Red Hat Enterprise Linux 9

ネットワークのセキュリティー保護

セキュリティー保護されたネットワークおよびネットワーク通信の設定

Last Updated: 2022-08-08

Red Hat Enterprise Linux 9 ネットワークのセキュリティー保護

セキュリティー保護されたネットワークおよびネットワーク通信の設定

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

本書は、管理者が、様々な攻撃からネットワーク、接続されているマシン、およびネットワーク通信のセキュリティーを保護する方法を説明します。

目次

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、弊社の CTO、Chris Wright のメッセージ を参照してください。

RED HAT ドキュメントへのフィードバックの提供

ご意見ご要望をお聞かせください。ドキュメントの改善点はございませんか。

- 特定の部分についての簡単なコメントをお寄せいただく場合は、以下をご確認ください。
    1. ドキュメントの表示が **Multi-page HTML** 形式になっていて、ドキュメントの右上隅に **Feedback** ボタンがあることを確認してください。
    2. マウスカーソルで、コメントを追加する部分を強調表示します。
    3. そのテキストの下に表示される **Add Feedback** ポップアップをクリックします。
    4. 表示される手順に従ってください。
- Bugzilla を介してフィードバックを送信するには、新しいチケットを作成します。
    1. Bugzilla の Web サイトに移動します。
    2. Component で **Documentation** を選択します。
    3. **Description** フィールドに、ドキュメントの改善に関するご意見を記入してください。ドキュメントの該当部分へのリンクも記入してください。
    4. **Submit Bug** をクリックします。

第1章 2 台のシステム間で OPENSSH を使用した安全な通信の使用

SSH (Secure Shell) は、クライアント/サーバーアーキテクチャーを使用する 2 つのシステム間で安全な通信を提供し、ユーザーがリモートでサーバーホストシステムにログインできるようにするプロトコルです。FTP、Telnet などの他のリモート通信プロトコルとは異なり、SSH はログインセッションを暗号化するため、侵入者が接続して暗号化されていないパスワードを入手するのが困難になります。

Red Hat Enterprise Linux には、基本的な **OpenSSH** パッケージ (一般的な **openssh** パッケージ、**openssh-server** パッケージ、および **openssh-clients** パッケージ) が含まれます。**OpenSSH** パッケージには、**OpenSSL** パッケージ (**openssl-libs**) が必要です。このパッケージは、重要な暗号化ライブラリーをいくつかインストールして、暗号化通信を提供する **OpenSSH** を有効にします。

1.1. SSH と OPENSSH

SSH (Secure Shell) は、リモートマシンにログインしてそのマシンでコマンドを実行するプログラムです。SSH プロトコルは、安全でないネットワーク上で、信頼されていないホスト間で安全な通信を提供します。また、X11 接続と任意の TCP/IP ポートを安全なチャンネルで転送することもできます。

SSH プロトコルは、リモートシェルのログインやファイルコピー用に使用する場合に、システム間の通信の傍受や特定ホストの偽装など、セキュリティーの脅威を軽減します。これは、SSH クライアントとサーバーがデジタル署名を使用してそれぞれの ID を確認するためです。さらに、クライアントシステムとサーバーシステムとの間の通信はすべて暗号化されます。

ホストキーは、SSH プロトコルのホストを認証します。ホスト鍵は、OpenSSH の初回インストール時、またはホストの初回起動時に自動的に生成される暗号鍵です。

OpenSSH は、Linux、UNIX、および同様のオペレーティングシステムでサポートされている SSH プロトコルの実装です。OpenSSH クライアントとサーバー両方に必要なコアファイルが含まれます。OpenSSH スイートは、以下のユーザー空間ツールで構成されます。

- **SSH** は、リモートログインプログラム (SSH クライアント) です。
- **sshd** は、OpenSSH SSH デーモンです。
- **scp** は、安全なリモートファイルコピープログラムです。
- **sftp** は、安全なファイル転送プログラムです。
- **ssh-agent** は、秘密鍵をキャッシュする認証エージェントです。
- **ssh-add** は、秘密鍵の ID を **ssh-agent** に追加します。
- **ssh-keygen** が、**ssh** の認証キーを生成、管理、および変換します。
- **ssh-copy-id** は、ローカルの公開鍵をリモート SSH サーバーの **authorized_keys** ファイルに追加するスクリプトです。
- **ssh-keyscan** - SSH パブリックホストキーを収集します。

注記

RHEL 9 では、Secure copy protocol (SCP) がデフォルトで SSH File Transfer Protocol (SFTP) に置き換えられています。これは、CVE-2020-15778 など、SCP が原因のセキュリティーの問題が発生しているためです。

使用しているシナリオで SFTP が利用できない場合や互換性がない場合は、**-O** オプションを使用して、元の SCP/RCP プロトコルを強制的に使用できます。

追加情報は Red Hat Enterprise Linux 9 の記事の OpenSSH SCP プロトコルが非推奨に を参照してください。

現在、SSH のバージョンには、バージョン 1 と新しいバージョン 2 の 2 つがあります。RHEL の OpenSSH スイートは、SSH バージョン 2 のみをサポートします。このスイートは、バージョン 1 で知られているエクスプロイトに対して脆弱ではない拡張キー交換アルゴリズムを備えています。

RHEL コア暗号化サブシステムの 1 つである OpenSSH は、システム全体の暗号化ポリシーを使用します。これにより、弱い暗号スイートおよび暗号化アルゴリズムがデフォルト設定で無効になります。ポリシーを変更するには、管理者が **update-crypto-policies** コマンドを使用して設定を調節するか、システム全体の暗号化ポリシーを手動でオプトアウトする必要があります。

OpenSSH スイートは、2 セットの設定ファイルを使用します。1 つはクライアントプログラム (つまり、**ssh**、**scp**、および **sftp**) 用で、もう 1 つはサーバー (**sshd** デーモン) 用です。

システム全体の SSH 設定情報が **/etc/ssh/** ディレクトリーに保存されます。ユーザー固有の SSH 設定情報は、ユーザーのホームディレクトリーの **~/.ssh/** に保存されます。OpenSSH 設定ファイルの詳細な一覧は、**sshd (8)** の man ページの **FILES** セクションを参照してください。

関連情報

- **man -k ssh** コマンドを使用して一覧表示される man ページ
- システム全体の暗号化ポリシーの使用

1.2. OPENSSH サーバーの設定および起動

お使いの環境と OpenSSH サーバーの起動に必要となる基本設定には、以下の手順を使用します。デフォルトの RHEL インストールを行うと、**sshd** デーモンがすでに起動し、サーバーのホスト鍵が自動的に作成されることに注意してください。

前提条件

- **openssh-server** パッケージがインストールされている。

手順

1. 現行セッションで **sshd** デーモンを開始し、ブート時に自動的に起動するように設定します。

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

2. デフォルトの**0.0.0.0** (IPv4) または **::** とは異なるアドレスを指定するには、以下を行います。(IPv6) **/etc/ssh/sshd_config** 設定ファイルの **ListenAddress** ディレクティブ、および低速な動的ネットワーク設定を使用するには、**network-online.target** ターゲットユニットの依存関係

を**sshd.service** ユニットファイルに追加します。これを行うには、以下の内容で **/etc/systemd/system/sshd.service.d/local.conf** ファイルを作成します。

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

3. **/etc/ssh/sshd_config** 設定ファイルの OpenSSH サーバーの設定がシナリオの要件を満たしているかどうかを確認します。

4. 必要に応じて、**/etc/issue** ファイルを編集して、クライアント認証を行う前に OpenSSH サーバーに表示される welcome メッセージを変更します。以下に例を示します。

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

Banner オプションが **/etc/ssh/sshd_config** でコメントアウトされず、その値に **/etc/issue** が含まれていることを確認します。

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

ログインに成功すると表示されるメッセージを変更するには、サーバーの **/etc/motd** ファイルを編集する必要があります。詳細は、man ページの **pam_motd** を参照してください。

5. **systemd** 設定を再読み込みし、**sshd** を再起動して変更を適用します。

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

検証

1. **sshd** デーモンが実行していることを確認します。

```
# 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. SSH クライアントを使用して SSH サーバーに接続します。

```
# 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:
```

関連情報

- **sshd(8)** および **sshd_config(5)** の man ページ。

1.3. 鍵ベースの認証用の OPENSSH サーバーの設定

システムのセキュリティーを強化するには、OpenSSH サーバーでパスワード認証を無効にして鍵ベースの認証を有効にします。

前提条件

- **openssh-server** パッケージがインストールされている。
- サーバーで **sshd** デーモンが実行している。

手順

1. テキストエディターで **/etc/ssh/sshd_config** 設定を開きます。以下に例を示します。

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

2. **PasswordAuthentication** オプションを **no** に変更します。

```
PasswordAuthentication no
```

新しいデフォルトインストール以外のシステムで **PubkeyAuthentication no** が設定されていないことと、**ChallengeResponseAuthentication** ディレクティブが **no** に設定されていることを確認します。リモートで接続している場合は、コンソールもしくは帯域外アクセスを使用せず、パスワード認証を無効にする前に、鍵ベースのログインプロセスをテストします。

3. NFS がマウントされたホームディレクトリーで鍵ベースの認証を使用するには、SELinux ブール値 **use_nfs_home_dirs** を有効にします。

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

4. **sshd** デーモンを再読み込みし、変更を適用します。

```
# systemctl reload sshd
```

関連情報

- **sshd(8)**、**sshd_config(5)**、および **setsebool(8)** の man ページ。

1.4. SSH 鍵ペアの生成

以下の手順を使用して、ローカルシステムに SSH 鍵ペアを生成し、生成された公開鍵を OpenSSH サーバーにコピーします。サーバーが正しく設定されている場合は、パスワードなしで OpenSSH サーバーにログインできます。

重要

root で次の手順を完了すると、鍵を使用できるのは **root** だけとなります。

手順

1. SSH プロトコルのバージョン 2 用の ECDSA 鍵ペアを生成するには、次のコマンドを実行します。

```
$ 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]-----+
```

ssh-keygen コマンドまたは Ed25519 鍵ペアに **-t rsa** オプションを指定して RSA 鍵ペアを生成するには、**ssh-keygen -t ed25519** コマンドを実行します。

2. 公開鍵をリモートマシンにコピーするには、次のコマンドを実行します。

```
$ 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.
```

セッションで **ssh-agent** プログラムを使用しない場合は、上記のコマンドで、最後に変更した **~/.ssh/id*.pub** 公開鍵をコピーします (インストールされていない場合)。別の公開キーファイルを指定したり、**ssh-agent** により、メモリーにキャッシュされた鍵よりもファイル内の鍵の方が優先順位を高くするには、**-i** オプションを指定して **ssh-copy-id** コマンドを使用します。

注記

システムを再インストールする際に、生成しておいた鍵ペアを引き続き使用する場合は、~/**.ssh**/ ディレクトリーのバックアップを作成します。再インストール後に、このディレクトリーをホームディレクトリーにコピーします。これは、(**root** を含む) システムの全ユーザーで実行できます。

検証

1. パスワードなしで OpenSSH サーバーにログインします。

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

関連情報

- **ssh-keygen (1)** および **ssh-copy-id (1)** の man ページ

1.5. スマートカードに保存された SSH 鍵の使用

Red Hat Enterprise Linux では、OpenSSH クライアントでスマートカードに保存されている RSA 鍵および ECDSA 鍵を使用できるようになりました。この手順に従って、パスワードの代わりにスマートカードを使用した認証を有効にします。

前提条件

- クライアントで、**opensc** パッケージをインストールして、**pcscd** サービスを実行している。

手順

1. PKCS #11 の URI を含む OpenSC PKCS #11 モジュールが提供する鍵の一覧を表示し、その出力を **keys.pub** ファイルに保存します。

```
$ 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. リモートサーバー (**example.com**) でスマートカードを使用した認証を有効にするには、公開鍵をリモートサーバーに転送します。前の手順で作成された **keys.pub** で **ssh-copy-id** コマンドを使用します。

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

3. 手順 1 の **ssh-keygen -D** コマンドの出力にある ECDSA 鍵を使用して **example.com** に接続するには、鍵を一意に参照する URI のサブセットのみを使用できます。以下に例を示します。

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

4. ~/**.ssh**/**config** ファイルで同じ URI 文字列を使用して、設定を永続化できます。

```
$ 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] $
```

OpenSSH は **p11-kit-proxy** ラッパーを使用し、OpenSC PKCS #11 モジュールが PKCS#11 キットに登録されているため、以前のコマンドを簡素化できます。

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

PKCS #11 の URI の **id=** の部分を飛ばすと、OpenSSH が、プロキシーモジュールで利用可能な鍵をすべて読み込みます。これにより、必要な入力の量を減らすことができます。

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

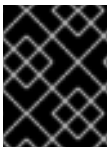
関連情報

- Fedora 28:Better smart card support in OpenSSH
- **p11-kit(8)**、**opensc.conf(5)**、**pcscd(8)**、**ssh(1)**、および **ssh-keygen(1)** の man ページ

1.6. OPENSSH のセキュリティーの強化

以下のヒントは、OpenSSH を使用する際にセキュリティーを高めるのに役に立ちます。OpenSSH 設定ファイル **/etc/ssh/sshd_config** を変更するには、**sshd** デーモンを再読み込みして有効にする必要があることに注意してください。

```
# systemctl reload sshd
```



重要

ほとんどのセキュリティー強化の設定変更により、最新のアルゴリズムまたは暗号スイートに対応していないクライアントとの互換性が低下します。

安全ではない接続プロトコルの無効化

- SSH を本当の意味で有効なものにするため、OpenSSH スイートに置き換えられる安全ではない接続プロトコルを使用しないようにします。このような接続プロトコルを使用すると、ユーザーのパスワード自体は SSH を使用した 1 回のセッションで保護されても、その後に Telnet を使用してログインした時に傍受されてしまうためです。このため、telnet、rsh、rlogin、ftp などの安全ではないプロトコルを無効にすることを検討してください。

鍵ベースの認証の有効化およびパスワードベースの認証の無効化

- 認証用パスワードを無効にして鍵のペアのみを許可すると、攻撃対象領域が減ってユーザーの時間を節約できる可能性があります。クライアントにおいて、**ssh-keygen** ツールを使用して鍵のペアを生成し、**ssh-copy-id** ユーティリティーを使用して OpenSSH サーバーのクライアントから公開鍵をコピーします。OpenSSH サーバーでパスワードベースの認証を無効にするには、**/etc/ssh/sshd_config** の **PasswordAuthentication** オプションを **no** に変更します。

```
PasswordAuthentication no
```

鍵のタイプ

- **ssh-keygen** コマンドは、デフォルトで RSA 鍵のペアを生成しますが、**-t** オプションを使用して ECDSA 鍵または Ed25519 鍵を生成するように指定できます。ECDSA (Elliptic Curve Digital Signature Algorithm) は、同等の対称鍵強度で RSA よりも優れたパフォーマンスを提供します。また、短いキーも生成します。Ed25519 公開鍵アルゴリズムは、RSA、DSA、および ECDSA より安全で高速な歪曲エドワーズ曲線の実装です。
サーバーホストの鍵の RSA、ECDSA、および Ed25519 がない場合は、OpenSSH が自動的に作成します。RHEL でホストの鍵の作成を設定するには、インスタンス化したサービス **sshd-keygen@.service** を使用します。たとえば、RSA 鍵タイプの自動作成を無効にするには、次のコマンドを実行します。

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

- SSH 接続の特定の鍵タイプを除外するには、**/etc/ssh/sshd_config** で該当行をコメントアウトして **sshd** サービスを再読み込みします。たとえば、Ed25519 ホストキーだけを許可するには、次のコマンドを実行します。

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

デフォルト以外のポート

- デフォルトでは、**sshd** デーモンは TCP ポート 22 をリッスンします。ポートを変更すると、自動化したネットワークスキャンに基づく攻撃にシステムがさらされる可能性が減るため、あいまいさによりセキュリティーが向上します。ポートは、**/etc/ssh/sshd_config** 設定ファイルの **Port** ディレクティブを使用して指定できます。
また、デフォルト以外のポートを使用できるように、デフォルトの SELinux ポリシーも更新する必要があります。そのためには、**policycoreutils-python-utils** パッケージの **semanage** ツールを使用します。

