ce qu'on appelle *manual keying*. Vous pouvez également configurer une clé manuelle à l'aide des commandes **ip xfrm**, mais cela est fortement déconseillé pour des raisons de sécurité. Libreswan communique avec le noyau Linux à l'aide de l'interface Netlink. Le noyau effectue le cryptage et le décryptage des paquets.

Libreswan utilise la bibliothèque cryptographique NSS (Network Security Services). NSS est certifié pour une utilisation avec la publication 140-2 de *Federal Information Processing Standard* (*FIPS*).



IMPORTANT

Les VPN IKE/IPsec, mis en œuvre par Libreswan et le noyau Linux, sont la seule technologie VPN dont l'utilisation est recommandée dans RHEL. N'utilisez aucune autre technologie VPN sans en comprendre les risques.

Dans RHEL, Libreswan suit **system-wide cryptographic policies** par défaut. Cela garantit que Libreswan utilise des paramètres sécurisés pour les modèles de menace actuels, y compris IKEv2 comme protocole par défaut. Pour plus d'informations, reportez-vous à la section Utilisation de stratégies cryptographiques à l'échelle du système.

Libreswan n'utilise pas les termes " source " et " destination " ou " serveur " et " client " car les protocoles IKE/IPsec sont des protocoles d'égal à égal. Il utilise plutôt les termes "gauche" et "droite" pour désigner les points finaux (les hôtes). Cela vous permet également d'utiliser la même configuration sur les deux points d'extrémité dans la plupart des cas. Cependant, les administrateurs choisissent généralement de toujours utiliser "left" pour l'hôte local et "right" pour l'hôte distant.

Les options **leftid** et **rightid** servent à identifier les hôtes respectifs dans le processus d'authentification. Voir la page de manuel **ipsec.conf(5)** pour plus d'informations.

6.2. MÉTHODES D'AUTHENTIFICATION DANS LIBRESWAN

Libreswan prend en charge plusieurs méthodes d'authentification, chacune correspondant à un scénario différent.

Clé pré-partagée (PSK)

Pre-Shared Key (PSK) est la méthode d'authentification la plus simple. Pour des raisons de sécurité, n'utilisez pas de PSK plus courts que 64 caractères aléatoires. En mode FIPS, les PSK doivent respecter une exigence de force minimale en fonction de l'algorithme d'intégrité utilisé. Vous pouvez définir le PSK en utilisant la connexion **authby=secret**.

Clés RSA brutes

Raw RSA keys sont généralement utilisés pour les configurations IPsec statiques d'hôte à hôte ou de sous-réseau à sous-réseau. Chaque hôte est configuré manuellement avec les clés publiques RSA de tous les autres hôtes, et Libreswan met en place un tunnel IPsec entre chaque paire d'hôtes. Cette méthode n'est pas adaptée à un grand nombre d'hôtes.

Vous pouvez générer une clé RSA brute sur un hôte à l'aide de la commande **ipsec newhostkey**. Vous pouvez dresser la liste des clés générées à l'aide de la commande **ipsec showhostkey**. La ligne **leftrsasigkey=** est nécessaire pour les configurations de connexion qui utilisent des clés CKA ID. Utilisez l'option de connexion **authby=rsasig** pour les clés RSA brutes.

Certificats X.509

X.509 certificates sont couramment utilisés pour les déploiements à grande échelle avec des hôtes qui se connectent à une passerelle IPsec commune. Une autorité centrale *certificate authority* (CA) signe les certificats RSA pour les hôtes ou les utilisateurs. Cette autorité centrale est chargée de relayer la confiance, y compris les révocations d'hôtes ou d'utilisateurs individuels.

Par exemple, vous pouvez générer des certificats X.509 à l'aide de la commande **openssl** et de la commande NSS **certutil**. Libreswan lit les certificats d'utilisateur à partir de la base de données NSS en utilisant le pseudonyme des certificats dans l'option de configuration **leftcert=**. Vous devez donc fournir un pseudonyme lors de la création d'un certificat.

Si vous utilisez un certificat d'autorité de certification personnalisé, vous devez l'importer dans la base de données NSS (Network Security Services). Vous pouvez importer n'importe quel certificat au format PKCS #12 dans la base de données Libreswan NSS à l'aide de la commande **ipsec import**.

AVERTISSEMENT

Libreswan requiert un identifiant de pair IKE (Internet Key Exchange) comme nom alternatif de sujet (SAN) pour chaque certificat de pair, comme décrit dans la section 3.1 de la RFC 4945 . La désactivation de cette vérification en modifiant l'option **require-id-on-certificated=** peut rendre le système vulnérable aux attaques de type man-in-the-middle.

Utilisez l'option de connexion **authby=rsasig** pour l'authentification basée sur des certificats X.509 utilisant RSA avec SHA-2. Vous pouvez la limiter davantage pour les signatures numériques ECDSA utilisant SHA-2 en définissant **authby=** sur **ecdsa** et pour l'authentification basée sur les signatures numériques RSA Probabilistic Signature Scheme (RSASSA-PSS) avec SHA-2 via **authby=rsa-sha2**. La valeur par défaut est **authby=rsasig,ecdsa**.

Les certificats et les méthodes de signature **authby=** doivent correspondre. Cela permet d'améliorer l'interopérabilité et de préserver l'authentification dans un seul système de signature numérique.

Authentification NULL

NULL authentication est utilisé pour obtenir un chiffrement du maillage sans authentification. Elle protège contre les attaques passives mais pas contre les attaques actives. Toutefois, comme IKEv2 autorise les méthodes d'authentification asymétriques, l'authentification NULL peut également être utilisée pour l'IPsec opportuniste à l'échelle de l'internet. Dans ce modèle, les clients authentifient le serveur, mais les serveurs n'authentifient pas le client. Ce modèle est similaire à celui des sites web sécurisés utilisant TLS. Utilisez **authby=null** pour l'authentification NULL.

Protection contre les ordinateurs quantiques

Outre les méthodes d'authentification mentionnées précédemment, vous pouvez utiliser la méthode *Post-quantum Pre-shared Key* (PPK) pour vous protéger contre les attaques éventuelles d'ordinateurs quantiques. Les clients individuels ou les groupes de clients peuvent utiliser leur propre PPK en spécifiant un ID PPK qui correspond à une clé pré-partagée configurée hors bande.

L'utilisation d'IKEv1 avec des clés pré-partagées offre une protection contre les attaquants quantiques. La nouvelle conception d'IKEv2 n'offre pas cette protection de manière native. Libreswan propose l'utilisation de *Post-quantum Pre-shared Key* (PPK) pour protéger les connexions IKEv2 contre les attaques quantiques.

Pour activer la prise en charge optionnelle du PPK, ajoutez **ppk=yes** à la définition de la connexion. Pour exiger le PPK, ajoutez **ppk=insist**. Ensuite, chaque client peut recevoir un identifiant PPK avec une valeur secrète qui est communiquée hors bande (et de préférence sans risque quantique). Les PPK doivent être très aléatoires et ne pas être basés sur des mots du dictionnaire. L'ID PPK et les données PPK sont stockées sur **ipsec.secrets**, par exemple :

```
@west @east : PPKS "user1" "thestringismeanttobearandomstr"
```

L'option **PPKS** se réfère aux PPK statiques. Cette fonction expérimentale utilise des PPK dynamiques basés sur un pavé à usage unique. À chaque connexion, une nouvelle partie du tampon à usage unique est utilisée comme PPK. Lorsqu'elle est utilisée, la partie du PPK dynamique contenue dans le fichier est écrasée par des zéros afin d'empêcher sa réutilisation. S'il n'y a plus de matériel de tampon à usage unique, la connexion échoue. Voir la page de manuel **ipsec.secrets(5)** pour plus d'informations.

AVERTISSEMENT

L'implémentation des PPK dynamiques est fournie en tant qu'aperçu technologique non pris en charge. À utiliser avec précaution.

6.3. INSTALLATION DE LIBRESWAN

Cette procédure décrit les étapes d'installation et de démarrage de l'implémentation du VPN IPsec/IKE de Libreswan.

Conditions préalables

- Le référentiel **AppStream** est activé.

Procédure

1. Installez les paquets **libreswan**:

```
# dnf install libreswan
```

2. Si vous réinstallez Libreswan, supprimez ses anciens fichiers de base de données et créez une nouvelle base de données :

```
# systemctl stop ipsec
# rm /var/lib/ipsec/nss/*db
# ipsec initnss
```

3. Démarrez le service **ipsec** et activez le service pour qu'il soit démarré automatiquement au démarrage :

```
# systemctl enable ipsec --now
```

4. Configurez le pare-feu pour qu'il autorise les ports 500 et 4500/UDP pour les protocoles IKE, ESP et AH en ajoutant le service **ipsec**:

```
# firewall-cmd --add-service="ipsec"
# firewall-cmd --runtime-to-permanent
```

6.4. CRÉATION D'UN VPN D'HÔTE À HÔTE

Pour configurer [application]Libreswan afin de créer un VPN IPsec d'hôte à hôte entre deux hôtes appelés *left* et *right* en utilisant l'authentification par clés RSA brutes, entrez les commandes suivantes sur les deux hôtes :

Conditions préalables

- Libreswan est installé et le service **ipsec** est démarré sur chaque nœud.

Procédure

1. Générer une paire de clés RSA brutes sur chaque hôte :

```
# ipsec newhostkey
```

2. L'étape précédente a renvoyé la clé générée à **ckaid**. Utilisez cette **ckaid** avec la commande suivante sur *left*, par exemple :

```
# ipsec showhostkey --left --ckaid 2d3ea57b61c9419dfd6cf43a1eb6cb306c0e857d
```

   La sortie de la commande précédente a généré la ligne **leftrsasigkey=** nécessaire à la configuration. Faites de même sur le second hôte (*right*) :

```
# ipsec showhostkey --right --ckaid a9e1f6ce9ecd3608c24e8f701318383f41798f03
```

3. Dans le répertoire **/etc/ipsec.d/**, créez un nouveau fichier **my_host-to-host.conf**. Inscrivez dans le nouveau fichier les clés d'hôte RSA provenant de la sortie des commandes **ipsec showhostkey** de l'étape précédente. Par exemple :

```
conn mytunnel
    leftid=@west
    left=192.1.2.23
    leftrsasigkey=0sAQOrlo+hOafUZDlCQmXFrje/oZm [...] W2n417C/4urYHQkCvuIQ==
    rightid=@east
    right=192.1.2.45
    rightrsasigkey=0sAQO3fwC6nSSGgt64DWiYZzuHbc4 [...] D/v8t5YTQ==
    authby=rsasig
```

4. Après l'importation des clés, redémarrez le service **ipsec**:

```
# systemctl restart ipsec
```

5. Charger la connexion :

```
# ipsec auto --add mytunnel
```

6. Établir le tunnel :

```
# ipsec auto --up mytunnel
```

7. Pour démarrer automatiquement le tunnel lorsque le service **ipsec** est démarré, ajoutez la ligne suivante à la définition de la connexion :

```
auto=start
```

6.5. CONFIGURATION D'UN VPN SITE À SITE

Pour créer un VPN IPsec de site à site, en joignant deux réseaux, un tunnel IPsec est créé entre les deux hôtes. Les hôtes agissent donc comme des points d'extrémité, qui sont configurés pour permettre au trafic d'un ou de plusieurs sous-réseaux de passer. On peut donc considérer les hôtes comme des passerelles vers la partie distante du réseau.

La configuration du VPN site à site ne diffère de celle du VPN hôte à hôte que par le fait qu'un ou plusieurs réseaux ou sous-réseaux doivent être spécifiés dans le fichier de configuration.

Conditions préalables

- Un VPN d'hôte à hôte est déjà configuré.

Procédure

1. Copiez le fichier contenant la configuration de votre VPN hôte à hôte dans un nouveau fichier, par exemple :

```
# cp /etc/ipsec.d/my_host-to-host.conf /etc/ipsec.d/my_site-to-site.conf
```

2. Ajoutez la configuration du sous-réseau au fichier créé à l'étape précédente, par exemple :

```
conn mysubnet
    also=mytunnel
    leftsubnet=192.0.1.0/24
    rightsubnet=192.0.2.0/24
    auto=start

conn mysubnet6
    also=mytunnel
    leftsubnet=2001:db8:0:1::/64
    rightsubnet=2001:db8:0:2::/64
    auto=start

# the following part of the configuration file is the same for both host-to-host and site-to-site connections:

conn mytunnel
   leftid=@west
   left=192.1.2.23
   leftrsasigkey=0sAQOrlo+hOafUZDlCQmXFrje/oZm [...] W2n417C/4urYHQkCvuIQ==
   rightid=@east
   right=192.1.2.45
   rightrsasigkey=0sAQO3fwC6nSSGgt64DWiYZzuHbc4 [...] D/v8t5YTQ==
   authby=rsasig
```

6.6. CONFIGURATION D'UN VPN D'ACCÈS À DISTANCE

Les guerriers de la route sont des utilisateurs itinérants disposant de clients mobiles et d'une adresse IP attribuée de manière dynamique. Les clients mobiles s'authentifient à l'aide de certificats X.509.

L'exemple suivant montre la configuration pour **IKEv2** et évite d'utiliser le protocole XAUTH de **IKEv1**.

