Red Hat Enterprise Linux 8

SELinux の使用

SELinux (Security-Enhanced Linux) の基本設定および高度な設定

Last Updated: 2020-09-04

Red Hat Enterprise Linux 8 SELinux の使用

SELinux (Security-Enhanced Linux) の基本設定および高度な設定

法律上の通知

概要

本書は、ユーザーおよび管理者向けに、SELinux の基礎および基本を紹介し、さまざまなサービスを設定および構成する実用的な作業を説明します。

目次

RED HAT ドキュメントへのフィードバック (英語のみ)

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。改善点を報告する場合は、以下のように行います。

- 特定の文章に簡単なコメントを記入する場合は、以下の手順を行います。
    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端に **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。
- より詳細なフィードバックを行う場合は、Bugzilla のチケットを作成します。
    1. Bugzilla の Web サイトにアクセスします。
    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 SELINUX の使用

1.1. SELINUX の概要

Security Enhanced Linux (SELinux) は、新たにシステムセキュリティーの層を提供します。SELinux は、基本的に **May <subject> do <action> to <object>?**の形式の問い (たとえば「**May a web server access files in users' home directories?** (Web サーバーは、ユーザーのホームディレクトリーのファイルにアクセスできますか?)」) に答えていきます。

ユーザー、グループ、およびその他のアクセス権に基づいた標準のアクセスポリシーは Discretionary Access Control (DAC) として知られており、システム管理者が、包括的で詳細なセキュリティーポリシー (たとえば、特定のアプリケーションではログファイルの表示だけを許可し、その他のアプリケーションではログファイルに新しいデータを追加するのを許可するなど) を作成することはできません。

SELinux は、Mandatory Access Control (MAC) を実装しています。それぞれのプロセスおよびシステムリソースには、**SELinux コンテキスト** と呼ばれる特別なセキュリティーラベルがあります。SELinux コンテキストは **SELinux ラベル** として参照されることがありますが、システムレベルの詳細を抽象化し、エンティティーのセキュリティープロパティーに焦点を当てた識別子です。これにより、SELinux ポリシーでオブジェクトを参照する方法に一貫性を持たせ、他の識別方法に含まれる曖昧さがなくなりました。たとえば、バインドマウントを使用するシステムで、ファイルに、有効なパス名を複数設定できます。

SELinux ポリシーは、プロセスが、互いに、またはさまざまなシステムリソースと相互作用する方法を定義する一連のルールにこのコンテキストを使用します。デフォルトでは、最初にルールが明示的にアクセスを許可し、その後ポリシーが任意の対話を許可します。

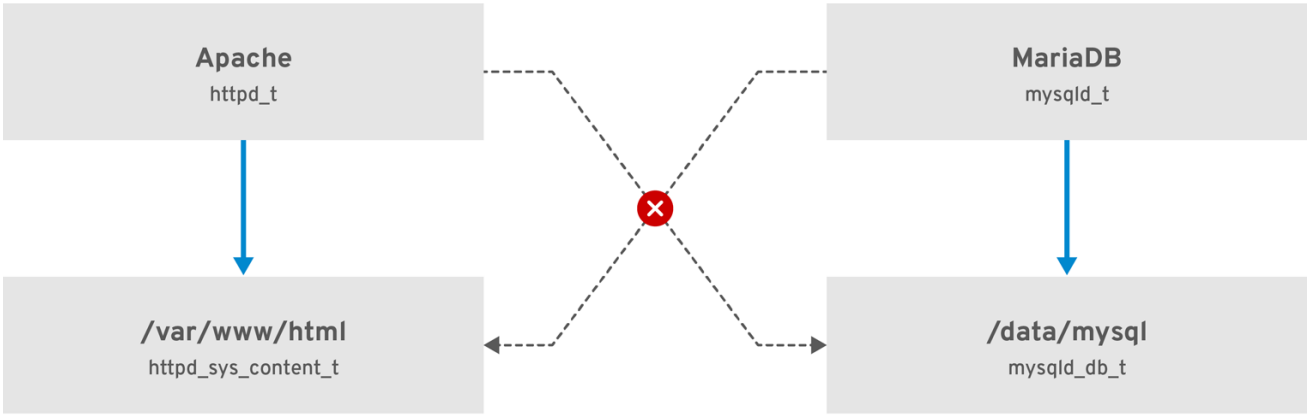
注記

SELinux ポリシールールが DAC ルールの後に確認されている点に注意してください。DAC ルールがアクセスを拒否すると、SELinux ポリシールールは使用されません。これは、従来の DAC ルールがそのアクセスを拒否すると、SELinux 拒否がログに記録されないということを示しています。

SELinux コンテキストには、複数のフィールド (ユーザー、ロール、タイプ、セキュリティーレベル) があります。プロセスとシステムリソースとの間で許可される相互作用を定義する最も一般的なポリシールールは、完全な SELinux コンテキストではなく、SELinux タイプを使用するため、SELinux ポリシーでは、SELinux のタイプ情報がおそらく最も重要です。SELinux のタイプの名前は、最後に **_t** が付きます。たとえば、Web サーバーのタイプ名は **httpd_t** です。**/var/www/html/** にあるファイルおよびディレクトリーのタイプコンテキストは、通常 **httpd_sys_content_t** です。**/tmp** および **/var/tmp/** にあるファイルおよびディレクトリーに対するタイプコンテクストは、通常 **tmp_t** です。Web サーバーポートのタイプコンテキストは **http_port_t** です。

Apache (**httpd_t** として実行する Web サーバープロセス) を許可するポリシールールがあります。このルールでは、通常 **/var/www/html/** にあるコンテキストを持つファイルおよびディレクトリーと、その他の Web サーバーディレクトリー (**httpd_sys_content_t**) へのアクセスを許可します。通常、**/tmp** および **/var/tmp/** に含まれるファイルのポリシーには、許可ルールがないため、アクセスは許可されません。SELinux を使用すれば、Apache が危険にさらされ、悪意のあるスクリプトがアクセスを得た場合でも、**/tmp** ディレクトリーにアクセスすることはできなくなります。

図1.1 Apache と MariaDB を安全に実行する SELinux の例



このスキーマが示すように、SELinux は、**httpd_t** として実行している Apache プロセスが **/var/www/html/** ディレクトリーにアクセスするのは許可しますが、同じ Apache プロセスが **/data/mysql/** ディレクトリーにアクセスするのは拒否します。これは、**httpd_t** タイプコンテキストと **mysqld_db_t** タイプコンテキストに許可ルールがないのが原因です。一方、**mysqld_t** として実行する MariaDB プロセスは **/data/mysql/** ディレクトリーにアクセスできますが、SELinux により、**mysqld_t** タイプを持つプロセスが、**httpd_sys_content_t** とラベルが付いた **/var/www/html/** ディレクトリーにアクセスするのは拒否されます。

関連情報

詳細は、以下のドキュメントを参照してください。

- **apropos selinux** コマンドで表示される man ページの **selinux(8)**
- **selinux-policy-doc** パッケージをインストールしている場合は、**man -k _selinux** コマンドで表示された man ページ。
- The SELinux Coloring Book、SELinux の基本概念をよりよく理解するのに役立ちます。
- SELinux Wiki FAQ

