Red Hat Enterprise Linux 9

Déploiement de serveurs web et de serveurs mandataires (reverse proxies)

Mise en place et configuration de serveurs web et de proxys inversés dans Red Hat Enterprise Linux 9

Last Updated: 2024-02-15

Red Hat Enterprise Linux 9 Déploiement de serveurs web et de serveurs mandataires (reverse proxies)

Mise en place et configuration de serveurs web et de proxys inversés dans Red Hat Enterprise Linux 9

Notice légale

Résumé

Configurez et exécutez le serveur web Apache HTTP, le serveur web NGINX ou le serveur proxy de mise en cache Squid sur Red Hat Enterprise Linux 9. Configurez le cryptage TLS. Configurer l'authentification Kerberos pour le serveur web Apache HTTP. Configurer NGINX en tant que proxy inverse pour le trafic HTTP ou en tant qu'équilibreur de charge HTTP. Configurer Squid comme proxy de mise en cache sans authentification, avec authentification LDAP ou avec authentification Kerberos.

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. CONFIGURATION DU SERVEUR WEB APACHE HTTP

1.1. INTRODUCTION AU SERVEUR WEB APACHE HTTP

Un site *web server* est un service de réseau qui fournit du contenu à un client sur le web. Il s'agit généralement de pages web, mais tout autre document peut également être servi. Les serveurs web sont également connus sous le nom de serveurs HTTP, car ils utilisent le protocole *hypertext transport protocol* (**HTTP**).

Le **Apache HTTP Server**, **httpd**, est un serveur web open source développé par la Apache Software Foundation.

Si vous mettez à niveau une version précédente de Red Hat Enterprise Linux, vous devez mettre à jour la configuration du service **httpd** en conséquence. Cette section passe en revue certaines des fonctionnalités nouvellement ajoutées et vous guide dans la mise à jour des fichiers de configuration antérieurs.

1.2. CHANGEMENTS NOTABLES DANS LE SERVEUR HTTP APACHE

RHEL 9 fournit la version 2.4.48 du serveur HTTP Apache. Les changements notables par rapport à la version 2.4.37 distribuée avec RHEL 8 sont les suivants :

- Interface de contrôle du serveur HTTP Apache (**apachectl**) :
  - Le pager **systemctl** est maintenant désactivé pour la sortie **apachectl status**.
  - La commande **apachectl** échoue maintenant au lieu de donner un avertissement si vous passez des arguments supplémentaires.
  - La commande **apachectl graceful-stop** revient maintenant immédiatement.
  - La commande **apachectl configtest** exécute désormais la commande **httpd -t** sans modifier le contexte SELinux.
  - La page de manuel **apachectl(8)** de RHEL documente désormais pleinement les différences avec la version amont **apachectl**.
- Outil Apache eXtenSion (**apxs**) :
  - La commande **/usr/bin/apxs** n'utilise ni n'expose plus les drapeaux d'optimisation du compilateur tels qu'ils sont appliqués lors de la construction du paquet **httpd**. Vous pouvez maintenant utiliser la commande **/usr/lib64/httpd/build/vendor-apxs** pour appliquer les mêmes drapeaux de compilateur que ceux utilisés pour construire **httpd**. Pour utiliser la commande **vendor-apxs**, vous devez d'abord installer le paquetage **redhat-rpm-config**.
- Modules Apache :
  - Le module **mod_lua** est désormais fourni dans un paquet séparé.
- Modifications de la syntaxe de configuration :
  - Dans la directive obsolète **Allow** fournie par le module **mod_access_compat**, un commentaire (le caractère **#**) déclenche désormais une erreur de syntaxe au lieu d'être ignoré silencieusement.

- Autres modifications :
  - Les identifiants des threads du noyau sont désormais utilisés directement dans les messages du journal des erreurs, ce qui les rend à la fois précis et plus concis.
  - Nombreuses améliorations mineures et corrections de bugs.
  - Un certain nombre de nouvelles interfaces sont à la disposition des auteurs de modules.

L'API du module **httpd** n'a fait l'objet d'aucune modification rétrocompatible depuis RHEL 8.

Apache HTTP Server 2.4 est la version initiale de ce flux d'applications, que vous pouvez installer facilement sous la forme d'un paquetage RPM.

1.3. LES FICHIERS DE CONFIGURATION D'APACHE

Par défaut, le site **httpd** lit les fichiers de configuration après le démarrage. Vous pouvez voir la liste des emplacements des fichiers de configuration dans le tableau ci-dessous.

Tableau 1.1. Les fichiers de configuration du service httpd

Chemin d'accès	Description
/etc/httpd/conf/httpd.conf	Le fichier de configuration principal.
/etc/httpd/conf.d/	Un répertoire auxiliaire pour les fichiers de configuration qui sont inclus dans le fichier de configuration principal.
/etc/httpd/conf.modules.d/	Un répertoire auxiliaire pour les fichiers de configuration qui chargent les modules dynamiques installés dans Red Hat Enterprise Linux. Dans la configuration par défaut, ces fichiers de configuration sont traités en premier.

Bien que la configuration par défaut convienne à la plupart des situations, vous pouvez également utiliser d'autres options de configuration. Pour que les modifications soient prises en compte, redémarrez d'abord le serveur web.

Pour vérifier que la configuration ne contient pas d'erreurs, tapez ce qui suit à l'invite d'un shell :

```
# apachectl configtest
Syntax OK
```

Pour faciliter la récupération des erreurs, faites une copie du fichier original avant de le modifier.

1.4. GESTION DU SERVICE HTTPD

Cette section décrit comment démarrer, arrêter et redémarrer le service **httpd**.

Conditions préalables

- Le serveur HTTP Apache est installé.

Procédure

- Pour démarrer le service **httpd**, entrez

```
# systemctl start httpd
```

- Pour arrêter le service **httpd**, entrez

```
# systemctl stop httpd
```

- Pour redémarrer le service **httpd**, entrez :

```
# systemctl restart httpd
```

1.5. CONFIGURATION D'UNE INSTANCE UNIQUE DU SERVEUR HTTP APACHE

Cette section décrit comment configurer un serveur HTTP Apache à instance unique pour servir du contenu HTML statique.

Suivez la procédure décrite dans cette section si le serveur web doit fournir le même contenu à tous les domaines associés au serveur. Si vous souhaitez fournir un contenu différent pour différents domaines, configurez des hôtes virtuels basés sur le nom. Pour plus d'informations, voir Configuration des hôtes virtuels Apache basés sur le nom.

Procédure

1. Installez le paquetage **httpd**:

```
# dnf install httpd
```

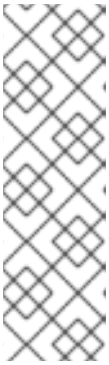
2. Si vous utilisez **firewalld**, ouvrez le port TCP **80** dans le pare-feu local :

```
# firewall-cmd --permanent --add-port=80/tcp
# firewall-cmd --reload
```

3. Activez et démarrez le service **httpd**:

```
# systemctl enable --now httpd
```

4. Facultatif : Ajoutez des fichiers HTML au répertoire **/var/www/html/**.



NOTE

Lors de l'ajout de contenu à **/var/www/html/**, les fichiers et les répertoires doivent être lisibles par l'utilisateur sous lequel **httpd** s'exécute par défaut. Le propriétaire du contenu peut être soit l'utilisateur **root** et le groupe d'utilisateurs **root**, soit un autre utilisateur ou groupe au choix de l'administrateur. Si le propriétaire du contenu est l'utilisateur **root** et le groupe d'utilisateurs **root**, les fichiers doivent pouvoir être lus par d'autres utilisateurs. Le contexte SELinux pour tous les fichiers et répertoires doit être **httpd_sys_content_t**, qui est appliqué par défaut à tout le contenu du répertoire **/var/www**.

Verification steps

- Se connecter avec un navigateur web pour **http://*server_IP_or_host_name*/**.
  Si le répertoire **/var/www/html/** est vide ou ne contient pas de fichier **index.html** ou **index.htm**, Apache affiche le répertoire **Red Hat Enterprise Linux Test Page**. Si **/var/www/html/** contient des fichiers HTML portant un nom différent, vous pouvez les charger en saisissant l'URL de ce fichier, par exemple **http://*server_IP_or_host_name*/*example.html***.

Ressources supplémentaires

- Manuel Apache : Manuel d'installation du serveur HTTP Apache .
- Voir la page de manuel **httpd.service(8)**.

1.6. CONFIGURATION DES HÔTES VIRTUELS APACHE BASÉS SUR LE NOM

Les hôtes virtuels basés sur le nom permettent à Apache de servir un contenu différent pour différents domaines qui se résolvent à l'adresse IP du serveur.

La procédure de cette section décrit la mise en place d'un hôte virtuel pour les domaines **example.com** et **example.net** avec des répertoires racine distincts. Les deux hôtes virtuels servent un contenu HTML statique.

Conditions préalables

- Les clients et le serveur web résolvent les domaines **example.com** et **example.net** en fonction de l'adresse IP du serveur web.
  Notez que vous devez ajouter manuellement ces entrées à votre serveur DNS.