Sur le serveur :

```
conn roadwarriors
   ikev2=insist
   # support (roaming) MOBIKE clients (RFC 4555)
   mobike=yes
   fragmentation=yes
```

```
    left=1.2.3.4
    # if access to the LAN is given, enable this, otherwise use 0.0.0.0/0
    # leftsubnet=10.10.0.0/16
    leftsubnet=0.0.0.0/0
    leftcert=gw.example.com
    leftid=%fromcert
    leftxauthserver=yes
    leftmodecfgserver=yes
    right=%any
    # trust our own Certificate Agency
    rightca=%same
    # pick an IP address pool to assign to remote users
    # 100.64.0.0/16 prevents RFC1918 clashes when remote users are behind NAT
    rightaddresspool=100.64.13.100-100.64.13.254
    # if you want remote clients to use some local DNS zones and servers
    modecfgdns="1.2.3.4, 5.6.7.8"
    modecfgdomains="internal.company.com, corp"
    rightxauthclient=yes
    rightmodecfgclient=yes
    authby=rsasig
    # optionally, run the client X.509 ID through pam to allow or deny client
    # pam-authorize=yes
    # load connection, do not initiate
    auto=add
    # kill vanished roadwarriors
    dpddelay=1m
    dpdtimeout=5m
    dpdaction=clear
```

Sur le client mobile, l'appareil du guerrier de la route, utilisez une légère variation de la configuration précédente :

```
conn to-vpn-server
    ikev2=insist
    # pick up our dynamic IP
    left=%defaultroute
    leftsubnet=0.0.0.0/0
    leftcert=myname.example.com
    leftid=%fromcert
    leftmodecfgclient=yes
    # right can also be a DNS hostname
    right=1.2.3.4
    # if access to the remote LAN is required, enable this, otherwise use 0.0.0.0/0
    # rightsubnet=10.10.0.0/16
    rightsubnet=0.0.0.0/0
    fragmentation=yes
    # trust our own Certificate Agency
    rightca=%same
    authby=rsasig
    # allow narrowing to the server's suggested assigned IP and remote subnet
    narrowing=yes
    # support (roaming) MOBIKE clients (RFC 4555)
    mobike=yes
    # initiate connection
    auto=start
```

6.7. CONFIGURATION D'UN VPN MAILLÉ

Un réseau VPN maillé, également connu sous le nom de VPN *any-to-any*, est un réseau dont tous les nœuds communiquent à l'aide d'IPsec. La configuration prévoit des exceptions pour les nœuds qui ne peuvent pas utiliser IPsec. Le réseau VPN maillé peut être configuré de deux manières :

- Pour exiger IPsec.
- Pour privilégier IPsec tout en autorisant un retour à la communication en texte clair.

L'authentification entre les nœuds peut être basée sur des certificats X.509 ou sur des extensions de sécurité DNS (DNSSEC).

La procédure suivante utilise des certificats X.509. Ces certificats peuvent être générés à l'aide de n'importe quel système de gestion d'autorité de certification (CA), tel que le système de certificats Dogtag. Dogtag suppose que les certificats de chaque nœud sont disponibles au format PKCS #12 (fichiers .p12), qui contient la clé privée, le certificat du nœud et le certificat de l'autorité de certification racine utilisé pour valider les certificats X.509 des autres nœuds.

Chaque nœud a une configuration identique à l'exception de son certificat X.509. Cela permet d'ajouter de nouveaux nœuds sans reconfigurer les nœuds existants du réseau. Les fichiers PKCS #12 requièrent un "nom convivial", pour lequel nous utilisons le nom "nœud" afin que les fichiers de configuration faisant référence au nom convivial soient identiques pour tous les nœuds.

Conditions préalables

- Libreswan est installé et le service **ipsec** est démarré sur chaque nœud.

Procédure

1. Sur chaque nœud, importer les fichiers PKCS #12. Cette étape nécessite le mot de passe utilisé pour générer les fichiers PKCS #12 :

```
# ipsec import nodeXXX.p12
```

2. Créez les trois définitions de connexion suivantes pour les profils **IPsec required** (privé), **IPsec optional** (privé ou clair) et **No IPsec** (clair) :

```
# cat /etc/ipsec.d/mesh.conf
conn clear
 auto=ondemand
 type=passthrough
 authby=never
 left=%defaultroute
 right=%group

conn private
 auto=ondemand
 type=transport
 authby=rsasig
 failureshunt=drop
 negotiationshunt=drop
 # left
 left=%defaultroute
 leftcert=nodeXXXX
```

```
 leftid=%fromcert
       leftrsasigkey=%cert
 # right
 rightrsasigkey=%cert
 rightid=%fromcert
 right=%opportunisticgroup

conn private-or-clear
 auto=ondemand
 type=transport
 authby=rsasig
 failureshunt=passthrough
 negotiationshunt=passthrough
 # left
 left=%defaultroute
 leftcert=nodeXXXX
 leftid=%fromcert
       leftrsasigkey=%cert
 # right
 rightrsasigkey=%cert
 rightid=%fromcert
 right=%opportunisticgroup
```

3. Ajoutez l'adresse IP du réseau dans la catégorie appropriée. Par exemple, si tous les nœuds résident dans le réseau 10.15.0.0/16 et que tous les nœuds doivent utiliser le cryptage IPsec :

```
# echo "10.15.0.0/16" >> /etc/ipsec.d/policies/private
```

4. Pour permettre à certains nœuds, par exemple 10.15.34.0/24, de fonctionner avec et sans IPsec, ajoutez ces nœuds au groupe privé ou clair en utilisant :

```
# echo "10.15.34.0/24" >> /etc/ipsec.d/policies/private-or-clear
```

5. Pour définir un hôte, par exemple 10.15.1.2, qui n'est pas capable d'IPsec dans le groupe clair, utilisez :

```
# echo "10.15.1.2/32" >> /etc/ipsec.d/policies/clear
```

   Les fichiers du répertoire **/etc/ipsec.d/policies** peuvent être créés à partir d'un modèle pour chaque nouveau nœud, ou peuvent être approvisionnés à l'aide de Puppet ou Ansible.

   Notez que chaque nœud a la même liste d'exceptions ou des attentes différentes en matière de flux de trafic. Deux nœuds peuvent donc ne pas être en mesure de communiquer parce que l'un d'eux nécessite IPsec et que l'autre ne peut pas l'utiliser.

6. Redémarrez le nœud pour l'ajouter au maillage configuré :

```
# systemctl restart ipsec
```

7. Une fois l'ajout de nœuds terminé, une commande **ping** suffit pour ouvrir un tunnel IPsec. Pour savoir quels tunnels un nœud a ouvert :

```
# ipsec trafficstatus
```

6.8. DÉPLOYER UN VPN IPSEC CONFORME AUX NORMES FIPS

Utilisez cette procédure pour déployer une solution VPN IPsec conforme aux normes FIPS basée sur Libreswan. Les étapes suivantes vous permettent également d'identifier les algorithmes cryptographiques disponibles et ceux qui sont désactivés pour Libreswan en mode FIPS.

Conditions préalables

- Le référentiel **AppStream** est activé.

Procédure

1. Installez les paquets **libreswan**:

```
# dnf install libreswan
```

2. Si vous réinstallez Libreswan, supprimez son ancienne base de données NSS :

```
# systemctl stop ipsec

# rm /var/lib/ipsec/nss/*db
```

3. Démarrez le service **ipsec** et activez le service pour qu'il soit démarré automatiquement au démarrage :

```
# systemctl enable ipsec --now
```

4. Configurez le pare-feu pour qu'il autorise les ports 500 et 4500/UDP pour les protocoles IKE, ESP et AH en ajoutant le service **ipsec**:

```
# firewall-cmd --add-service="ipsec"
# firewall-cmd --runtime-to-permanent
```

5. Passer le système en mode FIPS :

```
# fips-mode-setup --enable
```

6. Redémarrez votre système pour permettre au noyau de passer en mode FIPS :

```
# reboot
```

Vérification

1. Pour confirmer que Libreswan fonctionne en mode FIPS :

```
# ipsec whack --fipsstatus
000 FIPS mode enabled
```

2. Il est également possible de vérifier les entrées de l'unité **ipsec** dans le journal **systemd**:

```
$ journalctl -u ipsec
...
Jan 22 11:26:50 localhost.localdomain pluto[3076]: FIPS Mode: YES
```

3. Pour connaître les algorithmes disponibles en mode FIPS :

```
# ipsec pluto --selftest 2>&1 | head -6
Initializing NSS using read-write database "sql:/var/lib/ipsec/nss"
FIPS Mode: YES
NSS crypto library initialized
FIPS mode enabled for pluto daemon
NSS library is running in FIPS mode
FIPS HMAC integrity support [disabled]
```

4. Pour interroger les algorithmes désactivés en mode FIPS :

```
# ipsec pluto --selftest 2>&1 | grep disabled
Encryption algorithm CAMELLIA_CTR disabled; not FIPS compliant
Encryption algorithm CAMELLIA_CBC disabled; not FIPS compliant
Encryption algorithm NULL disabled; not FIPS compliant
Encryption algorithm CHACHA20_POLY1305 disabled; not FIPS compliant
Hash algorithm MD5 disabled; not FIPS compliant
PRF algorithm HMAC_MD5 disabled; not FIPS compliant
PRF algorithm AES_XCBC disabled; not FIPS compliant
Integrity algorithm HMAC_MD5_96 disabled; not FIPS compliant
Integrity algorithm HMAC_SHA2_256_TRUNCBUG disabled; not FIPS compliant
Integrity algorithm AES_XCBC_96 disabled; not FIPS compliant
DH algorithm MODP1536 disabled; not FIPS compliant
DH algorithm DH31 disabled; not FIPS compliant
```

5. Pour dresser la liste de tous les algorithmes et chiffrements autorisés en mode FIPS :

```
# ipsec pluto --selftest 2>&1 | grep ESP | grep FIPS | sed "s/^.*FIPS//"
aes_ccm, aes_ccm_c
aes_ccm_b
aes_ccm_a
NSS(CBC)  3des
NSS(GCM)  aes_gcm, aes_gcm_c
NSS(GCM)  aes_gcm_b
NSS(GCM)  aes_gcm_a
NSS(CTR)  aesctr
NSS(CBC)  aes
aes_gmac
NSS       sha, sha1, sha1_96, hmac_sha1
NSS       sha512, sha2_512, sha2_512_256, hmac_sha2_512
NSS       sha384, sha2_384, sha2_384_192, hmac_sha2_384
NSS       sha2, sha256, sha2_256, sha2_256_128, hmac_sha2_256
aes_cmac
null
NSS(MODP) null, dh0
NSS(MODP) dh14
NSS(MODP) dh15
NSS(MODP) dh16
NSS(MODP) dh17
NSS(MODP) dh18
```

```
NSS(ECP)  ecp_256, ecp256
NSS(ECP)  ecp_384, ecp384
NSS(ECP)  ecp_521, ecp521
```

Ressources supplémentaires

- Utilisation de politiques cryptographiques à l'échelle du système .

6.9. PROTÉGER LA BASE DE DONNÉES IPSEC NSS PAR UN MOT DE PASSE

Par défaut, le service IPsec crée sa base de données NSS (Network Security Services) avec un mot de passe vide lors du premier démarrage. Ajoutez une protection par mot de passe en suivant les étapes suivantes.

Conditions préalables

- Le répertoire **/var/lib/ipsec/nss/** contient les fichiers de la base de données NSS.

Procédure

1. Activer la protection par mot de passe de la base de données **NSS** pour Libreswan :

```
# certutil -N -d sql:/var/lib/ipsec/nss
Enter Password or Pin for "NSS Certificate DB":
Enter a password which will be used to encrypt your keys.
The password should be at least 8 characters long,
and should contain at least one non-alphabetic character.

Enter new password:
```

2. Créez le fichier **/etc/ipsec.d/nsspassword** contenant le mot de passe que vous avez défini à l'étape précédente, par exemple :

```
# cat /etc/ipsec.d/nsspassword
NSS Certificate DB:MyStrongPasswordHere
```

   Notez que le fichier **nsspassword** utilise la syntaxe suivante :

```
token_1_name:the_password
token_2_name:the_password
```

   Le jeton par défaut du logiciel NSS est **NSS Certificate DB**. Si votre système fonctionne en mode FIPS, le nom du jeton est **NSS FIPS 140-2 Certificate DB**.

3. Selon votre scénario, démarrez ou redémarrez le service **ipsec** après avoir terminé le fichier **nsspassword**:

```
# systemctl restart ipsec
```

Vérification

1. Vérifiez que le service **ipsec** fonctionne après avoir ajouté un mot de passe non vide à sa base de données NSS :

```
# systemctl status ipsec
● ipsec.service - Internet Key Exchange (IKE) Protocol Daemon for IPsec
  Loaded: loaded (/usr/lib/systemd/system/ipsec.service; enabled; vendor preset: disable>
  Active: active (running)...
```

2. En option, vérifiez que le journal **Journal** contient des entrées confirmant la réussite de l'initialisation :

```
# journalctl -u ipsec
...
pluto[6214]: Initializing NSS using read-write database "sql:/var/lib/ipsec/nss"
pluto[6214]: NSS Password from file "/etc/ipsec.d/nsspassword" for token "NSS Certificate
DB" with length 20 passed to NSS
pluto[6214]: NSS crypto library initialized
...
```

Ressources supplémentaires

- **certutil(1)** page de manuel.
- Article de la base de connaissances sur les normes gouvernementales.