1.2. SELINUX を実行する利点

SELinux は、次のような利点を提供します。

- プロセスとファイルにはすべてラベルが付いています。SELinux ポリシーにより、プロセスがファイルと相互作用する方法と、プロセスが互いに相互作用する方法が定義されます。アクセスは、それを特別に許可する SELinux ポリシールールが存在する場合に限り許可されます。
- アクセス制御がより詳細に設定できるようになりました。SELinux のアクセスは、ユーザーの裁量と、Linux のユーザー ID およびグループ ID に基づいて制御される従来の UNIX アクセス権だけでなく、SELinux のユーザー、ロール、タイプなど (必要に応じてセキュリティーレベルも) の、入手可能なすべての情報に基づいて決定されます。
- SELinux ポリシーは管理者が定義し、システム全体に適用されます。
- 権限昇格攻撃に対する軽減策が向上しました。プロセスはドメインで実行するため、互いに分離しています。SELinux ポリシールールは、プロセスがどのようにファイルやその他のプロセスにアクセスするかを定義します。プロセスへのアクセスが不正に行われても、攻撃者は、そのプロセスの通常の機能と、そのプロセスがアクセスするように設定されているファイルにし

かアクセスできません。たとえば、Apache HTTP Server へのアクセスが不正に行われても、そのアクセスを許可する特別な SELinux ポリシールールが追加されたり、設定された場合を除き、ユーザーのホームディレクトリーにあるファイルを読み込むプロセスを攻撃者が利用することはできません。

- SELinux は、データの機密性と完全性、並びに信頼されていない入力からの保護プロセスを強化するのに使用できます。

ただし、SELinux は以下の機能とは異なります。

- ウイルス対策ソフトウェア
- パスワード、ファイアウォールなどのセキュリティーシステムの代替
- 一体型のセキュリティーソリューション

SELinux は、既存のセキュリティーソリューションを強化するために作られており、代わりに使用されるものではありません。SELinux を実行している場合でも、ソフトウェアを最新の状態にする、推測が困難なパスワードを使用する、ファイアウォールを使用するなど、優れたセキュリティー対策を続けることが重要です。

1.3. SELINUX の例

以下の例は、SELinux がどのようにセキュリティーを向上するかを説明します。

- デフォルトのアクションは「拒否」です。アクセスを許可する SELinux のポリシールール (ファイルを開くプロセスなど) が存在しない場合は、アクセスが拒否されます。
- SELinux は、Linux ユーザーに制限をかけられます。SELinux ポリシーには、制限がかけられた SELinux ユーザーが多数含まれます。Linux ユーザーを、制限がかけられた SELinux ユーザーにマッピングして、SELinux ユーザーに適用されているセキュリティールールおよびメカニズムを利用できます。たとえば Linux ユーザーを、SELinux の **user_u** にマッピングすると、その Linux ユーザーは、(許可が設定されていない限り) **sudo** や **su** などの setuid (set user ID) アプリケーションを実行したり、そのユーザーのホームディレクトリーにある、害を及ぼす可能性があるファイルやアプリケーションを実行したりできなくなります。
- プロセスとデータの分離が向上します。SELinux **ドメイン** の概念により、特定のファイルやディレクトリーにアクセスできるプロセスの定義が可能になります。たとえば、SELinux を実行している場合に、(許可が設定されていない限り) 攻撃者は Samba サーバーを危険にさらすことはできず、その Samba サーバーを攻撃ベクトルとして使用して、その他のプロセス (MariaDB など) が使用するファイルの読み書きを行うことはできません。
- SELinux は、設定ミスによるダメージを軽減します。Domain Name System (DNS) サーバーはゾーン転送として知られている機能で、互いに頻繁に情報を複製します。攻撃者は、ゾーン転送を使用して、虚偽の情報で DNS サーバーを更新できます。Red Hat Enterprise Linux で BIND (Berkeley Internet Name Domain) を DNS サーバーとして実行すると、ゾーン転送を実行できるサーバーを管理者が制限した場合でも、デフォルトの SELinux ポリシーによりゾーンファイルが阻止されます。[1] BIND **named** デーモン自体、およびその他のプロセスにより、ゾーン転送を使用して更新済みです。

1.4. SELINUX のアーキテクチャーおよびパッケージ

SELinux は、Linux カーネルに組み込まれる Linux セキュリティーモジュール (LSM) です。カーネルの SELinux サブシステムは、管理者が制御し、システムの起動時に読み込まれるセキュリティーポリシーにより動作します。システムにおけるセキュリティー関連の、カーネルレベルのアクセス操作はすべて

SELinux により傍受され、読み込んだセキュリティーポリシーのコンテキストに従って検討されます。読み込んだポリシーが操作を許可すると、その操作は継続します。許可しないと、その操作はブロックされ、プロセスがエラーを受け取ります。

アクセスの許可、拒否などの SELinux の結果はキャッシュされます。このキャッシュは、アクセスベクトルキャッシュ (AVC) として知られています。このように結果がキャッシュされると、確認が必要な量が減るため、SELinux ポリシーのパフォーマンスが向上します。DAC ルールがアクセスを拒否した場合は、SELinux ポリシールールが適用されないことに注意してください。未加工の監査メッセージのログは、行頭に **type=AVC** 文字列が追加されて、**/var/log/audit/audit.log** に記録されます。

Red Hat Enterprise Linux 8 では、システムサービスは **systemd** デーモンにより制御されます。すべてのサービスの開始および停止は **systemd** が行い、ユーザーとプロセスは、**systemctl** を介して **systemd** とやりとりします。**systemd** デーモンは、SELinux ポリシーを調べ、呼び出しているプロセスのラベルと、呼び出し元が管理するユニットファイルのラベルを確認してから、呼び出し元のアクセスを許可するかどうかを SELinux に確認します。このアプローチにより、システムサービスの開始や停止などの、重要なシステム機能へのアクセス制御が強化されます。

また、**systemd** デーモンは SELinux Access Manager としても起動します。 **systemctl** を実行しているプロセス、または **systemd** に **D-Bus** メッセージを送信したプロセスのラベルを取得します。次に、デーモンは、プロセスが設定するユニットファイルのラベルを探します。最後に、SELinux ポリシーでプロセスラベルとユニットファイルラベルとの間で特定のアクセスが許可されている場合、**systemd** はカーネルから情報を取得できます。これは、特定のサービスに対して、**systemd** と相互作用を必要とする、危険にさらされたアプリケーションをSELinux が制限できることを意味します。ポリシー作成者は、このような粒度の細かい制御を使用して、管理者を制限することもできます。

重要

SELinux ラベリングが誤っているために問題が発生するのを回避するには、**systemctl start** コマンドを使用してサービスを開始するようにしてください。

Red Hat Enterprise Linux 8 では、以下のような SELinux を操作するパッケージが提供されます。

- ポリシー - **selinux-policy-targeted**、**selinux-policy-mls**
- ツール - **policycoreutils**、**policycoreutils-gui**、**libselinux-utils**、**policycoreutils-python-utils**、**setools-console**, **checkpolicy**