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

さらに、**firewalld** 設定を更新します。

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

このコマンドで、**port_number** を、**Port** ディレクティブで指定された新しいポート番号に置き換えます。

Root ログイン

- **PermitRootLogin** はデフォルトで **prohibit-password** に設定されています。これにより、root としてログインしてパスワードを使用する代わりに鍵ベースの認証が使用され、ブルートフォース攻撃を防ぐことでリスクが軽減します。

  注意

  root ユーザーとしてログインを有効にすることは、どのユーザーがどの特権コマンドを実行するかを監査できないため、安全ではありません。管理コマンドを使用するには、ログインして、代わりに **sudo** を使用します。

X セキュリティー拡張機能の使用

- Red Hat Enterprise Linux クライアントの X サーバーは、X セキュリティー拡張を提供しません。そのため、クライアントは X11 転送を使用して信頼できない SSH サーバーに接続するときに別のセキュリティー層を要求できません。ほとんどのアプリケーションは、この拡張機能を有効にしても実行できません。
  デフォルトでは、**/etc/ssh/ssh_config.d/05-redhat.conf** ファイルの **ForwardX11Trusted** オプションが **yes** に設定され、**ssh -X remote_machine** コマンド (信頼できないホスト) と **ssh -Y remote_machine** コマンド (信頼できるホスト) には違いがありません。

  シナリオで X11 転送機能を必要としない場合は、**/etc/ssh/sshd_config** 設定ファイルの **X11Forwarding** ディレクティブを **no** に設定します。

特定のユーザー、グループ、またはドメインへのアクセス制限

- **/etc/ssh/sshd_config** 設定ファイルの **AllowUsers** ディレクティブおよび **AllowGroups** ディレクティブを使用すると、特定のユーザー、ドメイン、またはグループのみが OpenSSH サーバーに接続することを許可できます。**AllowUsers** および **AllowGroups** を組み合わせて、アクセスをより正確に制限できます。以下に例を示します。

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

  この設定行は、192.168.1.* サブネットおよび 10.0.0.* のサブネットのシステムの全ユーザーからの接続を許可します (192.168.1.2 アドレスのシステムを除く)。すべてのユーザーは、**example-group** グループに属している必要があります。OpenSSH サーバーは、その他のすべての接続を拒否します。

  許可リストは、許可されていない新しいユーザーまたはグループもブロックするため、許可リスト (Allow で始まるディレクティブ) の使用は、拒否リスト (Deny で始まるオプション) を使用するよりも安全です。

システム全体の暗号化ポリシーの変更

- OpenSSH は、RHEL のシステム全体の暗号化ポリシーを使用し、デフォルトのシステム全体の暗号化ポリシーレベルは、現在の脅威モデルに安全な設定を提供します。暗号化の設定をより厳格にするには、現在のポリシーレベルを変更します。

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

- OpenSSH サーバーに対するシステム全体の暗号化ポリシーを除外するには、**/etc/sysconfig/sshd** ファイルの **CRYPTO_POLICY=** 変数行のコメントを除外します。この変更後、**/etc/ssh/sshd_config** ファイルの **Ciphers** セクション、**MACs** セクショ

ン、**KexAlgoritms** セクション、および **GSSAPIKexAlgorithms** セクションで指定した値は上書きされません。このタスクには、暗号化オプションの設定に関する深い専門知識が必要になることに注意してください。

- 詳細は、Security hardening の Using system-wide cryptographic policies を参照してください

関連情報

- **sshd_config(5)**、**ssh-keygen(1)**、**crypto-policies(7)**、および **update-crypto-policies(8)** の man ページ

1.7. SSH ジャンプホストを使用してリモートサーバーに接続

この手順に従って、ジャンプホストとも呼ばれる中間サーバーを介してローカルシステムをリモートサーバーに接続します。

前提条件

- ジャンプホストでローカルシステムからの SSH 接続に対応している。
- リモートサーバーが、ジャンプホストからのみ SSH 接続を受け入れる。

手順

1. ローカルシステムの **~/.ssh/config** ファイルを編集してジャンプホストを定義します。以下に例を示します。

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

   - **Host** パラメーターは、**ssh** コマンドで使用できるホストの名前またはエイリアスを定義します。値は実際のホスト名と一致可能ですが、任意の文字列にすることもできます。
   - **HostName** パラメーターは、ジャンプホストの実際のホスト名または IP アドレスを設定します。

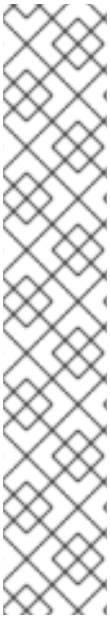
2. **ProxyJump** ディレクティブを使用してリモートサーバーのジャンプ設定を、ローカルシステムの **~/.ssh/config** ファイルに追加します。以下に例を示します。

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

3. ローカルシステムを使用して、ジャンプサーバー経由でリモートサーバーに接続します。

```
$ ssh remote-server
```

   このコマンドは、設定手順 1 および 2 を省略したときの **ssh -J jump-server1 remote-server** コマンドと同じです。



注記

ジャンプサーバーをさらに指定することもできます。また、完全なホスト名を指定する場合は、設定ファイルへのホスト定義の追加を飛ばすこともできます。以下に例を示します。

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

ジャンプサーバーのユーザー名または SSH ポートが、リモートサーバーの名前およびポートと異なる場合は、上記のコマンドのホスト名のみの表記を変更します。以下に例を示します。

```
$ ssh -J
johndoe@jump1.example.com:75,johndoe@jump2.example.com:75,johndoe@ju
mp3.example.com:75 joesec@remote1.example.com:220
```

関連情報

- **ssh_config(5)** および **ssh(1)** の man ページ

1.8. SSH-AGENT を使用して SSH キーでリモートマシンに接続する手順

パスフレーズを SSH 接続を開始するたびに入力しなくて済むようにするには、**ssh-agent** ユーティリティーを使用して SSH 秘密鍵をキャッシュします。秘密鍵とパスフレーズのセキュリティーが確保されます。

前提条件

- SSH デーモンが実行中で、ネットワーク経由で到達可能なリモートホストがある。
- リモートホストにログインするための IP アドレスまたはホスト名および認証情報を把握している。
- パスフレーズで SSH キーペアを生成し、公開鍵をリモートマシンに転送している。

手順

1. オプション:キーを使用してリモートホストに対して認証できることを確認します。
   a. SSH を使用してリモートホストに接続します。

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

   b. 秘密鍵へのアクセス権を付与する鍵の作成時に指定したパスフレーズを入力します。

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

2. **ssh-agent** を起動します。

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

3. **ssh-agent** にキーを追加します。

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

検証

- オプション:SSH を使用してホストマシンにログインします。

```
$ ssh example.user1@198.51.100.1

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

パスフレーズを入力する必要がないことに注意してください。

1.9. 関連情報

- **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)**、および **crypto-policies(7)** の manページ
- OpenSSH のホームページ
- 非標準設定でアプリケーションとサービスの SELinux の設定
- Controlling network traffic using firewalld

第2章 SSH システムロールを使用した安全な通信の設定

管理者は、SSHD システムロールを使用して SSH サーバーを設定し、SSH システムロールを使用し、Ansible Core パッケージを使用して同時に任意の数の RHEL システムで SSH クライアントを一貫して設定できます。

2.1. SSH SERVER のシステムロール変数

SSH Server システムロール Playbook では、設定と制限に応じて、SSH 設定ファイルのパラメーターを定義できます。

これらの変数が設定されていない場合には、システムロールは RHEL のデフォルト値と同じ **sshd_config** ファイルを作成します。

どのような場合でも、ブール値は **sshd** 設定で適切に **yes** と **no** としてレンダリングされます。一覧を使用して複数行の設定項目を定義できます。以下に例を示します。

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

レンダリングは以下のようになります。

```
ListenAddress 0.0.0.0
ListenAddress ::
```

SSH Server システムのロールの変数

sshd_enable

False に設定すると、ロールは完全に無効になります。デフォルトは **True** です。

sshd_skip_defaults

True に設定すると、システムロールではデフォルト値が適用されません。代わりに、**sshd** dict または **sshd_Key** 変数のいずれかを使用して、設定のデフォルト値をすべて指定します。デフォルトは **False** です。

sshd_manage_service

False に設定すると、サービスは管理対象ではなくなるので、起動時に有効化されず、起動または再読み込みされません。Ansible サービスモジュールが現在 AIX で **enabled** になっていないため、コンテナーまたは AIX 内で実行する時以外はデフォルトで **True** に設定されます。

sshd_allow_reload

False に設定すると、設定の変更後に **sshd** は再読み込みされません。これはトラブルシューティングで役立ちます。変更した設定を適用するには、**sshd** を手動で再読み込みします。AIX を除き、**sshd_manage_service** と同じ値にデフォルト設定されます。ここで、**sshd_manage_service** はデフォルトで **False** に設定されますが、**sshd_allow_reload** はデフォルトで **True** に設定されます。

sshd_install_service

True に設定すると、ロールは **sshd** サービスのサービスファイルをインストールします。これにより、オペレーティングシステムで提供されるファイルが上書きされます。2 つ目のインスタンスを設定し、**sshd_service** 変数も変更しない限り、**True** に設定しないでください。デフォルトは **False** です。
ロールは、以下の変数でテンプレートとして参照するファイルを使用します。

-

```
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

この変数により **sshd** サービス名が変更されます。これは、2 つ目の **sshd** サービスインスタンスを設定するのに役立ちます。

sshd

設定が含まれる dict。以下に例を示します。

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

sshd_OptionName

dict の代わりに、**sshd_** プレフィックスとオプション名で構成される単純な変数を使用してオプションを定義できます。簡単な変数は、**sshd** dict の値を上書きします。以下に例を示します。

```
sshd_Compression: no
```

sshd_match and **sshd_match_1** to **sshd_match_9**

dict のリスト、または Match セクションの dict のみ。これらの変数は、**sshd** dict で定義されている一致するブロックを上書きしないことに注意してください。すべてのソースは作成された設定ファイルに反映されます。

SSH Server システムロールのセカンダリー変数

これらの変数を使用して、サポートされている各プラットフォームに対応するデフォルトを上書きすることができます。

sshd_packages

この変数を使用して、インストール済みパッケージのデフォルト一覧を上書きできます。

sshd_config_owner、sshd_config_group、sshd_config_mode

このロールは、これらの変数を使用して生成する **openssh** 設定ファイルの所有権およびパーミッションを設定できます。

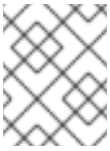
sshd_config_file

このロールが作成した **openssh** サーバー設定を保存するパス。

sshd_config_namespace

この変数のデフォルト値は null です。これは、ロールがシステムのデフォルトを含む設定ファイルの内容全体を定義することを意味します。または、この変数を使用して、他のロールから、またはドロップインディレクトリーをサポートしないシステムの 1 つの Playbook 内の複数の場所から、このロールを呼び出すことができます。**sshd_skip_defaults** 変数は無視され、この場合、システムのデフォルトは使用されません。

この変数が設定されている場合、ロールは指定された namespace の下の既存の設定ファイルの設定スニペットに指定する設定を配置します。シナリオにロールを複数回適用する必要がある場合は、アプリケーションごとに異なる namespace を選択する必要があります。



注記

openssh 設定ファイルの制限は引き続き適用されます。たとえば、設定ファイルで指定した最初のオプションだけが、ほとんどの設定オプションで有効です。

技術的には、ロールは他の一致ブロックが含まれていない限り、スニペットを "Match all" ブロックに配置し、既存の設定ファイル内の以前の一致ブロックに関係なく適用されるようにします。これにより、異なるロール呼び出しから競合しないオプションを設定できます。

sshd_binary

openssh の **sshd** 実行可能ファイルへのパス。

sshd_service

sshd サービスの名前。デフォルトでは、この変数には、ターゲットプラットフォームが使用する **sshd** サービスの名前が含まれます。ロールが **sshd_install_service** 変数を使用する場合は、これを使用してカスタムの **sshd** サービスの名前を設定することもできます。

sshd_verify_hostkeys

デフォルトは **auto** です。**auto** に設定すると、生成された設定ファイルに存在するホストキーがすべて一覧表示され、存在しないパスが生成されます。また、パーミッションおよびファイルの所有者はデフォルト値に設定されます。これは、ロールがデプロイメント段階で使用され、サービスが最初の試行で起動できるようにする場合に便利です。このチェックを無効にするには、この変数を空のリスト **[]** に設定します。

sshd_hostkey_owner, sshd_hostkey_group, sshd_hostkey_mode

これらの変数を使用して、**sshd_verify_hostkeys** からホストキーの所有権とパーミッションを設定します。

sshd_sysconfig

RHEL ベースのシステムでは、この変数は **sshd** サービスに関する追加情報を設定します。**true** に設定すると、このロールは以下の設定に基づいて **/etc/sysconfig/sshd** 設定ファイルも管理します。デフォルトは **false** です。

sshd_sysconfig_override_crypto_policy

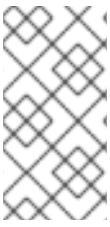
RHEL では、**true** に設定すると、この変数はシステム全体の暗号化ポリシーを上書きします。デフォルトは **false** です。

sshd_sysconfig_use_strong_rng

RHEL ベースのシステムでは、この変数により、**sshd** は、引数として指定されたバイト数を使用して、**openssl** 乱数ジェネレーターを強制的に再シードすることができます。デフォルトは **0** で、この機能を無効にします。システムにハードウェア乱数ジェネレーターがない場合は、この機能を有効にしないでください。

2.2. SSH SERVER システムロールを使用した OPENSSH サーバーの設定

SSH Server システムロールを使用して、Ansible Playbook を実行することで複数の SSH サーバーを設定できます。

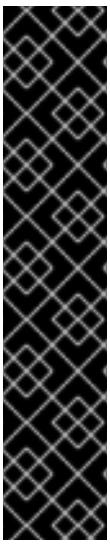


注記

SSH Server システムロールは、SSH および SSHD 設定を変更する他のシステムロール (ID 管理 RHEL システムロールなど) とともに使用できます。設定が上書きされないようにするには、SSH Server ロールがネームスペース (RHEL 8 以前のバージョン) またはドロップインディレクトリー (RHEL 9) を使用していることを確認してください。

前提条件

- 1 つ以上の **管理対象ノード** (SSHD システムロールで設定するシステム) へのアクセスおよびパーミッション。
- **コントロールノード** (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
  コントロールノードでは、
  - **ansible-core** パッケージおよび **rhel-system-roles** パッケージがインストールされている。



重要

RHEL 8.0-8.5 では、別の Ansible リポジトリーへのアクセス権を指定されており、Ansible をベースにする自動化用の Ansible Engine 2.9 が含まれています。Ansible Engine には、**ansible**、**ansible-playbook** などのコマンドラインユーティリティー、**docker** や **podman** などのコネクター、プラグインとモジュールが多く含まれています。Ansible Engine を入手してインストールする方法については、ナレッジベースの How to download and install Red Hat Ansible Engine を参照してください。

RHEL 8.6 および 9.0 では、Ansible Core (**ansible-core** パッケージとして提供) が導入されました。これには、Ansible コマンドラインユーティリティー、コマンド、およびビルトイン Ansible プラグインのセットが含まれています。RHEL は、AppStream リポジトリーを介してこのパッケージを提供し、サポート範囲は限定的です。詳細については、ナレッジベースの Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories を参照してください。

- 管理対象ノードが記載されているインベントリーファイルがある。

手順

1. SSH Server システムロールのサンプル Playbook をコピーします。

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

2. 以下の例のように、テキストエディターでコピーした Playbook を開きます。

```
# 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
```

Playbook は、以下のように、管理対象ノードを SSH サーバーとして設定します。

- パスワードと **root** ユーザーのログインが無効である
- **192.0.2.0/24** のサブネットからのパスワードおよび **root** ユーザーのログインのみが有効である

設定に合わせて変数を変更できます。詳細は、SSHD Server System Role variables を参照してください。

3. オプション:Playbook の構文を確認します。

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

4. インベントリーファイルで Playbook を実行します。

```
# 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
```