6.10. CONFIGURER UN VPN IPSEC POUR UTILISER TCP

Libreswan prend en charge l'encapsulation TCP des paquets IKE et IPsec, comme décrit dans la RFC 8229. Grâce à cette fonctionnalité, vous pouvez établir des VPN IPsec sur des réseaux qui empêchent le trafic transmis via UDP et Encapsulating Security Payload (ESP). Vous pouvez configurer les serveurs et les clients VPN pour qu'ils utilisent TCP soit comme protocole de secours, soit comme protocole de transport VPN principal. L'encapsulation TCP étant plus coûteuse en termes de performances, n'utilisez TCP comme protocole VPN principal que si l'UDP est bloqué en permanence dans votre scénario.

Conditions préalables

- Un VPN d'accès à distance est déjà configuré.

Procédure

1. Ajoutez l'option suivante au fichier **/etc/ipsec.conf** dans la section **config setup**:

```
listen-tcp=yes
```

2. Pour utiliser l'encapsulation TCP comme option de repli lorsque la première tentative via UDP échoue, ajoutez les deux options suivantes à la définition de la connexion du client :

```
enable-tcp=fallback
tcp-remoteport=4500
```

Sinon, si vous savez que UDP est bloqué en permanence, utilisez les options suivantes dans la configuration de la connexion du client :

```
enable-tcp=yes
tcp-remoteport=4500
```

Ressources supplémentaires

- IETF RFC 8229 : Encapsulation TCP des paquets IKE et IPsec .

6.11. CONFIGURATION DE LA DÉTECTION AUTOMATIQUE ET DE L'UTILISATION DE LA DÉCHARGE MATÉRIELLE ESP POUR ACCÉLÉRER UNE CONNEXION IPSEC

Le déchargement de l'Encapsulating Security Payload (ESP) vers le matériel accélère les connexions IPsec sur Ethernet. Par défaut, Libreswan détecte si le matériel prend en charge cette fonctionnalité et, par conséquent, active le délestage matériel de l'ESP. Dans le cas où la fonctionnalité a été désactivée ou explicitement activée, vous pouvez revenir à la détection automatique.

Conditions préalables

- La carte réseau prend en charge la décharge matérielle ESP.
- Le pilote de réseau prend en charge la décharge matérielle ESP.
- La connexion IPsec est configurée et fonctionne.

Procédure

1. Modifiez le fichier de configuration Libreswan dans le répertoire **/etc/ipsec.d/** de la connexion qui doit utiliser la détection automatique de la prise en charge du délestage matériel ESP.
2. Assurez-vous que le paramètre **nic-offload** n'est pas défini dans les paramètres de la connexion.
3. Si vous avez supprimé **nic-offload**, redémarrez le service **ipsec**:

```
# systemctl restart ipsec
```

Vérification

Si la carte réseau prend en charge la décharge matérielle ESP, procédez comme suit pour vérifier le résultat :

1. Affiche les compteurs **tx_ipsec** et **rx_ipsec** du périphérique Ethernet utilisé par la connexion IPsec :

```
# ethtool -S enp1s0 | egrep "_ipsec"
    tx_ipsec: 10
    rx_ipsec: 10
```

2. Envoyer du trafic à travers le tunnel IPsec. Par exemple, envoyer un ping à une adresse IP distante :

```
# ping -c 5 remote_ip_address
```

3. Affichez à nouveau les compteurs **tx_ipsec** et **rx_ipsec** de l'appareil Ethernet :

```
# ethtool -S enp1s0 | egrep "_ipsec"
    tx_ipsec: 15
    rx_ipsec: 15
```

Si les valeurs des compteurs ont augmenté, le délestage matériel ESP fonctionne.

Ressources supplémentaires

- Configuration d'un VPN avec IPsec

6.12. CONFIGURATION DE LA DÉCHARGE MATÉRIELLE ESP SUR UNE LIAISON POUR ACCÉLÉRER UNE CONNEXION IPSEC

Le déchargement de la charge utile d'encapsulation de sécurité (ESP) vers le matériel accélère les connexions IPsec. Si vous utilisez un lien réseau pour des raisons de basculement, les exigences et la procédure de configuration du déchargement matériel de l'ESP sont différentes de celles qui utilisent un périphérique Ethernet ordinaire. Par exemple, dans ce scénario, vous activez la prise en charge du délestage sur le lien et le noyau applique les paramètres aux ports du lien.

Conditions préalables

- Toutes les cartes réseau de la liaison prennent en charge la décharge matérielle ESP.
- Le pilote réseau prend en charge le délestage matériel ESP sur un périphérique de liaison. Dans RHEL, seul le pilote **ixgbe** prend en charge cette fonctionnalité.
- La liaison est configurée et fonctionne.
- La liaison utilise le mode **active-backup**. Le pilote de liaison ne prend pas en charge d'autres modes pour cette fonctionnalité.
- La connexion IPsec est configurée et fonctionne.

Procédure

1. Activer la prise en charge de la décharge matérielle ESP sur la liaison réseau :

```
# nmcli connection modify bond0 ethtool.feature-esp-hw-offload on
```

Cette commande active la prise en charge du délestage matériel ESP sur la connexion **bond0**.

2. Réactiver la connexion **bond0**:

```
# nmcli connection up bond0
```

3. Modifiez le fichier de configuration de Libreswan dans le répertoire **/etc/ipsec.d/** de la connexion qui doit utiliser le délestage matériel ESP, et ajoutez l'instruction **nic-offload=yes** à l'entrée de la connexion :

```
conn example
    ...
    nic-offload=yes
```

–

4. Redémarrez le service **ipsec**:

```
# systemctl restart ipsec
```

Vérification

1. Affiche le port actif de la liaison :

```
# grep "Currently Active Slave" /proc/net/bonding/bond0
Currently Active Slave: enp1s0
```

2. Affiche les compteurs **tx_ipsec** et **rx_ipsec** du port actif :

```
# ethtool -S enp1s0 | egrep "_ipsec"
     tx_ipsec: 10
     rx_ipsec: 10
```

3. Envoyer du trafic à travers le tunnel IPsec. Par exemple, envoyer un ping à une adresse IP distante :

```
# ping -c 5 remote_ip_address
```

4. Affichez à nouveau les compteurs **tx_ipsec** et **rx_ipsec** du port actif :

```
# ethtool -S enp1s0 | egrep "_ipsec"
     tx_ipsec: 15
     rx_ipsec: 15
```

Si les valeurs des compteurs ont augmenté, le délestage matériel ESP fonctionne.

Ressources supplémentaires

- Configuration de la liaison réseau
- Configuration d'un VPN avec IPsec

6.13. CONFIGURATION DES CONNEXIONS IPSEC QUI NE RESPECTENT PAS LES POLITIQUES CRYPTOGRAPHIQUES DU SYSTÈME

Remplacer les politiques cryptographiques du système pour une connexion

Les politiques cryptographiques du système RHEL créent une connexion spéciale appelée **Þfault**. Cette connexion contient les valeurs par défaut des options **ikev2**, **esp** et **ike**. Toutefois, vous pouvez remplacer les valeurs par défaut en spécifiant l'option mentionnée dans le fichier de configuration de la connexion.

Par exemple, la configuration suivante autorise les connexions qui utilisent IKEv1 avec AES et SHA-1 ou SHA-2, et IPsec (ESP) avec AES-GCM ou AES-CBC :

```
conn MyExample
```

```
...
ikev2=never
ike=aes-sha2,aes-sha1;modp2048
esp=aes_gcm,aes-sha2,aes-sha1
...
```

Notez que l'AES-GCM est disponible pour IPsec (ESP) et pour IKEv2, mais pas pour IKEv1.

Désactivation des stratégies cryptographiques à l'échelle du système pour toutes les connexions

Pour désactiver les stratégies cryptographiques à l'échelle du système pour toutes les connexions IPsec, commentez la ligne suivante dans le fichier **/etc/ipsec.conf**:

```
include /etc/crypto-policies/back-ends/libreswan.config
```

Ajoutez ensuite l'option **ikev2=never** à votre fichier de configuration de la connexion.

Ressources supplémentaires

- Utilisation de politiques cryptographiques à l'échelle du système .

6.14. DÉPANNAGE DES CONFIGURATIONS VPN IPSEC

Les problèmes liés aux configurations de VPN IPsec surviennent le plus souvent pour plusieurs raisons principales. Si vous rencontrez de tels problèmes, vous pouvez vérifier si la cause du problème correspond à l'un des scénarios suivants et appliquer la solution correspondante.

Dépannage de base des connexions

La plupart des problèmes liés aux connexions VPN surviennent lors de nouveaux déploiements, lorsque les administrateurs ont configuré les points d'extrémité avec des options de configuration inadaptées. De même, une configuration fonctionnelle peut soudainement cesser de fonctionner, souvent en raison de valeurs incompatibles nouvellement introduites. Cela peut résulter d'une modification de la configuration par un administrateur. Il se peut également qu'un administrateur ait installé une mise à jour du micrologiciel ou une mise à jour du paquetage avec des valeurs par défaut différentes pour certaines options, telles que les algorithmes de chiffrement.

Pour confirmer qu'une connexion VPN IPsec est établie :

```
# ipsec trafficstatus
006 #8: "vpn.example.com"[1] 192.0.2.1, type=ESP, add_time=1595296930, inBytes=5999,
outBytes=3231, id='@vpn.example.com', lease=100.64.13.5/32
```

Si la sortie est vide ou ne montre pas d'entrée avec le nom de la connexion, le tunnel est rompu.

Vérifier que le problème se situe au niveau de la connexion :

1. Rechargez la connexion *vpn.example.com*:

   ```
   # ipsec auto --add vpn.example.com
   002 added connection description "vpn.example.com"
   ```

2. Ensuite, lancez la connexion VPN :

```
# ipsec auto --up vpn.example.com
```

Problèmes liés au pare-feu

Le problème le plus courant est qu'un pare-feu situé sur l'un des points d'extrémité IPsec ou sur un routeur entre les points d'extrémité bloque tous les paquets IKE (Internet Key Exchange).

- Pour IKEv2, une sortie similaire à l'exemple suivant indique un problème avec un pare-feu :

```
# ipsec auto --up vpn.example.com
181 "vpn.example.com"[1] 192.0.2.2 #15: initiating IKEv2 IKE SA
181 "vpn.example.com"[1] 192.0.2.2 #15: STATE_PARENT_I1: sent v2I1, expected v2R1
010 "vpn.example.com"[1] 192.0.2.2 #15: STATE_PARENT_I1: retransmission; will wait 0.5
seconds for response
010 "vpn.example.com"[1] 192.0.2.2 #15: STATE_PARENT_I1: retransmission; will wait 1
seconds for response
010 "vpn.example.com"[1] 192.0.2.2 #15: STATE_PARENT_I1: retransmission; will wait 2
seconds for
...
```

- Pour IKEv1, la sortie de la commande d'initiation se présente comme suit :

```
# ipsec auto --up vpn.example.com
002 "vpn.example.com" #9: initiating Main Mode
102 "vpn.example.com" #9: STATE_MAIN_I1: sent MI1, expecting MR1
010 "vpn.example.com" #9: STATE_MAIN_I1: retransmission; will wait 0.5 seconds for
response
010 "vpn.example.com" #9: STATE_MAIN_I1: retransmission; will wait 1 seconds for
response
010 "vpn.example.com" #9: STATE_MAIN_I1: retransmission; will wait 2 seconds for
response
...
```

Le protocole IKE, utilisé pour configurer IPsec, étant crypté, l'outil **tcpdump** ne permet de résoudre qu'un nombre limité de problèmes. Si un pare-feu bloque des paquets IKE ou IPsec, vous pouvez essayer d'en trouver la cause à l'aide de l'utilitaire **tcpdump**. Cependant, **tcpdump** ne peut pas diagnostiquer d'autres problèmes liés aux connexions VPN IPsec.

- Pour capturer la négociation du VPN et toutes les données cryptées sur l'interface **eth0**:

```
# tcpdump -i eth0 -n -n esp or udp port 500 or udp port 4500 or tcp port 4500
```

Algorithmes, protocoles et politiques inadaptés

Les connexions VPN exigent que les algorithmes IKE, les algorithmes IPsec et les plages d'adresses IP des points d'extrémité correspondent. En cas de non-concordance, la connexion échoue. Si vous identifiez une incohérence à l'aide de l'une des méthodes suivantes, corrigez-la en alignant les algorithmes, les protocoles ou les stratégies.

- Si l'extrémité distante n'exécute pas IKE/IPsec, un paquet ICMP l'indique. Par exemple :

```
# ipsec auto --up vpn.example.com
...
000 "vpn.example.com"[1] 192.0.2.2 #16: ERROR: asynchronous network error report on
```

```
wlp2s0 (192.0.2.2:500), complainant 198.51.100.1: Connection refused [errno 111, origin
ICMP type 3 code 3 (not authenticated)]
...
```

- Exemple d'algorithmes IKE non compatibles :

```
# ipsec auto --up vpn.example.com
...
003 "vpn.example.com"[1] 193.110.157.148 #3: dropping unexpected IKE_SA_INIT message
containing NO_PROPOSAL_CHOSEN notification; message payloads: N; missing payloads:
SA,KE,Ni
```

- Exemple d'algorithmes IPsec non compatibles :

```
# ipsec auto --up vpn.example.com
...
182 "vpn.example.com"[1] 193.110.157.148 #5: STATE_PARENT_I2: sent v2I2, expected
v2R2 {auth=IKEv2 cipher=AES_GCM_16_256 integ=n/a prf=HMAC_SHA2_256
group=MODP2048}
002 "vpn.example.com"[1] 193.110.157.148 #6: IKE_AUTH response contained the error
notification NO_PROPOSAL_CHOSEN
```

Une version IKE inadaptée peut également entraîner l'abandon de la demande par le point d'extrémité distant sans réponse. Cette situation est identique à celle d'un pare-feu qui abandonne tous les paquets IKE.