1.5. SELINUX のステータスおよびモード

SELinux は、3 つのモード (Enforcing、Permissive、または Disabled) のいずれかで実行できます。

- Enforcing モードは、デフォルトのモードで、推奨される動作モードです。SELinux は、Enforcing モードでは正常に動作し、読み込んだセキュリティーポリシーをシステム全体に強制します。
- Permissive モードでは、システムは、読み込んだセキュリティーポリシーを SELinux が強制しているように振る舞い、オブジェクトのラベリングや、アクセスを拒否したエントリーをログに出力しますが、実際に操作を拒否しているわけではありません。Permissive モードは、実稼働システムで使用することは推奨されませんが、SELinux ポリシーの開発やデバッグには役に立ちます。
- Disabled モードを使用することは推奨されません。システムは、SELinux ポリシーの強制を回避するだけでなく、ファイルなどの任意の永続オブジェクトにラベルを付けなくなり、将来的に SELinux を有効にすることが難しくなります。

Enforcing モードと Permissive モードとの間を切り替えるには **setenforce** ユーティリティーを使用してください。**setenforce** で行った変更は、システムを再起動すると元に戻ります。Enforcing モードに変更するには、Linux の root ユーザーで、**setenforce 1** コマンドを実行します。Permissive モードに変更するには、**setenforce 0** コマンドを実行します。**getenforce** ユーティリティーを使用して、現在の SELinux モードを表示します。

```
# getenforce
Enforcing
```

```
# setenforce 0
# getenforce
Permissive
```

```
# setenforce 1
# getenforce
Enforcing
```

Red Hat Enterprise Linux では、システムを Enforcing モードで実行している場合に、個々のドメインを Permissive モードに設定できます。たとえば、**httpd_t** ドメインを Permissive に設定するには、以下のコマンドを実行します。

```
# semanage permissive -a httpd_t
```

Permissive ドメインは、システムのセキュリティーを侵害できる強力なツールです。Red Hat は、特定のシナリオのデバッグ時など、Permissive ドメインを使用する場合は注意することを推奨します。

---

[1] DNS サーバーで使用される IP アドレスマッピングへのホスト名などの情報が含まれるテキストファイル。

第2章 SELINUX のステータスおよびモードの変更

SELinux が有効になっている場合は、Enforcing モードまたは Permissive モードのいずれかで実行できます。以下のセクションでは、これらのモードに永続的に変更する方法を説明します。

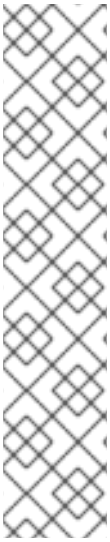
2.1. SELINUX のステータスおよびモードの永続的変更

SELinux のステータスおよびモード で説明されているように、SELinux は有効または無効にできます。有効にした場合の SELinux のモードには、Enforcing および Permissive の 2 つがあります。

getenforce コマンド、または **sestatus** コマンドを使用して、SELinux が実行しているモードを確認できます。**getenforce** コマンドは、**Enforcing**、**Permissive**、または **Disabled** を返します。

sestatus コマンドは SELinux のステータスと、使用されている SELinux ポリシーを返します。

```
$ sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Memory protection checking:     actual (secure)
Max kernel policy version:      31
```



注記

Permissive モードで SELinux を実行すると、ユーザーやプロセスにより、さまざまなファイルシステムオブジェクトのラベルが間違って設定される可能性があります。SELinux が無効になっている間に作成されたファイルシステムのオブジェクトには、ラベルが追加されません。ただし、SELinux では、ファイルシステムオブジェクトのラベルが正しいことが必要になるため、これにより Enforcing モードに変更したときに問題が発生します。

ラベルが誤って設定されていたり、ファイルにラベルが付いていないために問題が発生するのを防ぐために、Disabled 状態から Permissive モードまたは Enforcing モードに変更すると、ファイルシステムのラベルが自動的に再設定されます。Permissive モードでは、root で **fixfiles -F onboot** コマンドを使用して、**-F** オプションを含む **/.autorelabel** ファイルを作成し、次回のシステムの再起動時にファイルに再ラベル付けされるようにします。

2.2. PERMISSIVE モードへの変更

以下の手順を使用して、SELinux モードを永続的に Permissive に変更します。SELinux を Permissive モードで実行していると、SELinux ポリシーは強制されません。システムは動作し続け、SELinux がオペレーションを拒否せず AVC メッセージをログに記録できるため、このログを使用して、トラブルシューティングやデバッグ、ならびに SELinux ポリシーの改善に使用できます。この場合、各 AVC は一度だけログに記録されます。

前提条件

- **selinux-policy-targeted** パッケージ、**libselinux-utils** パッケージ、および **policycoreutils** パッケージがインストールされている。
- **selinux=0** または **enforcing=0** カーネルパラメーターは使用されません。

手順

1. 任意のテキストエディターで **/etc/selinux/config** ファイルを開きます。以下に例を示します。

```
# vi /etc/selinux/config
```

2. **SELINUX=permissive** オプションを設定します。

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

3. システムを再起動します。

```
# reboot
```

検証手順

1. システムの再起動後に、**getenforce** コマンドが **Permissive** を返すことを確認します。

```
$ getenforce
Permissive
```

2.3. ENFORCING モードへの変更

以下の手順に従って、SELinux を Enforcing モードに切り替えます。SELinux を Enforcing モードで実行している場合は、SELinux ポリシーが強制され、SELinux ポリシールールに基づいてアクセスが拒否されます。RHEL では、システムに SELinux を最初にインストールした時に、Enforcing モードがデフォルトで有効になります。

前提条件

- **selinux-policy-targeted** パッケージ、**libselinux-utils** パッケージ、および **policycoreutils** パッケージがインストールされている。
- **selinux=0** または **enforcing=0** カーネルパラメーターは使用されません。

手順

1. 任意のテキストエディターで **/etc/selinux/config** ファイルを開きます。以下に例を示します。

```
# vi /etc/selinux/config
```

2. **SELINUX=enforcing** オプションを設定します。

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

3. 変更を保存して、システムを再起動します。

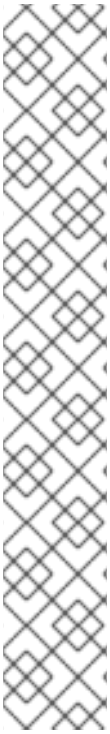
```
# reboot
```

次にシステムを起動する際に、SELinux はシステム内のファイルおよびディレクトリーのラベルを再設定し、SELinux が無効になっている間に作成したファイルおよびディレクトリーに SELinux コンテキストを追加します。

検証手順

1. システムの再起動後に、**getenforce** コマンドが **Enforcing** を返すことを確認します。

```
$ getenforce
Enforcing
```



注記

Enforcing モードに変更したあと、SELinux ポリシールールが間違っていたか、設定されていなかったため、SELinux が一部のアクションを拒否する場合があります。SELinux に拒否されるアクションを表示するには、root で以下のコマンドを実行します。

```
# ausearch -m AVC,USER_AVC,SELINUX_ERR,USER_SELINUX_ERR -ts today
```

setroubleshoot-server パッケージがインストールされている場合は、次のコマンドも使用できます。

```
# grep "SELinux is preventing" /var/log/messages
```