Procédure

1. Installez le paquetage **httpd**:

   ```
   # dnf install httpd
   ```

2. Modifiez le fichier **/etc/httpd/conf/httpd.conf**:

   a. Ajoutez la configuration d'hôte virtuel suivante pour le domaine **example.com**:

   ```
   <VirtualHost *:80>
     DocumentRoot "/var/www/example.com/"
     ServerName example.com
     CustomLog /var/log/httpd/example.com_access.log combined
     ErrorLog /var/log/httpd/example.com_error.log
   </VirtualHost>
   ```

   Ces paramètres permettent de configurer les éléments suivants :

   - Tous les paramètres de la directive **<VirtualHost *:80>** sont spécifiques à cet hôte virtuel.
   - **DocumentRoot** définit le chemin d'accès au contenu web de l'hôte virtuel.

- **ServerName** définit les domaines pour lesquels cet hôte virtuel sert du contenu. Pour définir plusieurs domaines, ajoutez le paramètre **ServerAlias** à la configuration et indiquez les domaines supplémentaires en les séparant par un espace dans ce paramètre.
- **CustomLog** définit le chemin d'accès au journal d'accès de l'hôte virtuel.
- **ErrorLog** définit le chemin d'accès au journal des erreurs de l'hôte virtuel.

NOTE

Apache utilise le premier hôte virtuel trouvé dans la configuration également pour les requêtes qui ne correspondent à aucun domaine défini dans les paramètres **ServerName** et **ServerAlias**. Cela inclut également les requêtes envoyées à l'adresse IP du serveur.

3. Ajoutez une configuration d'hôte virtuel similaire pour le domaine **example.net**:

```
<VirtualHost *:80>
    DocumentRoot "/var/www/example.net/"
    ServerName example.net
    CustomLog /var/log/httpd/example.net_access.log combined
    ErrorLog /var/log/httpd/example.net_error.log
</VirtualHost>
```

4. Créez les racines des documents pour les deux hôtes virtuels :

```
# mkdir /var/www/example.com/
# mkdir /var/www/example.net/
```

5. Si vous définissez des chemins dans les paramètres **DocumentRoot** qui ne se trouvent pas dans **/var/www/**, définissez le contexte **httpd_sys_content_t** sur les deux racines du document :

```
# semanage fcontext -a -t httpd_sys_content_t "/srv/example.com(/.*)?"
# restorecon -Rv /srv/example.com/
# semanage fcontext -a -t httpd_sys_content_t "/srv/example.net(/.\*)?"
# restorecon -Rv /srv/example.net/
```

Ces commandes définissent le contexte **httpd_sys_content_t** sur les répertoires **/srv/example.com/** et **/srv/example.net/**.

Notez que vous devez installer le paquetage **policycoreutils-python-utils** pour exécuter la commande **restorecon**.

6. Si vous utilisez **firewalld**, ouvrez le port **80** dans le pare-feu local :

```
# firewall-cmd --permanent --add-port=80/tcp
# firewall-cmd --reload
```

7. Activez et démarrez le service **httpd**:

```
# systemctl enable --now httpd
```

Verification steps

1. Créez un fichier d'exemple différent dans la racine du document de chaque hôte virtuel :

```
# echo "vHost example.com" > /var/www/example.com/index.html
# echo "vHost example.net" > /var/www/example.net/index.html
```

2. Utilisez un navigateur et connectez-vous à **http://example.com**. Le serveur web affiche le fichier d'exemple de l'hôte virtuel **example.com**.

3. Utilisez un navigateur et connectez-vous à **http://example.net**. Le serveur web affiche le fichier d'exemple de l'hôte virtuel **example.net**.

Ressources supplémentaires

- Manuel d'installation du serveur Apache HTTP - Hôtes virtuels

1.7. CONFIGURATION DE L'AUTHENTIFICATION KERBEROS POUR LE SERVEUR WEB APACHE HTTP

Pour effectuer l'authentification Kerberos dans le serveur web HTTP Apache, RHEL 9 utilise le module Apache **mod_auth_gssapi**. Le Generic Security Services API (**GSSAPI**) est une interface pour les applications qui demandent à utiliser des bibliothèques de sécurité, telles que Kerberos. Le service **gssproxy** permet de mettre en œuvre la séparation des privilèges pour le serveur **httpd**, ce qui optimise ce processus du point de vue de la sécurité.

NOTE

Le module **mod_auth_gssapi** remplace le module **mod_auth_kerb** qui a été retiré.

Conditions préalables

- Les **httpd**, **mod_auth_gssapi** et **gssproxy** sont installés.
- Le serveur web Apache est installé et le service **httpd** est en cours d'exécution.

1.7.1. Mise en place de GSS-Proxy dans un environnement IdM

Cette procédure décrit comment configurer **GSS-Proxy** pour effectuer l'authentification Kerberos dans le serveur web Apache HTTP.

Procédure

1. Autoriser l'accès au fichier **keytab** du principal HTTP/<SERVER_NAME>@realm en créant le principal de service :

```
# ipa service-add HTTP/<SERVER_NAME>
```

2. Récupérer le site **keytab** pour le mandant stocké dans le fichier **/etc/gssproxy/http.keytab**:

```
# ipa-getkeytab -s $(awk '/^server =/ {print $3}' /etc/ipa/default.conf) -k
/etc/gssproxy/http.keytab -p HTTP/$(hostname -f)
```

Cette étape définit les autorisations à 400, de sorte que seul l'utilisateur **root** a accès au fichier **keytab**. L'utilisateur **apache** n'y a pas accès.

3. Créez le fichier **/etc/gssproxy/80-httpd.conf** avec le contenu suivant :

```
[service/HTTP]
  mechs = krb5
  cred_store = keytab:/etc/gssproxy/http.keytab
  cred_store = ccache:/var/lib/gssproxy/clients/krb5cc_%U
  euid = apache
```

4. Redémarrez et activez le service **gssproxy**:

```
# systemctl restart gssproxy.service
# systemctl enable gssproxy.service
```

Ressources supplémentaires

- **gssproxy(8)** pages de manuel
- **gssproxy-mech(8)** pages de manuel
- **gssproxy.conf(5)** pages de manuel

1.7.2. Configuration de l'authentification Kerberos pour un répertoire partagé par le serveur web Apache HTTP

Cette procédure décrit comment configurer l'authentification Kerberos pour le répertoire **/var/www/html/private/**.

Conditions préalables

- Le service **gssproxy** est configuré et fonctionne.

Procédure

1. Configurer le module **mod_auth_gssapi** pour protéger le répertoire **/var/www/html/private/**:

```
<Location /var/www/html/private>
  AuthType GSSAPI
  AuthName "GSSAPI Login"
  Require valid-user
</Location>
```

2. Créez le fichier **/etc/systemd/system/httpd.service** avec le contenu suivant :

```
.include /lib/systemd/system/httpd.service
[Service]
Environment=GSS_USE_PROXY=1
```

3. Recharger la configuration de **systemd**:

```
# systemctl daemon-reload
```

4. Redémarrez le service **httpd**:

```
# systemctl restart httpd.service
```

Verification steps

1. Obtenir un ticket Kerberos :

```
# kinit
```

2. Ouvrez l'URL du répertoire protégé dans un navigateur.

1.8. CONFIGURATION DU CRYPTAGE TLS SUR UN SERVEUR HTTP APACHE

Par défaut, Apache fournit du contenu aux clients en utilisant une connexion HTTP non chiffrée. Cette section explique comment activer le chiffrement TLS et configurer les paramètres de chiffrement les plus fréquemment utilisés sur un serveur HTTP Apache.

Conditions préalables

- Le serveur HTTP Apache est installé et fonctionne.

1.8.1. Ajouter le cryptage TLS à un serveur HTTP Apache

Cette section décrit comment activer le cryptage TLS sur un serveur HTTP Apache pour le domaine **example.com**.

Conditions préalables

- Le serveur HTTP Apache est installé et fonctionne.
- La clé privée est stockée dans le fichier **/etc/pki/tls/private/example.com.key**.
  Pour plus de détails sur la création d'une clé privée et d'une demande de signature de certificat (CSR), ainsi que sur la manière de demander un certificat à une autorité de certification (AC), reportez-vous à la documentation de votre AC. Par ailleurs, si votre autorité de certification prend en charge le protocole ACME, vous pouvez utiliser le module **mod_md** pour automatiser la récupération et le provisionnement des certificats TLS.
- Le certificat TLS est stocké dans le fichier **/etc/pki/tls/certs/example.com.crt**. Si vous utilisez un chemin différent, adaptez les étapes correspondantes de la procédure.
- Le certificat CA est stocké dans le fichier **/etc/pki/tls/certs/ca.crt**. Si vous utilisez un chemin différent, adaptez les étapes correspondantes de la procédure.
- Les clients et le serveur web transforment le nom d'hôte du serveur en adresse IP du serveur web.

Procédure

1. Installez le paquetage **mod_ssl**:

```
# dnf install mod_ssl
```

2. Modifiez le fichier **/etc/httpd/conf.d/ssl.conf** et ajoutez les paramètres suivants à la directive **<VirtualHost _default_:443>**:

   a. Définir le nom du serveur :

   ```
   ServerName example.com
   ```

   IMPORTANT

   Le nom du serveur doit correspondre à l'entrée définie dans le champ **Common Name** du certificat.

   b. Facultatif : Si le certificat contient des noms d'hôtes supplémentaires dans le champ **Subject Alt Names** (SAN), vous pouvez configurer **mod_ssl** pour qu'il fournisse également un cryptage TLS pour ces noms d'hôtes. Pour ce faire, ajoutez le paramètre **ServerAliases** avec les noms correspondants :

   ```
   ServerAlias www.example.com server.example.com
   ```

   c. Définissez les chemins d'accès à la clé privée, au certificat du serveur et au certificat de l'autorité de certification :

   ```
   SSLCertificateKeyFile "/etc/pki/tls/private/example.com.key"
   SSLCertificateFile "/etc/pki/tls/certs/example.com.crt"
   SSLCACertificateFile "/etc/pki/tls/certs/ca.crt"
   ```

3. Pour des raisons de sécurité, configurez l'accès au fichier de la clé privée uniquement pour l'utilisateur **root**:

   ```
   # chown root:root /etc/pki/tls/private/example.com.key
   # chmod 600 /etc/pki/tls/private/example.com.key
   ```

   AVERTISSEMENT

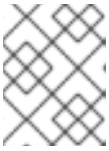
   Si des utilisateurs non autorisés ont eu accès à la clé privée, révoquez le certificat, créez une nouvelle clé privée et demandez un nouveau certificat. Sinon, la connexion TLS n'est plus sécurisée.