- Exemple de plages d'adresses IP non concordantes pour IKEv2 (appelées sélecteurs de trafic - TS) :

```
# ipsec auto --up vpn.example.com
...
1v2 "vpn.example.com" #1: STATE_PARENT_I2: sent v2I2, expected v2R2 {auth=IKEv2
cipher=AES_GCM_16_256 integ=n/a prf=HMAC_SHA2_512 group=MODP2048}
002 "vpn.example.com" #2: IKE_AUTH response contained the error notification
TS_UNACCEPTABLE
```

- Exemple de plages d'adresses IP non concordantes pour IKEv1 :

```
# ipsec auto --up vpn.example.com
...
031 "vpn.example.com" #2: STATE_QUICK_I1: 60 second timeout exceeded after 0
retransmits.  No acceptable response to our first Quick Mode message: perhaps peer likes
no proposal
```

- Lors de l'utilisation de PreSharedKeys (PSK) dans IKEv1, si les deux parties ne mettent pas le même PSK, l'ensemble du message IKE devient illisible :

```
# ipsec auto --up vpn.example.com
...
003 "vpn.example.com" #1: received Hash Payload does not match computed value
223 "vpn.example.com" #1: sending notification INVALID_HASH_INFORMATION to
192.0.2.23:500
```

- Dans le cadre d'IKEv2, l'erreur de mauvaise concordance des paquets se traduit par l'envoi d'un message AUTHENTICATION_FAILED :

```
# ipsec auto --up vpn.example.com
...
002 "vpn.example.com" #1: IKE SA authentication request rejected by peer:
AUTHENTICATION_FAILED
```

Unité de transmission maximale

Outre les pare-feu qui bloquent les paquets IKE ou IPsec, la cause la plus fréquente des problèmes de réseau est l'augmentation de la taille des paquets cryptés. Le matériel réseau fragmente les paquets dont la taille est supérieure à l'unité de transmission maximale (MTU), par exemple 1500 octets. Souvent, les fragments sont perdus et les paquets ne parviennent pas à se réassembler. Cela entraîne des défaillances intermittentes, lorsqu'un test ping, qui utilise des paquets de petite taille, fonctionne mais que d'autres trafics échouent. Dans ce cas, vous pouvez établir une session SSH, mais le terminal se fige dès que vous l'utilisez, par exemple en entrant la commande 'ls -al /usr' sur l'hôte distant.

Pour contourner le problème, réduisez la taille du MTU en ajoutant l'option **mtu=1400** au fichier de configuration du tunnel.

Sinon, pour les connexions TCP, activez une règle iptables qui modifie la valeur MSS :

```
# iptables -I FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --clamp-mss-to-pmtu
```

Si la commande précédente ne résout pas le problème dans votre scénario, spécifiez directement une taille inférieure dans le paramètre **set-mss**:

```
# iptables -I FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --set-mss 1380
```

Traduction d'adresses de réseau (NAT)

Lorsqu'un hôte IPsec sert également de routeur NAT, il peut accidentellement remapper des paquets. L'exemple de configuration suivant illustre le problème :

```
conn myvpn
    left=172.16.0.1
    leftsubnet=10.0.2.0/24
    right=172.16.0.2
    rightsubnet=192.168.0.0/16
…
```

Le système avec l'adresse 172.16.0.1 a une règle NAT :

```
iptables -t nat -I POSTROUTING -o eth0 -j MASQUERADE
```

Si le système à l'adresse 10.0.2.33 envoie un paquet à 192.168.0.1, le routeur traduit la source 10.0.2.33 en 172.16.0.1 avant d'appliquer le cryptage IPsec.

Le paquet avec l'adresse source 10.0.2.33 ne correspond plus à la configuration **conn myvpn**, et IPsec ne crypte pas ce paquet.

Pour résoudre ce problème, insérez des règles qui excluent la NAT pour les plages de sous-réseaux IPsec cibles sur le routeur, dans cet exemple :

```
iptables -t nat -I POSTROUTING -s 10.0.2.0/24 -d 192.168.0.0/16 -j RETURN
```

Bogues du sous-système IPsec du noyau

Le sous-système IPsec du noyau peut échouer, par exemple, lorsqu'un bogue provoque une désynchronisation de l'espace utilisateur IKE et du noyau IPsec. Pour vérifier l'existence de tels problèmes, procédez comme suit

```
$ cat /proc/net/xfrm_stat
XfrmInError               0
XfrmInBufferError         0
...
```

Toute valeur non nulle dans la sortie de la commande précédente indique un problème. Si vous rencontrez ce problème, ouvrez un nouveau dossier d'assistance et joignez la sortie de la commande précédente ainsi que les journaux IKE correspondants.

Bûches Libreswan

Par défaut, Libreswan enregistre les données en utilisant le protocole **syslog**. Vous pouvez utiliser la commande **journalctl** pour trouver les entrées du journal relatives à IPsec. Comme les entrées correspondantes dans le journal sont envoyées par le démon IKE de **pluto**, recherchez le mot-clé "pluto", par exemple :

```
$ journalctl -b | grep pluto
```

Pour afficher un journal en direct pour le service **ipsec**:

```
$ journalctl -f -u ipsec
```

Si le niveau de journalisation par défaut ne révèle pas votre problème de configuration, activez les journaux de débogage en ajoutant l'option **plutodebug=all** à la section **config setup** du fichier **/etc/ipsec.conf**.

Notez que la journalisation de débogage produit beaucoup d'entrées et qu'il est possible que le service **journald** ou **syslogd** limite le débit des messages **syslog**. Pour vous assurer d'avoir des journaux complets, redirigez la journalisation vers un fichier. Modifiez le site **/etc/ipsec.conf** et ajoutez le site **logfile=/var/log/pluto.log** dans la section **config setup**.

Ressources supplémentaires

- Résolution des problèmes à l'aide des fichiers journaux .
- **tcpdump(8)** et **ipsec.conf(5)**.
- Utilisation et configuration de firewalld

6.15. RESSOURCES SUPPLÉMENTAIRES

- **ipsec(8)**, **ipsec.conf(5)**, **ipsec.secrets(5)**, **ipsec_auto(8)**, et **ipsec_rsasigkey(8)**.
- **/usr/share/doc/libreswan-*version*/** répertoire.
- Le site web du projet en amont .

- Le projet Libreswan Wiki.
- Toutes les pages de manuel de Libreswan .
- Publication spéciale 800-77 du NIST : Guide to IPsec VPNs .

CHAPITRE 7. CONFIGURATION DES CONNEXIONS VPN AVEC IPSEC À L'AIDE DU RÔLE SYSTÈME VPN RHEL

Avec le rôle de système **vpn**, vous pouvez configurer des connexions VPN sur des systèmes RHEL en utilisant Red Hat Ansible Automation Platform. Vous pouvez l'utiliser pour configurer des configurations d'hôte à hôte, de réseau à réseau, de serveur d'accès à distance VPN et de maillage.

Pour les connexions d'hôte à hôte, le rôle établit un tunnel VPN entre chaque paire d'hôtes dans la liste de **vpn_connections** en utilisant les paramètres par défaut, y compris la génération de clés si nécessaire. Vous pouvez également le configurer pour créer une configuration de maillage opportuniste entre tous les hôtes de la liste. Le rôle suppose que les noms des hôtes sous **hosts** sont les mêmes que les noms des hôtes utilisés dans l'inventaire Ansible, et que vous pouvez utiliser ces noms pour configurer les tunnels.

NOTE

Le rôle de système RHEL **vpn** ne prend actuellement en charge que Libreswan, qui est une implémentation IPsec, en tant que fournisseur de VPN.

7.1. CRÉATION D'UN VPN D'HÔTE À HÔTE AVEC IPSEC À L'AIDE DU RÔLE DE SYSTÈME VPN

Vous pouvez utiliser le rôle de système **vpn** pour configurer les connexions d'hôte à hôte en exécutant un manuel de jeu Ansible sur le nœud de contrôle, qui configurera tous les nœuds gérés répertoriés dans un fichier d'inventaire.

Conditions préalables

- Accès et autorisations à un ou plusieurs *managed nodes*, qui sont des systèmes que vous souhaitez configurer avec le rôle de système **vpn**.
- Accès et permissions à un *control node*, qui est un système à partir duquel Red Hat Ansible Core configure d'autres systèmes.
  Sur le nœud de contrôle :
  - Les paquets **ansible-core** et **rhel-system-roles** sont installés.

IMPORTANT

RHEL 8.0-8.5 donne accès à un dépôt Ansible distinct qui contient Ansible Engine 2.9 pour l'automatisation basée sur Ansible. Ansible Engine contient des utilitaires de ligne de commande tels que **ansible**, **ansible-playbook**, des connecteurs tels que **docker** et **podman**, ainsi que de nombreux plugins et modules. Pour plus d'informations sur la manière d'obtenir et d'installer Ansible Engine, consultez l'article de la base de connaissances Comment télécharger et installer Red Hat Ansible Engine .

RHEL 8.6 et 9.0 ont introduit Ansible Core (fourni en tant que paquetage **ansible-core**), qui contient les utilitaires de ligne de commande Ansible, les commandes et un petit ensemble de plugins Ansible intégrés. RHEL fournit ce paquetage par l'intermédiaire du dépôt AppStream, et sa prise en charge est limitée. Pour plus d'informations, consultez l'article de la base de connaissances intitulé Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories (Portée de la prise en charge du package Ansible Core inclus dans les dépôts AppStream RHEL 9 et RHEL 8.6 et versions ultérieures).

- Un fichier d'inventaire qui répertorie les nœuds gérés.

Procédure

1. Créez un nouveau fichier ***playbook.yml*** avec le contenu suivant :

```
- name: Host to host VPN
  hosts: managed_node1, managed_node2
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - hosts:
          managed_node1:
          managed_node2:
    vpn_manage_firewall: true
    vpn_manage_selinux: true
```

   Ce playbook configure la connexion ***managed_node1*-to-*managed_node2*** en utilisant l'authentification par clé pré-partagée avec des clés générées automatiquement par le rôle système. Puisque **vpn_manage_firewall** et **vpn_manage_selinux** sont tous deux définis sur true, le rôle **vpn** utilisera les rôles **firewall** et **selinux** pour gérer les ports utilisés par le rôle **vpn**.

2. Facultatif : Configurez les connexions entre les hôtes gérés et les hôtes externes qui ne sont pas répertoriés dans le fichier d'inventaire en ajoutant la section suivante à la liste d'hôtes **vpn_connections**:

```
    vpn_connections:
      - hosts:
          managed_node1:
          managed_node2:
          external_node:
            hostname: 192.0.2.2
```

   Cela permet de configurer deux connexions supplémentaires : ***managed_node1*-to-*external_node*** et ***managed_node2*-to-*external_node***.

NOTE

Les connexions sont configurées uniquement sur les nœuds gérés et non sur le nœud externe.

1. Facultatif : vous pouvez spécifier plusieurs connexions VPN pour les nœuds gérés en utilisant des sections supplémentaires dans **vpn_connections**, par exemple un plan de contrôle et un plan de données :

```
- name: Multiple VPN
  hosts: managed_node1, managed_node2
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - name: control_plane_vpn
        hosts:
          managed_node1:
            hostname: 192.0.2.0 # IP for the control plane
          managed_node2:
            hostname: 192.0.2.1
      - name: data_plane_vpn
        hosts:
          managed_node1:
            hostname: 10.0.0.1 # IP for the data plane
          managed_node2:
            hostname: 10.0.0.2
```

2. Facultatif : vous pouvez modifier les variables selon vos préférences. Pour plus de détails, voir le fichier **/usr/share/doc/rhel-system-roles/vpn/README.md**.

3. Facultatif : Vérifier la syntaxe du playbook.

```
# ansible-playbook --syntax-check /path/to/file/playbook.yml -i /path/to/file/inventory_file
```

4. Exécutez le playbook sur votre fichier d'inventaire :

```
# ansible-playbook -i /path/to/file/inventory_file /path/to/file/playbook.yml
```

Vérification

1. Sur les nœuds gérés, confirmez que la connexion est chargée avec succès :

```
# ipsec status | grep connection.name
```

Remplacez *connection.name* par le nom de la connexion de ce nœud, par exemple **managed_node1-to-managed_node2**.

NOTE

Par défaut, le rôle génère un nom descriptif pour chaque connexion qu'il crée du point de vue de chaque système. Par exemple, lors de la création d'une connexion entre **managed_node1** et **managed_node2**, le nom descriptif de cette connexion sur **managed_node1** est **managed_node1-to-managed_node2** mais sur **managed_node2** la connexion est nommée **managed_node2-to-managed_node1**.

1. Sur les nœuds gérés, confirmez que la connexion a été lancée avec succès :

```
# ipsec trafficstatus | grep connection.name
```

2. Facultatif : si une connexion n'a pas été chargée avec succès, ajoutez-la manuellement en entrant la commande suivante. Vous obtiendrez ainsi des informations plus précises sur la raison pour laquelle la connexion n'a pas pu être établie :

```
# ipsec auto --add connection.name
```

NOTE

Toutes les erreurs qui ont pu se produire au cours du processus de chargement et de démarrage de la connexion sont signalées dans les journaux, qui se trouvent à l'adresse **/var/log/pluto.log**. Comme ces journaux sont difficiles à analyser, essayez d'ajouter manuellement la connexion afin d'obtenir les messages de journaux à partir de la sortie standard à la place.