検証

1. SSH サーバーにログインします。

```
$ ssh user1@10.1.1.1
```

ここで、

- **user1** は、SSH サーバーのユーザーです。
- **10.1.1.1** は、SSH サーバーの IP アドレスです。

2. SSH サーバーの **sshd_config** ファイルの内容を確認します。

```
$ vim /etc/ssh/sshd_config

# Ansible managed
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
AcceptEnv LANG LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE LC_MONETARY
LC_MESSAGES
AcceptEnv LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE LC_MEASUREMENT
AcceptEnv LC_IDENTIFICATION LC_ALL LANGUAGE
AcceptEnv XMODIFIERS
AuthorizedKeysFile .ssh/authorized_keys
ChallengeResponseAuthentication no
GSSAPIAuthentication yes
GSSAPICleanupCredentials no
```

```
PasswordAuthentication no
PermitRootLogin no
PrintMotd no
Subsystem sftp /usr/libexec/openssh/sftp-server
SyslogFacility AUTHPRIV
UsePAM yes
X11Forwarding yes
Match Address 192.0.2.0/24
  PasswordAuthentication yes
  PermitRootLogin yes
```

3. **192.0.2.0/24** サブネットから root としてサーバーに接続できることを確認します。

   a. IP アドレスを確認します。

   ```
   $ hostname -I
   192.0.2.1
   ```

   IP アドレスが **192.0.2.1** - **192.0.2.254** 範囲にある場合は、サーバーに接続できます。

   b. **root** でサーバーに接続します。

   ```
   $ ssh root@10.1.1.1
   ```

関連情報

- **/usr/share/doc/rhel-system-roles/sshd/README.md** ファイル。
- **ansible-playbook(1)** man ページ。

2.3. SSH CLIENT システムロール変数

SSH Client システムロール Playbook では、設定および制限に応じて、クライアント SSH 設定ファイルのパラメーターを定義できます。

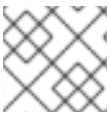
これらの変数が設定されていない場合には、システムロールは RHEL のデフォルト値と同じグローバル **ssh_config** ファイルを作成します。

どのような場合でも、ブール値は **ssh** 設定で適切に **yes** または **no** とレンダリングされます。一覧を使用して複数行の設定項目を定義できます。以下に例を示します。

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

レンダリングは以下のようになります。

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



注記

設定オプションでは、大文字と小文字が区別されます。

SSH Client システムロールの変数

ssh_user

システムロールでユーザー固有の設定を変更するように、既存のユーザー名を定義できます。ユーザー固有の設定は、指定したユーザーの **~/.ssh/config** に保存されます。デフォルト値は null で、すべてのユーザーに対するグローバル設定を変更します。

ssh_skip_defaults

デフォルトは **auto** です。**auto** に設定すると、システムロールはシステム全体の設定ファイル **/etc/ssh/ssh_config** を読み取り、そこで定義した RHEL のデフォルトを保持します。**ssh_drop_in_name** 変数を定義してドロップイン設定ファイルを作成すると、**ssh_skip_defaults** 変数が自動的に無効化されます。

ssh_drop_in_name

システム全体のドロップインディレクトリーに置かれたドロップイン設定ファイルの名前を定義します。この名前は、変更する設定ファイルを参照するテンプレート **/etc/ssh/ssh_config.d/{ssh_drop_in_name}.conf** で使用されます。システムがドロップインディレクトリーに対応していない場合、デフォルト値は null です。システムがドロップインディレクトリーに対応している場合、デフォルト値は **00-ansible** です。

警告

システムがドロップインディレクトリーに対応していない場合は、このオプションを設定すると、プレイに失敗します。

推奨される形式は **NN-name** です。**NN** は、設定ファイルの指定に使用する 2 桁の番号で、 **name** はコンテンツまたはファイルの所有者を示す名前になります。

ssh

設定オプションとその値が含まれる dict。

ssh_OptionName

dict の代わりに、**ssh_** プレフィックスとオプション名で構成される単純な変数を使用してオプションを定義できます。簡単な変数は、**ssh** dict の値を上書きします。

ssh_additional_packages

このロールは、一般的なユースケースに必要な **openssh パッケージ および openssh-clients** パッケージを自動的にインストールします。ホストベースの認証用に **openssh-keysign** などの追加のパッケージをインストールする必要がある場合は、この変数で指定できます。

ssh_config_file

ロールが生成した設定ファイルを保存するパス。デフォルト値:

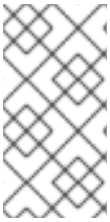
- システムにドロップインディレクトリーがある場合、デフォルト値は **/etc/ssh/ssh_config.d/{ssh_drop_in_name}.conf** テンプレートで定義されます。
- システムにドロップインディレクトリーがない場合、デフォルト値は **/etc/ssh/ssh_config** になります。
- **ssh_user** 変数が定義されている場合、デフォルト値は **~/.ssh/config** になります。

ssh_config_owner, ssh_config_group, ssh_config_mode

作成した設定ファイルの所有者、グループ、およびモード。デフォルトでは、ファイルの所有者は **root:root** で、モードは **0644** です。**ssh_user** が定義されている場合、モードは **0600** で、owner と group は **ssh_user** 変数で指定したユーザー名から派生します。

2.4. SSH CLIENT システムロールを使用した OPENSSH クライアントの設定

SSH Client システムロールを使用して、Ansible Playbook を実行して複数の SSH クライアントを設定できます。



注記

SSH Client システムロールは、SSH および SSHD 設定を変更する他のシステムロール (ID 管理 RHEL システムロールなど) とともに使用できます。設定が上書きされないようにするには、SSH Client ロールがドロップインディレクトリー (RHEL 8 から デフォルト) を使用していることを確認してください。

前提条件

- 1 つ以上の **管理対象ノード** (SSH Client システムロールで設定するシステム) へのアクセスおよびパーミッション。
- **コントロールノード** (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
  コントロールノードでは、
  - **ansible-core** パッケージおよび **rhel-system-roles** パッケージがインストールされている。

重要

RHEL 8.0-8.5 では、別の Ansible リポジトリーへのアクセス権を指定されており、Ansible をベースにする自動化用の Ansible Engine 2.9 が含まれています。Ansible Engine には、**ansible**、**ansible-playbook** などのコマンドラインユーティリティー、**docker** や **podman** などのコネクター、プラグインとモジュールが多く含まれています。Ansible Engine を入手してインストールする方法については、ナレッジベースの How to download and install Red Hat Ansible Engine を参照してください。

RHEL 8.6 および 9.0 では、Ansible Core (**ansible-core** パッケージとして提供) が導入されました。これには、Ansible コマンドラインユーティリティー、コマンド、およびビルトイン Ansible プラグインのセットが含まれています。RHEL は、AppStream リポジトリーを介してこのパッケージを提供し、サポート範囲は限定的です。詳細については、ナレッジベースの Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories を参照してください。

- 管理対象ノードが記載されているインベントリーファイルがある。

手順

1. 以下の内容を含む新しい **playbook.yml** ファイルを作成します。

```
---
- 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
```

この Playbook は、以下の設定を使用して、管理対象ノードで **root** ユーザーの SSH クライアント設定を行います。

- 圧縮が有効になっている。
- ControlMaster multiplexing が **auto** に設定されている。
- **example.com** ホストに接続するための **example** エイリアスが **user1** である。
- ホストエイリアスの **example** が作成されている。(これはユーザー名が **user1** の **example.com** ホストへの接続を表します。)
- X11 転送が無効化されている。

必要に応じて、これらの変数は設定に合わせて変更できます。詳細は、SSH System Role variables を参照してください。

2. オプション:Playbook の構文を確認します。

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

3. インベントリーファイルで Playbook を実行します。

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

検証

- テキストエディターで SSH 設定ファイルを開いて、管理対象ノードが正しく設定されていることを確認します。以下に例を示します。

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

上記の Playbook の例の適用後に、設定ファイルの内容は以下のようになるはずです。

```
# Ansible managed
Compression yes
ControlMaster auto
ControlPath ~/.ssh/.cm%C
ForwardX11 no
```

```
GSSAPIAuthentication no
Host example
  Hostname example.com
  User user1
```

2.5. 非排他的設定での SSH サーバーシステムロールの使用

通常、SSH Server システムロールを適用すると、設定全体が上書きされます。これは、たとえば別のシステムロールや Playbook などを使用して、以前に設定を調整している場合に問題が発生する可能性があります。他のオプションをそのまま維持しながら、選択した設定オプションのみに SSH Server システムロールを適用するには、非排他的設定を使用できます。

RHEL 8 以前では、設定スニペットを使用して非排他的設定を適用することができます。詳細は、RHEL 8 ドキュメントの Using the SSH Server System Role for non-exclusive configuration を参照してください。

RHEL 9 では、ドロップインディレクトリーのファイルを使用して、非排他的設定を適用することができます。デフォルトの設定ファイルは、**/etc/ssh/sshd_config.d/00-ansible_system_role.conf** としてドロップインディレクトリーにすでに配置されています。

前提条件

- 1 つ以上の **管理対象ノード** (SSH Server システムロールで設定するシステム) へのアクセスおよびパーミッション。
- **コントロールノード** (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
  コントロールノードでは、
  - **ansible-core** パッケージがインストールされている。
  - 管理対象ノードが記載されているインベントリーファイルがある。
  - 別の RHEL システムロールの Playbook。詳細は、Applying a role を参照してください。

手順

1. **sshd_config_file** 変数を含む設定スニペットを Playbook に追加します。

```
---
- 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
```

sshd_config_file 変数で、SSH Server システムロールが設定オプションを書き込む **.conf** ファイルを定義します。

設定ファイルが適用される順序を指定するには、2 桁のプレフィックス（例: **42-**）を使用します。

Playbook をインベントリーに適用すると、ロールは **sshd_config_file** 変数で定義されるファイルに次の設定オプションを追加します。

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

検証

- オプション:Playbook の構文を確認します。

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

関連情報

- **/usr/share/doc/rhel-system-roles/sshd/README.md** ファイル。
- **ansible-playbook(1)** man ページ。

第3章 TLS の計画および実施

TLS (トランスポート層セキュリティー) は、ネットワーク通信のセキュリティー保護に使用する暗号化プロトコルです。優先する鍵交換プロトコル、認証方法、および暗号化アルゴリズムを設定してシステムのセキュリティー設定を強化する際には、対応するクライアントの範囲が広ければ広いほど、セキュリティーのレベルが低くなることを認識しておく必要があります。反対に、セキュリティー設定を厳密にすると、クライアントとの互換性が制限され、システムからロックアウトされるユーザーが出てくる可能性もあります。可能な限り厳密な設定を目指し、互換性に必要な場合に限り、設定を緩めるようにしてください。

3.1. SSL プロトコルおよび TLS プロトコル

Secure Sockets Layer (SSL) プロトコルは、元々はインターネットを介した安全な通信メカニズムを提供するために、Netscape Corporation により開発されました。その後、このプロトコルは、Internet Engineering Task Force (IETF) により採用され、Transport Layer Security (TLS) に名前が変更になりました。

TLS プロトコルは、アプリケーションプロトコル層と、TCP/IP などの信頼性の高いトランスポート層の間にあります。これは、アプリケーションプロトコルから独立しているため、さまざまなプロトコルの下に階層化できます。(HTTP、FTP、SMTP など)

プロトコルのバージョン	推奨される使用方法
SSL v2	使用しないでください。深刻なセキュリティー上の脆弱性があります。RHEL 7 以降、コア暗号ライブラリーから削除されました。
SSL v3	使用しないでください。深刻なセキュリティー上の脆弱性があります。RHEL 8 以降、コア暗号ライブラリーから削除されました。
TLS 1.0	使用は推奨されません。相互運用性を保証した方法では軽減できない既知の問題があり、最新の暗号スイートには対応しません。RHEL 9 では、すべての暗号化ポリシーで無効になります。
TLS 1.1	必要に応じて相互運用性の目的で使用します。最新の暗号スイートには対応しません。RHEL 9 では、すべての暗号化ポリシーで無効になります。
TLS 1.2	最新の AEAD 暗号スイートに対応します。このバージョンは、システム全体のすべての暗号化ポリシーで有効になっていますが、このプロトコルの必須ではない部分に脆弱性があります。また、TLS 1.2 では古いアルゴリズムも使用できます。
TLS 1.3	推奨されるバージョン。TLS 1.3 は、既知の問題があるオプションを取り除き、より多くのネゴシエーションハンドシェイクを暗号化することでプライバシーを強化し、最新の暗号アルゴリズムをより効果的に使用することで速度を速めることができます。TLS 1.3 は、システム全体のすべての暗号化ポリシーでも有効になっています。

関連情報

- IETF: The Transport Layer Security (TLS) Protocol Version 1.3

3.2. RHEL 9 における TLS のセキュリティー上の検討事項

RHEL 9 では、TLS 設定はシステム全体の暗号化ポリシーメカニズムを使用して実行されます。1.2 未満の TLS バージョンはサポートされなくなりました。**DEFAULT**、**FUTURE**、および **LEGACY** 暗号化ポリシーでは TLS 1.2 および 1.3 のみを使用できます。詳細は、Using system-wide cryptographic policies を参照してください。

RHEL 9 に含まれるライブラリーが提供するデフォルト設定は、ほとんどのデプロイメントで十分に安全です。TLS 実装は、可能な場合は、安全なアルゴリズムを使用する一方で、レガシーなクライアントまたはサーバーとの間の接続は妨げません。セキュリティーが保護されたアルゴリズムまたはプロトコルに対応しないレガシーなクライアントまたはサーバーの接続が期待できないまたは許可されない場合に、厳密なセキュリティー要件の環境で、強化設定を適用します。

TLS 設定を強化する最も簡単な方法は、**update-crypto-policies --set FUTURE** コマンドを実行して、システム全体の暗号化ポリシーレベルを **FUTURE** に切り替えます。

警告

LEGACY 暗号化ポリシーで無効にされているアルゴリズムは、Red Hat の RHEL 9 セキュリティーのビジョンに準拠しておらず、それらのセキュリティプロパティーは信頼できません。これらのアルゴリズムを再度有効化するのではなく、使用しないようにすることを検討してください。たとえば、古いハードウェアとの相互運用性のためにそれらを再度有効化することを決めた場合は、それらを安全でないものとして扱い、ネットワークの相互作用を個別のネットワークセグメントに分離するなどの追加の保護手段を適用します。パブリックネットワーク全体では使用しないでください。

RHEL システム全体の暗号化ポリシーに従わない場合、またはセットアップに適したカスタム暗号化ポリシーを作成する場合は、カスタム設定で必要なプロトコル、暗号スイート、および鍵の長さについて、以下の推奨事項を使用します。

3.2.1. プロトコル

最新バージョンの TLS は、最高のセキュリティーメカニズムを提供します。TLS 1.2 は、**LEGACY** 暗号化ポリシーを使用する場合でも最小バージョンになりました。古いプロトコルバージョンを再度有効にするには、暗号化ポリシーをオプトアウトするか、カスタムポリシーを提供することで可能ですが、この結果生成される設定はサポートされません。

RHEL 9 は TLS バージョン 1.3 をサポートしていますが、このプロトコルのすべての機能が RHEL 9 コンポーネントで完全にサポートされているわけではない点に注意してください。たとえば、接続レイテンシーを短縮する 0-RTT (Zero Round Trip Time) 機能は、Apache Web サーバーではまだ完全にはサポートされていません。

3.2.2. 暗号化スイート

旧式で、安全ではない暗号化スイートではなく、最近の、より安全なものを使用してください。暗号化スイートの eNULL および aNULL は、暗号化や認証を提供しないため、常に無効にしてください。RC4 や HMAC-MD5 をベースとした暗号化スイートには深刻な欠陥があるため、可能な場合はこれも無効にしてください。いわゆるエクスポート暗号化スイートも同様です。エクスポート暗号化スイートは意図的に弱くなっているため、侵入が容易になっています。