SELinux が有効で、Audit デーモン (**auditd**) がシステムで実行していない場合は、**dmesg** コマンドの出力で SELinux メッセージを検索します。

```
# dmesg | grep -i -e type=1300 -e type=1400
```

詳細は「Troubleshooting problems related to SELinux」 を参照してください。

2.4. 以前無効にしたシステムで SELINUX を有効にする

SELinux が無効になっていたシステムで SELinux を有効にする際に、システムが起動できない、プロセスが失敗するなどの問題を回避するには、この手順に従ってください。

手順

1. SELinux を Permissive モードで有効にします。詳細は「Permissive モードへの変更」を参照してください。
2. システムを再起動します。

```
# reboot
```

3. SELinux 拒否メッセージを確認します。詳細は「SELinux 拒否の特定」を参照してください。
4. 拒否がない場合は、Enforcing モードに切り替えます。詳細は「システムの起動時に SELinux モードの変更」を参照してください。

検証手順

1. システムの再起動後に、**getenforce** コマンドが **Enforcing** を返すことを確認します。

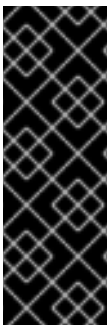
```
$ getenforce
Enforcing
```

関連情報

- Enforcing モードで SELinux を使用してカスタムアプリケーションを実行するには、次のいずれかのシナリオを選択してください。
  - **unconfined_service_t** ドメインでアプリケーションを実行します。
  - アプリケーションに新しいポリシーを記述します。詳細は、ナレッジベースの記事「Writing Custom SELinux Policy」を参照してください。
- モードの一時的な変更は、「SELinux のステータスおよびモード」で説明されています。

2.5. SELINUX の無効化

以下の手順に従って、SELinux を永続的に無効にします。



重要

SELinux が無効になっていると、SELinux ポリシーは読み込まれません。ポリシーは強制されず、AVC メッセージはログに記録されません。したがって、SELinux を実行する利点 はすべて失われます。

Red Hat は、SELinux を永続的に無効にする代わりに、Permissive モードを使用することを強く推奨します。Permissive モードの詳細は「Permissive モードへの変更」を参照してください。

警告

/etc/selinux/config で **SELINUX=disabled** オプションを使用して SELinux を無効にすると、カーネルが SELinux を有効にして起動し、その後のブートプロセスで無効化モードに切り替わります。メモリーリークおよび競合状態によりカーネルパニックが発生する可能性があるため、SELinux を完全に無効にする必要がある場合に 「システムの起動時に SELinux モードの変更」 で説明されているように、**selinux=0** パラメーターをカーネルコマンドラインに追加して SELinux を無効にすることが推奨されます。

手順

1. 任意のテキストエディターで **/etc/selinux/config** ファイルを開きます。以下に例を示します。

```
# vi /etc/selinux/config
```

2. **SELINUX=disabled** オプションを設定します。

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

3. 変更を保存して、システムを再起動します。

```
# reboot
```

検証手順

1. 再起動したら、**getenforce** コマンドが **Disabled** を返すことを確認します。

```
$ getenforce
Disabled
```

2.6. システムの起動時に SELINUX モードの変更

システムの起動時に、SELinux の実行方法を変更するカーネルパラメーターを設定できます。

enforcing=0

このパラメータを設定すると、システムを起動する際に、Permissive モードで起動します。これは、問題のトラブルシューティングを行うときに便利です。ファイルシステムの破損がひどい場合は、Permissive モードを使用することが、問題を検出するための唯一の選択肢となるかもしれませ

ん。また、Permissive モードでは、ラベルの作成が適切に行われます。このモードで作成した AVC メッセージは、Enforcing モードと同じになるとは限りません。

Permissive モードでは、一連の同じ拒否の最初の拒否のみが報告されます。一方、Enforcing モードでは、ディレクトリーの読み込みに関する拒否が発生し、アプリケーションが停止する場合がします。Permissive モードでは、表示される AVC メッセージは同じですが、アプリケーションは、ディレクトリー内のファイルを読み続け、拒否が発生するたびに AVC を取得します。

selinux=0

このパラメーターにより、カーネルは、SELinux インフラストラクチャーのどの部分も読み込まないようになります。init スクリプトは、システムが **selinux=0** パラメーターで起動してるのを認識し、**/.autorelabel** ファイルのタイムスタンプを変更します。これにより、次回 SELinux を有効にしてシステムを起動する際にシステムのラベルが自動的に再設定されます。

重要

Red Hat では、**selinux=0** パラメーターを使用することは推奨されません。システムをデバッグする場合は、Permissive モードを使用することが推奨されます。

autorelabel=1

このパラメーターにより、システムで、以下のコマンドと同様の再ラベルが強制的に行われます。

```
# touch /.autorelabel
# reboot
```

ファイルシステムに間違ったラベルが付いたオブジェクトが大量に含まれる場合は、システムを Permissive モードで起動して自動再ラベルプロセスを正常に実行します。

関連情報

- **checkreqprot** などの追加の SELinux 関連のカーネル起動パラメーターは、 **kernel-doc** パッケージと一緒にインストールされる **/usr/share/doc/kernel-doc-<KERNEL_VER>/Documentation/admin-guide/kernel-parameters.txt** ファイルを参照してください。**<KERNEL_VER>** 文字列をインストール済みカーネルのバージョン番号に置き換えます。以下に例を示します。

```
# yum install kernel-doc
$ less /usr/share/doc/kernel-doc-4.18.0/Documentation/admin-guide/kernel-parameters.txt
```

第3章 制限のあるユーザーおよび制限のないユーザーの管理

以下のセクションでは、Linux ユーザーの SELinux ユーザーへのマッピング、基本的な制限のあるユーザードメイン、新しいユーザーの SELinux ユーザーへのマッピングを説明します。

3.1. 制限のあるユーザーおよび制限のないユーザー

各 Linux ユーザーは、SELinux ポリシーを使用して SELinux ユーザーにマッピングされます。これにより、Linux ユーザーは SELinux ユーザーの制限を継承できます。

システムで SELinux ユーザーマッピングを表示するには、root で **semanage login -l** コマンドを使用します。

```
# semanage login -l
Login Name           SELinux User         MLS/MCS Range        Service

__default__          unconfined_u         s0-s0:c0.c1023       *
root                 unconfined_u         s0-s0:c0.c1023       *
```

Red Hat Enterprise Linux では、Linux ユーザーはデフォルトで SELinux の **default** ログインにマッピングされ、SELinux **unconfined_u** ユーザーにマッピングされます。以下の行は、デフォルトのマッピングを定義します。

```
__default__          unconfined_u         s0-s0:c0.c1023       *
```

制限のある Linux ユーザー、および制限のない Linux ユーザーは、実行可能なメモリーチェックおよび書き込み可能なメモリーチェックに依存し、また MCS または MLS によっても制限されます。