7.2. CRÉATION D'UNE CONNEXION VPN MAILLÉE OPPORTUNISTE AVEC IPSEC EN UTILISANT LE RÔLE DE SYSTÈME VPN

Vous pouvez utiliser le rôle système **vpn** pour configurer une connexion VPN maillée opportuniste qui utilise des certificats pour l'authentification en exécutant un livre de jeu Ansible sur le nœud de contrôle, qui configurera tous les nœuds gérés répertoriés dans un fichier d'inventaire.

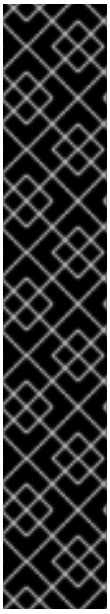
L'authentification par certificat est configurée en définissant le paramètre **auth_method: cert** dans le playbook. Le rôle de système **vpn** suppose que la bibliothèque cryptographique IPsec Network Security Services (NSS), définie dans le répertoire **/etc/ipsec.d**, contient les certificats nécessaires. Par défaut, le nom du nœud est utilisé comme pseudonyme du certificat. Dans cet exemple, il s'agit de **managed_node1**. Vous pouvez définir d'autres noms de certificats en utilisant l'attribut **cert_name** dans votre inventaire.

Dans l'exemple de procédure suivant, le nœud de contrôle, qui est le système à partir duquel vous exécuterez le playbook Ansible, partage le même numéro de routage interdomaine sans classe (CIDR) que les deux nœuds gérés (192.0.2.0/24) et possède l'adresse IP 192.0.2.7. Par conséquent, le nœud de contrôle relève de la stratégie privée qui est automatiquement créée pour le CIDR 192.0.2.0/24.

Pour éviter la perte de connexion SSH pendant la pièce, une politique claire pour le nœud de contrôle est incluse dans la liste des politiques. Notez qu'il y a également un élément dans la liste des stratégies où le CIDR est égal à la valeur par défaut. Cela s'explique par le fait que ce playbook remplace la règle de la stratégie par défaut pour la rendre privée au lieu de private-or-clear.

Conditions préalables

- Accès et autorisations à un ou plusieurs *managed nodes*, qui sont des systèmes que vous souhaitez configurer avec le rôle de système **vpn**.
  - Sur tous les nœuds gérés, la base de données NSS dans le répertoire **/etc/ipsec.d** contient tous les certificats nécessaires à l'authentification des pairs. Par défaut, le nom du nœud est utilisé comme pseudonyme du certificat.
- Accès et permissions à un *control node*, qui est un système à partir duquel Red Hat Ansible Core configure d'autres systèmes.
  Sur le nœud de contrôle :
  - Les paquets **ansible-core** et **rhel-system-roles** sont installés.



IMPORTANT

RHEL 8.0-8.5 donne accès à un dépôt Ansible distinct qui contient Ansible Engine 2.9 pour l'automatisation basée sur Ansible. Ansible Engine contient des utilitaires de ligne de commande tels que **ansible**, **ansible-playbook**, des connecteurs tels que **docker** et **podman**, ainsi que de nombreux plugins et modules. Pour plus d'informations sur la manière d'obtenir et d'installer Ansible Engine, consultez l'article de la base de connaissances Comment télécharger et installer Red Hat Ansible Engine .

RHEL 8.6 et 9.0 ont introduit Ansible Core (fourni en tant que paquetage **ansible-core**), qui contient les utilitaires de ligne de commande Ansible, les commandes et un petit ensemble de plugins Ansible intégrés. RHEL fournit ce paquetage par l'intermédiaire du dépôt AppStream, et sa prise en charge est limitée. Pour plus d'informations, consultez l'article de la base de connaissances intitulé Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories (Portée de la prise en charge du package Ansible Core inclus dans les dépôts AppStream RHEL 9 et RHEL 8.6 et versions ultérieures).

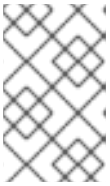
- Un fichier d'inventaire qui répertorie les nœuds gérés.

Procédure

1. Créez un nouveau fichier ***playbook.yml*** avec le contenu suivant :

```
- name: Mesh VPN
  hosts: managed_node1, managed_node2, managed_node3
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - opportunistic: true
        auth_method: cert
        policies:
          - policy: private
            cidr: default
          - policy: private-or-clear
            cidr: 198.51.100.0/24
          - policy: private
            cidr: 192.0.2.0/24
          - policy: clear
```

```
        cidr: 192.0.2.7/32
    vpn_manage_firewall: true
    vpn_manage_selinux: true
```



NOTE

Puisque **vpn_manage_firewall** et **vpn_manage_selinux** sont tous deux définis sur true, le rôle **vpn** utilisera les rôles **firewall** et **selinux** pour gérer les ports utilisés par le rôle **vpn**.

2. Facultatif : vous pouvez modifier les variables selon vos préférences. Pour plus de détails, voir le fichier **/usr/share/doc/rhel-system-roles/vpn/README.md**.

3. Facultatif : Vérifier la syntaxe du playbook.

```
# ansible-playbook --syntax-check playbook.yml
```

4. Exécutez le playbook sur votre fichier d'inventaire :

```
# ansible-playbook -i inventory_file /path/to/file/playbook.yml
```

7.3. RESSOURCES SUPPLÉMENTAIRES

- Pour plus de détails sur les paramètres utilisés dans le rôle de système **vpn** et des informations supplémentaires sur le rôle, voir le fichier **/usr/share/doc/rhel-system-roles/vpn/README.md**.
- Pour plus de détails sur la commande **ansible-playbook**, voir la page de manuel **ansible-playbook(1)**.

CHAPITRE 8. SÉCURISATION DES SERVICES DE RÉSEAU

Red Hat Enterprise Linux 9 prend en charge de nombreux types de serveurs réseau. Leurs services réseau peuvent exposer la sécurité du système aux risques de divers types d'attaques, tels que les attaques par déni de service (DoS), les attaques par déni de service distribué (DDoS), les attaques par vulnérabilité de script et les attaques par débordement de mémoire tampon.

Pour renforcer la sécurité du système contre les attaques, il est important de surveiller les services réseau actifs que vous utilisez. Par exemple, lorsqu'un service réseau est en cours d'exécution sur une machine, son démon écoute les connexions sur les ports réseau, ce qui peut réduire la sécurité. Pour limiter l'exposition aux attaques sur le réseau, tous les services qui ne sont pas utilisés doivent être désactivés.

8.1. SÉCURISER LE SERVICE RPCBIND

Le service **rpcbind** est un démon dynamique d'attribution de ports pour les services d'appel de procédure à distance (RPC) tels que Network Information Service (NIS) et Network File System (NFS). Étant donné que ses mécanismes d'authentification sont faibles et qu'il peut attribuer un large éventail de ports aux services qu'il contrôle, il est important de sécuriser **rpcbind**.

Vous pouvez sécuriser **rpcbind** en limitant l'accès à tous les réseaux et en définissant des exceptions spécifiques à l'aide de règles de pare-feu sur le serveur.

NOTE

- Le service **rpcbind** est requis sur les serveurs **NFSv3**.
- **NFSv4** ne nécessite pas que le service **rpcbind** écoute le réseau.

Conditions préalables

- Le paquet **rpcbind** est installé.
- Le paquetage **firewalld** est installé et le service est en cours d'exécution.

Procédure

1. Ajouter des règles de pare-feu, par exemple :
   - Limiter les connexions TCP et n'accepter que les paquets provenant de l'hôte **192.168.0.0/24** via le port **111**:

     ```
     # firewall-cmd --add-rich-rule='rule family="ipv4" port port="111" protocol="tcp" source address="192.168.0.0/24" invert="True" drop'
     ```

   - Limiter les connexions TCP et n'accepter que les paquets provenant de l'hôte local via le port **111**:

     ```
     # firewall-cmd --add-rich-rule='rule family="ipv4" port port="111" protocol="tcp" source address="127.0.0.1" accept'
     ```

   - Limiter les connexions UDP et n'accepter que les paquets provenant de l'hôte **192.168.0.0/24** via le port **111**:

```
# firewall-cmd --permanent --add-rich-rule='rule family="ipv4" port port="111"
protocol="udp" source address="192.168.0.0/24" invert="True" drop'
```

Pour rendre les paramètres du pare-feu permanents, utilisez l'option **--permanent** lors de l'ajout de règles de pare-feu.

2. Rechargez le pare-feu pour appliquer les nouvelles règles :

```
# firewall-cmd --reload
```

Verification steps

- Listez les règles du pare-feu :

```
# firewall-cmd --list-rich-rule
rule family="ipv4" port port="111" protocol="tcp" source address="192.168.0.0/24"
invert="True" drop
rule family="ipv4" port port="111" protocol="tcp" source address="127.0.0.1" accept
rule family="ipv4" port port="111" protocol="udp" source address="192.168.0.0/24"
invert="True" drop
```

Ressources supplémentaires

- Pour plus d'informations sur les serveurs **NFSv4-only**, voir la section Configuration d'un serveur NFSv4 uniquement.
- Utilisation et configuration de firewalld

8.2. SÉCURISER LE SERVICE RPC.MOUNTD

Le démon **rpc.mountd** implémente le côté serveur du protocole de montage NFS. Le protocole de montage NFS est utilisé par la version 3 de NFS (RFC 1813).

Vous pouvez sécuriser le service **rpc.mountd** en ajoutant des règles de pare-feu au serveur. Vous pouvez restreindre l'accès à tous les réseaux et définir des exceptions spécifiques à l'aide de règles de pare-feu.

Conditions préalables

- Le paquet **rpc.mountd** est installé.
- Le paquetage **firewalld** est installé et le service est en cours d'exécution.

Procédure

1. Ajouter des règles de pare-feu au serveur, par exemple :
   - Accepter les connexions **mountd** à partir de l'hôte **192.168.0.0/24**:

```
# firewall-cmd --add-rich-rule 'rule family="ipv4" service name="mountd" source
address="192.168.0.0/24" invert="True" drop'
```

   - Accepter les connexions **mountd** à partir de l'hôte local :

```
-
```

```
# firewall-cmd --permanent --add-rich-rule 'rule family="ipv4" source address="127.0.0.1"
service name="mountd" accept'
```

Pour rendre les paramètres du pare-feu permanents, utilisez l'option **--permanent** lors de l'ajout de règles de pare-feu.

2. Rechargez le pare-feu pour appliquer les nouvelles règles :

```
# firewall-cmd --reload
```

Verification steps

- Listez les règles du pare-feu :

```
# firewall-cmd --list-rich-rule
rule family="ipv4" service name="mountd" source address="192.168.0.0/24" invert="True"
drop
rule family="ipv4" source address="127.0.0.1" service name="mountd" accept
```

Ressources supplémentaires

- Utilisation et configuration de firewalld

8.3. SÉCURISER LE SERVICE NFS

Vous pouvez sécuriser la version 4 du système de fichiers réseau (NFSv4) en authentifiant et en chiffrant toutes les opérations du système de fichiers à l'aide de Kerberos. Lorsque vous utilisez NFSv4 avec un NAT (Network Address Translation) ou un pare-feu, vous pouvez désactiver les délégations en modifiant le fichier **/etc/default/nfs**. La délégation est une technique par laquelle le serveur délègue la gestion d'un fichier à un client.

En revanche, NFSv3 n'utilise pas Kerberos pour le verrouillage et le montage des fichiers.

Le service NFS envoie le trafic en utilisant TCP dans toutes les versions de NFS. Le service prend en charge l'authentification des utilisateurs et des groupes Kerberos, dans le cadre du module du noyau **RPCSEC_GSS**.

NFS permet aux hôtes distants de monter des systèmes de fichiers sur un réseau et d'interagir avec ces systèmes de fichiers comme s'ils étaient montés localement. Vous pouvez fusionner les ressources sur des serveurs centralisés et personnaliser les options de montage NFS dans le fichier **/etc/nfsmount.conf** lors du partage des systèmes de fichiers.

8.3.1. Options d'exportation pour sécuriser un serveur NFS

Le serveur NFS détermine une structure de liste de répertoires et d'hôtes concernant les systèmes de fichiers à exporter vers les hôtes dans le fichier **/etc/exports**.

AVERTISSEMENT

Des espaces supplémentaires dans la syntaxe du fichier d'exportation peuvent entraîner des changements importants dans la configuration.

Dans l'exemple suivant, le répertoire **/tmp/nfs/** est partagé avec l'hôte **bob.example.com** et dispose d'autorisations de lecture et d'écriture.

```
/tmp/nfs/     bob.example.com(rw)
```

L'exemple suivant est le même que le précédent, mais il partage le même répertoire avec l'hôte **bob.example.com** avec des autorisations de lecture seule et le partage avec l'hôte *world* avec des autorisations de lecture et d'écriture en raison d'un seul caractère d'espace après le nom d'hôte.

```
/tmp/nfs/     bob.example.com (rw)
```

Vous pouvez vérifier les répertoires partagés de votre système en entrant la commande **showmount -e <hostname>**.