128 ビット未満のセキュリティーしか提供しない暗号化スイートでは直ちにセキュリティーが保護され

なくなるというわけではありませんが、使用できる期間が短いため考慮すべきではありません。アルゴリズムが 128 ビット以上のセキュリティーを使用している場合は、少なくとも数年間は解読不可能であることが期待されているため、強く推奨されます。3DES 暗号は 168 ビットを使用していると言われていますが、実際に提供されているのは 112 ビットのセキュリティーであることに注意してください。

サーバーの鍵が危険にさらされた場合でも、暗号化したデータの機密性を保証する (完全な) 前方秘匿性 (PFS) に対応する暗号スイートを常に優先します。ここでは、速い RSA 鍵交換は除外されますが、ECDHE および DHE は使用できます。この 2 つを比べると、ECDHE の方が速いため推奨されます。

また、AES-GCM などの AEAD 暗号は、パディングオラクル攻撃の影響は受けないため、CBC モード暗号よりも推奨されます。さらに、多くの場合、特にハードウェアに AES 用の暗号化アクセラレーターがある場合、AES-GCM は CBC モードの AES よりも高速です。

ECDSA 証明書で ECDHE 鍵交換を使用すると、トランザクションは純粋な RSA 鍵交換よりもさらに高速になります。レガシークライアントに対応するため、サーバーには証明書と鍵のペアを 2 つ (新しいクライアント用の ECDSA 鍵と、レガシー用の RSA 鍵) インストールできます。

3.2.3. 公開鍵の長さ

RSA 鍵を使用する際は、SHA-256 以上で署名され、鍵の長さが 3072 ビット以上のものが常に推奨されます (これは、実際に 128 ビットであるセキュリティーに対して十分な大きさです)。

警告

システムのセキュリティー強度は、チェーンの中の最も弱いリンクが示すものと同じになります。たとえば、強力な暗号化だけではすぐれたセキュリティーは保証されません。鍵と証明書も同様に重要で、認証機関 (CA) が鍵の署名に使用するハッシュ機能と鍵もまた重要になります。

3.3. アプリケーションで TLS 設定の強化

RHEL では、システム全体の暗号化ポリシー は、暗号化ライブラリーを使用するアプリケーションが、既知の安全でないプロトコル、暗号化、またはアルゴリズムを許可しないようにするための便利な方法を提供します。

暗号化設定をカスタマイズして、TLS 関連の設定を強化する場合は、このセクションで説明する暗号化設定オプションを使用して、必要最小量でシステム全体の暗号化ポリシーを上書きできます。

いずれの設定を選択しても、サーバーアプリケーションが強制的に **サーバー側が指定した順序** で暗号を利用することを確認し、使用される暗号化スイートの選択がサーバでの設定順に行われるように設定してください。

3.3.1. Apache HTTP サーバー の設定

Apache HTTP Server は、TLS のニーズに **OpenSSL** ライブラリーおよび **NSS** ライブラリーの両方を使用できます。RHEL 9 では、mod_ss パッケージで **mod_ssl** 機能が提供されます。

```
# dnf install mod_ssl
```

mod_ssl パッケージは、**/etc/httpd/conf.d/ssl.conf** 設定ファイルをインストールします。これは、**Apache HTTP Server** の TLS 関連の設定を変更するのに使用できます。

httpd-manual パッケージをインストールして、TLS 設定を含む **Apache HTTP Server** の完全ドキュメントを取得します。**/etc/httpd/conf.d/ssl.conf** 設定ファイルで利用可能なディレクティブの詳細は、**/usr/share/httpd/manual/mod/mod_ssl.html** を参照してください。さまざまな設定の例は、**/usr/share/httpd/manual/ssl/ssl_howto.html** ファイルに記載されています。

/etc/httpd/conf.d/ssl.conf 設定ファイルの設定を修正する場合は、少なくとも下記の 3 つのディレクティブを確認してください。

SSLProtocol

このディレクティブを使用して、許可する TLS または SSL のバージョンを指定します。

SSLCipherSuite

優先する暗号化スイートを指定する、もしくは許可しないスイートを無効にするディレクティブです。

SSLHonorCipherOrder

コメントを解除して、このディレクティブを **on** に設定すると、接続先のクライアントは指定した暗号化の順序に従います。

たとえば、TLS 1.2 プロトコルおよび 1.3 プロトコルだけを使用する場合は、以下を実行します。

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

詳細は、Deploying web servers and reverse proxies の Configuring TLS encryption on an Apache HTTP Server の章を参照してください。

3.3.2. Nginx HTTP およびプロキシーサーバーの設定

Nginx で TLS 1.3 サポートを有効にするには、**/etc/nginx/nginx.conf** 設定ファイルの **server** セクションで、**ssl_protocols** オプションに **TLSv1.3** 値を追加します。

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

詳細は、Deploying web servers and reverse proxies の Adding TLS encryption to an Nginx web server の章を参照してください。

3.3.3. Dovecot メールサーバーの設定

Dovecot メールサーバーのインストールが TLS を使用するように設定するには、**/etc/dovecot/conf.d/10-ssl.conf** 設定ファイルを修正します。このファイルで利用可能な基本的な設定ディレクティブの一部は、**/usr/share/doc/dovecot/wiki/SSL.DovecotConfiguration.txt** ファイルで説明されています。このファイルは **Dovecot** の標準インストールに含まれています。

/etc/dovecot/conf.d/10-ssl.conf 設定ファイルの設定を修正する場合は、少なくとも下記の 3 つのディレクティブを確認してください。

ssl_protocols

このディレクティブを使用して、許可または無効にする TLS または SSL のバージョンを指定します。

ssl_cipher_list

優先する暗号化スイートを指定する、もしくは許可しないスイートを無効にするディレクティブです。

ssl_prefer_server_ciphers

コメントを解除して、このディレクティブを **yes** に設定すると、接続先のクライアントは指定した暗号化の順序に従います。

たとえば、**/etc/dovecot/conf.d/10-ssl.conf** 内の次の行が、TLS 1.1 以降だけを許可します。

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

関連情報

- Web サーバーとリバースプロキシーのデプロイ
- **config(5)** および **ciphers(1)** の man ページ
- Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)
- Mozilla SSL Configuration Generator
- SSL Server Test

第4章 IPSEC を使用した VPN の設定

RHEL 9 では、仮想プライベートネットワーク(VPN)は**IPsec**プロトコルを使用して設定できます。これは、**Libreswan** アプリケーションによりサポートされます。

4.1. IPSEC VPN 実装としての LIBRESWAN

RHEL では、仮想プライベートネットワーク(VPN)は IPsec プロトコルを使用して設定できます。これは、Libreswan アプリケーションによりサポートされます。Libreswan は、Openswan アプリケーションの延長であり、Openswan ドキュメントの多くの例は Libreswan でも利用できます。

VPN の IPsec プロトコルは、IKE (Internet Key Exchange) プロトコルを使用して設定されます。IPsec と IKE は同義語です。IPsec VPN は、IKE VPN、IKEv2 VPN、XAUTH VPN、Cisco VPN、または IKE/IPsec VPN とも呼ばれます。Layer 2 Tunneling Protocol(L2TP)も使用する IPsec VPN のバリアントは、通常 L2TP/IPsec VPN と呼ばれます。これには、**オプション** のリポジトリーが提供する **xl2tpd** パッケージが必要です。

Libreswan は、オープンソースのユーザー空間の IKE 実装です。IKE v1 および v2 は、ユーザーレベルのデーモンとして実装されます。IKE プロトコルも暗号化されています。IPsec プロトコルは Linux カーネルで実装され、Libreswan は、VPN トンネル設定を追加および削除するようにカーネルを設定します。

IKE プロトコルは、UDP ポート 500 および 4500 を使用します。IPsec プロトコルは、以下の 2 つのプロトコルで構成されます。

- 暗号セキュリティーペイロード (ESP) (プロトコル番号が 50)
- 認証ヘッダー (AH) (プロトコル番号 51)

AH プロトコルの使用は推奨されていません。AH のユーザーは、null 暗号化で ESP に移行することが推奨されます。

IPsec プロトコルは、以下の 2 つの操作モードを提供します。

- トンネルモード (デフォルト)
- トランスポートモード

IKE を使用せずに IPsec を使用してカーネルを設定できます。これは、**手動キーリング** と呼ばれます。また、**ip xfrm** コマンドを使用して手動キーを設定できますが、これはセキュリティー上の理由からは強く推奨されません。Libreswan では、netlink を使用する Linux カーネルで相互作用が行われます。Linux カーネルでパケットの暗号化と復号が行われます。

Libreswan は、ネットワークセキュリティーサービス (NSS) 暗号化ライブラリーを使用します。Libreswan および NSS はともに、**連邦情報処理標準** (**FIPS**) の公開文書 140-2 での使用が認定されています。

重要

Libreswan および Linux カーネルが実装する IKE/IPsec の VPN は、RHEL で使用することが推奨される唯一の VPN 技術です。その他の VPN 技術は、そのリスクを理解せずに使用しないでください。

RHEL では、Libreswan はデフォルトで **システム全体の暗号化ポリシー** に従います。これにより、Libreswan は、デフォルトのプロトコルとして IKEv2 を含む現在の脅威モデルに対して安全な設定を使用するようになります。詳細は、Using system-wide crypto policies を参照してください。

IKE/IPsec はピアツーピアプロトコルであるため、Libreswan では、「ソース」および「宛先」、または「サーバー」および「クライアント」という用語を使用しません。終了点 (ホスト) を参照する場合は、代わりに「左」と「右」という用語を使用します。これにより、ほとんどの場合、両方の終了点で同じ設定も使用できます。ただし、管理者は通常、ローカルホストに「左」を使用し、リモートホストに「右」を使用します。

leftid と **rightid** オプションは、認証プロセス内の各ホストの識別として機能します。詳細は、man ページの **ipsec.conf(5)** を参照してください。

4.2. LIBRESWAN の認証方法

Libreswan は複数の認証方法をサポートしますが、それぞれは異なるシナリオとなっています。

事前共有キー（PSK）

事前共有キー (PSK) は、最も簡単な認証メソッドです。セキュリティー上の理由から、PSK は 64 文字未満は使用しないでください。FIPSモードでは、PSKは、使用される整合性アルゴリズムに応じて、、最低強度の要件に準拠する必要があります。**authby=secret** 接続を使用して PSK を設定できます。

Raw RSA 鍵

Raw RSA 鍵 は、静的なホスト間またはサブネット間の IPsec 設定で一般的に使用されます。各ホストは、他のすべてのホストのパブリック RSA 鍵を使用して手動で設定され、Libreswan はホストの各ペア間で IPsec トンネルを設定します。この方法は、多数のホストでは適切にスケーリングされません。

ipsec newhostkey コマンドを使用して、ホストで Raw RSA 鍵を生成できます。 **ipsec showhostkey** コマンドを使用して、生成された鍵を一覧表示できます。**leftrsasigkey=** の行は、CKA ID キーを使用する接続設定に必要です。Raw RSA 鍵に **authby=rsasig** 接続オプションを使用します。

X.509 証明書

X.509 証明書 は、共通の IPsec ゲートウェイに接続するホストが含まれる大規模なデプロイメントに一般的に使用されます。中央の **認証局** (CA) は、ホストまたはユーザーの RSA 証明書に署名します。この中央 CA は、個別のホストまたはユーザーの取り消しを含む、信頼のリレーを行います。

たとえば、**openssl** コマンドおよび NSS **certutil** コマンドを使用して X.509 証明書を生成できます。Libreswan は、**leftcert=** 設定オプションの証明書のニックネームを使用して NSS データベースからユーザー証明書を読み取るため、証明書の作成時にニックネームを指定します。

カスタム CA 証明書を使用する場合は、これを Network Security Services(NSS)データベースにインポートする必要があります。**ipsec import** コマンドを使用して、PKCS #12 形式の証明書を Libreswan NSS データベースにインポートできます。

警告

Libreswan は、section 3.1 of RFC 4945 で説明されているように、すべてのピア証明書のサブジェクト代替名(SAN)としてインターネット鍵 Exchange(IKE)ピア ID を必要とします。**require-id-on-certificated=** オプションを変更してこのチェックを無効にすると、システムが中間者攻撃に対して脆弱になる可能性があります。

SHA-2 で RSA を使用した X.509 証明書に基づく認証に **authby=rsasig** 接続オプションを使用します。**authby=**を **ecdsa** に設定し、**authby=rsa-sha2** を介した SHA-2 による RSA Probabilistic Signature Scheme (RSASSA-PSS) デジタル署名ベースの認証を設定することにより、SHA-2を使用する ECDSA デジタル署名に対してさらに制限することができます。デフォルト値は **authby=rsasig,ecdsa** です。

証明書と **authby=** 署名メソッドが一致する必要があります。これにより、相互運用性が向上し、1 つのデジタル署名システムでの認証が維持されます。

NULL 認証

null 認証 は、認証なしでメッシュの暗号化を取得するために使用されます。これは、パッシブ攻撃は防ぎますが、アクティブ攻撃は防ぎません。ただし、IKEv2 は非対称認証メソッドを許可するため、NULL 認証はインターネットスケールの日和見 IPsec にも使用できます。このモデルでは、クライアントはサーバーを認証しますが、サーバーはクライアントを認証しません。このモデルは、TLS を使用して Web サイトのセキュリティーを保護するのと似ています。NULL 認証に **authby=null** を使用します。

量子コンピューターに対する保護

上記の認証方法に加えて、**Post-quantum Pre-shared Key** (PPK) メソッドを使用して、量子コンピューターによる潜在的な攻撃から保護することができます。個々のクライアントまたはクライアントグループは、帯域外で設定された事前共有鍵に対応する PPK ID を指定することにより、独自の PPK を使用できます。

事前共有鍵が設定されている IKEv1 を使用すると、量子攻撃者からの保護が提供されます。IKEv2 の再設計は、この保護をネイティブに提供しません。Libreswan は、**Post-quantum Pre-shared Key** (PPK) を使用して、量子攻撃に対して IKEv2 接続を保護します。

任意の PPK 対応を有効にする場合は、接続定義に **ppk=yes** を追加します。PPK が必要な場合は **ppk=insist** を追加します。次に、各クライアントには、帯域外で通信する (および可能であれば量子攻撃に対して安全な) シークレット値を持つ PPK ID を付与できます。PPK はランダム性において非常に強力で、辞書の単語に基づいていません。PPK ID および PPK データは **ipsec.secrets** に保存されます。以下に例を示します。

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

PPKS オプションは、静的な PPK を参照します。実験的な関数は、ワンタイムパッドに基づいた動的 PPK を使用します。各接続では、ワンタイムパッドの新しい部分が PPK として使用されます。これを使用すると、ファイル内の動的な PPK の部分がゼロで上書きされ、再利用を防ぐことができます。複数のタイムパッドマテリアルが残っていないと、接続は失敗します。詳細は、man ページの **ipsec.secrets(5)** を参照してください。

警告

動的 PPK の実装はサポート対象外のテクノロジープレビューとして提供されます。注意して使用してください。

4.3. LIBRESWAN のインストール

この手順では、Libreswan IPsec/IKE VPN 実装をインストールおよび起動を行う手順を説明します。

前提条件

- **AppStream** リポジトリーが有効になっている。

手順

1. **libreswan** パッケージをインストールします。

```
# dnf install libreswan
```

2. Libreswan を再インストールする場合は、古いデータベースファイルを削除し、新しいデータベースを作成します。

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

3. **ipsec** サービスを開始して有効にし、システムの起動時にサービスを自動的に開始できるようにします。

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

4. ファイアウォールで、**ipsec** サービスを追加して、IKE プロトコル、ESP プロトコル、および AH プロトコルの 500/UDP ポートおよび 4500/UDP ポートを許可するように設定します。

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

4.4. ホスト間の VPN の作成

Libreswan が、Raw RSA 鍵による認証を使用して、**left** と **right**と呼ばれる 2 つのホスト間にホスト間の IPsec VPN を作成するように設定するには、両方のホストに以下のコマンドを入力します。