利用可能な SELinux ユーザーを一覧表示するには、以下のコマンドを実行します。

```
$ seinfo -u
Users: 8
   guest_u
   root
   staff_u
   sysadm_u
   system_u
   unconfined_u
   user_u
   xguest_u
```

seinfo コマンドは、**setools-console** パッケージにより提供されることに注意してください。デフォルトではインストールされません。

unconfined_t ドメインから自身の制限のあるドメインに移行できるものとして SELinux ポリシーが定義するアプリケーションを実行すると、制限のない Linux ユーザーは制限のあるドメインの制限を受けます。このセキュリティー上の利点は、Linux ユーザーが制限なしで実行している場合でも、アプリケーションは制限されたままになります。したがって、アプリケーションにおける不具合の悪用はポリシーによって制限できます。

同様に、これらのチェックを制限のあるユーザーに適用できます。制限のある各ユーザーは、制限のあるユーザードメインによって制限されます。SELinux ポリシーは、制限のあるユーザードメインから自身のターゲット制限のあるドメインへの移行を定義することもできます。このような場合、制限のある

ユーザーはそのターゲットの制限のあるドメインの制限を受けます。主な点として、特別な権限は、ロールに応じて制限のあるユーザーに関連付けられる点が挙げられます。

3.2. SELINUX ユーザー機能

以下の表は、Red Hat Enterprise Linux における Linux ユーザーの基本的な制限のあるドメインの例です。

表3.1 SELinux ユーザー機能

ユーザー	ロール	ドメイン	X Window System	su または sudo	ホームディレクトリーおよび /tmp (デフォルト) での実行	ネットワーク
sysadm_u	sysadm_r	sysadm_t	はい	su および sudo	はい	はい
staff_u	staff_r	staff_t	はい	sudo のみ	はい	はい
user_u	user_r	user_t	はい	いいえ	はい	はい
guest_u	guest_r	guest_t	いいえ	いいえ	はい	いいえ
xguest_u	xguest_r	xguest_t	はい	いいえ	はい	Firefox のみ

- **user_t** ドメイン、**guest_t** ドメイン、および **xguest_t** ドメイン内の Linux ユーザーは、SELinux ポリシーが許可した場合 (**passwd** など)、設定したユーザー ID (setuid) アプリケーションのみを実行できます。これらのユーザーは、**su** および **sudo** setuid アプリケーションを実行できないため、これらのアプリケーションを使用して root にすることはできません。

- **sysadm_t** ドメイン、**staff_t** ドメイン、**user_t** ドメイン、および **xguest_t** ドメインの Linux ユーザーは、X Window System およびターミナルを使用してログインできます。

- デフォルトでは、**staff_t** ドメイン、**user_t** ドメイン、**guest_t** ドメイン、および **xguest_t** ドメインの Linux ユーザーは、ホームディレクトリーと **/tmp** でアプリケーションを実行できます。
  ユーザーのパーミッションを継承するアプリケーションの実行を防ぐには、**guest_exec_content** および **xguest_exec_content** ブール値を **off** に設定します。これにより、不具合のあるアプリケーションや悪意のあるアプリケーションがユーザーのファイルを変更できなくなります。

- **xguest_t** ドメインの Linux ユーザーにアクセスできる唯一のネットワークアクセスは、Web ページに接続する Firefox です。

- **sysadm_u** ユーザーは、SSH を使用して直接ログインできません。**sysadm_u** の SSH ログインを有効にするには、**ssh_sysadm_login** ブール値を **on** に設定します。

```
# setsebool -P ssh_sysadm_login on
```

system_u は、システムプロセスおよびオブジェクトに対する特別なユーザー ID であることに注意し

てください。Linux ユーザーに関連付けることはできません。また、**unconfined_u** および **root** は制限のないユーザーです。このような理由により、SELinux ユーザー機能の以前の表には含まれていません。

上記の SELinux ユーザーとともに、**semanage user** コマンドを使用して、これらのユーザーにマップできる特別なロールがあります。これらのロールは、SELinux でユーザーに許可するものを決定します。

- **webadm_r** は、Apache HTTP Server に関連する SELinux タイプの処理のみが可能です。
- **dm_r** は、MariaDB データベースおよび PostgreSQL データベース管理システムに関連する SELinux タイプの処理のみが可能です。
- **logadm_r** は、**syslog** および **auditlog** プロセスに関連する SELinux タイプの処理のみが可能です。
- **secadm_r** は SELinux の処理のみが可能です。
- **auditadm_r** は、Audit サブシステムに関連するプロセスのみを管理できます。

利用可能なロールの一覧を表示するには、**seinfo -r** コマンドを実行します。

```
$ seinfo -r
Roles: 14
   auditadm_r
   dbadm_r
   guest_r
   logadm_r
   nx_server_r
   object_r
   secadm_r
   staff_r
   sysadm_r
   system_r
   unconfined_r
   user_r
   webadm_r
   xguest_r
```

seinfo コマンドは、**setools-console** パッケージにより提供されることに注意してください。デフォルトではインストールされません。

関連情報

- 詳細は、man ページの **seinfo(1)**、**semanage-login(8)**、および **xguest_selinux(8)** を参照してください。

3.3. SELINUX の UNCONFINED_U ユーザーに自動的にマッピングされた新規ユーザーの追加

以下の手順では、新しい Linux ユーザーをシステムに追加する方法を説明します。ユーザーは自動的に SELinux **unconfined_u** ユーザーにマッピングされます。

前提条件

- **root** ユーザーは、Red Hat Enterprise Linux でデフォルトで実行するため、制限なしで実行されています。

手順

1. 以下のコマンドを実行して **example.user** という名前の新規 Linux ユーザーを作成します。

```
# useradd example.user
```

2. Linux の **example.user** ユーザーにパスワードを割り当てるには、次のコマンドを実行します。

```
# passwd example.user
Changing password for user example.user.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

3. 現在のセッションからログアウトします。

4. Linux の **example.user** ユーザーとしてログインします。ログインすると、PAM モジュール **pam_selinux** は Linux ユーザーを SELinux ユーザー (この場合は **unconfined_u**) に自動的にマッピングし、作成された SELinux コンテキストを設定します。Linux ユーザーのシェルはこのコンテキストで起動します。

検証手順

1. **example.user** ユーザーとしてログインしたら、Linux ユーザーのコンテキストを確認します。

```
$ id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

関連情報

- 詳細は、man ページの **pam_selinux(8)** を参照してください。

3.4. 新規ユーザーを SELINUX で制限されたユーザーとして追加

新規 SELinux が限定したユーザーをシステムに追加するには、以下の手順に従います。この手順では、ユーザーアカウントを作成するコマンドを使用して、ユーザーを SELinux の **staff_u** ユーザー権限にマッピングします。

前提条件

- **root** ユーザーは、Red Hat Enterprise Linux でデフォルトで実行するため、制限なしで実行されています。

手順

1. 以下のコマンドを実行して **example.user** という名前の新規 Linux ユーザーを作成し、SELinux **staff_u** ユーザーにマップします。

```
# useradd -Z staff_u example.user
```

2. Linux の **example.user** ユーザーにパスワードを割り当てるには、次のコマンドを実行します。

```
# passwd example.user
Changing password for user example.user.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

3. 現在のセッションからログアウトします。

4. Linux の **example.user** ユーザーとしてログインします。ユーザーのシェルは **staff_u** コンテキストで起動します。

検証手順

1. **example.user** ユーザーとしてログインしたら、Linux ユーザーのコンテキストを確認します。