Utilisez les options d'exportation suivantes sur le fichier **/etc/exports**:

AVERTISSEMENT

Exporter un système de fichiers entier, car l'exportation d'un sous-répertoire d'un système de fichiers n'est pas sécurisée. Un pirate peut éventuellement accéder à la partie non exportée d'un système de fichiers partiellement exporté.

ro

Utilisez l'option **ro** pour exporter le volume NFS en lecture seule.

rw

Utilisez l'option **rw** pour autoriser les demandes de lecture et d'écriture sur le volume NFS. Utilisez cette option avec prudence, car le fait d'autoriser l'accès en écriture augmente le risque d'attaques.

NOTE

Si votre scénario nécessite de monter les répertoires avec l'option **rw**, assurez-vous qu'ils ne sont pas accessibles en écriture à tous les utilisateurs afin de réduire les risques éventuels.

racine_squash

Utilisez l'option **root_squash** pour faire correspondre les requêtes provenant de **uid/gid** 0 avec les requêtes anonymes **uid/gid**. Cela ne s'applique pas aux autres **uids** ou **gids** qui pourraient être tout aussi sensibles, comme l'utilisateur **bin** ou le groupe **staff**.

no_root_squash

Utilisez l'option **no_root_squash** pour désactiver l'écrasement de la racine. Par défaut, les partages NFS remplacent l'utilisateur **root** par l'utilisateur **nobody**, qui est un compte d'utilisateur non privilégié. Le propriétaire de tous les fichiers créés par **root** devient **nobody**, ce qui empêche le téléchargement de programmes dont le bit **setuid** est défini. Lorsque l'option **no_root_squash** est utilisée, les utilisateurs root distants peuvent modifier n'importe quel fichier sur le système de fichiers partagé et laisser des applications infectées par des chevaux de Troie à d'autres utilisateurs.

sécurisé

L'option **secure** permet de limiter les exportations aux ports réservés. Par défaut, le serveur n'autorise la communication avec les clients que par l'intermédiaire des ports réservés. Cependant, sur de nombreux réseaux, il est facile pour n'importe qui de devenir un utilisateur **root** sur un client, de sorte qu'il est rarement sûr pour le serveur de supposer que la communication via un port réservé est privilégiée. Par conséquent, la restriction aux ports réservés n'a qu'une valeur limitée ; il est préférable de s'appuyer sur Kerberos, les pare-feu et la restriction des exportations à des clients particuliers.

En outre, il convient de tenir compte des meilleures pratiques suivantes lors de l'exportation d'un serveur NFS :

- L'exportation des répertoires personnels présente un risque car certaines applications stockent les mots de passe en texte clair ou dans un format faiblement crypté. Vous pouvez réduire ce risque en révisant et en améliorant le code de l'application.
- Certains utilisateurs ne définissent pas de mots de passe sur les clés SSH, ce qui entraîne à nouveau des risques avec les répertoires personnels. Vous pouvez réduire ces risques en imposant l'utilisation de mots de passe ou en utilisant Kerberos.
- Limitez les exportations NFS aux seuls clients nécessaires. Utilisez la commande **showmount -e** sur le serveur NFS pour vérifier ce que le serveur exporte. N'exportez rien qui ne soit pas spécifiquement requis.
- Pour réduire le risque d'attaques, n'autorisez pas les utilisateurs inutiles à se connecter à un serveur. Vous pouvez vérifier périodiquement qui et quoi peut accéder au serveur.

Ressources supplémentaires

- Sécuriser NFS avec Kerberos lors de l'utilisation de Red Hat Identity Management
- Configuration du serveur NFS.
- **exports(5)** et **nfs(5)** pages de manuel

8.3.2. Options de montage pour sécuriser un client NFS

Vous pouvez ajouter les options suivantes à la commande **mount** afin d'améliorer la sécurité des clients NFS :

nosuid

Utilisez l'option **nosuid** pour désactiver les bits **set-user-identifier** ou **set-group-identifier**. Cela empêche les utilisateurs distants d'obtenir des privilèges plus élevés en exécutant un programme **setuid** et vous pouvez utiliser cette option à l'inverse de l'option **setuid**.

noexec

L'option **noexec** permet de désactiver tous les fichiers exécutables sur le client. Cette option permet d'éviter que les utilisateurs n'exécutent accidentellement des fichiers placés dans le système de fichiers partagés.

nodev

Utilisez l'option **nodev** pour empêcher le client de traiter les fichiers de périphérique comme un périphérique matériel.

resvport

Utilisez l'option **resvport** pour restreindre la communication à un port réservé et vous pouvez utiliser un port source privilégié pour communiquer avec le serveur. Les ports réservés sont réservés aux utilisateurs et processus privilégiés tels que l'utilisateur **root**.

sec

Utilisez l'option **sec** sur le serveur NFS pour choisir la saveur de sécurité RPCGSS pour l'accès aux fichiers sur le point de montage. Les saveurs de sécurité valides sont **none**, **sys**, **krb5**, **krb5i** et **krb5p**.

IMPORTANT

Les bibliothèques MIT Kerberos fournies par le paquetage **krb5-libs** ne prennent pas en charge l'algorithme Data Encryption Standard (DES) dans les nouveaux déploiements. Pour des raisons de sécurité et de compatibilité, l'algorithme DES est déprécié et désactivé par défaut dans les bibliothèques Kerberos. Utilisez des algorithmes plus récents et plus sûrs au lieu de DES, à moins que votre environnement n'exige DES pour des raisons de compatibilité.

Ressources supplémentaires

- Options de montage NFS courantes .

8.3.3. Sécuriser NFS avec un pare-feu

Pour sécuriser le pare-feu d'un serveur NFS, ne laissez ouverts que les ports nécessaires. N'utilisez pas les numéros de port de la connexion NFS pour un autre service.

Conditions préalables

- Le paquet **nfs-utils** est installé.
- Le paquet **firewalld** est installé et fonctionne.

Procédure

- Sur NFSv4, le pare-feu doit ouvrir le port TCP **2049**.
- Sur NFSv3, ouvrez quatre ports supplémentaires avec **2049**:

1. **rpcbind** attribue les ports NFS de manière dynamique, ce qui peut poser des problèmes lors de la création de règles de pare-feu. Pour simplifier ce processus, utilisez le fichier **/etc/nfs.conf** pour spécifier les ports à utiliser :

   a. Définir le port TCP et UDP pour **mountd** (**rpc.mountd**) dans la section **[mountd]** au format **port=*<value>*** format.

b. Définir le port TCP et UDP pour **statd** (**rpc.statd**) dans la section **[statd]** au format **port=*<value>*** format.

2. Définissez les ports TCP et UDP pour le gestionnaire de verrous NFS (**nlockmgr**) dans le fichier **/etc/nfs.conf**:

a. Définissez le port TCP pour **nlockmgr** (**rpc.statd**) dans la section **[lockd]** au format **port=value**. Vous pouvez également utiliser l'option **nlm_tcpport** dans le fichier **/etc/modprobe.d/lockd.conf**.

b. Définissez le port UDP pour **nlockmgr** (**rpc.statd**) dans la section **[lockd]** au format **udp-port=value**. Vous pouvez également utiliser l'option **nlm_udpport** dans le fichier **/etc/modprobe.d/lockd.conf**.

Verification steps

- Liste des ports actifs et des programmes RPC sur le serveur NFS :

```
$ rpcinfo -p
```

Ressources supplémentaires

- Sécuriser NFS avec Kerberos lors de l'utilisation de Red Hat Identity Management
- **exports(5)** et **nfs(5)** pages de manuel

8.4. SÉCURISER LE SERVICE FTP

Vous pouvez utiliser le protocole de transfert de fichiers (FTP) pour transférer des fichiers sur un réseau. Étant donné que toutes les transactions FTP avec le serveur, y compris l'authentification de l'utilisateur, ne sont pas cryptées, vous devez vous assurer que la configuration est sécurisée.

RHEL 9 fournit deux serveurs FTP :

- Red Hat Content Accelerator (tux) - un serveur web en espace noyau avec des capacités FTP.
- Very Secure FTP Daemon (vsftpd) - une implémentation autonome du service FTP, axée sur la sécurité.

Les consignes de sécurité suivantes concernent la mise en place du service FTP **vsftpd**.

8.4.1. Sécurisation de la bannière d'accueil FTP

Lorsqu'un utilisateur se connecte au service FTP, ce dernier affiche une bannière d'accueil qui, par défaut, contient des informations sur la version qui pourraient être utiles aux pirates pour identifier les faiblesses d'un système. Vous pouvez empêcher les pirates d'accéder à ces informations en modifiant la bannière par défaut.

Vous pouvez définir une bannière personnalisée en modifiant le fichier **/etc/banners/ftp.msg** pour inclure directement un message d'une seule ligne ou pour faire référence à un fichier séparé, qui peut contenir un message de plusieurs lignes.

Procédure

- Pour définir un message d'une seule ligne, ajoutez l'option suivante au fichier **/etc/vsftpd/vsftpd.conf**:

```
ftpd_banner=Hello, all activity on ftp.example.com is logged.
```

- Pour définir un message dans un fichier séparé :
  - Créez un fichier **.msg** qui contient le message de la bannière, par exemple **/etc/banners/*ftp*.msg**:

```
######### Hello, all activity on ftp.example.com is logged. #########
```

    Pour simplifier la gestion de plusieurs bannières, placez toutes les bannières dans le répertoire **/etc/banners/**.

  - Ajoutez le chemin d'accès au fichier de la bannière à l'option **banner_file** du fichier **/etc/vsftpd/vsftpd.conf**:

```
banner_file=/etc/banners/ftp.msg
```

Vérification

- Afficher la bannière modifiée :

```
$ ftp localhost
Trying ::1…
Connected to localhost (::1).
Hello, all activity on ftp.example.com is logged.
```

8.4.2. Empêcher les accès et les téléchargements anonymes dans FTP

Par défaut, l'installation du paquet **vsftpd** crée le répertoire **/var/ftp/** et une arborescence de répertoires pour les utilisateurs anonymes avec des autorisations de lecture seule sur les répertoires. Comme les utilisateurs anonymes peuvent accéder aux données, ne stockez pas de données sensibles dans ces répertoires.

Pour renforcer la sécurité du système, vous pouvez configurer le serveur FTP de manière à permettre aux utilisateurs anonymes de télécharger des fichiers dans un répertoire spécifique et à empêcher les utilisateurs anonymes de lire les données. Dans la procédure suivante, l'utilisateur anonyme doit pouvoir télécharger des fichiers dans le répertoire appartenant à l'utilisateur **root**, mais ne doit pas pouvoir le modifier.

Procédure

- Créez un répertoire en écriture seule dans le répertoire **/var/ftp/pub/**:

```
# mkdir /var/ftp/pub/upload
# chmod 730 /var/ftp/pub/upload
# ls -ld /var/ftp/pub/upload
drwx-wx---. 2 root ftp 4096 Nov 14 22:57 /var/ftp/pub/upload
```

- Ajoutez les lignes suivantes au fichier **/etc/vsftpd/vsftpd.conf**:

```
anon_upload_enable=YES
anonymous_enable=YES
```

- Facultatif : si SELinux est activé et appliqué sur votre système, activez les attributs booléens SELinux **allow_ftpd_anon_write** et **allow_ftpd_full_access**.

AVERTISSEMENT

En permettant à des utilisateurs anonymes de lire et d'écrire dans des répertoires, le serveur peut devenir un dépôt de logiciels volés.

8.4.3. Sécurisation des comptes d'utilisateurs pour FTP

Le protocole FTP transmet les noms d'utilisateur et les mots de passe en clair sur des réseaux non sécurisés à des fins d'authentification. Vous pouvez améliorer la sécurité du protocole FTP en refusant aux utilisateurs du système l'accès au serveur à partir de leurs comptes d'utilisateur.

Effectuez autant d'étapes que nécessaire en fonction de votre configuration.

Procédure

- Désactivez tous les comptes d'utilisateurs du serveur **vsftpd** en ajoutant la ligne suivante au fichier **/etc/vsftpd/vsftpd.conf**:

```
local_enable=NO
```

- Désactivez l'accès FTP pour des comptes ou des groupes de comptes spécifiques, tels que l'utilisateur **root** et les utilisateurs disposant de privilèges **sudo**, en ajoutant les noms d'utilisateur au fichier de configuration PAM **/etc/pam.d/vsftpd**.
- Désactiver les comptes d'utilisateurs en ajoutant les noms d'utilisateurs au fichier **/etc/vsftpd/ftpusers**.

8.4.4. Ressources supplémentaires

- **ftpd_selinux(8)** page de manuel

8.5. SÉCURISATION DES SERVEURS HTTP

8.5.1. Améliorations de la sécurité dans httpd.conf

Vous pouvez renforcer la sécurité du serveur HTTP Apache en configurant les options de sécurité dans le fichier **/etc/httpd/conf/httpd.conf**.

Vérifiez toujours que tous les scripts exécutés sur le système fonctionnent correctement avant de les mettre en production.

Assurez-vous que seul l'utilisateur **root** dispose des droits d'écriture sur tout répertoire contenant des scripts ou des Common Gateway Interfaces (CGI). Pour changer la propriété du répertoire en faveur de l'utilisateur **root** avec des droits d'écriture, entrez les commandes suivantes :

```
# chown root directory-name
# chmod 755 directory-name
```

Dans le fichier **/etc/httpd/conf/httpd.conf**, vous pouvez configurer les options suivantes :

FollowSymLinks

Cette directive est activée par défaut et suit les liens symboliques dans le répertoire.

Index

Cette directive est activée par défaut. Désactivez cette directive pour empêcher les visiteurs de parcourir les fichiers du serveur.