前提条件

- Libreswan がインストールされ、**ipsec** サービスが各ノードで開始している。

手順

1. 各ホストで Raw RSA 鍵ペアを生成します。

```
# ipsec newhostkey
```

2. 前の手順で生成した鍵の **ckaid** を返します。**左** で次のコマンドを実行して、その **ckaid** を使用します。以下に例を示します。

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

上のコマンドの出力により、設定に必要な **leftrsasigkey=** 行が生成されます。次のホスト (**右**) でも同じ操作を行います。

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

3. **/etc/ipsec.d/** ディレクトリーで、新しい **my_host-to-host.conf** ファイルを作成します。上の手順の **ipsec showhostkey** コマンドの出力から、RSA ホストの鍵を新規ファイルに書き込みます。以下に例を示します。

```
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. 鍵をインポートしたら、**ipsec** サービスを再起動します。

```
# systemctl restart ipsec
```

5. 接続を読み込みます。

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

6. トンネルを確立します。

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

7. **ipsec** サービスの開始時に自動的にトンネルを開始するには、以下の行を接続定義に追加します。

```
auto=start
```

4.5. サイト間 VPN の設定

2 つのネットワークを結合してサイト間の IPsec VPN を作成する場合は、その 2 つのホスト間の IPsec トンネルを作成します。これにより、ホストは終了点として動作し、1 つまたは複数のサブネットからのトラフィックが通過できるように設定されます。したがって、ホストを、ネットワークのリモート部分にゲートウェイとして見なすことができます。

サイト間の VPN の設定は、設定ファイル内で複数のネットワークまたはサブネットを指定する必要がある点のみが、ホスト間の VPN とは異なります。

前提条件

- ホスト間の VPN が設定されている。

手順

1. ホスト間の VPN の設定が含まれるファイルを、新規ファイルにコピーします。以下に例を示します。

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

2. 上の手順で作成したファイルに、サブネット設定を追加します。以下に例を示します。

```
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
```

4.6. リモートアクセスの VPN の設定

ロードウォーリアーとは、モバイルクライアントと動的に割り当てられた IP アドレスを使用する移動するユーザーのことです。モバイルクライアントは、X.509 証明書を使用して認証します。

以下の例では、**IKEv2** の設定を示しています。**IKEv1** XAUTH プロトコルは使用していません。

サーバー上では以下の設定になります。

```
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
```

ロードウォーリアーのデバイスであるモバイルクライアントでは、上記の設定に多少変更を加えて使用します。

```
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
```

4.7. メッシュ VPN の設定

any-to-any VPN とも呼ばれるメッシュ VPN ネットワークは、全ノードが IPsec を使用して通信するネットワークです。この設定では、IPsec を使用できないノードの例外が許可されます。メッシュの VPN ネットワークは、以下のいずれかの方法で設定できます。

- IPSec を必要とする。
- IPsec を優先するが、平文通信へのフォールバックを可能にする。

ノード間の認証は、X.509 証明書または DNSSEC (DNS Security Extensions) を基にできます。

以下の手順では、X.509 証明書を使用します。これらの証明書は、Dongtag Certificate System などのいかなる種類の認証局 (CA) 管理システムを使用して生成できます。Dogtag は、各ノードの証明書が PKCS #12 形式 (.p12 ファイル) で利用可能であることを前提としています。これには、秘密鍵、ノード

証明書、およびその他のノードの X.509 証明書を検証するのに使用されるルート CA 証明書が含まれます。

各ノードでは、その X.509 証明書を除いて、同じ設定を使用します。これにより、ネットワーク内で既存ノードを再設定せずに、新規ノードを追加できます。PKCS #12 ファイルには「分かりやすい名前」が必要であるため、名前には「node」を使用します。これにより、すべてのノードに対して、この名前を参照する設定ファイルが同一になります。

前提条件

- Libreswan がインストールされ、**ipsec** サービスが各ノードで開始している。

手順

1. 各ノードで PKCS #12 ファイルをインポートします。この手順では、PKCS #12 ファイルの生成に使用するパスワードが必要になります。

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

2. **IPsec required** (private)、**IPsec optional** (private-or-clear)、および **No IPsec** (clear) プロファイルに、以下のような 3 つの接続定義を作成します。

```
# 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. 適切なカテゴリーに、ネットワークの IP アドレスを追加します。たとえば、すべてのノードが 10.15.0.0/16 ネットワークにある場合は、すべてのノードに IPsec 暗号が必要です。

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

4. 特定のノード (例: 10.15.34.0/24) を、IPsec を使用または使用せずに機能させるには、以下の設定を使用して、これらのノードを private-or-clear グループに追加します。

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

5. ホストを、10.15.1.2 など、IPsec の機能がない clear グループに定義する場合は、次のコマンドを実行します。

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

/etc/ipsec.d/policies ディレクトリーのファイルは、各新規ノードのテンプレートから作成することも、Puppet または Ansible を使用してプロビジョニングすることもできます。

すべてのノードでは、例外の一覧が同じか、異なるトラフィックフローが期待される点に注意してください。したがって、あるノードで IPsec が必要になり、別のノードで IPsec を使用できないために、ノード間の通信ができない場合もあります。

6. ノードを再起動して、設定したメッシュに追加します。

```
# systemctl restart ipsec
```

7. ノードを追加したら、**ping** コマンドで IPsec トンネルを開くだけで十分です。ノードが開くトンネルを確認するには、次のコマンドを実行します。

```
# ipsec trafficstatus
```

4.8. FIPS 準拠の IPSEC VPN のデプロイメント

この手順を使用して、Libreswan に基づく FIPS 準拠の IPsec VPN ソリューションをデプロイします。次の手順では、FIPS モードの Libreswan で使用可能な暗号化アルゴリズムと無効になっている暗号化アルゴリズムを識別することもできます。

前提条件

- **AppStream** リポジトリーが有効になっている。

手順

1. **libreswan** パッケージをインストールします。

```
# dnf install libreswan
```

2. Libreswan を再インストールする場合は、古い NSS データベースを削除します。

```
# systemctl stop ipsec

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

3. **ipsec** サービスを開始して有効にし、システムの起動時にサービスを自動的に開始できるようにします。

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

4. ファイアウォールで、**ipsec** サービスを追加して、IKE プロトコル、ESP プロトコル、および AH プロトコルの 500/UDP ポートおよび 4500/UDP ポートを許可するように設定します。

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

5. システムを FIPS モードに切り替えます。

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

6. システムを再起動して、カーネルを FIPS モードに切り替えます。

```
# reboot
```

検証

1. Libreswan が FIPS モードで実行していることを確認するには、次のコマンドを実行します。

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

2. または、**systemd** ジャーナルで **ipsec** ユニットのエントリーを確認します。

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

3. FIPS モードで使用可能なアルゴリズムを表示するには、次のコマンドを実行します。

```
# 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. FIPS モードで無効化されたアルゴリズムをクエリーするには、次のコマンドを実行します。

```
# 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. FIPS モードで許可されているすべてのアルゴリズムと暗号の一覧を表示するには、次のコマンドを実行します。

```
# 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
```

関連情報

- Using system-wide cryptographic policies .

4.9. パスワードによる IPSEC NSS データベースの保護

デフォルトでは、IPsec サービスは、初回起動時に空のパスワードを使用して Network Security Services (NSS) データベースを作成します。以下の手順を使用して、パスワード保護を追加します。

前提条件

- **/var/lib/ipsec/nss/** ディレクトリーには NSS データベースファイルが含まれます。

手順

1. Libreswan の **NSS** データベースのパスワード保護を有効にします。

```
# 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. 前の手順で設定したパスワードが含まれる **/etc/ipsec.d/nsspassword** ファイルを作成します。以下に例を示します。

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

nsspassword ファイルは以下の構文を使用することに注意してください。

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

デフォルトの NSS ソフトウェアトークンは **NSS Certificate DB** です。システムが FIPS モードで実行し場合は、トークンの名前が **NSS FIPS 140-2 Certificate DB** になります。

3. 選択したシナリオに応じて、**nsspassword** ファイルの完了後に **ipsec** サービスを起動または再起動します。

```
# systemctl restart ipsec
```

検証

1. NSS データベースに空でないパスワードを追加した後に、**ipsec** サービスが実行中であることを確認します。

```
# 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. 必要に応じて、**Journal** ログに、初期化が成功したことを確認するエントリーが含まれていることを確認します。

```
# 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
...
```

関連情報

- **certutil(1)** man ページ。
- Government Standards ナレッジベースアーティクル

4.10. TCP を使用するように IPSEC VPN を設定

Libreswan は、RFC 8229 で説明されているように、IKE パケットおよび IPsec パケットの TCP カプセル化に対応します。この機能により、UDP 経由でトラフィックが転送されないように、IPsec VPN をネットワークに確立し、セキュリティーのペイロード (ESP) を強化できます。フォールバックまたはメインの VPN トランスポートプロトコルとして TCP を使用するように VPN サーバーおよびクライアントを設定できます。TCP カプセル化にはパフォーマンスコストが大きくなるため、UDP がシナリオで永続的にブロックされている場合に限り、TCP を主な VPN プロトコルとして使用してください。

前提条件

- リモートアクセス VPN が設定されている。

手順

1. **config setup** セクションの **/etc/ipsec.conf** ファイルに以下のオプションを追加します。

```
listen-tcp=yes
```

2. UDP で最初の試行に失敗した場合に TCP カプセル化をフォールバックオプションとして使用するには、クライアントの接続定義に以下の 2 つのオプションを追加します。

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

または、UDP を永続的にブロックしている場合は、クライアントの接続設定で以下のオプションを使用します。

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

関連情報

- IETF RFC 8229: TCP Encapsulation of IKE and IPsec Packets

4.11. IPSEC 接続を高速化するために、ESP ハードウェアオフロードの自動検出と使用を設定

Encapsulating Security Payload (ESP) をハードウェアにオフロードすると、Ethernet で IPsec 接続が加速します。デフォルトでは、Libreswan は、ハードウェアがこの機能に対応しているかどうかを検出するため、ESP ハードウェアのオフロードを有効にします。この手順では、機能が無効または明示的に有効になっている場合に自動検出を有効にする方法を説明します。

前提条件

- ネットワークカードは、ESP ハードウェアオフロードに対応します。
- ネットワークドライバーは、ESP ハードウェアのオフロードに対応します。

- IPsec 接続が設定され、動作します。

手順

1. ESP ハードウェアオフロードサポートの自動検出を使用する接続の **/etc/ipsec.d/** ディレクトリーにある Libreswan 設定ファイルを編集します。
2. 接続の設定で **nic-offload** パラメーターが設定されていないことを確認します。
3. **nic-offload** を削除した場合は、**ipsec** を再起動します。

```
# systemctl restart ipsec
```

検証

ネットワークカードが ESP ハードウェアオフロードサポートに対応している場合は、以下の手順に従って結果を検証します。

1. IPsec 接続が使用するイーサネットデバイスの **tx_ipsec** カウンターおよび **rx_ipsec** カウンターを表示します。

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

2. IPsec トンネルを介してトラフィックを送信します。たとえば、リモート IP アドレスに ping します。

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

3. イーサネットデバイスの **tx_ipsec** カウンターおよび **rx_ipsec** カウンターを再度表示します。

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

カウンターの値が増えると、ESP ハードウェアオフロードが動作します。

関連情報

- IPsec を使用した VPN の設定

4.12. IPSEC 接続を加速化するためにボンディングでの ESP ハードウェアオフロードの設定

Encapsulating Security Payload (ESP) をハードウェアにオフロードすると、IPsec 接続が加速します。フェイルオーバーの理由でネットワークボンディングを使用する場合、ESP ハードウェアオフロードを設定する要件と手順は、通常のイーサーネットデバイスを使用する要件と手順とは異なります。たとえば、このシナリオでは、ボンディングでオフロードサポートを有効にし、カーネルはボンディングのポートに設定を適用します。

前提条件

- ボンディングのすべてのネットワークカードは、ESP ハードウェアオフロードをサポートしています。
- ネットワークドライバーは、ボンドデバイスで ESP ハードウェアオフロードに対応しています。RHEL では、**ixgbe** ドライバーのみがこの機能をサポートしています。
- ボンディングが設定されており、動作します。
- ボンディングは **active-backup** モードを使用します。ボンディングドライバーは、この機能の他のモードをサポートしません。
- IPsec 接続が設定され、動作します。

手順

1. ネットワークボンディングで ESP ハードウェアオフロードのサポートを有効にします。

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

このコマンドにより、**bond0** 接続での ESP ハードウェアオフロードのサポートが有効になります。

2. **bond0** 接続を再度アクティブにします。

```
# nmcli connection up bond0
```

3. ESP ハードウェアオフロードを使用する接続の **/etc/ipsec.d/** ディレクトリーの Libreswan 設定ファイルを編集し、**nic-offload=yes** ステートメントを接続エントリーに追加します。

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

4. **ipsec** サービスを再起動します。

```
# systemctl restart ipsec
```

検証

1. ボンディングのアクティブなポートを表示します。

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

2. アクティブなポートの **tx_ipsec** カウンターおよび **rx_ipsec** カウンターを表示します。

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

3. IPsec トンネルを介してトラフィックを送信します。たとえば、リモート IP アドレスに ping します。

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

4. アクティブなポートの **tx_ipsec** カウンターおよび **rx_ipsec** カウンターを再度表示します。

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

カウンターの値が増えると、ESP ハードウェアオフロードが動作します。

関連情報

- ネットワークボンディングの設定
- 『**ネットワークのセキュリティー保護**』ドキュメントの「IPsec を使用した VPNの設定」のセクション
- 『ネットワークのセキュリティー保護』ドキュメントの「IPsec を使用した VPNの設定」の章

4.13. システム全体の暗号化ポリシーをオプトアウトする IPSEC 接続の設定

接続のシステム全体の crypto-policies のオーバーライド

RHEL のシステム全体の暗号化ポリシーは、**%default** と呼ばれる特別な接続を作成します。この接続には、**ikev2** オプション、**esp** オプション、および **ike** オプションのデフォルト値が含まれます。ただし、接続設定ファイルに上記のオプションを指定すると、デフォルト値を上書きできます。

たとえば、次の構成では、AES および SHA-1 または SHA-2 で IKEv1 を使用し、AES-GCM または AES-CBC で IPsec (ESP) を使用する接続が許可されます。

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

AES-GCM は IPsec (ESP) および IKEv2 で利用できますが、IKEv1 では利用できません。

すべての接続でのシステム全体の暗号化ポリシーの無効化

すべての IPsec 接続のシステム全体の暗号化ポリシーを無効にするには、**/etc/ipsec.conf** ファイルで次の行をコメントアウトします。

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

次に、接続設定ファイルに **ikev2=never** オプションを追加してください。

関連情報

- Using system-wide cryptographic policies.

4.14. IPSEC VPN 設定のトラブルシューティング

IPsec VPN 設定に関連する問題は主に、一般的な理由が原因で発生する可能性が高くなっています。このような問題が発生した場合は、問題の原因が以下のシナリオのいずれかに該当するかを確認して、対応するソリューションを適用します。

基本的な接続のトラブルシューティング

VPN 接続関連の問題の多くは、管理者が不適当な設定オプションを指定してエンドポイントを設定した新しいデプロイメントで発生します。また、互換性のない値が新たに実装された場合に、機能していた設定が突然動作が停止する可能性があります。管理者が設定を変更した場合など、このような結果になることがあります。また、管理者が暗号化アルゴリズムなど、特定のオプションに異なるデフォルト値を使用して、ファームウェアまたはパッケージの更新をインストールした場合などです。

IPsec VPN 接続が確立されていることを確認するには、次のコマンドを実行します。

```
# 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
```

出力が空の場合や、エントリーで接続名が表示されない場合など、トンネルが破損します。

接続に問題があることを確認するには、以下を実行します。

1. **vpn.example.com** 接続をもう一度読み込みます。

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

2. 次に、VPN 接続を開始します。

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

ファイアウォール関連の問題

最も一般的な問題は、IPSec エンドポイントの 1 つ、またはエンドポイント間にあるルーターにあるファイアウォールで Internet Key Exchange (IKE) パケットがドロップされるという点が挙げられます。

- IKEv2 の場合には、以下の例のような出力は、ファイアウォールに問題があることを示しています。

```
# 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
...
```

- IKEv1 の場合は、最初のコマンドの出力は以下のようになります。