```
$ id -Z
uid=1000(example.user) gid=1000(example.user) groups=1000(example.user)
context=staff_u:staff_r:staff_t:s0-s0:c0.c1023
```

関連情報

- 詳細は、man ページの **pam_selinux(8)** を参照してください。

3.5. SELINUX ユーザーを制限するためのシステムの設定

デフォルトでは、管理者権限を持つユーザーを含め、Red Hat Enterprise Linux のすべての Linux ユーザーは、制限のない SELinux ユーザー **unconfined_u** にマッピングされます。SELinux の制限のあるユーザーにユーザーを割り当てることで、システムのセキュリティーを強化できます。これは、「V-71971 Security Technical Implementation Guide」に準拠するのに役立ちます。制限のあるユーザーおよび制限のないユーザーの詳細は、「制限のあるユーザーおよび制限のないユーザーの管理」を参照してください。

3.5.1. 一般ユーザーの設定

SELinux ユーザー **user_u** にマッピングすることで、システムのすべての一般ユーザーを制限できます。

手順

1. SELinux ログインレコードの一覧を表示します。この一覧には、SELinux ユーザーへの Linux ユーザーのマッピングが表示されます。

```
# semanage login -l

Login Name    SELinux User  MLS/MCS Range   Service

__default__   unconfined_u  s0-s0:c0.c1023      *
root          unconfined_u  s0-s0:c0.c1023      *
```

2. 明示的なマッピングのないすべてのユーザーを表す __default__ ユーザーを、SELinux ユーザー **user_u** にマッピングします。

```
# semanage login -m -s user_u -r s0 __default__
```

検証手順

1. _default__ ユーザーが、SELinux ユーザー **user_u** にマッピングされていることを確認します。

```
# semanage login -l

Login Name    SELinux User   MLS/MCS Range    Service

__default__   user_u         s0               *
root          unconfined_u   s0-s0:c0.c1023   *
```

2. 新規ユーザーのプロセスが SELinux コンテキスト **user_u:user_r:user_t:s0** で実行されることを確認します。

   a. 新しいユーザーを作成します。

   ```
   # adduser example.user
   ```

   b. **example.user** のパスワードを定義します。

   ```
   # passwd example.user
   ```

   c. **root** としてログアウトし、新規ユーザーとしてログインします。

   d. ユーザーの ID のセキュリティーコンテキストを表示します。

   ```
   [example.user@localhost ~]$ id -Z
   user_u:user_r:user_t:s0
   ```

   e. ユーザーの現在のプロセスのセキュリティーコンテキストを表示します。

   ```
   [example.user@localhost ~]$ ps axZ
   LABEL                             PID TTY      STAT   TIME COMMAND
   -                                   1 ?        Ss     0:05 /usr/lib/systemd/systemd --switched-root --
   system --deserialize 18
   -                                3729 ?        S      0:00 (sd-pam)
   user_u:user_r:user_t:s0          3907 ?        Ss     0:00 /usr/lib/systemd/systemd --user
   -                                3911 ?        S      0:00 (sd-pam)
   user_u:user_r:user_t:s0          3918 ?        S      0:00 sshd: example.user@pts/0
   user_u:user_r:user_t:s0          3922 pts/0    Ss     0:00 -bash
   user_u:user_r:user_dbusd_t:s0  3969 ?        Ssl    0:00 /usr/bin/dbus-daemon --session --
   address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
   user_u:user_r:user_t:s0          3971 pts/0    R+     0:00 ps axZ
   ```

3.5.2. 管理者ユーザーの設定

以下の方法のいずれかを使用して、管理者ユーザーを制限できます。

3.5.2.1. sysadm_u へのマッピングによる管理者の制約

SELinux ユーザー **sysadm_u** に直接マッピングすることで、管理者権限を持つユーザーを制限できます。ユーザーがログインすると、セッションは SELinux コンテキスト **sysadm_u:sysadm_r:sysadm_t** で実行されます。

前提条件

- **root** ユーザーは制限なしで実行します。これは、Red Hat Enterprise Linux のデフォルトです。

手順

1. 必要に応じて、**sysadm_u** ユーザーが SSH を使用してシステムに接続できるようにするには、次のコマンドを実行します。

```
# setsebool -P ssh_sysadm_login on
```

2. 新しいユーザーを作成し、そのユーザーを **wheel** ユーザーグループに追加し、そのユーザーを SELinux ユーザー **sysadm_u** にマッピングします。

```
# adduser -G wheel -Z sysadm_u example.user
```

3. 必要に応じて、既存のユーザーを SELinux ユーザー **sysadm_u** にマッピングし、そのユーザーを **wheel** ユーザーグループに追加します。

```
# usermod -G wheel -Z sysadm_u example.user
```

検証手順

1. **example.user** が、SELinux ユーザー **sysadm_u** にマッピングされていることを確認します。

```
# semanage login -l | grep example.user
example.user     sysadm_u    s0-s0:c0.c1023   *
```

2. SSH などを使用して **example.user** としてログインし、ユーザーのセキュリティーコンテキストを表示します。

```
[example.user@localhost ~]$ id -Z
sysadm_u:sysadm_r:sysadm_t:s0-s0:c0.c1023
```

3. **root** ユーザーに切り替えます。

```
$ sudo -i
[sudo] password for example.user:
```

4. セキュリティーコンテキストが変更されていないことを確認します。

```
# id -Z
sysadm_u:sysadm_r:sysadm_t:s0-s0:c0.c1023
```

5. **sshd** サービスを再起動するなど、管理タスクを実行します。

```
# systemctl restart sshd
```

出力がない場合は、コマンドが正常に完了します。

コマンドが正常に完了しない場合は、以下のメッセージが表示されます。

```
Failed to restart sshd.service: Access denied
See system logs and 'systemctl status sshd.service' for details.
```

3.5.2.2. sudo および sysadm_r ロールを使用した管理者の制約

管理者権限を持つ特定のユーザーを SELinux ユーザー **staff_u** にマッピングし、ユーザーが SELinux 管理者ロール **sysadm_r** を取得できるように **sudo** を設定できます。このロールにより、SELinux 拒否なしで管理タスクを実行できます。ユーザーがログインすると、セッションは、SELinux コンテキスト **staff_u:staff_r:staff_t** で実行されますが、**sudo** を使用してコマンドを入力すると、セッションは **staff_u:sysadm_r:sysadm_t** コンテキストに変わります。

前提条件

- **root** ユーザーは制限なしで実行します。これは、Red Hat Enterprise Linux のデフォルトです。

手順

1. 新規ユーザーを作成し、そのユーザーを **wheel** ユーザーグループに追加し、そのユーザーを SELinux ユーザー **staff_u** にマッピングします。

```
# adduser -G wheel -Z staff_u example.user
```

2. 必要に応じて、既存のユーザーを SELinux ユーザー **staff_u** にマッピングし、そのユーザーを **wheel** ユーザーグループに追加します。