Dossier de l'utilisateur

Cette directive est désactivée par défaut car elle peut confirmer la présence d'un compte utilisateur sur le système. Pour activer la consultation des répertoires d'utilisateurs autres que **/root/**, utilisez les directives racine **UserDir enabled** et **UserDir disabled**. Pour ajouter des utilisateurs à la liste des comptes désactivés, ajoutez une liste d'utilisateurs délimitée par des espaces sur la ligne **UserDir disabled**.

Jetons de serveur

Cette directive contrôle le champ d'en-tête de la réponse du serveur qui est renvoyée aux clients. Vous pouvez utiliser les paramètres suivants pour personnaliser les informations :

ServerTokens Complet

fournit toutes les informations disponibles, telles que le numéro de version du serveur web, les détails du système d'exploitation du serveur, les modules Apache installés, par exemple :

```
Apache/2.4.37 (Red Hat Enterprise Linux) MyMod/1.2
```

ServerTokens Full-Release

fournit toutes les informations disponibles sur les versions, par exemple :

```
Apache/2.4.37 (Red Hat Enterprise Linux) (Release 41.module+el8.5.0+11772+c8e0c271)
```

ServerTokens Prod / ServerTokens ProductOnly

fournit le nom du serveur web, par exemple :

```
Apache
```

ServerTokens Major

fournit la version majeure du serveur web, par exemple :

```
Apache/2
```

ServerTokens Minor

fournit la version mineure du serveur web, par exemple :

```
Apache/2.4
```

ServerTokens Min / ServerTokens Minimal

fournit la version minimale du serveur web, par exemple :

```
Apache/2.4.37
```

ServerTokens OS

fournit la version du serveur web et le système d'exploitation, par exemple :

```
Apache/2.4.37 (Red Hat Enterprise Linux)
```

Utilisez l'option **ServerTokens Prod** pour réduire le risque que des pirates obtiennent des informations précieuses sur votre système.

IMPORTANT

Ne supprimez pas la directive **IncludesNoExec**. Par défaut, le module Server Side Includes (SSI) ne peut pas exécuter de commandes. Modifier cela peut permettre à un attaquant d'entrer des commandes sur le système.

Suppression des modules httpd

Vous pouvez supprimer les modules **httpd** pour limiter les fonctionnalités du serveur HTTP. Pour ce faire, modifiez les fichiers de configuration dans le répertoire **/etc/httpd/conf.modules.d/** ou **/etc/httpd/conf.d/**. Par exemple, pour supprimer le module proxy :

```
echo '# All proxy modules disabled' > /etc/httpd/conf.modules.d/00-proxy.conf
```

Ressources supplémentaires

- Le serveur HTTP Apache
- Personnalisation de la politique SELinux pour le serveur HTTP Apache

8.5.2. Sécuriser la configuration du serveur Nginx

Nginx est un serveur HTTP et proxy très performant. Vous pouvez renforcer votre configuration Nginx avec les options de configuration suivantes.

Procédure

- Pour désactiver les chaînes de version, modifiez l'option de configuration **server_tokens**:

```
server_tokens off;
```

Cette option arrête l'affichage de détails supplémentaires tels que le numéro de version du serveur. Cette configuration n'affiche que le nom du serveur dans toutes les requêtes servies par Nginx, par exemple :

```
$ curl -sI http://localhost | grep Server
Server: nginx
```

- Ajouter des en-têtes de sécurité supplémentaires qui atténuent certaines vulnérabilités connues des applications web dans des fichiers conf spécifiques de **/etc/nginx/**:
  - Par exemple, l'option d'en-tête **X-Frame-Options** interdit à toute page extérieure à votre domaine d'encadrer tout contenu servi par Nginx, ce qui atténue les attaques par détournement de clics :

    ```
    add_header X-Frame-Options "SAMEORIGIN";
    ```

  - Par exemple, l'en-tête **x-content-type** empêche le reniflage du type MIME dans certains navigateurs plus anciens :

    ```
    add_header X-Content-Type-Options nosniff;
    ```

  - Par exemple, l'en-tête **X-XSS-Protection** permet le filtrage des scripts intersites (XSS), qui empêche les navigateurs de rendre un contenu potentiellement malveillant inclus dans une réponse de Nginx :

    ```
    add_header X-XSS-Protection "1; mode=block";
    ```

- Vous pouvez limiter les services exposés au public et limiter ce qu'ils font et acceptent des visiteurs, par exemple :

  ```
  limit_except GET {
    allow 192.168.1.0/32;
    deny  all;
  }
  ```

  L'extrait limitera l'accès à toutes les méthodes, à l'exception de **GET** et **HEAD**.

- Vous pouvez désactiver les méthodes HTTP, par exemple :

  ```
  # Allow GET, PUT, POST; return "405 Method Not Allowed" for all others.
  if ( $request_method !~ ^(GET|PUT|POST)$ ) {
    return 405;
  }
  ```

- Vous pouvez configurer SSL pour protéger les données servies par votre serveur web Nginx, en envisageant de les servir uniquement via HTTPS. En outre, vous pouvez générer un profil de configuration sécurisé pour activer SSL dans votre serveur Nginx à l'aide du générateur de configuration SSL de Mozilla. La configuration générée garantit que les protocoles vulnérables connus (par exemple, SSLv2 et SSLv3), les algorithmes de chiffrement et de hachage (par exemple, 3DES et MD5) sont désactivés. Vous pouvez également utiliser le test du serveur SSL pour vérifier que votre configuration répond aux exigences de sécurité modernes.

Ressources supplémentaires

- Générateur de configuration SSL de Mozilla
- Test du serveur SSL

8.6. SÉCURISER POSTGRESQL EN LIMITANT L'ACCÈS AUX UTILISATEURS LOCAUX AUTHENTIFIÉS

PostgreSQL est un système de gestion de base de données (SGBD) objet-relationnel. Dans Red Hat Enterprise Linux, PostgreSQL est fourni par le paquetage **postgresql-server**.

Vous pouvez réduire les risques d'attaques en configurant l'authentification du client. Le fichier de configuration **pg_hba.conf** stocké dans le répertoire de données du cluster de bases de données contrôle l'authentification du client. Suivez la procédure pour configurer PostgreSQL pour l'authentification basée sur l'hôte.

Procédure

1. Installer PostgreSQL :

```
# yum install postgresql-server
```

2. Initialiser une zone de stockage de la base de données à l'aide de l'une des options suivantes :

   a. Utilisation de l'utilitaire **initdb**:

   ```
   $ initdb -D /home/postgresql/db1/
   ```

   La commande **initdb** avec l'option **-D** crée le répertoire que vous spécifiez s'il n'existe pas déjà, par exemple ***/home/postgresql/db1/***. Ce répertoire contient alors toutes les données stockées dans la base de données ainsi que le fichier de configuration de l'authentification du client.

   b. Utilisation du script **postgresql-setup**:

   ```
   $ postgresql-setup --initdb
   ```

   Par défaut, le script utilise le répertoire **/var/lib/pgsql/data/**. Ce script aide les administrateurs système à gérer les bases de données en cluster.

3. Pour permettre à n'importe quel utilisateur local authentifié d'accéder à n'importe quelle base de données avec son nom d'utilisateur, modifiez la ligne suivante dans le fichier **pg_hba.conf**:

```
local   all             all                                     trust
```

   Cela peut s'avérer problématique lorsque vous utilisez des applications en couches qui créent des utilisateurs de base de données et aucun utilisateur local. Si vous ne souhaitez pas contrôler explicitement tous les noms d'utilisateur du système, supprimez la ligne **local** du fichier **pg_hba.conf**.

4. Redémarrez la base de données pour appliquer les modifications :

```
# systemctl restart postgresql
```

   La commande précédente met à jour la base de données et vérifie également la syntaxe du fichier de configuration.

8.7. SÉCURISER LE SERVICE MEMCACHED

Memcached est un système de mise en cache d'objets en mémoire distribuée, open source et très performant. Il peut améliorer les performances des applications web dynamiques en réduisant la charge de la base de données.

Memcached est un magasin de valeurs clés en mémoire pour de petits morceaux de données arbitraires, telles que des chaînes de caractères et des objets, provenant des résultats d'appels de bases de données, d'appels d'API ou de rendus de pages. Memcached permet d'affecter la mémoire des zones sous-utilisées aux applications qui ont besoin de plus de mémoire.

En 2018, des vulnérabilités d'attaques par amplification DDoS en exploitant des serveurs Memcached exposés à l'internet public ont été découvertes. Ces attaques ont tiré parti de la communication Memcached utilisant le protocole UDP pour le transport. L'attaque était efficace en raison du taux d'amplification élevé où une requête d'une taille de quelques centaines d'octets pouvait générer une réponse d'une taille de quelques mégaoctets, voire de centaines de mégaoctets.

Dans la plupart des cas, le service **memcached** n'a pas besoin d'être exposé à l'Internet public. Une telle exposition peut poser ses propres problèmes de sécurité, en permettant à des attaquants distants de divulguer ou de modifier des informations stockées dans Memcached.

Suivez la section pour renforcer le système utilisant le service Memcached contre d'éventuelles attaques DDoS.

8.7.1. Renforcer Memcached contre les DDoS

Pour réduire les risques de sécurité, effectuez autant d'étapes que possible en fonction de votre configuration.

Procédure

- Configurez un pare-feu dans votre réseau local. Si votre serveur Memcached ne doit être accessible que dans votre réseau local, n'acheminez pas le trafic externe vers les ports utilisés par le service **memcached**. Par exemple, supprimez le port par défaut **11211** de la liste des ports autorisés :

```
# firewall-cmd --remove-port=11211/udp
# firewall-cmd --runtime-to-permanent
```

- Si vous utilisez un seul serveur Memcached sur la même machine que votre application, configurez **memcached** pour qu'il n'écoute que le trafic local. Modifiez la valeur de **OPTIONS** dans le fichier **/etc/sysconfig/memcached**:

```
OPTIONS="-l 127.0.0.1,::1"
```

- Activer l'authentification SASL (Simple Authentication and Security Layer) :

  1. Modifier ou ajouter le fichier **/etc/sasl2/memcached.conf**:

```
sasldb_path: /path.to/memcached.sasldb
```

  2. Ajouter un compte dans la base de données SASL :

```
# saslpasswd2 -a memcached -c cacheuser -f /path.to/memcached.sasldb
```

  3. Assurez-vous que la base de données est accessible à l'utilisateur et au groupe **memcached**:

```
# chown memcached:memcached /path.to/memcached.sasldb
```

4. Activez le support SASL dans Memcached en ajoutant la valeur **-S** au paramètre **OPTIONS** dans le fichier **/etc/sysconfig/memcached**:

```
OPTIONS="-S"
```

5. Redémarrez le serveur Memcached pour appliquer les modifications :

```
# systemctl restart memcached
```

6. Ajoutez le nom d'utilisateur et le mot de passe créés dans la base de données SASL à la configuration du client Memcached de votre application.

- Crypter les communications entre les clients et les serveurs Memcached avec TLS :

1. Activez la communication cryptée entre les clients et les serveurs Memcached avec TLS en ajoutant la valeur **-Z** au paramètre **OPTIONS** dans le fichier **/etc/sysconfig/memcached**:

```
OPTIONS="-Z"
```

2. Ajoutez le chemin d'accès au fichier de la chaîne de certificats au format PEM à l'aide de l'option **-o ssl_chain_cert**.

3. Ajoutez un chemin d'accès au fichier de clé privée à l'aide de l'option **-o ssl_key**.

CHAPITRE 9. UTILISATION DE MACSEC POUR CRYPTER LE TRAFIC DE COUCHE 2 DANS LE MÊME RÉSEAU PHYSIQUE

Vous pouvez utiliser MACsec pour sécuriser la communication entre deux appareils (point à point). Par exemple, si votre succursale est reliée au bureau central par une connexion Metro-Ethernet, vous pouvez configurer MACsec sur les deux hôtes qui relient les bureaux afin d'accroître la sécurité.

Media Access Control Security (MACsec) est un protocole de couche 2 qui sécurise différents types de trafic sur les liaisons Ethernet, notamment :

- protocole de configuration dynamique de l'hôte (DHCP)
- le protocole de résolution d'adresses (ARP)
- Protocole Internet version 4 / 6 (**IPv4** / **IPv6**) et
- tout trafic sur IP tel que TCP ou UDP

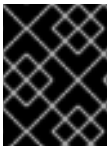
MACsec chiffre et authentifie tout le trafic dans les réseaux locaux, par défaut avec l'algorithme GCM-AES-128, et utilise une clé pré-partagée pour établir la connexion entre les hôtes participants. Si vous souhaitez modifier la clé pré-partagée, vous devez mettre à jour la configuration NM sur tous les hôtes du réseau qui utilisent MACsec.

Une connexion MACsec utilise un périphérique Ethernet, tel qu'une carte réseau Ethernet, un VLAN ou un périphérique tunnel, comme parent. Vous pouvez soit définir une configuration IP uniquement sur le périphérique MACsec pour communiquer avec d'autres hôtes uniquement à l'aide de la connexion cryptée, soit définir également une configuration IP sur le périphérique parent. Dans ce dernier cas, vous pouvez utiliser le dispositif parent pour communiquer avec d'autres hôtes en utilisant une connexion non chiffrée et le dispositif MACsec pour les connexions chiffrées.

MACsec ne nécessite pas de matériel particulier. Par exemple, vous pouvez utiliser n'importe quel commutateur, sauf si vous souhaitez crypter le trafic uniquement entre un hôte et un commutateur. Dans ce cas, le commutateur doit également prendre en charge MACsec.