```
# 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
...
```

IPsec の設定に使用される IKE プロトコルは暗号化されているため、**tcpdump** ツールを使用して、トラブルシューティングできるサブセットは一部のみです。ファイアウォールが IKE パケットまたは IPsec パケットをドロップしている場合は、**tcpdump** ユーティリティーを使用して原因を見つけることができます。ただし、**tcpdump** は IPsec VPN 接続に関する他の問題を診断できません。

- **eth0** インターフェースで VPN および暗号化データすべてのネゴシエーションを取得するには、次のコマンドを実行します。

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

アルゴリズム、プロトコル、およびポリシーが一致しない場合

VPN 接続では、エンドポイントが IKE アルゴリズム、IPsec アルゴリズム、および IP アドレス範囲に一致する必要があります。不一致が発生した場合には接続は失敗します。以下の方法のいずれかを使用して不一致を特定した場合は、アルゴリズム、プロトコル、またはポリシーを調整して修正します。

- リモートエンドポイントが IKE/IPsec を実行していない場合は、そのパケットを示す ICMP パケットが表示されます。以下に例を示します。

```
# 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)]
...
```

- IKE アルゴリズムが一致しない例:

```
# 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
```

- IPsec アルゴリズムが一致しない例:

```
# 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
```

また、IKE バージョンが一致しないと、リモートエンドポイントが応答なしの状態でリクエストをドロップする可能性がありました。これは、すべての IKE パケットをドロップするファイアウォールと同じです。

- IKEv2 (Traffic Selectors - TS) の IP アドレス範囲が一致しない例:

```
# 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
```

- IKEv1 の IP アドレス範囲で一致しない例:

```
# 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
```

- IKEv1 で PreSharedKeys (PSK) を使用する場合には、どちらでも同じ PSK に配置されなければ、IKE メッセージ全体の読み込みができなくなります。

```
# 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
```

- IKEv2 では、 mismatched-PSK エラーが原因で AUTHENTICATION_FAILED メッセージが表示されます。

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

最大伝送単位 (MTU)

ファイアウォールが IKE または IPSec パケットをブロックする以外で、ネットワークの問題の原因として、暗号化パケットのパケットサイズの増加が最も一般的です。ネットワークハードウェアは、最大伝送単位 (MTU) を超えるパケットを 1500 バイトなどのサイズに断片化します。多くの場合、断片化されたパケットは失われ、パケットの再アセンブルに失敗します。これにより、小さいサイズのパケットを使用する ping テスト時には機能し、他のトラフィックでは失敗するなど、断続的な問題が発生します。このような場合に、SSH セッションを確立できますが、リモートホストに 'ls -al /usr' コマンドに入力した場合など、すぐにターミナルがフリーズします。

この問題を回避するには、トンネル設定ファイルに **mtu=1400** のオプションを追加して、MTU サイズを縮小します。

または、TCP 接続の場合は、MSS 値を変更する iptables ルールを有効にします。

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

各シナリオで上記のコマンドを使用して問題が解決されない場合は、**set-mss** パラメーターで直接サイズを指定します。

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

ネットワークアドレス変換 (NAT)

IPsec ホストが NAT ルーターとしても機能すると、誤ってパケットが再マッピングされる可能性があります。以下の設定例はこの問題について示しています。

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

アドレスが 172.16.0.1 のシステムには NAT ルールが 1 つあります。

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

アドレスが 10.0.2.33 のシステムがパケットを 192.168.0.1 に送信する場合に、ルーターは IPsec 暗号化を適用する前にソースを 10.0.2.33 から 172.16.0.1 に変換します。

次に、ソースアドレスが 10.0.2.33 のパケットは **conn myvpn** 設定と一致しなくなるので、IPsec ではこのパケットが暗号化されません。

この問題を解決するには、ルーターのターゲット IPsec サブネット範囲の NAT を除外するルールを挿入します。以下に例を示します。

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

カーネル IPsec サブシステムのバグ

たとえば、バグが原因で IKE ユーザー空間と IPsec カーネルの同期が解除される場合など、カーネル IPsec サブシステムに問題が発生する可能性があります。このような問題がないかを確認するには、以下を実行します。

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

上記のコマンドの出力でゼロ以外の値が表示されると、問題があることを示しています。この問題が発生した場合は、新しい サポートケース を作成し、1 つ前のコマンドの出力と対応する IKE ログを添付してください。

Libreswan のログ

デフォルトでは、Libreswan は **syslog** プロトコルを使用してログに記録します。**journalctl** コマンドを使用して、IPsec に関連するログエントリーを検索できます。ログへの対応するエントリーは **pluto** IKE デーモンにより送信されるため、以下のように、キーワード「pluto」を検索します。

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

ipsec サービスのライブログを表示するには、次のコマンドを実行します。

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

ロギングのデフォルトレベルで設定問題が解決しない場合は、**/etc/ipsec.conf** ファイルの **config setup** セクションに **plutodebug=all** オプションを追加してデバッグログを有効にします。

デバッグロギングは多くのエントリーを生成し、**journald** サービスまたは **syslogd** サービスレートのいずれかが **syslog** メッセージを制限する可能性があることに注意してください。完全なログを取得するには、ロギングをファイルにリダイレクトします。**/etc/ipsec.conf** を編集し、**config setup** セクションに **logfile=/var/log/pluto.log** を追加します。

関連情報

- ログファイルを使用した問題のトラブルシューティング
- **tcpdump(8)** および **ipsec.conf(5)** の man ページ
- firewalld の使用および設定

4.15. 関連情報

- **ipsec(8)**、**ipsec.conf(5)**、**ipsec.secrets(5)**、**ipsec_auto(8)**、および **ipsec_rsasigkey(8)** の man ページ
- **/usr/share/doc/libreswan-version/** ディレクトリー
- アップストリームプロジェクトの Web サイト
- The Libreswan プロジェクトの Wiki
- All Libreswan のすべての man ページ
- NIST Special Publication 800-77: Guide to IPsec VPNs

第5章 VPN RHEL システムロールを使用した IPSEC との VPN 接続の設定

VPNシステムロールを使用すると、Red Hat Ansible Automation Platformを使用してRHELシステムでVPN接続を構成できます。これを使用して、ホスト間、ネットワーク間、VPNリモートアクセスサーバー、およびメッシュ構成をセットアップできます。

ホスト間接続の場合、ロールは、必要に応じてキーを生成するなど、デフォルトのパラメーターを使用して、**vpn_connections**のリスト内のホストの各ペア間にVPNトンネルを設定します。または、リストされているすべてのホスト間に日和見メッシュ構成を作成するように構成することもできます。この役割は、**hosts**の下にあるホストの名前がAnsibleインベントリで使用されているホストの名前と同じであり、それらの名前を使用してトンネルを構成できることを前提としています。

注記

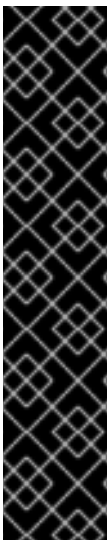
VPN RHEL システムロールは現在、VPN プロバイダーとして IPsec 実装である Libreswan のみをサポートしています。

5.1. VPNシステムロールを使用してIPSECでホスト間VPNの作成

VPNシステムロールを使用して、コントロールノードでAnsible Playbookを実行することにより、ホスト間接続を構成できます。これにより、インベントリファイルにリストされているすべての管理対象ノードが構成されます。

前提条件

- 1つ以上の **管理対象ノード** (VPN システムロールで設定するシステム) へのアクセスおよびパーミッション。
- **コントロールノード** (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
  コントロールノードでは、
  - **ansible-core** パッケージおよび **rhel-system-roles** パッケージがインストールされている。



重要

RHEL 8.0-8.5 では、別の Ansible リポジトリーへのアクセス権を指定されており、Ansible をベースにする自動化用の Ansible Engine 2.9 が含まれています。Ansible Engine には、**ansible**、**ansible-playbook** などのコマンドラインユーティリティー、**docker** や **podman** などのコネクター、プラグインとモジュールが多く含まれています。Ansible Engine を入手してインストールする方法については、ナレッジベースの How to download and install Red Hat Ansible Engine を参照してください。

RHEL 8.6 および 9.0 では、Ansible Core (**ansible-core** パッケージとして提供) が導入されました。これには、Ansible コマンドラインユーティリティー、コマンド、およびビルトイン Ansible プラグインのセットが含まれています。RHEL は、AppStream リポジトリーを介してこのパッケージを提供し、サポート範囲は限定的です。詳細については、ナレッジベースの Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories を参照してください。

- 管理対象ノードが記載されているインベントリーファイルがある。

手順

1. 以下の内容を含む新しい **playbook.yml** ファイルを作成します。

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

このPlaybookは、システムロールによって自動生成されたキーを使用した事前共有キー認証を使用して、**managed_node1からmanaged_node2**への接続を設定します。

2. オプション:ホストの **vpn_connections** リストに次のセクションを追加して、管理対象ホストから、インベントリファイルに記述されていない外部ホストへの接続を設定します。

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

これは、追加の接続（**managed_node1からexternal_node**）へと（**managed_node2からexternal_node**）を設定します。

注記

接続は管理対象ノードでのみ設定され、外部ノードでは設定されません。

1. オプション:**vpn_connections** 内の追加セクション (コントロールプレーンやデータプレーンなど) を使用して、管理対象ノードに複数の VPN 接続を指定できます。

```
- 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. オプション:設定に合わせて変数を変更できます。詳細は、**/usr/share/doc/rhel-system-roles/vpn/README.md**ファイルを参照してください。

3. オプション:Playbook の構文を確認します。

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

4. インベントリーファイルで Playbook を実行します。

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

検証

1. 管理対象ノードで、接続が正常にロードされていることを確認します。

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

connection.nameを、このノードからの接続の名前（たとえば、**managed_node1-to-managed_node2**）に置き換えます。

注記

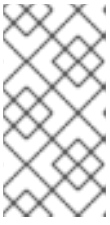
デフォルトでは、ロールは、各システムの観点から作成する接続ごとにわかりやすい名前を生成します。たとえば、**managed_node1**と**managed_node2**との間の接続を作成するときに、**managed_node1**上のこの接続のわかりやすい名前は**managed_node1-to-managed_node2**ですが、**managed_node2** では、この接続の名前は**managed_node2-to-managed_node1**となります。

1. 管理対象ノードで、接続が正常に開始されたことを確認します。

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

2. オプション:接続が正常に読み込まれなかった場合は、次のコマンドを入力して手動で接続を追加します。これにより、接続の確立に失敗した理由を示す、より具体的な情報が提供されます。

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



注記

接続の読み込みおよび開始のプロセス中に発生した可能性のあるエラーは、ログに報告されます。ログは、**/var/log/pluto.log**にあります。これらのログは解析が難しいため、代わりに接続を手動で追加して、標準出力からログメッセージを取得してみてください。

5.2. VPNシステムロールを使用してIPSECで日和見メッシュVPN接続の作成

VPNシステムロールを使用して、コントロールノードでAnsible Playbookを実行することにより、認証に証明書を使用する日和見メッシュVPN接続を構成できます。これにより、インベントリファイルにリストされているすべての管理対象ノードが構成されます。

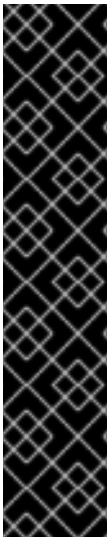
証明書による認証は、Playbookで**auth_method: cert**パラメーターを定義することによって設定されます。VPNシステムロールは、**/etc/ipsec.d**ディレクトリで定義されているIPsecネットワークセキュリティサービス（NSS）暗号ライブラリに必要な証明書が含まれていることを前提としています。デフォルトでは、ノード名が証明書のニックネームとして使用されます。この例では、これは**managed_node1**です。インベントリで**cert_name**属性を使用して、さまざまな証明書名を定義できます。

次の手順例では、Ansible Playbook を実行するシステムであるコントロールノードは、両方の管理対象ノード（192.0.2.0/24）と同じクラスレスドメイン間ルーティング（CIDR）番号を共有し、IPアドレスは192.0.2.7になります。したがって、コントロールノードは、CIDR 192.0.2.0/24用に自動的に作成されるプライベートポリシーに該当します。

再生中のSSH接続の損失を防ぐために、コントロールノードの明確なポリシーがポリシーのリストに含まれています。ポリシーリストには、CIDRがデフォルトと等しい項目もあることに注意してください。これは、このPlaybookがデフォルトポリシーのルールを上書きして、private-or-clearではなくprivateにするためです。

前提条件

- 1 つ以上の **管理対象ノード** (VPN システムロールで設定するシステム) へのアクセスおよびパーミッション。
  - すべての管理対象ノードで、**/etc/ipsec.d**ディレクトリのNSSデータベースには、ピア認証に必要なすべての証明書が含まれています。デフォルトでは、ノード名が証明書のニックネームとして使用されます。
- **コントロールノード** (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
  コントロールノードでは、
  - **ansible-core** パッケージおよび **rhel-system-roles** パッケージがインストールされている。



重要

RHEL 8.0-8.5 では、別の Ansible リポジトリーへのアクセス権を指定されており、Ansible をベースにする自動化用の Ansible Engine 2.9 が含まれています。Ansible Engine には、**ansible**、**ansible-playbook** などのコマンドラインユーティリティー、**docker** や **podman** などのコネクター、プラグインとモジュールが多く含まれています。Ansible Engine を入手してインストールする方法については、ナレッジベースの How to download and install Red Hat Ansible Engine を参照してください。

RHEL 8.6 および 9.0 では、Ansible Core (**ansible-core** パッケージとして提供) が導入されました。これには、Ansible コマンドラインユーティリティー、コマンド、およびビルトイン Ansible プラグインのセットが含まれています。RHEL は、AppStream リポジトリーを介してこのパッケージを提供し、サポート範囲は限定的です。詳細については、ナレッジベースの Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories を参照してください。

- 管理対象ノードが記載されているインベントリーファイルがある。

手順

1. 以下の内容を含む新しい **playbook.yml** ファイルを作成します。

```
- 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
```

2. オプション:設定に合わせて変数を変更できます。詳細は、**/usr/share/doc/rhel-system-roles/vpn/README.md**ファイルを参照してください。

3. オプション:Playbook の構文を確認します。

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

4. インベントリーファイルで Playbook を実行します。

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

5.3. 関連情報

- VPN システムロールで使用するパラメーターの詳細と、VPN システムロールに関する追加情報は、**/usr/share/doc/rhel-system-roles/vpn/README.md** ファイルを参照してください。
- **ansible-playbook** コマンドの詳細は、**ansible-playbook(1)** の man ページを参照してください。

第6章 ネットワークサービスのセキュリティー保護

Red Hat Enterprise Linux 9 は、さまざまな種類のネットワークサーバーをサポートしています。RHEL 9 ネットワークサービスを使用すると、システムのセキュリティーが DoS 攻撃 (Denial of Service)、DDoS 攻撃 (Distributed Denial of Service)、スクリプト脆弱性攻撃、バッファーオーバーフロー攻撃など、さまざまな種類の攻撃のリスクにさらされる可能性があります。

攻撃に対するシステムのセキュリティーを強化するには、使用しているアクティブなネットワークサービスを監視することが重要です。たとえば、ネットワークサービスがマシンで実行されている場合に、そのデーモンはネットワークポートでの接続をリッスンするのでセキュリティーが低下する可能性があります。ネットワークに対する攻撃に対する公開を制限するには、未使用のすべてのサービスをオフにする必要があります。

6.1. RPCBIND サービスのセキュリティー保護

rpcbind サービスは、Network Information Service (NIS)や Network File System (NFS)などの Remote Procedure Calls (RPC)サービス用の動的ポート割り当てデーモンです。その認証メカニズムは弱く、制御するサービスに幅広いポート範囲を割り当てる可能性があるため、**rpcbind** をセキュア化することが重要です。

すべてのネットワークへのアクセスを制限し、サーバーのファイアウォールルールを使用して特定の例外を定義することにより、**rpcbind** のセキュリティーを確保できます。

注記

- **NFSv3** サーバーでは、**rpcbind** サービスが必要です。
- **NFSv4** では、**rpcbind** サービスがネットワークをリッスンする必要がありません。