```
# usermod -G wheel -Z staff_u example.user
```

3. **example.user** が SELinux 管理者ロールを取得できるようにするには、**/etc/sudoers.d/** ディレクトリーに新規ファイルを作成します。以下に例を示します。

```
# visudo -f /etc/sudoers.d/example.user
```

4. 以下の行を新規ファイルに追加します。

```
example.user  ALL=(ALL) TYPE=sysadm_t ROLE=sysadm_r ALL
```

検証手順

1. **example.user** が SELinux ユーザー **staff_u** にマッピングされていることを確認します。

```
# semanage login -l | grep example.user
example.user     staff_u    s0-s0:c0.c1023   *
```

2. SSH などを使用して **example.user** としてログインし、**root** ユーザーに切り替えます。

```
[example.user@localhost ~]$ sudo -i
[sudo] password for example.user:
```

3. **root** セキュリティーコンテキストを表示します。

```
# id -Z
staff_u:sysadm_r:sysadm_t:s0-s0:c0.c1023
```

4. **sshd** サービスを再起動するなど、管理タスクを実行します。

```
# systemctl restart sshd
```

出力がない場合は、コマンドが正常に完了します。

コマンドが正常に完了しない場合は、以下のメッセージが表示されます。

```
Failed to restart sshd.service: Access denied
See system logs and 'systemctl status sshd.service' for details.
```

3.5.3. 関連情報

- 追加のオプションは、ナレッジベースアーティクル「「How to set up a system with SELinux confined users」を参照してください。
- 詳細は、man ページの **user_selinux(8)**、**staff_selinux(8)**、および **sysadm_selinux(8)** を参照してください。

3.6. 関連情報

- 詳細は、man ページの **unconfined_selinux(8)** を参照してください。

第4章 非標準設定でアプリケーションとサービスの SELINUX の設定

SELinux が Enforcing モードの場合、デフォルトのポリシーはターゲットポリシーになります。以下のセクションでは、ポート、データベースの場所、プロセスのファイルシステムパーミッションなどの設定のデフォルトを変更した後に、さまざまなサービスに対して SELinux ポリシーを設定および構成する方法を説明します。

以下の手順では、非標準ポート用に SELinux タイプを変更し、デフォルトディレクトリーの変更に関する間違ったラベルを特定して修正し、SELinux ブール値を使用してポリシーを調整する方法を説明します。

4.1. 非標準設定での APACHE HTTP サーバーの SELINUX ポリシーのカスタマイズ

Apache HTTP サーバーを設定して、別のポートでリッスンし、デフォルト以外のディレクトリーにコンテンツを提供できます。SELinux の拒否を防止するには、以下の手順に従い、システムの SELinux ポリシーを調整します。

前提条件

- **httpd** パッケージがインストールされ、Apache HTTP サーバーが TCP ポート 3131 をリッスンし、デフォルトの **/var/www/** ディレクトリーの代わりに **/var/test_www/** ディレクトリーを使用するように設定されています。
- **policycoreutils-python-utils** パッケージおよび **setroubleshoot-server** パッケージがシステムにインストールされている。

手順

1. **httpd** サービスを起動して、ステータスを確認します。

```
# systemctl start httpd
# systemctl status httpd
...
httpd[14523]: (13)Permission denied: AH00072: make_sock: could not bind to address
[::]:3131
...
systemd[1]: Failed to start The Apache HTTP Server.
...
```

2. SELinux ポリシーは、**httpd** がポート 80 で実行していることを前提としています。

```
# semanage port -l | grep http
http_cache_port_t              tcp      8080, 8118, 8123, 10001-10010
http_cache_port_t              udp      3130
http_port_t                    tcp      80, 81, 443, 488, 8008, 8009, 8443, 9000
pegasus_http_port_t            tcp      5988
pegasus_https_port_t           tcp      5989
```

3. ポート 3131 の SELinux タイプを、ポート 80 に一致させるように変更します。

```
# semanage port -a -t http_port_t -p tcp 3131
```

4. **httpd** を再開します。

```
# systemctl start httpd
```

5. ただし、コンテンツにはアクセスできません。

```
# wget localhost:3131/index.html
...
HTTP request sent, awaiting response... 403 Forbidden
...
```

sealert ツールの理由を確認します。

```
# sealert -l "*"
...
SELinux is preventing httpd from getattr access on the file /var/test_www/html/index.html.
...
```

6. **matchpathcon** ツールを使用して、標準パスと新規パスの SELinux タイプを比較します。

```
# matchpathcon /var/www/html /var/www/html system_u:object_r:httpd_sys_content_t:s0
# matchpathcon /var/test_www/html /var/test_www/html system_u:object_r:var_t:s0
```

7. 新しい /**var**/**test_www**/**html**/ コンテンツディレクトリーの SELinux タイプを、デフォルトの /**var**/ ディレクトリーのタイプに変更します。

```
# semanage fcontext -a -e /var/www /var/test_www
```

8. 再帰的に、/**var** ディレクトリーのラベルを再設定します。

```
# restorecon -Rv /var/
...
Relabeled /var/test_www/html from unconfined_u:object_r:var_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
Relabeled /var/test_www/html/index.html from unconfined_u:object_r:var_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
```

検証手順

1. **httpd** サービスが実行していることを確認します。

```
# systemctl status httpd
...
Active: active (running)
...
systemd[1]: Started The Apache HTTP Server.
httpd[14888]: Server configured, listening on: port 3131
...
```

2. Apache HTTP サーバーが提供するコンテンツがアクセスできることを確認します。

```
# wget localhost:3131/index.html
...
```

```
HTTP request sent, awaiting response... 200 OK
Length: 0 [text/html]
Saving to: ‘index.html’
...
```

関連情報

- man ページの **semanage(8)**、**matchpathcon(8)**、および **sealert(8)**

4.2. SELINUX ブール値で NFS ボリュームおよび CIFS ボリュームの共有に関するポリシーの調整

SELinux ポリシーの記述に関する知識がなくても、ブール値を使用して、ランタイム時に SELinux ポリシーの一部を変更できます。これにより、SELinux ポリシーの再読み込みや再コンパイルを行わずに、サービスが NFS ボリュームにアクセスするのを許可するなどの変更が可能になります。以下の手順では、SELinux ブール値の一覧を表示して、ポリシーで必要な変更を実現するように設定します。

クライアント側の NFS マウントは、NFS ボリュームのポリシーで定義されたデフォルトのコンテキストでラベル付けされます。RHEL では、このデフォルトのコンテキストは **nfs_t** タイプを使用します。また、クライアント側にマウントされた Samba 共有には、ポリシーで定義されたデフォルトのコンテキストがラベル付けされます。このデフォルトのコンテキストは **cifs_t** タイプを使用します。ブール値を有効または無効にすると、**nfs_t** タイプおよび **cifs_t** タイプにアクセスできるサービスを制御できます。

Apache HTTP サーバーサービス (**httpd**) による NFS ボリュームおよび CIFS ボリュームへのアクセスおよび共有を許可するには、以下の手順を実行します。

前提条件

- 必要に応じて、**selinux-policy-devel** パッケージをインストールして、**semanage boolean -l** コマンドの出力で SELinux ブール値の詳細を、より明確で詳細な形で取得します。

手順

1. NFS、CIFS、および Apache に関する SELinux ブール値を特定します。