En d'autres termes, il existe deux méthodes courantes pour configurer MACsec ;

- d'hôte à hôte et
- hôte à basculer puis basculer vers d'autres hôtes



IMPORTANT

Vous ne pouvez utiliser MACsec qu'entre des hôtes qui se trouvent dans le même réseau local (physique ou virtuel).

9.1. CONFIGURATION D'UNE CONNEXION MACSEC À L'AIDE DE NMCLI

Vous pouvez configurer les interfaces Ethernet pour utiliser MACsec à l'aide de l'utilitaire **nmcli**. Par exemple, vous pouvez créer une connexion MACsec entre deux hôtes connectés par Ethernet.

Procédure

1. Sur le premier hôte sur lequel vous configurez MACsec :

- Créez la clé d'association de connectivité (CAK) et le nom de la clé d'association de connectivité (CKN) pour la clé pré-partagée :

  a. Créer un CAK hexadécimal de 16 octets :

  ```
  # dd if=/dev/urandom count=16 bs=1 2> /dev/null | hexdump -e '1/2 "%04x"'
  50b71a8ef0bd5751ea76de6d6c98c03a
  ```

  b. Créer un CKN hexadécimal de 32 octets :

  ```
  # dd if=/dev/urandom count=32 bs=1 2> /dev/null | hexdump -e '1/2 "%04x"'
  f2b4297d39da7330910a74abc0449feb45b5c0b9fc23df1430e1898fcf1c4550
  ```

2. Sur les deux hôtes, vous souhaitez établir une connexion MACsec :

3. Créer la connexion MACsec :

   ```
   # nmcli connection add type macsec con-name macsec0 ifname macsec0
   connection.autoconnect yes macsec.parent enp1s0 macsec.mode psk macsec.mka-
   cak 50b71a8ef0bd5751ea76de6d6c98c03a macsec.mka-ckn
   f2b4297d39da7330910a7abc0449feb45b5c0b9fc23df1430e1898fcf1c4550
   ```

   Utilisez les CAK et CKN générés à l'étape précédente dans les paramètres **macsec.mka-cak** et **macsec.mka-ckn**. Les valeurs doivent être identiques sur chaque hôte du réseau protégé par MACsec.

4. Configurez les paramètres IP de la connexion MACsec.

   a. Configurez les paramètres **IPv4**. Par exemple, pour définir une adresse **IPv4** statique, un masque de réseau, une passerelle par défaut et un serveur DNS pour la connexion **macsec0**, entrez :

   ```
   # nmcli connection modify macsec0 ipv4.method manual ipv4.addresses
   '192.0.2.1/24' ipv4.gateway '192.0.2.254' ipv4.dns '192.0.2.253'
   ```

   b. Configurez les paramètres **IPv6**. Par exemple, pour définir une adresse **IPv6** statique, un masque de réseau, une passerelle par défaut et un serveur DNS pour la connexion **macsec0**, entrez :

   ```
   # nmcli connection modify macsec0 ipv6.method manual ipv6.addresses
   '2001:db8:1::1/32' ipv6.gateway '2001:db8:1::fffe' ipv6.dns '2001:db8:1::fffd'
   ```

5. Activer la connexion :

   ```
   # nmcli connection up macsec0
   ```

Vérification

1. Vérifiez que le trafic est crypté :

   ```
   # tcpdump -nn -i enp1s0
   ```

2. Facultatif : Affichez le trafic non crypté :

```
# tcpdump -nn -i macsec0
```

3. Affiche les statistiques MACsec :

```
# ip macsec show
```

4. Afficher les compteurs individuels pour chaque type de protection : intégrité seule (cryptage désactivé) et cryptage (cryptage activé)

```
# ip -s macsec show
```

9.2. RESSOURCES SUPPLÉMENTAIRES

- MACsec : une solution différente pour crypter le trafic réseau blog.

CHAPITRE 10. SÉCURISER LE SERVICE POSTFIX

Postfix est un agent de transfert de courrier (MTA) qui utilise le protocole SMTP (Simple Mail Transfer Protocol) pour transmettre des messages électroniques entre d'autres MTA et à des clients de messagerie ou à des agents de livraison. Bien que les MTA puissent crypter le trafic entre eux, ils peuvent ne pas le faire par défaut. Vous pouvez également réduire les risques d'attaques diverses en remplaçant les paramètres par des valeurs plus sûres.

10.1. RÉDUIRE LES RISQUES DE SÉCURITÉ LIÉS AU RÉSEAU POSTFIX

Pour réduire le risque d'intrusion de pirates dans votre système via le réseau, effectuez le plus grand nombre possible des tâches suivantes.

- Ne partagez pas le répertoire spool de **/var/spool/postfix/** sur un volume partagé NFS (Network File System). NFSv2 et NFSv3 n'assurent pas le contrôle des identifiants d'utilisateur et de groupe. Par conséquent, si deux utilisateurs ou plus ont le même UID, ils peuvent recevoir et lire le courrier des autres, ce qui constitue un risque pour la sécurité.

NOTE

Cette règle ne s'applique pas à NFSv4 utilisant Kerberos, car le module du noyau **SECRPC_GSS** n'utilise pas l'authentification basée sur l'UID. Toutefois, pour réduire les risques de sécurité, vous ne devez pas placer le répertoire du spool de courrier sur les volumes partagés NFS.

- Pour réduire la probabilité d'exploitation du serveur Postfix, les utilisateurs de messagerie doivent accéder au serveur Postfix à l'aide d'un programme de courrier électronique. N'autorisez pas les comptes shell sur le serveur de messagerie et définissez tous les shells des utilisateurs dans le fichier **/etc/passwd** sur **/sbin/nologin** (à l'exception peut-être de l'utilisateur **root**).

- Pour protéger Postfix d'une attaque réseau, il est configuré par défaut pour n'écouter que l'adresse loopback locale. Vous pouvez le vérifier en consultant la ligne **inet_interfaces = localhost** dans le fichier **/etc/postfix/main.cf**. Cela garantit que Postfix n'accepte que les messages (tels que les rapports de travail **cron**) provenant du système local et non du réseau. Il s'agit du paramètre par défaut, qui protège Postfix d'une attaque réseau. Pour supprimer la restriction relative à l'hôte local et permettre à Postfix d'écouter sur toutes les interfaces, définissez le paramètre **inet_interfaces** à **all** dans **/etc/postfix/main.cf**.

10.2. OPTIONS DE CONFIGURATION DE POSTFIX POUR LIMITER LES ATTAQUES DOS

Un attaquant peut inonder le serveur de trafic ou envoyer des informations qui déclenchent une panne, provoquant ainsi une attaque par déni de service (DoS). Vous pouvez configurer votre système pour réduire le risque de telles attaques en définissant des limites dans le fichier **/etc/postfix/main.cf**. Vous pouvez modifier la valeur des directives existantes ou ajouter de nouvelles directives avec des valeurs personnalisées dans le format *<directive> = <value>*.

Utilisez la liste suivante de directives pour limiter une attaque DoS :

smtpd_client_connection_rate_limit

Cette directive limite le nombre maximum de tentatives de connexion qu'un client peut faire à ce service par unité de temps. La valeur par défaut est **0**, ce qui signifie qu'un client peut effectuer autant de connexions par unité de temps que Postfix peut en accepter. Par défaut, la directive exclut

les clients des réseaux de confiance.

unité_de_taux_de_temps_de_l'enclume

Cette directive indique l'unité de temps utilisée pour calculer la limite de vitesse. La valeur par défaut est **60** seconds.

smtpd_client_event_limit_exceptions

Cette directive exclut les clients des commandes de connexion et de limitation de débit. Par défaut, la directive exclut les clients des réseaux de confiance.

smtpd_client_message_rate_limit

Cette directive définit le nombre maximal de livraisons de messages du client à la demande par unité de temps (que Postfix accepte ou non ces messages).

limite_de_processus_par_défaut

Cette directive définit le nombre maximal par défaut de processus enfants Postfix qui fournissent un service donné. Vous pouvez ignorer cette règle pour des services spécifiques dans le fichier **master.cf**. Par défaut, la valeur est **100**.

queue_minfree

Cette directive définit l'espace libre minimum requis pour recevoir du courrier dans le système de fichiers de la file d'attente. Elle est actuellement utilisée par le serveur SMTP Postfix pour décider s'il accepte ou non du courrier. Par défaut, le serveur SMTP Postfix rejette les commandes **MAIL FROM** lorsque l'espace libre est inférieur à 1,5 fois la valeur **message_size_limit**. Pour spécifier une limite minimale d'espace libre plus élevée, spécifiez une valeur **queue_minfree** qui est au moins 1,5 fois la valeur **message_size_limit**. Par défaut, la valeur **queue_minfree** est **0**.

limite_de_taille_de_l'en-tête

Cette directive définit la quantité maximale de mémoire en octets pour le stockage d'un en-tête de message. Si l'en-tête est trop volumineux, le système rejette l'en-tête excédentaire. Par défaut, la valeur est de **102400** bytes.

limite_de_taille_du_message

Cette directive définit la taille maximale d'un message, y compris les informations de l'enveloppe, en octets. Par défaut, la valeur est de **10240000** bytes.

10.3. CONFIGURATION DE POSTFIX POUR L'UTILISATION DE SASL

Postfix prend en charge l'authentification SMTP (AUTH) basée sur la couche d'authentification et de sécurité simple (SASL). SMTP AUTH est une extension du protocole de transfert de courrier simple. Actuellement, le serveur SMTP Postfix prend en charge les implémentations SASL de la manière suivante :

Dovecot SASL

Le serveur SMTP Postfix peut communiquer avec l'implémentation SASL de Dovecot en utilisant soit une socket UNIX-domain, soit une socket TCP. Utilisez cette méthode si les applications Postfix et Dovecot tournent sur des machines séparées.

Cyrus SASL

Lorsque cette option est activée, les clients SMTP doivent s'authentifier auprès du serveur SMTP à l'aide d'une méthode d'authentification prise en charge et acceptée à la fois par le serveur et le client.

Conditions préalables

- Le paquet **dovecot** est installé sur le système

Procédure

Procédure

1. Configurer Dovecot :

   a. Inclure les lignes suivantes dans le fichier **/etc/dovecot/conf.d/10-master.conf**:

   ```
   service auth {
     unix_listener /var/spool/postfix/private/auth {
       mode = 0660
       user = postfix
       group = postfix
     }
   }
   ```

   L'exemple précédent utilise des sockets de domaine UNIX pour la communication entre Postfix et Dovecot. L'exemple suppose également des paramètres de serveur SMTP Postfix par défaut, qui incluent la file d'attente de courrier située dans le répertoire **/var/spool/postfix/**, et l'application fonctionnant sous l'utilisateur et le groupe **postfix**.

   b. Optionnel : Configurer Dovecot pour qu'il écoute les demandes d'authentification de Postfix via TCP :

   ```
   service auth {
     inet_listener {
         port = port-number
     }
   }
   ```

   c. Spécifiez la méthode que le client de messagerie utilise pour s'authentifier avec Dovecot en éditant le paramètre **auth_mechanisms** dans le fichier **/etc/dovecot/conf.d/10-auth.conf**:

   ```
   auth_mechanisms = plain login
   ```

   Le paramètre **auth_mechanisms** prend en charge différentes méthodes d'authentification en texte clair et en texte non clair.

2. Configurez Postfix en modifiant le fichier **/etc/postfix/main.cf**:

   a. Activer l'authentification SMTP sur le serveur SMTP Postfix :

   ```
   smtpd_sasl_auth_enable = yes
   ```

   b. Activer l'utilisation de l'implémentation SASL de Dovecot pour l'authentification SMTP :

   ```
   smtpd_sasl_type = dovecot
   ```

   c. Indiquez le chemin d'authentification relatif au répertoire de la file d'attente Postfix. Notez que l'utilisation d'un chemin relatif garantit que la configuration fonctionne indépendamment du fait que le serveur Postfix tourne sur **chroot** ou non :

   ```
   smtpd_sasl_path = private/auth
   ```

   Cette étape utilise des sockets de domaine UNIX pour la communication entre Postfix et Dovecot.

Pour configurer Postfix afin qu'il recherche Dovecot sur une autre machine dans le cas où vous utilisez des sockets TCP pour la communication, utilisez des valeurs de configuration similaires à ce qui suit :

```
smtpd_sasl_path = inet : ip-address: port-number
```

Dans l'exemple précédent, remplacez *ip-address* par l'adresse IP de la machine Dovecot et *port-number* par le numéro de port spécifié dans le fichier **/etc/dovecot/conf.d/10-master.conf** de Dovecot.

d. Spécifiez les mécanismes SASL que le serveur SMTP Postfix met à la disposition des clients. Notez que vous pouvez spécifier des mécanismes différents pour les sessions cryptées et non cryptées.

```
smtpd_sasl_security_options = noanonymous, noplaintext
smtpd_sasl_tls_security_options = noanonymous
```

Les directives précédentes précisent que lors des sessions non chiffrées, aucune authentification anonyme n'est autorisée et qu'aucun mécanisme transmettant des noms d'utilisateur ou des mots de passe non chiffrés n'est autorisé. Pour les sessions cryptées utilisant TLS, seuls les mécanismes d'authentification non anonymes sont autorisés.

Ressources supplémentaires

- Politique du serveur SMTP Postfix - Propriétés du mécanisme SASL
- Postfix et Dovecot SASL
- Configuration de l'authentification SASL dans le serveur SMTP Postfix