前提条件

- **rpcbind** パッケージがインストールされている。
- **Firewalld** パッケージがインストールされ、サービスが実行されている。

手順

1. 次に、ファイアウォールルールを追加します。
   - TCP 接続を制限し、**111** ポート経由の **192.168.0.0/24** ホストからのパッケージだけを受け入れます。

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

   - TCP 接続を制限し、**111** ポート経由のローカルホストからのパッケージだけを受け入れます。

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

   - UDP 接続を制限し、**111** ポート経由の **192.168.0.0/24** ホストからのパッケージだけを受け入れます。

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

ファイアウォール設定を永続化するには、ファイアウォールルールを追加するときに **--permanent** オプションを使用します。

2. ファイアウォールをリロードして、新しいルールを適用します。

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

検証手順

- ファイアウォールルールをリストします。

```
# 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
```

関連情報

- **NFSv4-only** サーバーの詳細は、Configuring an NFSv4-only server セクションを参照してください。
- firewalld の使用および設定

6.2. RPC.MOUNTD サービスのセキュリティー保護

rpc.mountd デーモンは、NFS マウントプロトコルのサーバー側を実装します。NFS マウントプロトコルは、NFS バージョン 3 (RFC 1813) で使用されます。

rpc.mountd サービスは、サーバーにファイアウォールルールを追加することでセキュリティー保護できます。すべてのネットワークへのアクセスを制限し、ファイアウォールルールを使用して特定の例外を定義できます。

前提条件

- **rpc.mountd** パッケージがインストールされている。
- **Firewalld** パッケージがインストールされ、サービスが実行されている。

手順

1. 以下のように、サーバーにファイアウォールルールを追加します。
   - **192.168.0.0/24**ホストからの**mountd**接続を許可します。

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

   - ローカルホストからの**mountd** 接続を受け入れます。

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

ファイアウォール設定を永続化するには、ファイアウォールルールを追加するときに **--permanent** オプションを使用します。

2. ファイアウォールをリロードして、新しいルールを適用します。

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

検証手順

- ファイアウォールルールをリストします。

```
# 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
```

関連情報

- firewalld の使用および設定

6.3. NFS サービスの保護

Kerberos を使用してすべてのファイルシステム操作を認証および暗号化して、ネットワークファイルシステムバージョン 4 (NFSv4) のセキュリティーを保護できます。ネットワークアドレス変換 (NAT) またはファイアウォールで NFSv4 を使用する場合に、**/etc/default/nfs** ファイルを変更することで委譲をオフにできます。委譲は、サーバーがファイルの管理をクライアントに委譲する手法です。

対照的に、NFSv3 ではファイルのロックとマウントに Kerberos は使用されません。

NFS サービスは、すべてのバージョンの NFS で TCP を使用してトラフィックを送信します。このサービスは、**RPCSEC_GSS** カーネルモジュールの一部として Kerberos ユーザーおよびグループ認証をサポートします。

NFS を利用すると、リモートのホストがネットワーク経由でファイルシステムをマウントし、そのファイルシステムを、ローカルにマウントしているファイルシステムのように操作できるようになります。集約サーバーのリソースを統合して、ファイルシステムを共有するときに **/etc/nfsmount.conf** ファイルの NFS マウントオプションをさらにカスタマイズできます。

6.3.1. NFS サーバーのセキュリティーを保護するエクスポートオプション

NFS サーバーは、**/etc/exports** ファイル内のどのファイルシステムにどのファイルシステムをエクスポートするかなど、ディレクトリーとホストのリスト構造を決定します。

警告

エクスポートファイルの構文に余分なスペースがあると、設定が大幅に変更される可能性があります。

以下の例では、**/tmp/nfs/** ディレクトリーは **bob.example.com** ホストと共有され、読み取りおよび書き込みのパーミッションを持ちます。

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

以下の例は上記と同じになりますが、同じディレクトリーを読み取り専用パーミッションで **bob.example.com** ホストに共有し、ホスト名の後の1つのスペース文字が原因で読み取りと書き込み権限で **すべてのユーザー** に共有します。

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

showmount -e <hostname> コマンドを入力すると、システム上の共有ディレクトリーを確認できます。

/etc/exports ファイルで次のエクスポートオプションを使用します。

警告

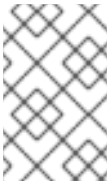
ファイルシステムのサブディレクトリーのエクスポートは安全ではないため、ファイルシステム全体をエクスポートします。攻撃者は、部分的にエクスポートされたファイルシステムで、エクスポートされていない部分にアクセスする可能性があります。

ro

ro オプションを使用して、NFS ボリュームを読み取り専用としてエクスポートします。

rw

rw オプションを使用して、NFS ボリュームに対する読み取りおよび書き込み要求の両方を許可します。書き込みアクセスが許可されると攻撃のリスクが高まるため、このオプションは注意して使用してください。



注記

シナリオとして **rw** オプションを使用してディレクトリーをマウントする必要がある場合は、起こりうるリスクを軽減するために、すべてのユーザーがディレクトリーを書き込み可能にしないようにしてください。

root_squash

root_squash オプションを使用して、**uid/gid** 0 からの要求を匿名の **uid/gid** にマッピングします。これは、**bin** ユーザーや **staff** グループなど、同様に機密である可能性の高い他の **uid** または **gid** には適用されません。

no_root_squash

root squashing をオフにするには、**no_root_squash** オプションを使用します。デフォルトでは、NFS 共有は **root** ユーザーを、非特権ユーザーである **nobody** ユーザーに変更します。これにより、**root**が作成したすべてのファイルの所有者が **nobody** に変更され、**setuid** ビットが設定されたプログラムのアップロードができなくなります。**no_root_squash** オプションを使用すると、リモートの root ユーザーは共有ファイルシステムの任意のファイルを変更し、他のユーザーに対してアプリケーションが Trojans に感染した状態のままにします。

secure

secure オプションを使用して、エクスポートを予約ポートに制限します。デフォルトでは、サーバーは予約済みポートからのクライアント通信のみを許可します。ただし、多くのネットワークで、クライアント上で **root** ユーザーになるのは簡単です。そのため、サーバーで予約されたポートからの通信が特権であると仮定することは安全ではありません。そのため、予約ポートの制限は効果が限定的です。Kerberos、ファイアウォール、および特定クライアントへのエクスポートを制限することに依存すると良いでしょう。

また、NFS サーバーをエクスポートする際に、以下のベストプラクティスを考慮してください。

- 一部のアプリケーションでは、パスワードをプレーンテキストまたは弱い暗号化形式で保存するため、ホームディレクトリーをエクスポートすることはリスクがあります。アプリケーションコードを確認して改善することで、リスクを軽減できます。
- 一部のユーザーは SSH キーにパスワードを設定していないため、この場合もホームディレクトリーによるリスクが発生します。パスワードの使用を強制するか、Kerberos を使用することで、これらのリスクを軽減できます。
- NFS エクスポートを必要なクライアントのみに制限します。NFS サーバーで **showmount -e** コマンドを使用して、サーバーのエクスポート内容を確認します。特に必要のないものはエクスポートしないでください。
- 攻撃のリスクを減らすために、不要なユーザーがサーバーにログインできないようにしてください。サーバーにアクセスできるユーザーを定期的に確認してください。

関連情報

- Secure NFS with Kerberos when using Red Hat Identity Management
- **exports(5)** および **nfs(5)** の man ページ

6.3.2. NFS クライアントのセキュリティーを保護するマウントオプション

mount コマンドに次のオプションを渡すと、NFS ベースのクライアントのセキュリティーを強化できます。

nosuid

nosuid オプションを使用して **set-user-identifier** または **set-group-identifier** ビットを無効にします。これにより、リモートユーザーが **setuid** プログラムを実行してより高い特権を取得するのを防ぎ、**setuid** オプションの反対となるこのオプションを使用できます。

noexec

noexec オプションを使用して、クライアント上の実行可能なファイルをすべて無効にします。これを使用して、ユーザーが共有ファイルシステムに配置されたファイルを誤って実行するのを防ぎます。

nodev

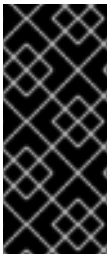
nodev オプションを使用して、クライアントがデバイスファイルをハードウェアデバイスとして処理するのを防ぎます。

resvport

resvport オプションを使用して、通信を予約済みポートに制限し、特権送信元ポートを使用してサーバーと通信できます。予約済みポートは、**root** ユーザーなどの特権ユーザーおよびプロセス用に予約されています。

秒

NFS サーバーの **sec** オプションを使用して、マウントポイント上のファイルにアクセスするための RPCGSS セキュリティーフレーバーを選択します。有効なセキュリティーフレーバーは、**none**、**sys**、**krb5**、**krb5i**、および **krb5p** です。



重要

krb5-libs パッケージが提供する MIT Kerberos ライブラリーは、新しいデプロイメントで Data Encryption Standard (DES) アルゴリズムに対応しなくなりました。DES は、セキュリティーと互換性の理由から、Kerberos ライブラリーでは非推奨であり、デフォルトで無効になっています。互換性の理由でご使用の環境で DES が必要な場合を除き、DES の代わりに新しくより安全なアルゴリズムを使用してください。

6.3.3. ファイアウォールでの NFS のセキュリティー保護

NFS サーバーでファイアウォールを保護するには、必要なポートのみを開いてください。他のサービスには NFS 接続ポート番号を使用しないでください。

前提条件

- **nfs-utils** パッケージがインストールされている。
- **Firewalld** パッケージがインストールされ、実行されている。

手順

- NFSv4 では、ファイアウォールは TCP ポート **2049** を開く必要があります。
- NFSv3 では、**2049** で 4 つのポートを追加で開きます。

1. **rpcbind** サービスは NFS ポートを動的に割り当て、ファイアウォールルールの作成時に問題が発生する可能性があります。このプロセスを簡素化するには、**/etc/nfs.conf** ファイルを使用して、使用するポートを指定します。

   a. **mountd** セクションの **mountd** (**rpc.mountd**) の TCP および UDP ポートを **port= <value>** 形式で設定します。

   b. **statd** セクションの **statd** (**rpc.statd**) の TCP および UDP ポートを **port= <value>** 形式で設定します。

2. **/etc/nfs.conf** ファイルで NFS ロックマネージャー (**nlockmgr**) の TCP および UDP ポートを設定します。

   a. **lockd** セクションの **nlockmgr** (**rpc.statd**) の TCP ポートを **port=value** 形式で設定します。または、**/etc/modprobe.d/lockd.conf** ファイルの **nlm_tcpport** オプションを使用することもできます。

b. **lockd** セクションの **nlockmgr** (**rpc.statd**) の UDP ポートを **udp-port=value** 形式で設定します。または、**/etc/modprobe.d/lockd.conf** ファイルの **nlm_udpport** オプションを使用することもできます。

検証手順

- NFS サーバー上のアクティブなポートと RPC プログラムを一覧表示します。

```
$ rpcinfo -p
```

関連情報

- Secure NFS with Kerberos when using Red Hat Identity Management
- **exports(5)** および **nfs(5)** の man ページ

6.4. FTP サービスのセキュリティー保護

ファイル転送プロトコル (FTP) を使用して、ネットワーク経由でファイルを転送できます。ユーザー認証を含むサーバーとの FTP トランザクションがすべて暗号化されているわけではないため、サーバーが安全に設定されていることを確認する必要があります。

RHEL 9 は、2 つの FTP サーバーを提供します。

- Red Hat Content Accelerator (tux): FTP 機能を持つカーネルスペースの Web サーバー。
- Very Secure FTP Daemon (vsftpd): スタンドアロンの、セキュリティー指向のFTP サービスの実装。

vsftpd FTP サービスをセットアップするためのセキュリティーガイドラインを以下に示します。

6.4.1. FTP グリーティングバナーのセキュリティー保護

ユーザーが FTP サービスに接続すると、FTP はグリーティングバナーを表示します。このバナーにはデフォルトで、バージョン情報が含まれており、攻撃者がシステムの弱点を特定するのに役立つ場合があります。デフォルトのバナーを変更することで、攻撃者がこの情報にアクセスできないようにします。

/etc/banners/ftp.msg ファイルを編集して、単一行のメッセージを直接含めるか、複数行のメッセージを含めることができる別のファイルを参照して、カスタムバナーを定義できます。

手順

- 1 行のメッセージを定義するには、次のオプションを **/etc/vsftpd/vsftpd.conf** ファイルに追加します。

```
ftpd_banner=Hello, all activity on ftp.example.com is logged.
```

- 別のファイルでメッセージを定義するには以下を実行します。
  - バナーメッセージを含む **.msg** ファイルを作成します。(例: **/etc/vendors/ftp .msg**)

```
######### Hello, all activity on ftp.example.com is logged. #########
```

複数のバナーの管理を簡素化するには、すべてのバナーを **/etc/vendors/** ディレクトリーに配置します。

- バナーファイルへのパスを **/etc/vsftpd/vsftpd.conf** ファイルの **banner_file** オプションに追加します。

```
banner_file=/etc/banners/ftp.msg
```

検証

- 変更されたバナーを表示します。

```
$ ftp localhost
Trying ::1…
Connected to localhost (::1).
Hello, all activity on ftp.example.com is logged.
```

6.4.2. FTP での匿名アクセスとアップロードの防止

デフォルトでは、**vsftpd** パッケージをインストールすると、**/var/ftp/** ディレクトリーと、ディレクトリーに対する読み取り専用権限を持つ匿名ユーザー用のディレクトリーツリーが作成されます。匿名ユーザーはデータにアクセスできるため、これらのディレクトリーに機密データを保存しないでください。

システムのセキュリティーを強化するために、匿名ユーザーが特定のディレクトリーにファイルをアップロードできるが、データは読み取れないように、FTP サーバーを設定できます。次の手順では、匿名ユーザーが **root** ユーザー所有のディレクトリーにファイルをアップロードできるが変更できないようにする必要があります。

手順

- **/var/ftp/pub/** ディレクトリーに書き込み専用ディレクトリーを作成します。

```
# 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
```

- **/etc/vsftpd/vsftpd.conf** ファイルに以下の行を追加します。

```
anon_upload_enable=YES
anonymous_enable=YES
```

- オプション:システムで SELinux が有効で Enforcing に設定されている場合には、SELinux ブール属性 **allow_ftpd_anon_write** および **allow_ftpd_full_access** を有効にします。

警告

匿名ユーザーによるディレクトリーの読み取りと書き込みを許可すると、盗まれたソフトウェアのリポジトリーになってしまう可能性があります。

6.4.3. FTP のユーザーアカウントのセキュリティー保護

FTP は、認証のために安全でないネットワークを介して暗号化されていないユーザー名とパスワードを送信します。システムユーザーが自分のユーザーアカウントからサーバーにアクセスできないようにして、FTP のセキュリティーを向上させることができます。

以下の手順のうち、お使いの設定に該当するものをできるだけ多く実行してください。

手順

- **/etc/vsftpd/vsftpd.conf** ファイルに次の行を追加して、**vsftpd** サーバーのすべてのユーザーアカウントを無効にします。

```
local_enable=NO
```

- **/etc/pam.d/vsftpd** PAM 設定ファイルにユーザー名を追加して、特定のアカウントまたは特定のアカウントグループ (**root** ユーザーや **sudo** 権限を持つユーザーなど) の FTP アクセスを無効にします。
- **/etc/vsftpd/ftpusers** ファイルにユーザー名を追加して、ユーザーアカウントを無効にします。

6.4.4. 関連情報

- **ftpd_selinux(8)** のman ページ

6.5. HTTP サーバーのセキュリティー保護

6.5.1. httpd.conf のセキュリティー強化

/etc/httpd/conf/httpd.conf ファイルでセキュリティーオプションを設定して、Apache HTTP のセキュリティーを強化できます。

システムで実行されているすべてのスクリプトが正しく機能することを常に確認してから、本番環境に移行してください。

root ユーザーのみが、スクリプトまたは Common Gateway Interface (CGI) を含むディレクトリーへの書き込み権限を持っていることを確認してください。ディレクトリーの所有権を書き込み権限を持つ **root** ユーザーに変更するには、次のコマンドを入力します。

```
# chown root directory-name
# chmod 755 directory-name
```