4. Si vous utilisez **firewalld**, ouvrez le port **443** dans le pare-feu local :

   ```
   # firewall-cmd --permanent --add-port=443/tcp
   # firewall-cmd --reload
   ```

5. Redémarrez le service **httpd**:

   ```
   # systemctl restart httpd
   ```



NOTE

Si vous avez protégé le fichier de clé privée par un mot de passe, vous devez saisir ce mot de passe à chaque démarrage du service **httpd**.

Verification steps

- Utilisez un navigateur et connectez-vous à **https://*example.com***.

Ressources supplémentaires

- Chiffrement SSL/TLS
- Considérations de sécurité pour TLS dans RHEL 9

1.8.2. Définition des versions du protocole TLS prises en charge sur un serveur HTTP Apache

Par défaut, le serveur HTTP Apache sur RHEL utilise la politique cryptographique du système qui définit des valeurs par défaut sûres, qui sont également compatibles avec les navigateurs récents. Par exemple, la politique **DEFAULT** définit que seules les versions des protocoles **TLSv1.2** et **TLSv1.3** sont activées dans Apache.

Cette section décrit comment configurer manuellement les versions du protocole TLS prises en charge par le serveur HTTP Apache. Suivez la procédure si votre environnement exige que seules certaines versions du protocole TLS soient activées, par exemple :

- Si votre environnement l'exige, les clients peuvent également utiliser le protocole faible **TLS1** (TLSv1.0) ou **TLS1.1**.
- Si vous souhaitez configurer Apache pour qu'il ne prenne en charge que le protocole **TLSv1.2** ou **TLSv1.3**.

Conditions préalables

- Le cryptage TLS est activé sur le serveur comme décrit dans Ajout du cryptage TLS à un serveur HTTP Apache.

Procédure

1. Modifiez le fichier **/etc/httpd/conf/httpd.conf** et ajoutez le paramètre suivant à la directive **<VirtualHost>** pour laquelle vous souhaitez définir la version du protocole TLS. Par exemple, pour activer uniquement le protocole **TLSv1.3**:

```
SSLProtocol -All TLSv1.3
```

2. Redémarrez le service **httpd**:

```
# systemctl restart httpd
```

Verification steps

1. Utilisez la commande suivante pour vérifier que le serveur prend en charge **TLSv1.3**:

```
# openssl s_client -connect example.com:443 -tls1_3
```

2. Utilisez la commande suivante pour vérifier que le serveur ne prend pas en charge **TLSv1.2**:

```
# openssl s_client -connect example.com:443 -tls1_2
```

Si le serveur ne prend pas en charge le protocole, la commande renvoie une erreur :

```
140111600609088:error:1409442E:Routines SSL:ssl3_read_bytes:version du protocole
d'alerte tlsv1:ssl/record/rec_layer_s3.c:1543:alerte SSL numéro 70
```

3. Facultatif : Répétez la commande pour d'autres versions du protocole TLS.

Ressources supplémentaires

- **update-crypto-policies(8)** page de manuel
- Utilisation de politiques cryptographiques à l'échelle du système .
- Pour plus de détails sur le paramètre **SSLProtocol**, reportez-vous à la documentation **mod_ssl** dans le manuel Apache : Manuel d'installation du serveur HTTP Ap ache.

1.8.3. Définition des algorithmes de chiffrement pris en charge sur un serveur HTTP Apache

Par défaut, le serveur HTTP Apache utilise la politique cryptographique du système qui définit des valeurs par défaut sûres, également compatibles avec les navigateurs récents. Pour obtenir la liste des algorithmes de chiffrement autorisés par la politique cryptographique du système, consultez le fichier **/etc/crypto-policies/back-ends/openssl.config**.

Cette section explique comment configurer manuellement les algorithmes de chiffrement pris en charge par le serveur HTTP Apache. Suivez la procédure si votre environnement requiert des algorithmes de chiffrement spécifiques.

Conditions préalables

- Le cryptage TLS est activé sur le serveur comme décrit dans Ajout du cryptage TLS à un serveur HTTP Apache.

Procédure

1. Modifiez le fichier **/etc/httpd/conf/httpd.conf** et ajoutez le paramètre **SSLCipherSuite** à la directive **<VirtualHost>** pour laquelle vous souhaitez définir les algorithmes TLS :

```
SSLCipherSuite "EECDH AESGCM:EDH AESGCM:AES256 EECDH:AES256
EDH:!SHA1:!SHA256"
```

Cet exemple n'active que les algorithmes de chiffrement **EECDH AESGCM**, **EDH AESGCM**, **AES256 EECDH** et **AES256 EDH** et désactive tous les algorithmes de chiffrement qui utilisent les codes d'authentification des messages (MAC) **SHA1** et **SHA256**.

2. Redémarrez le service **httpd**:

```
# systemctl restart httpd
```

Verification steps

1. Pour afficher la liste des algorithmes de chiffrement pris en charge par le serveur HTTP Apache :

   a. Installez le paquetage **nmap**:

   ```
   # dnf install nmap
   ```

   b. Utilisez l'utilitaire **nmap** pour afficher les algorithmes de chiffrement pris en charge :

   ```
   # nmap --script ssl-enum-ciphers -p 443 example.com
   ...
   PORT    STATE SERVICE
   443/tcp open  https
   | ssl-enum-ciphers:
   |   TLSv1.2:
   |     ciphers:
   |       TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (ecdh_x25519) - A
   |       TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (dh 2048) - A
   |       TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (ecdh_x25519) - A
   ...
   ```

Ressources supplémentaires

- **update-crypto-policies(8)** page de manuel
- Utilisation de politiques cryptographiques à l'échelle du système .
- SSLCipherSuite

1.9. CONFIGURATION DE L'AUTHENTIFICATION DU CERTIFICAT CLIENT TLS

L'authentification par certificat client permet aux administrateurs d'autoriser uniquement les utilisateurs qui s'authentifient à l'aide d'un certificat à accéder aux ressources du serveur web. Cette section explique comment configurer l'authentification par certificat client pour l'annuaire **/var/www/html/Example/**.

Si le serveur HTTP Apache utilise le protocole TLS 1.3, certains clients nécessitent une configuration supplémentaire. Par exemple, dans Firefox, définissez le paramètre **security.tls.enable_post_handshake_auth** dans le menu **about:config** sur **true**. Pour plus de détails, voir Transport Layer Security version 1.3 dans Red Hat Enterprise Linux 8 .

Conditions préalables

- Le cryptage TLS est activé sur le serveur comme décrit dans Ajout du cryptage TLS à un serveur HTTP Apache.

Procédure

1. Modifiez le fichier **/etc/httpd/conf/httpd.conf** et ajoutez les paramètres suivants à la directive **<VirtualHost>** pour laquelle vous souhaitez configurer l'authentification du client :

```
<Directory "/var/www/html/Example/">
  SSLVerifyClient require
</Directory>
```

Le paramètre **SSLVerifyClient require** définit que le serveur doit valider avec succès le certificat du client avant que ce dernier puisse accéder au contenu du répertoire **/var/www/html/Example/**.

2. Redémarrez le service **httpd**:

```
# systemctl restart httpd
```

Verification steps

1. Utilisez l'utilitaire **curl** pour accéder à l'URL **https://example.com/Example/** sans authentification du client :

```
$ curl https://example.com/Example/
curl: (56) OpenSSL SSL_read: error:1409445C:SSL routines:ssl3_read_bytes:tlsv13 **alert certificate required**, errno 0
```

L'erreur indique que le serveur web nécessite une authentification par certificat client.

2. Transmettez la clé privée et le certificat du client, ainsi que le certificat de l'autorité de certification à **curl** pour accéder à la même URL avec l'authentification du client :

```
$ curl --cacert ca.crt --key client.key --cert client.crt https://example.com/Example/
```

Si la demande aboutit, **curl** affiche le fichier **index.html** stocké dans le répertoire **/var/www/html/Example/**.

Ressources supplémentaires

- configuration de mod_ssl

1.10. SÉCURISATION DES APPLICATIONS WEB SUR UN SERVEUR WEB À L'AIDE DE MODSECURITY

ModSecurity est un pare-feu d'application web (WAF) open source supporté par divers serveurs web tels qu'Apache, Nginx et IIS, qui réduit les risques de sécurité dans les applications web. ModSecurity fournit des ensembles de règles personnalisables pour configurer votre serveur.

Le paquet **mod_security-crs** contient l'ensemble de règles de base (CRS) avec des règles contre les scripts intersites, les mauvais agents utilisateurs, les injections SQL, les chevaux de Troie, les détournements de session et d'autres exploits.

1.10.1. Déployer le pare-feu applicatif web ModSecurity pour Apache

Pour réduire les risques liés à l'exécution d'applications web sur votre serveur web en déployant ModSecurity, installez les paquets **mod_security** et **mod_security_crs** pour le serveur HTTP Apache.

Le paquet **mod_security_crs** fournit l'ensemble de règles de base (CRS) pour le module ModSecurity de pare-feu applicatif basé sur le web (WAF).

Procédure

1. Installez les paquets **mod_security**, **mod_security_crs**, et **httpd**:

```
# dnf install -y mod_security mod_security_crs httpd
```

2. Démarrez le serveur **httpd**:

```
# systemctl restart httpd
```

Vérification

1. Vérifiez que le pare-feu d'application web ModSecurity est activé sur votre serveur HTTP Apache :

```
# httpd -M | grep security
 security2_module (shared)
```

2. Vérifiez que le répertoire **/etc/httpd/modsecurity.d/activated_rules/** contient les règles fournies par **mod_security_crs**:

```
# ls /etc/httpd/modsecurity.d/activated_rules/
...
REQUEST-921-PROTOCOL-ATTACK.conf
REQUEST-930-APPLICATION-ATTACK-LFI.conf
...
```

Ressources supplémentaires

- Red Hat JBoss Core Services Guide ModSecurity
- Une introduction aux pare-feux d'application web pour les administrateurs système de Linux

1.10.2. Ajouter une règle personnalisée à ModSecurity

Si les règles contenues dans l'ensemble de règles de base (CRS) de ModSecurity ne correspondent pas à votre scénario et si vous souhaitez prévenir d'autres attaques possibles, vous pouvez ajouter vos propres règles à l'ensemble de règles utilisé par le pare-feu applicatif basé sur le web de ModSecurity. L'exemple suivant illustre l'ajout d'une règle simple. Pour créer des règles plus complexes, voir le manuel de référence sur le site web ModSecurity Wiki.

Conditions préalables

- ModSecurity pour Apache est installé et activé.

Procédure

1. Ouvrez le fichier **/etc/httpd/conf.d/mod_security.conf** dans un éditeur de texte de votre choix, par exemple :

```
# vi /etc/httpd/conf.d/mod_security.conf
```

2. Ajoutez l'exemple de règle suivant après la ligne commençant par **SecRuleEngine On**:

```
SecRule ARGS:data "@contains evil" "deny,status:403,msg:'param data contains evil data',id:1"
```

La règle précédente interdit l'utilisation de ressources à l'utilisateur si le paramètre **data** contient la chaîne **evil**.

3. Enregistrez les modifications et quittez l'éditeur.

4. Redémarrez le serveur **httpd**:

```
# systemctl restart httpd
```

Vérification

1. Créer une ***test*.html** page :

```
# echo "mod_security test" > /var/www/html/test.html
```

2. Redémarrez le serveur **httpd**:

```
# systemctl restart httpd
```

3. Demande ***test.html*** sans données malveillantes dans la variable **GET** de la requête HTTP :

```
$ curl http://localhost/test.html?data=good

mod_security test
```

4. Demande ***test.html*** avec des données malveillantes dans la variable **GET** de la requête HTTP :

```
$ curl localhost/test.html?data=xxxevilxxx

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
</body></html>
```

5. Vérifiez le fichier **/var/log/httpd/error_log** et localisez l'entrée du journal concernant le refus d'accès avec le message **param data containing an evil data**:

```
[Wed May 25 08:01:31.036297 2022] [:error] [pid 5839:tid 139874434791168] [client ::1:45658] [client ::1] ModSecurity: Access denied with code 403 (phase 2). String match "evil" at ARGS:data. [file \N- "/etc/httpd/conf.d/mod_security.conf\N"] [line \N- "4\N"] [id \N- "1\N"] [msg \N- "param data contains evil data\N"] [hostname \N- "localhost\N"] [uri \N- "/test.html\N"] [unique_id \N- "Yo4amwIdsBG3yZqSzh2GuwAAAIY\N"]
```

Ressources supplémentaires

- Wiki ModSecurity

1.11. INSTALLATION DU MANUEL DU SERVEUR HTTP APACHE

Cette section décrit comment installer le manuel du serveur HTTP Apache. Ce manuel fournit une documentation détaillée sur, par exemple :

- Paramètres et directives de configuration
- Optimisation des performances
- Paramètres d'authentification
- Modules
- Mise en cache du contenu
- Conseils de sécurité
- Configuration du cryptage TLS

Après avoir installé le manuel, vous pouvez l'afficher à l'aide d'un navigateur web.

Conditions préalables

- Le serveur HTTP Apache est installé et fonctionne.

Procédure

1. Installez le paquetage **httpd-manual**:

```
# dnf install httpd-manual
```

2. Facultatif : Par défaut, tous les clients se connectant au serveur HTTP Apache peuvent afficher le manuel. Pour restreindre l'accès à une plage IP spécifique, telle que le sous-réseau **192.0.2.0/24**, modifiez le fichier **/etc/httpd/conf.d/manual.conf** et ajoutez le paramètre **Require ip 192.0.2.0/24** à la directive **<Directory "/usr/share/httpd/manual">**:

```
<Directory "/usr/share/httpd/manual">
...
    **Require ip 192.0.2.0/24**
...
</Directory>
```

3. Redémarrez le service **httpd**:

```
# systemctl restart httpd
```

Verification steps

1. Pour afficher le manuel du serveur Apache HTTP, connectez-vous à l'aide d'un navigateur web à l'adresse suivante **http://*host_name_or_IP_address*/manual/**

1.12. TRAVAILLER AVEC LES MODULES APACHE

Le service **httpd** est une application modulaire, et vous pouvez l'étendre avec un certain nombre de *Dynamic Shared Objects* (**DSO**s). *Dynamic Shared Objects* sont des modules que vous pouvez charger ou décharger dynamiquement au moment de l'exécution, selon les besoins. Vous trouverez ces modules dans le répertoire **/usr/lib64/httpd/modules/**.

1.12.1. Chargement d'un module DSO

En tant qu'administrateur, vous pouvez choisir les fonctionnalités à inclure dans le serveur en configurant les modules que le serveur doit charger. Pour charger un module DSO particulier, utilisez la directive **LoadModule**. Notez que les modules fournis par un paquetage séparé ont souvent leur propre fichier de configuration dans le répertoire **/etc/httpd/conf.modules.d/**.

Conditions préalables

- Vous avez installé le paquetage **httpd**.

Procédure

1. Recherche du nom du module dans les fichiers de configuration du répertoire **/etc/httpd/conf.modules.d/**:

```
# grep mod_ssl.so /etc/httpd/conf.modules.d/*
```

2. Editez le fichier de configuration dans lequel le nom du module a été trouvé, et décommentez la directive **LoadModule** du module :

```
LoadModule ssl_module modules/mod_ssl.so
```

3. Si le module n'a pas été trouvé, par exemple parce qu'un paquetage RHEL ne fournit pas le module, créez un fichier de configuration, tel que **/etc/httpd/conf.modules.d/30-example.conf** avec la directive suivante :

```
LoadModule ssl_module modules/<custom_module>.so
```

4. Redémarrez le service **httpd**:

```
# systemctl restart httpd
```

1.12.2. Compilation d'un module Apache personnalisé

Vous pouvez créer votre propre module et le compiler à l'aide du paquetage **httpd-devel**, qui contient les fichiers include, les fichiers d'en-tête et l'utilitaire **APache eXtenSion** (**apxs**) nécessaires à la compilation d'un module.

Conditions préalables

- Vous avez installé le paquet **httpd-devel**.

Procédure

- Créez un module personnalisé à l'aide de la commande suivante :

```
# apxs -i -a -c module_name.c
```

Verification steps

- Chargez le module de la même manière que celle décrite dans Chargement d'un module DSO .

1.13. EXPORTATION D'UNE CLÉ PRIVÉE ET DE CERTIFICATS D'UNE BASE DE DONNÉES NSS POUR LES UTILISER DANS UNE CONFIGURATION DE SERVEUR WEB APACHE

Depuis RHEL 8, nous ne fournissons plus le module **mod_nss** pour le serveur web Apache, et Red Hat recommande d'utiliser le module **mod_ssl**. Si vous stockez votre clé privée et vos certificats dans une base de données NSS (Network Security Services), suivez cette procédure pour extraire la clé et les certificats au format PEM (Privacy Enhanced Mail).

1.14. RESSOURCES SUPPLÉMENTAIRES

- **httpd(8)** page de manuel
- **httpd.service(8)** page de manuel
- **httpd.conf(5)** page de manuel
- **apachectl(8)** page de manuel
- Authentification Kerberos sur un serveur HTTP Apache : Utilisation de GSS-Proxy pour le fonctionnement d'Apache httpd. L'utilisation de Kerberos est un moyen alternatif de renforcer l'autorisation du client sur un serveur HTTP Apache.
- Configurer des applications pour utiliser du matériel cryptographique via PKCS #11 .

CHAPITRE 2. MISE EN PLACE ET CONFIGURATION DE NGINX

NGINX est un serveur performant et modulaire que vous pouvez utiliser, par exemple, en tant que :

- Serveur web
- Proxy inversé
- Équilibreur de charge

Cette section décrit comment utiliser NGINX dans ces scénarios.