/etc/httpd/conf/httpd.conf ファイルで、次のオプションを設定できます。

FollowSymLinks

このディレクティブはデフォルトで有効になっており、ディレクトリー内のシンボリックリンクをたどります。

Indexes

このディレクティブはデフォルトで有効になっています。訪問者がサーバー上のファイルを閲覧できないようにするには、このディレクティブを削除してください。

UserDir

このディレクティブは、システム上にユーザーアカウントが存在することを確認できるため、デフォルトでは無効になっています。**/root/** 以外のすべてのユーザーディレクトリーのユーザーディレクトリーブラウジングをアクティブにするには、**UserDir enabled** と **UserDir disabled** の root ディレクティブを使用します。無効化されたアカウントのリストにユーザーを追加するには、**UserDir disabled** 行にスペースで区切られたユーザーのリストを追加します。

ServerTokens

このディレクティブは、クライアントに送り返されるサーバー応答ヘッダーフィールドを制御します。以下のパラメーターを使用するとログの出力をカスタマイズできます。

ServerTokens Full

以下のように、Web サーバーのバージョン番号、サーバーのオペレーティングシステムの詳細、インストールされている Apache モジュールなど、利用可能なすべての情報を提供します。

```
Apache/2.4.37 (Red Hat Enterprise Linux) MyMod/1.2
```

ServerTokens Full-Release

以下のように、利用可能なすべての情報をリリースバージョンとともに提供します。

```
Apache/2.4.37 (Red Hat Enterprise Linux) (Release 41.module+el8.5.0+11772+c8e0c271)
```

ServerTokens Prod / ServerTokens ProductOnly

以下のように、Web サーバー名を提供します。

```
Apache
```

ServerTokens Major

以下のように、Web サーバーのメジャーリリースバージョンを提供します。

```
Apache/2
```

ServerTokens Minor

以下のように、Web サーバーのマイナーリリースバージョンを提供します。

```
Apache/2.4
```

ServerTokens Min / ServerTokens Minimal

以下のように、Web サーバーの最小リリースバージョンを提供します。

```
Apache/2.4.37
```

ServerTokens OS

以下のように、Web サーバーのリリースバージョンとオペレーティングシステムを提供します。

```
Apache/2.4.37 (Red Hat Enterprise Linux)
```

ServerTokens Prod オプションを使用して、攻撃者がシステムに関する貴重な情報を入手するリスクを軽減します。

重要

IncludesNoExec ディレクティブを削除しないでください。デフォルトでは、Server-Side Includes (SSI) モジュールは、コマンドを実行できません。これを変更すると、攻撃者がシステムにコマンドを入力できるようになる可能性があります。

httpd モジュールの削除

httpd モジュールを削除して、HTTP サーバーの機能を制限できます。これを行うには、**/etc/httpd/conf.modules.d/** または **/etc/httpd/conf.d/** ディレクトリーの設定ファイルを編集します。たとえば、プロキシーモジュールを削除するためには、以下のコマンドを実行します。

```
echo '# All proxy modules disabled' > /etc/httpd/conf.modules.d/00-proxy.conf
```

関連情報

- Apache HTTP サーバー
- Apache HTTP サーバーの SELinux ポリシーのカスタマイズ

6.5.2. Nginx サーバー設定のセキュリティー保護

Nginx は、高性能の HTTP およびプロキシーサーバーです。次の設定オプションを使用して、Nginx 設定を強化できます。

手順

- バージョン文字列を無効にするには、**server_tokens** 設定オプションを変更します。

  ```
  server_tokens off;
  ```

  このオプションは、サーバーのバージョン番号などの追加の情報表示を停止します。以下のようにこの設定では、Nginx によって処理されるすべての要求のサーバー名のみが表示されます。

  ```
  $ curl -sI http://localhost | grep Server
  Server: nginx
  ```

- 特定の **/etc/nginx/** conf ファイルに、特定の既知の Web アプリケーションの脆弱性を軽減するセキュリティーヘッダーを追加します。
  - たとえば、**X-Frame-Options** ヘッダーオプションは、Nginx が提供するコンテンツのフレーム化がされないように、ドメイン外のページを拒否して、クリックジャッキング攻撃を軽減します。

```
add_header X-Frame-Options "SAMEORIGIN";
```

- たとえば、**x-content-type** ヘッダーは、特定の古いブラウザーでの MIME タイプのスニッフィングを防ぎます。

```
add_header X-Content-Type-Options nosniff;
```

- また、**X-XSS-Protection** ヘッダーは、クロスサイトスクリプティング (XSS) フィルタリングを有効にし、Nginx での応答に含まれる可能性がある、悪意のあるコンテンツをブラウザーがレンダリングしないようにします。

```
add_header X-XSS-Protection "1; mode=block";
```

- たとえば、一般に公開されるサービスを制限し、訪問者からのサービスと受け入れを制限できます。

```
limit_except GET {
    allow 192.168.1.0/32;
    deny  all;
}
```

スニペットは、**GET** と **HEAD** を除くすべてのメソッドへのアクセスを制限します。

- 以下のように、HTTP メソッドを無効にできます。

```
# Allow GET, PUT, POST; return "405 Method Not Allowed" for all others.
if ( $request_method !~ ^(GET|PUT|POST)$ ) {
    return 405;
}
```

- Nginx Web サーバーによって提供されるデータを保護するように SSL を設定できます。これは、HTTPS 経由でのみ提供することを検討してください。さらに、Mozilla SSL Configuration Generator を使用して、Nginx サーバーで SSL を有効にするための安全な設定プロファイルを生成できます。生成された設定により、既知の脆弱なプロトコル (SSLv2 や SSLv3 など)、暗号、ハッシュアルゴリズム (3DES や MD5 など) が確実に無効化されます。また、SSL サーバーテストを使用して、設定した内容が最新のセキュリティー要件を満たしていることを確認できます。

関連情報

- Mozilla SSL Configuration Generator
- SSL Server Test

6.6. 認証されたローカルユーザーへのアクセスを制限することによる POSTGRESQL のセキュリティー保護

PostgreSQLは、オブジェクトリレーショナルデータベース管理システム（DBMS）です。Red Hat Enterprise Linux では、PostgreSQL は **postgresql-server** パッケージで提供されます。

クライアント認証を設定して、攻撃のリスクを減らすことができます。データベースクラスターのデータディレクトリーに保存されている **pg_hba.conf** 設定ファイルは、クライアント認証を制御します。手順に従って、ホストベースの認証用に PostgreSQL を設定します。

手順

1. PostgreSQL をインストールします。

```
# yum install postgresql-server
```

2. 次のいずれかのオプションを使用して、データベースストレージ領域を初期化します。

   a. **initdb** ユーティリティーの使用:

```
$ initdb -D /home/postgresql/db1/
```

   -D オプションを指定した **initdb** コマンドを実行すると、指定したディレクトリーがまだ存在しない場合は作成します (例: **/home/postgresql/db1/**)。このディレクトリーには、データベースに保存されているすべてのデータと、クライアント認証設定ファイルが含まれています。

   b. **postgresql-setup** スクリプトの使用:

```
$ postgresql-setup --initdb
```

   デフォルトでは、スクリプトは **/var/lib/pgsql/data/** ディレクトリーを使用します。このスクリプトは、基本的なデータベースクラスター管理でシステム管理者を支援します。

3. 認証されたローカルユーザーが自分のユーザー名でデータベースにアクセスできるようにするには、**pg_hba.conf** ファイルの以下の行を変更します。

```
local   all             all                                     trust
```

   これは、データベースユーザーを作成し、ローカルユーザーを作成しないレイヤー型アプリケーションを使用する場合に、問題となることがあります。システム上のすべてのユーザー名を明示的に制御しない場合は、**pg_hba.conf** ファイルから **local** の行を削除してください。

4. データベースを再起動して、変更を適用します。

```
# systemctl restart postgresql
```

   前のコマンドはデータベースを更新し、設定ファイルの構文も検証します。

6.7. MEMCACHED サービスのセキュリティー保護

Memcached は、オープンソースの高性能分散メモリーオブジェクトキャッシングシステムです。データベースの負荷を軽減して、動的 Web アプリケーションのパフォーマンスを向上させることができます。

Memcached は、データベース呼び出し、API 呼び出し、またはページレンダリングの結果から、文字列やオブジェクトなどの任意のデータの小さなチャンクを格納するメモリー内のキーと値のストアです。Memcached を使用すると、十分に活用されていない領域から、より多くのメモリーを必要とするアプリケーションにメモリーを割り当てることができます。

2018 年に、パブリックインターネットに公開されている Memcached サーバーを悪用することによる DDoS 増幅攻撃の脆弱性が発見されました。これらの攻撃は、トランスポートに UDP プロトコルを使用する Memcached 通信を利用します。この攻撃は増幅率が高いため、効果的です。数百バイトのサイズの要求は、数メガバイトまたは数百メガバイトのサイズの応答を生成することができます。

ほとんどの場合、**memcached** サービスはパブリックインターネットに公開する必要はありません。このような弱点には、リモートの攻撃者が memcached に保存されている情報を漏洩または変更できるなど、独自のセキュリティー問題があります。

このセクションに従って、DDoS 攻撃の可能性に対して Memcached サービスを使用してシステムを強化します。

6.7.1. DDoS に対する Memcached の強化

セキュリティーリスクを軽減するために、以下の手順のうち、お使いの設定に該当するものをできるだけ多く実行してください。

手順

- LAN にファイアウォールを設定してください。Memcached サーバーにローカルネットワークだけでアクセスできるようにする必要がある場合は、**memcached** サービスで使用されるポートに外部トラフィックをルーティングしないでください。たとえば、許可されたポートのリストからデフォルトのポート **11211** を削除します。

```
# firewall-cmd --remove-port=11211/udp
# firewall-cmd --runtime-to-permanent
```

- アプリケーションと同じマシンで単一の memcached サーバーを使用する場合、ローカルホストトラフィックのみをリッスンするように **memcached** を設定します。**/etc/sysconfig/memcached** ファイルの **OPTIONS** 値を変更します。

```
OPTIONS="-l 127.0.0.1,::1"
```

- Simple Authentication and Security Layer (SASL) 認証を有効にします。

  1. **/etc/sasl2/memcached.conf** ファイルで、以下のように修正または追加します。

```
sasldb_path: /path.to/memcached.sasldb
```

  2. SASL データベースにアカウントを追加します。

```
# saslpasswd2 -a memcached -c cacheuser -f /path.to/memcached.sasldb
```

  3. **memcached** のユーザーとグループがデータベースにアクセスできることを確認します。

```
# chown memcached:memcached /path.to/memcached.sasldb
```

  4. **/etc/sysconfig/memcached** ファイルの **OPTIONS** パラメーターに **-S** 値を追加して、Memcached で SASL サポートを有効にします。

```
OPTIONS="-S"
```

  5. Memcached サーバーを再起動して、変更を適用します。

```
# systemctl restart memcached
```

  6. SASL データベースで作成したユーザー名とパスワードを、お使いのアプリケーションの Memcached クライアント設定に追加します。

- memcached クライアントとサーバー間の通信を TLS で暗号化します。

  1. **/etc/sysconfig/memcached** ファイルの **OPTIONS** パラメーターに **-Z** 値を追加して、TLS を使用した Memcached クライアントとサーバー間の暗号化通信を有効にします。

     ```
     OPTIONS="-Z"
     ```

  2. **-o ssl_chain_cert** オプションを使用して、証明書チェーンファイルパスを PEM 形式で追加します。

  3. **-o ssl_key** オプションを使用して、秘密鍵ファイルのパスを追加します。

第7章 MACSEC を使用した同じ物理ネットワーク内のレイヤー 2 トラフィックの暗号化

MACsec を使用して、2 つのデバイス間の通信を (ポイントツーポイントで) 保護できます。たとえば、ブランチオフィスがメトロイーサネット接続を介してセントラルオフィスに接続されている場合に、オフィスをつなぐ 2 つのホストで MACsec を設定して、セキュリティーを強化できます。

Media Access Control Security (MACsec) は、イーサーネットリンクで異なるトラフィックタイプを保護するレイヤー 2 プロトコルです。これには以下が含まれます。

- DHCP (Dynamic Host Configuration Protocol)
- アドレス解決プロトコル (ARP)
- インターネットプロトコルのバージョン 4 / 6 (**IPv4** / **IPv6**)
- TCP や UDP などの IP 経由のトラフィック

MACsec は、LAN 内のすべてのトラフィックを、デフォルトで GCM-AES-128 アルゴリズムで暗号化および認証し、事前共有キーを使用して参加者ホスト間の接続を確立します。事前共有キーを変更する場合は、MACsec を使用するネットワーク内のすべてのホストで NM 設定を更新する必要があります。

MACsec 接続は、親としてイーサーネットネットワークカード、VLAN、トンネルデバイスなどのイーサーネットデバイスを使用します。暗号化した接続のみを使用して他のホストと通信するように、MACsec デバイスでのみ IP 設定を設定するか、親デバイスに IP 設定を設定することもできます。後者の場合、親デバイスを使用して、暗号化されていない接続と暗号化された接続用のMACsecデバイスを使用して他のホストと通信できます。

MACsec には特別なハードウェアは必要ありません。たとえば、ホストとスイッチの間のトラフィックのみを暗号化する場合を除き、任意のスイッチを使用できます。このシナリオでは、スイッチが MACsec もサポートする必要があります。

つまり、MACsec を設定する方法は 2 つあります。

- ホスト対ホスト
- 他のホストに切り替えるホスト

重要

MACsec は、同じ (物理または仮想) LAN のホスト間でのみ使用することができます。

7.1. NMCLI を使用した MACSEC 接続の設定

nmcli ツールを使用して、MACsec を使用するようにイーサーネットインターフェースを設定できます。この手順では、イーサネット経由で接続されている 2 つのホスト間に MACsec 接続を作成する方法について説明します。

手順

1. MACsec を設定する最初のホストで以下を実行します。
   - 事前共有キー用の接続関連キー (CAK: connectivity association key) および接続関連キー名 (CKN: connectivity-association key name) を作成します。

a. 16 バイトの 16 進数の CAK を作成します。

```
# dd if=/dev/urandom count=16 bs=1 2> /dev/null | hexdump -e '1/2 "%04x"'
50b71a8ef0bd5751ea76de6d6c98c03a
```

b. 32 バイトの 16 進数の CKN を作成します。

```
# dd if=/dev/urandom count=32 bs=1 2> /dev/null | hexdump -e '1/2 "%04x"'
f2b4297d39da7330910a74abc0449feb45b5c0b9fc23df1430e1898fcf1c4550
```

2. 両方のホストで、MACsec 接続を介して接続します。

3. MACsec 接続を作成します。

```
# 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
```

前の手順で生成された CAK および CKN を **macsec.mka-cak** および **macsec.mka-ckn** パラメーターで使用します。この値は、MACsec で保護されるネットワーク内のすべてのホストで同じである必要があります。

4. MACsec 接続で IP を設定します。

a. **IPv4** を設定します。たとえば、静的 **IPv4** アドレス、ネットワークマスク、デフォルトゲートウェイ、および DNS サーバーを **macsec0** 接続に設定するには、以下のコマンドを実行します。

```
# 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. **IPv6** を設定します。たとえば、静的 **IPv6** アドレス、ネットワークマスク、デフォルトゲートウェイ、および DNS サーバーを **macsec0** 接続に設定するには、以下のコマンドを実行します。

```
# 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. 接続をアクティベートします。

```
# nmcli connection up macsec0
```

検証手順

1. トラフィックが暗号化されていることを確認します。

```
# tcpdump -nn -i enp1s0
```

2. オプション:暗号化されていないトラフィックを表示します。

```
# tcpdump -nn -i macsec0
```

3. MACsec の統計を表示します。

```
# ip macsec show
```

4. integrity-only (encrypt off) および encryption (encrypt on) の各タイプの保護に対して個々のカウンターを表示します。

```
# ip -s macsec show
```

7.2. 関連情報

- MACsec: a different solution to encrypt network traffic ブログ