2.1. INSTALLATION ET PRÉPARATION DE NGINX

Red Hat utilise Application Streams pour fournir différentes versions de NGINX. Cette section décrit comment :

- Sélectionner un flux et installer NGINX
- Ouvrir les ports nécessaires dans le pare-feu
- Activer et démarrer le service **nginx**

Avec la configuration par défaut, NGINX s'exécute en tant que serveur web sur le port **80** et fournit du contenu à partir du répertoire **/usr/share/nginx/html/**.

Conditions préalables

- RHEL 9 est installé.
- L'hôte est abonné au portail client de Red Hat.
- Le service **firewalld** est activé et démarré.

Procédure

1. Installez le paquetage **nginx**:
   - Pour installer NGINX 1.20 en tant que version initiale de ce flux d'application à partir d'un paquetage RPM :

     ```
     # dnf install nginx
     ```

     NOTE

     Si vous avez précédemment activé un flux de modules NGINX, cette commande installe la version de NGINX à partir du flux activé.

   - Pour installer une version ultérieure de NGINX à partir d'un flux de modules :
     a. Affiche les flux de modules NGINX disponibles :

     ```
     # dnf module list nginx
     ...
     ```

```
rhel-AppStream
Name       Stream       Profiles       Summary
nginx      1.22         common [d]     nginx webserver
...
Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

b. Activer le flux sélectionné :

```
# dnf module enable nginx:stream_version
```

c. Installez le paquetage nginx :

```
# dnf install nginx
```

2. Ouvrez les ports sur lesquels NGINX doit fournir son service dans le pare-feu. Par exemple, pour ouvrir les ports par défaut pour HTTP (port 80) et HTTPS (port 443) dans **firewalld**, entrez :

```
# firewall-cmd --permanent --add-port={80/tcp,443/tcp}
# firewall-cmd --reload
```

3. Activez le service **nginx** pour qu'il démarre automatiquement lorsque le système démarre :

```
# systemctl enable nginx
```

4. Si nécessaire, démarrez le service **nginx**:

```
# systemctl start nginx
```

Si vous ne souhaitez pas utiliser la configuration par défaut, ignorez cette étape et configurez NGINX en conséquence avant de démarrer le service.

Verification steps

1. Utilisez l'utilitaire **dnf** pour vérifier que le paquet **nginx** est installé.

- Dans le cas du paquetage RPM NGINX 1.20 :

```
# dnf list installed nginx
Installed Packages
nginx.x86_64    1:1.20.1-4.el9       @rhel-AppStream
```

- Dans le cas d'un flux de modules NGINX sélectionné :

```
# dnf list installed nginx
Installed Packages
nginx.x86_64    1:1.22.1-3.module+el9.2.0+17617+2f289c6c    @rhel-AppStream
```

2. Assurez-vous que les ports sur lesquels NGINX doit fournir son service sont ouverts dans le firewalld :

```
# firewall-cmd --list-ports
80/tcp 443/tcp
```

3. Vérifiez que le service **nginx** est activé :

```
# systemctl is-enabled nginx
enabled
```

Ressources supplémentaires

- Utilisation et configuration du gestionnaire d'abonnements
- Sécurisation des réseaux

2.2. CONFIGURER NGINX COMME UN SERVEUR WEB QUI FOURNIT UN CONTENU DIFFÉRENT POUR DIFFÉRENTS DOMAINES

Par défaut, NGINX agit comme un serveur web qui fournit le même contenu aux clients pour tous les noms de domaine associés aux adresses IP du serveur. Cette procédure explique comment configurer NGINX :

- Pour servir les requêtes au domaine **example.com** avec le contenu du répertoire **/var/www/example.com/**
- Pour servir les requêtes au domaine **example.net** avec le contenu du répertoire **/var/www/example.net/**
- Servir toutes les autres demandes, par exemple à l'adresse IP du serveur ou à d'autres domaines associés à l'adresse IP du serveur, avec le contenu du répertoire **/usr/share/nginx/html/**

Conditions préalables

- NGINX est installé
- Les clients et le serveur web résolvent les domaines **example.com** et **example.net** en fonction de l'adresse IP du serveur web.
  Notez que vous devez ajouter manuellement ces entrées à votre serveur DNS.

Procédure

1. Modifiez le fichier **/etc/nginx/nginx.conf**:
   a. Par défaut, le fichier **/etc/nginx/nginx.conf** contient déjà une configuration "catch-all". Si vous avez supprimé cette partie de la configuration, ajoutez à nouveau le bloc **server** au bloc **http** dans le fichier **/etc/nginx/nginx.conf**:

```
server {
    listen       80 default_server;
    listen       [::]:80 default_server;
    server_name  _;
    root         /usr/share/nginx/html;
}
```

   Ces paramètres permettent de configurer les éléments suivants :

   - La directive **listen** définit l'adresse IP et les ports que le service écoute. Dans ce cas, NGINX écoute sur le port **80** sur toutes les adresses IPv4 et IPv6. Le paramètre

default_server indique que NGINX utilise ce bloc **server** par défaut pour les requêtes correspondant aux adresses IP et aux ports.

- Le paramètre **server_name** définit les noms d'hôtes dont ce bloc **server** est responsable. La configuration de **server_name** à _ permet à NGINX d'accepter n'importe quel nom d'hôte pour ce bloc **server**.
- La directive **root** définit le chemin d'accès au contenu web pour ce bloc **server**.

b. Ajoutez au bloc **http** un bloc **server** similaire pour le domaine **example.com**:

```
server {
    server_name  example.com;
    root         /var/www/example.com/;
    access_log   /var/log/nginx/example.com/access.log;
    error_log    /var/log/nginx/example.com/error.log;
}
```

- La directive **access_log** définit un fichier journal d'accès distinct pour ce domaine.
- La directive **error_log** définit un fichier journal des erreurs distinct pour ce domaine.

c. Ajoutez au bloc **http** un bloc **server** similaire pour le domaine **example.net**:

```
server {
    server_name  example.net;
    root         /var/www/example.net/;
    access_log   /var/log/nginx/example.net/access.log;
    error_log    /var/log/nginx/example.net/error.log;
}
```

2. Créez les répertoires racine pour les deux domaines :

```
# mkdir -p /var/www/example.com/
# mkdir -p /var/www/example.net/
```

3. Définir le contexte **httpd_sys_content_t** sur les deux répertoires racine :

```
# semanage fcontext -a -t httpd_sys_content_t "/var/www/example.com(/.*)?"
# restorecon -Rv /var/www/example.com/
# semanage fcontext -a -t httpd_sys_content_t "/var/www/example.net(/.\*)?"
# restorecon -Rv /var/www/example.net/
```

Ces commandes définissent le contexte **httpd_sys_content_t** sur les répertoires **/var/www/example.com/** et **/var/www/example.net/**.

Notez que vous devez installer le paquetage **policycoreutils-python-utils** pour exécuter les commandes **restorecon**.

4. Créez les répertoires de journaux pour les deux domaines :

```
# mkdir /var/log/nginx/example.com/
# mkdir /var/log/nginx/example.net/
```

5. Redémarrez le service **nginx**:

```
# systemctl restart nginx
```

Verification steps

1. Créez un fichier d'exemple différent dans la racine du document de chaque hôte virtuel :

```
# echo "Content for example.com" > /var/www/example.com/index.html
# echo "Content for example.net" > /var/www/example.net/index.html
# echo "Catch All content" > /usr/share/nginx/html/index.html
```

2. Utilisez un navigateur et connectez-vous à **http://example.com**. Le serveur web affiche l'exemple de contenu du fichier **/var/www/example.com/index.html**.

3. Utilisez un navigateur et connectez-vous à **http://example.net**. Le serveur web affiche l'exemple de contenu du fichier **/var/www/example.net/index.html**.

4. Utilisez un navigateur et connectez-vous à **http://*IP_address_of_the_server***. Le serveur web affiche l'exemple de contenu du fichier **/usr/share/nginx/html/index.html**.

2.3. AJOUTER LE CRYPTAGE TLS À UN SERVEUR WEB NGINX

Cette section décrit comment activer le cryptage TLS sur un serveur web NGINX pour le domaine **example.com**.

Conditions préalables

- NGINX est installé.
- La clé privée est stockée dans le fichier **/etc/pki/tls/private/example.com.key**.
  Pour plus de détails sur la création d'une clé privée et d'une demande de signature de certificat (CSR), ainsi que sur la manière de demander un certificat à une autorité de certification (AC), consultez la documentation de votre AC.
- Le certificat TLS est stocké dans le fichier **/etc/pki/tls/certs/example.com.crt**. Si vous utilisez un chemin différent, adaptez les étapes correspondantes de la procédure.
- Le certificat de l'autorité de certification a été ajouté au fichier de certificats TLS du serveur.
- Les clients et le serveur web transforment le nom d'hôte du serveur en adresse IP du serveur web.
- Le port **443** est ouvert dans le pare-feu local.

Procédure

1. Modifiez le fichier **/etc/nginx/nginx.conf** et ajoutez le bloc **server** suivant au bloc **http** dans la configuration :

```
server {
    listen              443 ssl;
    server_name         example.com;
    root                /usr/share/nginx/html;
```

```
    ssl_certificate     /etc/pki/tls/certs/example.com.crt;
    ssl_certificate_key /etc/pki/tls/private/example.com.key;
}
```

2. Pour des raisons de sécurité, configurez l'accès au fichier de la clé privée uniquement pour l'utilisateur **root**:

```
# chown root:root /etc/pki/tls/private/example.com.key
# chmod 600 /etc/pki/tls/private/example.com.key
```

AVERTISSEMENT

Si des utilisateurs non autorisés ont eu accès à la clé privée, révoquez le certificat, créez une nouvelle clé privée et demandez un nouveau certificat. Sinon, la connexion TLS n'est plus sécurisée.

3. Redémarrez le service **nginx**:

```
# systemctl restart nginx
```

Verification steps

- Utilisez un navigateur et connectez-vous à **https://example.com**

Ressources supplémentaires

- Considérations de sécurité pour TLS dans RHEL

2.4. CONFIGURER NGINX COMME PROXY INVERSE POUR LE TRAFIC HTTP

Vous pouvez configurer le serveur web NGINX pour qu'il agisse comme un proxy inverse pour le trafic HTTP. Par exemple, vous pouvez utiliser cette fonctionnalité pour transférer des requêtes vers un sous-répertoire spécifique sur un serveur distant. Du point de vue du client, celui-ci charge le contenu de l'hôte auquel il accède. Cependant, NGINX charge le contenu réel à partir du serveur distant et le transmet au client.

Cette procédure explique comment transférer le trafic vers le répertoire **/example** du serveur web à l'URL **https://example.com**.

Conditions préalables

- NGINX est installé comme décrit dans la section Installation et préparation de NGINX.
- Facultatif : Le cryptage TLS est activé sur le proxy inverse.

Procédure

1. Modifiez le fichier **/etc/nginx/nginx.conf** et ajoutez les paramètres suivants au bloc **server** qui doit fournir le proxy inverse :

```
location /example {
    proxy_pass https://example.com;
}
```

Le bloc **location** définit que NGINX transmet toutes les demandes dans le répertoire **/example** à **https://example.com**.

2. Définissez le paramètre booléen SELinux **httpd_can_network_connect** sur **1** pour configurer SELinux de manière à ce qu'il autorise NGINX à transmettre le trafic :

```
# setsebool -P httpd_can_network_connect 1
```

3. Redémarrez le service **nginx**:

```
# systemctl restart nginx
```

Verification steps

- Utilisez un navigateur et connectez-vous à **http://*host_name*/example** et le contenu de **https://example.com** s'affiche.

2.5. CONFIGURER NGINX EN TANT QU'ÉQUILIBREUR DE CHARGE HTTP

Vous pouvez utiliser la fonction de proxy inverse de NGINX pour équilibrer le trafic. Cette procédure décrit comment configurer NGINX en tant qu'équilibreur de charge HTTP qui envoie des requêtes à différents serveurs, en fonction de celui d'entre eux qui a le moins de connexions actives. Si les deux serveurs ne sont pas disponibles, la procédure définit également un troisième hôte pour des raisons de repli.

Conditions préalables

- NGINX est installé comme décrit dans la section Installation et préparation de NGINX.

Procédure

1. Modifiez le fichier **/etc/nginx/nginx.conf** et ajoutez les paramètres suivants :

```
http {
    upstream backend {
        least_conn;
        server server1.example.com;
        server server2.example.com;
        server server3.example.com backup;
    }

    server {
        location / {
            proxy_pass http://backend;
```

```
        }
    }
}
```

La directive **least_conn** dans le groupe d'hôtes nommé **backend** définit que NGINX envoie des requêtes à **server1.example.com** ou **server2.example.com**, selon l'hôte qui a le moins de connexions actives. NGINX utilise **server3.example.com** uniquement comme solution de secours au cas où les deux autres hôtes ne seraient pas disponibles.

Avec la directive **proxy_pass** définie sur **http://backend**, NGINX agit comme un proxy inverse et utilise le groupe d'hôtes **backend** pour distribuer les requêtes en fonction des paramètres de ce groupe.

Au lieu de la méthode d'équilibrage de charge **least_conn**, vous pouvez spécifier :

- Pas de méthode permettant d'utiliser la méthode round robin et de répartir les demandes de manière égale entre les serveurs.
- **ip_hash** pour envoyer des requêtes d'une adresse client au même serveur sur la base d'un hachage calculé à partir des trois premiers octets de l'adresse IPv4 ou de l'adresse IPv6 complète du client.
- **hash** pour déterminer le serveur en fonction d'une clé définie par l'utilisateur, qui peut être une chaîne, une variable ou une combinaison des deux. Le paramètre **consistent** configure la distribution des requêtes par NGINX sur tous les serveurs en fonction de la valeur de la clé hachée définie par l'utilisateur.
- **random** pour envoyer des requêtes à un serveur choisi au hasard.

2. Redémarrez le service **nginx**:

```
# systemctl restart nginx
```

2.6. RESSOURCES SUPPLÉMENTAIRES

- Documentation officielle de NGINX . Notez que Red Hat ne maintient pas cette documentation et qu'elle peut ne pas fonctionner avec la version de NGINX que vous avez installée.
- Configurer des applications pour utiliser du matériel cryptographique via PKCS #11 .

CHAPITRE 3. CONFIGURATION DU SERVEUR PROXY DE MISE EN CACHE SQUID

Squid est un serveur proxy qui met en cache le contenu afin de réduire la bande passante et de charger les pages web plus rapidement. Ce chapitre décrit comment configurer Squid en tant que proxy pour les protocoles HTTP, HTTPS et FTP, ainsi que l'authentification et la restriction d'accès.

3.1. CONFIGURER SQUID COMME PROXY DE MISE EN CACHE SANS AUTHENTIFICATION

Cette section décrit une configuration de base de Squid en tant que proxy de mise en cache sans authentification. La procédure limite l'accès au proxy en fonction des plages d'adresses IP.

Conditions préalables

- La procédure suppose que le fichier **/etc/squid/squid.conf** est tel qu'il est fourni par le paquet **squid**. Si vous avez déjà modifié ce fichier, supprimez-le et réinstallez le paquet.

Procédure

1. Installez le paquetage **squid**:

```
# dnf install squid
```

2. Modifiez le fichier **/etc/squid/squid.conf**:

   a. Adaptez les listes de contrôle d'accès (ACL) de **localnet** pour qu'elles correspondent aux plages d'adresses IP qui doivent être autorisées à utiliser le proxy :

```
acl localnet src 192.0.2.0/24
acl localnet 2001:db8:1::/64
```

   Par défaut, le fichier **/etc/squid/squid.conf** contient la règle **http_access allow localnet** qui autorise l'utilisation du proxy à partir de toutes les plages d'adresses IP spécifiées dans les ACL **localnet**. Notez que vous devez spécifier toutes les ACL **localnet** avant la règle **http_access allow localnet**.

IMPORTANT

Supprimez toutes les entrées de **acl localnet** qui ne correspondent pas à votre environnement.

   b. L'ACL suivante existe dans la configuration par défaut et définit **443** comme un port qui utilise le protocole HTTPS :

```
acl SSL_ports port 443
```

   Si les utilisateurs doivent pouvoir utiliser le protocole HTTPS également sur d'autres ports, ajoutez une ACL pour chacun de ces ports :

```
acl SSL_ports port port_number
```

c. Mettez à jour la liste des règles **acl Safe_ports** pour configurer les ports sur lesquels Squid peut établir une connexion. Par exemple, pour configurer que les clients utilisant le proxy ne peuvent accéder aux ressources que sur les ports 21 (FTP), 80 (HTTP) et 443 (HTTPS), ne conservez que les déclarations **acl Safe_ports** suivantes dans la configuration :

```
acl Safe_ports port 21
acl Safe_ports port 80
acl Safe_ports port 443
```

Par défaut, la configuration contient la règle **http_access deny !Safe_ports** qui définit le refus d'accès aux ports qui ne sont pas définis dans les ACL **Safe_ports**.

d. Configurez le type de cache, le chemin d'accès au répertoire du cache, la taille du cache et d'autres paramètres spécifiques au type de cache dans le paramètre **cache_dir**:

```
cache_dir ufs /var/spool/squid 10000 16 256
```

Avec ces paramètres :

- Squid utilise le type de cache **ufs**.
- Squid stocke son cache dans le répertoire **/var/spool/squid/**.
- La mémoire cache peut atteindre **10000** MB.
- Squid crée des sous-répertoires de niveau 1 **16** dans le répertoire **/var/spool/squid/**.
- Squid crée des sous-répertoires **256** dans chaque répertoire de niveau 1.
  Si vous ne définissez pas de directive **cache_dir**, Squid stocke le cache en mémoire.

3. Si vous définissez un répertoire de cache différent de **/var/spool/squid/** dans le paramètre **cache_dir**:

a. Créer le répertoire du cache :

```
# mkdir -p path_to_cache_directory
```

b. Configurez les permissions pour le répertoire du cache :

```
# chown squid:squid path_to_cache_directory
```

c. Si vous utilisez SELinux en mode **enforcing**, définissez le contexte **squid_cache_t** pour le répertoire de cache :

```
# semanage fcontext -a -t squid_cache_t "path_to_cache_directory(/.*)?"
# restorecon -Rv path_to_cache_directory
```

Si l'utilitaire **semanage** n'est pas disponible sur votre système, installez le paquet **policycoreutils-python-utils**.

4. Ouvrez le port **3128** dans le pare-feu :

```
# firewall-cmd --permanent --add-port=3128/tcp
# firewall-cmd --reload
```

5. Activez et démarrez le service **squid**:

```
# systemctl enable --now squid
```

Verification steps

Pour vérifier que le proxy fonctionne correctement, téléchargez une page web à l'aide de l'utilitaire **curl**:

```
# curl -O -L "https://www.redhat.com/index.html" -x "proxy.example.com:3128"
```

Si **curl** n'affiche aucune erreur et que le fichier **index.html** a été téléchargé dans le répertoire actuel, le proxy fonctionne.

3.2. CONFIGURER SQUID COMME PROXY DE MISE EN CACHE AVEC AUTHENTIFICATION LDAP

Cette section décrit une configuration de base de Squid en tant que proxy de mise en cache qui utilise LDAP pour authentifier les utilisateurs. La procédure prévoit que seuls les utilisateurs authentifiés peuvent utiliser le proxy.

Conditions préalables

- La procédure suppose que le fichier **/etc/squid/squid.conf** est tel qu'il est fourni par le paquet **squid**. Si vous avez déjà modifié ce fichier, supprimez-le et réinstallez le paquet.
- Un utilisateur de service, tel que **uid=proxy_user,cn=users,cn=accounts,dc=example,dc=com**, existe dans l'annuaire LDAP. Squid utilise ce compte uniquement pour rechercher l'utilisateur qui s'authentifie. Si l'utilisateur authentifiant existe, Squid se lie à l'annuaire en tant que cet utilisateur pour vérifier l'authentification.

Procédure

1. Installez le paquetage **squid**:

```
# dnf install squid
```

2. Modifiez le fichier **/etc/squid/squid.conf**:

   a. Pour configurer l'utilitaire d'aide **basic_ldap_auth**, ajoutez l'entrée de configuration suivante au début de **/etc/squid/squid.conf**:

```
auth_param basic program /usr/lib64/squid/basic_ldap_auth -b
"cn=users,cn=accounts,dc=example,dc=com" -D
"uid=proxy_user,cn=users,cn=accounts,dc=example,dc=com" -W
/etc/squid/ldap_password -f "(&(objectClass=person)(uid=%s))" -ZZ -H
ldap://ldap_server.example.com:389
```

   Les paramètres transmis à l'utilitaire **basic_ldap_auth** dans l'exemple ci-dessus sont décrits ci-dessous :

   - **-b *base_DN*** définit la base de recherche LDAP.

- **-D *proxy_service_user_DN*** définit le nom distinctif (DN) du compte que Squid utilise pour rechercher l'utilisateur qui s'authentifie dans l'annuaire.
- **-W *path_to_password_file*** définit le chemin d'accès au fichier contenant le mot de passe de l'utilisateur du service proxy. L'utilisation d'un fichier de mot de passe permet d'éviter que le mot de passe soit visible dans la liste des processus du système d'exploitation.
- **-f *LDAP_filter*** spécifie le filtre de recherche LDAP. Squid remplace la variable **%s** par le nom d'utilisateur fourni par l'utilisateur qui s'authentifie.
  Le filtre **(&(objectClass=person)(uid=%s))** de l'exemple définit que le nom de l'utilisateur doit correspondre à la valeur définie dans l'attribut **uid** et que l'entrée du répertoire contient la classe d'objets **person**.
- **-ZZ** impose une connexion cryptée TLS sur le protocole LDAP à l'aide de la commande **STARTTLS**. Omettre le **-ZZ** dans les situations suivantes :
  - Le serveur LDAP ne prend pas en charge les connexions cryptées.
  - Le port spécifié dans l'URL utilise le protocole LDAPS.
- Le paramètre -H LDAP_URL spécifie le protocole, le nom d'hôte ou l'adresse IP et le port du serveur LDAP au format URL.

b. Ajoutez l'ACL et la règle suivantes pour configurer Squid de manière à ce que seuls les utilisateurs authentifiés puissent utiliser le proxy :

```
acl ldap-auth proxy_auth REQUIRED
http_access allow ldap-auth
```

IMPORTANT

Spécifiez ces paramètres avant la règle **http_access deny** all.

c. Supprimez la règle suivante pour désactiver le contournement de l'authentification par proxy à partir des plages d'adresses IP spécifiées dans les ACL **localnet**:

```
http_access allow localnet
```

d. L'ACL suivante existe dans la configuration par défaut et définit **443** comme un port qui utilise le protocole HTTPS :

```
acl SSL_ports port 443
```

Si les utilisateurs doivent pouvoir utiliser le protocole HTTPS également sur d'autres ports, ajoutez une ACL pour chacun de ces ports :

```
acl SSL_ports port port_number
```

e. Mettez à jour la liste des règles **acl Safe_ports** pour configurer les ports sur lesquels Squid peut établir une connexion. Par exemple, pour configurer que les clients utilisant le proxy ne peuvent accéder aux ressources que sur les ports 21 (FTP), 80 (HTTP) et 443 (HTTPS), ne conservez que les déclarations **acl Safe_ports** suivantes dans la configuration :

```
acl Safe_ports port 21
acl Safe_ports port 80
acl Safe_ports port 443
```

Par défaut, la configuration contient la règle **http_access deny !Safe_ports** qui définit le refus d'accès aux ports qui ne sont pas définis dans **Safe_ports ACLs**.

f. Configurez le type de cache, le chemin d'accès au répertoire du cache, la taille du cache et d'autres paramètres spécifiques au type de cache dans le paramètre **cache_dir**:

```
cache_dir ufs /var/spool/squid 10000 16 256
```

Avec ces paramètres :

- Squid utilise le type de cache **ufs**.
- Squid stocke son cache dans le répertoire **/var/spool/squid/**.
- La mémoire cache peut atteindre **10000** MB.
- Squid crée des sous-répertoires de niveau 1 **16** dans le répertoire **/var/spool/squid/**.
- Squid crée des sous-répertoires **256** dans chaque répertoire de niveau 1.
  Si vous ne définissez pas de directive **cache_dir**, Squid stocke le cache en mémoire.

3. Si vous définissez un répertoire de cache différent de **/var/spool/squid/** dans le paramètre **cache_dir**:

a. Créer le répertoire du cache :

```
# mkdir -p path_to_cache_directory
```

b. Configurez les permissions pour le répertoire du cache :

```
# chown squid:squid path_to_cache_directory
```

c. Si vous utilisez SELinux en mode **enforcing**, définissez le contexte **squid_cache_t** pour le répertoire de cache :

```
# semanage fcontext -a -t squid_cache_t "path_to_cache_directory(/.*)?"
# restorecon -Rv path_to_cache_directory
```

Si l'utilitaire **semanage** n'est pas disponible sur votre système, installez le paquet **policycoreutils-python-utils**.

4. Stockez le mot de passe de l'utilisateur du service LDAP dans le fichier **/etc/squid/ldap_password** et définissez les autorisations appropriées pour le fichier :

```
# echo "password" > /etc/squid/ldap_password
# chown root:squid /etc/squid/ldap_password
# chmod 640 /etc/squid/ldap_password
```

5. Ouvrez le port **3128** dans le pare-feu :

```
# firewall-cmd --permanent --add-port=3128/tcp
# firewall-cmd --reload
```

6. Activez et démarrez le service **squid**:

```
# systemctl enable --now squid
```

Verification steps

Pour vérifier que le proxy fonctionne correctement, téléchargez une page web à l'aide de l'utilitaire **curl**:

```
# curl -O -L "https://www.redhat.com/index.html" -x
"user_name:password@proxy.example.com:3128"
```

Si curl n'affiche aucune erreur et que le fichier **index.html** a été téléchargé dans le répertoire actuel, le proxy fonctionne.

Étapes de dépannage

Pour vérifier que l'utilitaire d'aide fonctionne correctement :

1. Démarrez manuellement l'utilitaire d'assistance avec les mêmes paramètres que ceux utilisés dans le paramètre **auth_param**:

```
# /usr/lib64/squid/basic_ldap_auth -b "cn=users,cn=accounts,dc=example,dc=com" -D
"uid=proxy_user,cn=users,cn=accounts,dc=example,dc=com" -W
/etc/squid/ldap_password -f "(&(objectClass=person)(uid=%s))" -ZZ -H
ldap://ldap_server.example.com:389
```

2. Saisissez un nom d'utilisateur et un mot de passe valides, puis appuyez sur **Entrée**:

```
user_name password
```

   Si l'utilitaire d'aide renvoie **OK**, l'authentification a réussi.

3.3. CONFIGURER SQUID COMME PROXY DE MISE EN CACHE AVEC AUTHENTIFICATION KERBEROS

Cette section décrit une configuration de base de Squid en tant que proxy de mise en cache qui authentifie les utilisateurs d'un Active Directory (AD) à l'aide de Kerberos. La procédure prévoit que seuls les utilisateurs authentifiés peuvent utiliser le proxy.

Conditions préalables

- La procédure suppose que le fichier **/etc/squid/squid.conf** est tel qu'il est fourni par le paquet **squid**. Si vous avez déjà modifié ce fichier, supprimez-le et réinstallez le paquet.
- Le serveur sur lequel vous voulez installer Squid est membre du domaine AD.

Procédure

1. Install the following packages:

```
# dnf install squid krb5-workstation
```

2. S'authentifier en tant qu'administrateur du domaine AD :

```
# kinit administrator@AD.EXAMPLE.COM
```

3. Créer un keytab pour Squid et le stocker dans le fichier **/etc/squid/HTTP.keytab**:

```
# export KRB5_KTNAME=FILE:/etc/squid/HTTP.keytab
# net ads keytab CREATE -U administrator
```

4. Ajouter le principal du service **HTTP** à la base de données de clés :

```
# net ads keytab ADD HTTP -U administrator
```

5. Définir le propriétaire du fichier keytab à l'utilisateur **squid**:

```
# chown squid /etc/squid/HTTP.keytab
```

6. En option, vérifiez que le fichier keytab contient le principal de service **HTTP** pour le nom de domaine complet (FQDN) du serveur proxy :

```
 klist -k /etc/squid/HTTP.keytab
Keytab name: FILE:/etc/squid/HTTP.keytab
KVNO Principal
---- --------------------------------------------------
...
   2 HTTP/proxy.ad.example.com@AD.EXAMPLE.COM
...
```

7. Modifiez le fichier **/etc/squid/squid.conf**:

   a. Pour configurer l'utilitaire d'aide **negotiate_kerberos_auth**, ajoutez l'entrée de configuration suivante au début de **/etc/squid/squid.conf**:

   ```
   auth_param negotiate program /usr/lib64/squid/negotiate_kerberos_auth -k
   /etc/squid/HTTP.keytab -s HTTP/proxy.ad.example.com@AD.EXAMPLE.COM
   ```

   Les paramètres transmis à l'utilitaire **negotiate_kerberos_auth** dans l'exemple ci-dessus sont décrits ci-dessous :

   - **-k *file*** définit le chemin d'accès au fichier key tab. L'utilisateur Squid doit avoir les droits de lecture sur ce fichier.
   - **-s HTTP/*host_name*@*kerberos_realm*** définit le principal Kerberos utilisé par Squid.

   En option, vous pouvez activer la journalisation en passant l'un ou les deux paramètres suivants à l'utilitaire d'aide :

   - **-i** enregistre les messages d'information, tels que l'authentification de l'utilisateur.
   - **-d** active la journalisation de débogage.
     Squid enregistre les informations de débogage de l'utilitaire d'aide dans le fichier **/var/log/squid/cache.log**.

b. Ajoutez l'ACL et la règle suivantes pour configurer Squid de manière à ce que seuls les utilisateurs authentifiés puissent utiliser le proxy :

```
acl kerb-auth proxy_auth REQUIRED
http_access allow kerb-auth
```

IMPORTANT

Spécifiez ces paramètres avant la règle **http_access deny all**.

c. Supprimez la règle suivante pour désactiver le contournement de l'authentification par proxy à partir des plages d'adresses IP spécifiées dans les ACL **localnet**:

```
http_access allow localnet
```

d. L'ACL suivante existe dans la configuration par défaut et définit **443** comme un port qui utilise le protocole HTTPS :

```
acl SSL_ports port 443
```

Si les utilisateurs doivent pouvoir utiliser le protocole HTTPS également sur d'autres ports, ajoutez une ACL pour chacun de ces ports :

```
acl SSL_ports port port_number
```

e. Mettez à jour la liste des règles **acl Safe_ports** pour configurer les ports sur lesquels Squid peut établir une connexion. Par exemple, pour configurer que les clients utilisant le proxy ne peuvent accéder aux ressources que sur les ports 21 (FTP), 80 (HTTP) et 443 (HTTPS), ne conservez que les déclarations **acl Safe_ports** suivantes dans la configuration :

```
acl Safe_ports port 21
acl Safe_ports port 80
acl Safe_ports port 443
```

Par défaut, la configuration contient la règle **http_access deny !Safe_ports** qui définit le refus d'accès aux ports qui ne sont pas définis dans les ACL **Safe_ports**.

f. Configurez le type de cache, le chemin d'accès au répertoire du cache, la taille du cache et d'autres paramètres spécifiques au type de cache dans le paramètre **cache_dir**:

```
cache_dir ufs /var/spool/squid 10000 16 256
```

Avec ces paramètres :

- Squid utilise le type de cache **ufs**.
- Squid stocke son cache dans le répertoire **/var/spool/squid/**.
- La mémoire cache peut atteindre **10000** MB.
- Squid crée des sous-répertoires de niveau 1 **16** dans le répertoire **/var/spool/squid/**.
- Squid crée des sous-répertoires **256** dans chaque répertoire de niveau 1.

Si vous ne définissez pas de directive **cache_dir**, Squid stocke le cache en mémoire.

8. Si vous définissez un répertoire de cache différent de **/var/spool/squid/** dans le paramètre **cache_dir**:

   a. Créer le répertoire du cache :

   ```
   # mkdir -p path_to_cache_directory
   ```

   b. Configurez les permissions pour le répertoire du cache :

   ```
   # chown squid:squid path_to_cache_directory
   ```

   c. Si vous utilisez SELinux en mode **enforcing**, définissez le contexte **squid_cache_t** pour le répertoire de cache :

   ```
   # semanage fcontext -a -t squid_cache_t "path_to_cache_directory(/.*)?"
   # restorecon -Rv path_to_cache_directory
   ```

   Si l'utilitaire **semanage** n'est pas disponible sur votre système, installez le paquet **policycoreutils-python-utils**.

9. Ouvrez le port **3128** dans le pare-feu :

   ```
   # firewall-cmd --permanent --add-port=3128/tcp
   # firewall-cmd --reload
   ```

10. Activez et démarrez le service **squid**:

    ```
    # systemctl enable --now squid
    ```

Verification steps

Pour vérifier que le proxy fonctionne correctement, téléchargez une page web à l'aide de l'utilitaire **curl**:

```
# curl -O -L "https://www.redhat.com/index.html" --proxy-negotiate -u : -x
"proxy.ad.example.com:3128"
```

Si **curl** n'affiche aucune erreur et que le fichier **index.html** existe dans le répertoire actuel, le proxy fonctionne.

Étapes de dépannage

Pour tester manuellement l'authentification Kerberos :

1. Obtenir un ticket Kerberos pour le compte AD :

   ```
   # kinit user@AD.EXAMPLE.COM
   ```

2. Il est possible d'afficher le ticket :

   ```
   # klist
   ```

3. Utilisez l'utilitaire **negotiate_kerberos_auth_test** pour tester l'authentification :

```
# /usr/lib64/squid/negotiate_kerberos_auth_test proxy.ad.example.com
```

Si l'utilitaire d'aide renvoie un jeton, l'authentification a réussi :

```
Token : YIIFtAYGKwYBBQUCoIIFqDC...
```

3.4. CONFIGURATION D'UNE LISTE DE REFUS DE DOMAINE DANS SQUID

Il arrive fréquemment que les administrateurs veuillent bloquer l'accès à des domaines spécifiques. Cette section décrit comment configurer une liste de refus de domaine dans Squid.

Conditions préalables

- Squid est configuré et les utilisateurs peuvent utiliser le proxy.

Procédure

1. Modifiez le fichier **/etc/squid/squid.conf** et ajoutez les paramètres suivants :

```
acl domain_deny_list dstdomain "/etc/squid/domain_deny_list.txt"
http_access deny all domain_deny_list
```

IMPORTANT

Ajoutez ces entrées avant la première déclaration **http_access allow** qui autorise l'accès aux utilisateurs ou aux clients.

2. Créez le fichier **/etc/squid/domain_deny_list.txt** et ajoutez les domaines que vous souhaitez bloquer. Par exemple, pour bloquer l'accès à **example.com**, y compris les sous-domaines, et pour bloquer **example.net**, ajoutez :

```
.example.com
example.net
```

IMPORTANT

Si vous avez fait référence au fichier **/etc/squid/domain_deny_list.txt** dans la configuration de Squid, ce fichier ne doit pas être vide. Si le fichier est vide, Squid ne démarre pas.

3. Redémarrez le service **squid**:

```
# systemctl restart squid
```

3.5. CONFIGURER LE SERVICE SQUID POUR QU'IL ÉCOUTE SUR UN PORT OU UNE ADRESSE IP SPÉCIFIQUE

Par défaut, le service proxy Squid écoute sur le port **3128** sur toutes les interfaces réseau. Cette section décrit comment modifier le port et configurer Squid pour qu'il écoute sur une adresse IP spécifique.

Conditions préalables

- Le paquet **squid** est installé.

Procédure

1. Modifiez le fichier **/etc/squid/squid.conf**:
   - Pour définir le port sur lequel le service Squid écoute, définissez le numéro de port dans le paramètre **http_port**. Par exemple, pour définir le port sur **8080**, définissez :

     ```
     http_port 8080
     ```

   - Pour configurer l'adresse IP sur laquelle le service Squid écoute, définissez l'adresse IP et le numéro de port dans le paramètre **http_port**. Par exemple, pour configurer l'écoute de Squid uniquement sur l'adresse IP **192.0.2.1** et le port **3128**, définissez :

     ```
     http_port 192.0.2.1:3128
     ```

     Ajoutez plusieurs paramètres **http_port** au fichier de configuration pour configurer l'écoute de Squid sur plusieurs ports et adresses IP :

     ```
     http_port 192.0.2.1:3128
     http_port 192.0.2.1:8080
     ```

2. Si vous avez configuré Squid pour qu'il utilise un port différent de celui par défaut (**3128**) :
   a. Ouvrez le port dans le pare-feu :

      ```
      # firewall-cmd --permanent --add-port=port_number/tcp
      # firewall-cmd --reload
      ```

   b. Si vous exécutez SELinux en mode "enforcing", affectez le port à la définition du type de port **squid_port_t**:

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

      Si l'utilitaire **semanage** n'est pas disponible sur votre système, installez le paquet **policycoreutils-python-utils**.

3. Redémarrez le service **squid**:

   ```
   # systemctl restart squid
   ```

3.6. RESSOURCES SUPPLÉMENTAIRES

- Paramètres de configuration **usr/share/doc/squid-<version>/squid.conf.documented**