```
# semanage boolean -l | grep 'nfs\|cifs' | grep httpd
httpd_use_cifs                 (off  ,  off)  Allow httpd to access cifs file systems
httpd_use_nfs                  (off  ,  off)  Allow httpd to access nfs file systems
```

2. ブール値の現在の状態を一覧表示します。

```
$ getsebool -a | grep 'nfs\|cifs' | grep httpd
httpd_use_cifs --> off
httpd_use_nfs --> off
```

3. 指定されたブール値を有効にします。

```
# setsebool httpd_use_nfs on
# setsebool httpd_use_cifs on
```

注記

setsebool で **-P** オプションを使用使用すると、システムを再起動しても変更が持続します。**setsebool -P** コマンドは、ポリシー全体を再構築する必要がありますが、設定によっては少し時間がかかる場合があります。

検証手順

1. ブール値が **on** になっていることを確認します。

```
$ getsebool -a | grep 'nfs\|cifs' | grep httpd
httpd_use_cifs --> on
httpd_use_nfs --> on
```

関連情報

- man ページの **semanage-boolean(8)**、**sepolicy-booleans(8)**、**getsebool(8)**、**setsebool(8)**、**booleans(5)**、および **booleans(8)**

4.3. 関連情報

- SELinux 拒否の特定と分析の詳細は、「SELinux 関連の問題のトラブルシューティング」を参照してください。

第5章 SELINUX 関連の問題のトラブルシューティング

SELinux が無効になっていたシステムで SELinux を有効にする場合や、標準以外の設定でサービスを実行した場合は、SELinux がブロックできる状況のトラブルシューティングを行う必要がある可能性があります。ほとんどの場合、SELinux の拒否は、設定間違いによるものになります。

5.1. SELINUX 拒否の特定

この手順で必要な手順のみを行います。ほとんどの場合は、ステップ 1 のみを実行する必要があります。

手順

1. SELinux がシナリオをブロックしたときに、最初に拒否に関する詳細情報を確認するのは **/var/log/audit/audit.log** ファイルとなります。Audit ログのクエリーには、**ausearch** ツールを使用します。アクセスを許可する、または許可しないといった SELinux の決定はキャッシュされ、このキャッシュは AVC (アクセスベクターキャッシュ) として知られています。たとえば、メッセージタイプパラメーターには **AVC** および **USER_AVC** の値を使用します。

   ```
   # ausearch -m AVC,USER_AVC,SELINUX_ERR,USER_SELINUX_ERR -ts recent
   ```

   一致するものがない場合は、Audit デーモンが実行しているかどうかを確認します。実行していない場合は、**auditd** を起動して Audit ログを再確認してから、拒否されたシナリオを繰り返します。

2. **auditd** が実行中で、**ausearch** の出力に一致がない場合は、**systemd** ジャーナルが出力するメッセージを確認してください。

   ```
   # journalctl -t setroubleshoot
   ```

3. SELinux が有効で、Audit デーモンがシステムで実行していない場合は、**dmesg** コマンドの出力で特定の SELinux メッセージを検索します。

   ```
   # dmesg | grep -i -e type=1300 -e type=1400
   ```

4. 以上の 3 つを確認しても、何も見つからない場合もあります。この場合、**dontaudit** ルールが原因で AVC 拒否を非表示にできます。
   一時的に **dontaudit** ルールを無効にし、すべての拒否をログに記録できるようにするには、以下のコマンドを実行します。

   ```
   # semodule -DB
   ```

   以前の手順で、拒否されたシナリオを再実行し、拒否メッセージを取得したら、次のコマンドはポリシーで **dontaudit** ルールを再度有効にします。

   ```
   # semodule -B
   ```

5. これまでの 4 つのステップをすべて試し、それでも問題が解決しない場合は、SELinux がシナリオをブロックするかどうかを検討してください。

   - Permissive モードに切り替えます。

```
# setenforce 0
$ getenforce
Permissive
```

- シナリオを繰り返します。

上記を試しても問題が発生する場合は、SELinux 以外に原因があります。

5.2. SELINUX 拒否メッセージの分析

SELinux がシナリオをブロックしていることを 特定 したら、修正する前に原因分析が必要になる場合があります。

前提条件

- **policycoreutils-python-utils** パッケージおよび **setroubleshoot-server** パッケージがシステムにインストールされている。

手順

1. 以下のように、**sealert** コマンドを実行して、ログに記録されている拒否の詳細を一覧表示します。

```
$ sealert -l "*"
SELinux is preventing /usr/bin/passwd from write access on the file
/root/test.

*****  Plugin leaks (86.2 confidence) suggests ******************************

If you want to ignore passwd trying to write access the test file,
because you believe it should not need this access.
Then you should report this as a bug.
You can generate a local policy module to dontaudit this access.
Do
# ausearch -x /usr/bin/passwd --raw | audit2allow -D -M my-passwd
# semodule -X 300 -i my-passwd.pp

*****  Plugin catchall (14.7 confidence) suggests **************************


...

Raw Audit Messages
type=AVC msg=audit(1553609555.619:127): avc:  denied  { write } for
pid=4097 comm="passwd" path="/root/test" dev="dm-0" ino=17142697
scontext=unconfined_u:unconfined_r:passwd_t:s0-s0:c0.c1023
tcontext=unconfined_u:object_r:admin_home_t:s0 tclass=file permissive=0

...

Hash: passwd,passwd_t,admin_home_t,file,write
```

2. 前の手順で取得した出力に明確な提案が含まれていない場合は、以下のコマンドを実行します。

- 完全パス監査を有効にして、アクセスしたオブジェクトの完全パスを表示し、追加の Linux

Audit イベントフィールドが表示されるようにします。

```
# auditctl -w /etc/shadow -p w -k shadow-write
```

- **setroubleshoot** キャッシュを削除します。

```
# rm -f /var/lib/setroubleshoot/setroubleshoot.xml
```

- 問題を再現します。
- ステップ 1 を繰り返します。

3. **sealert** が **catchall** 提案を返すか、**audit2allow** ツールを使用して新しいルールを追加するように提案した場合は、「Audit ログの SELinux 拒否」で説明されている例と問題を一致させます。

関連情報

- man ページの **sealert(8)**

5.3. 分析した SELINUX 拒否の修正

ほとんどの場合、**sealert** ツールが提供する提案により、SELinux ポリシーに関連する問題を修正するための適切なガイドが提供されます。「SELinux 拒否のメッセージを解析」を参照してください。**sealert** を使用して SELinux 拒否の解析方法を参照してください。

ツールが、**audit2allow** ツールを使用して設定変更を提案している場合は注意が必要です。**audit2allow** を使用して、SELinux 拒否を確認する際に、最初のオプションとしてローカルポリシーモジュールを生成することはできません。トラブルシューティングは、ラベル付けの問題があるかどうかを最初に確認します。2 番目に多いのが、SELinux が、プロセスの設定変更を認識していない場合です。

ラベル付けの問題

ラベル付けの問題の一般的な原因として、非標準ディレクトリーがサービスに使用される場合が挙げられます。たとえば、管理者が、Web サイト **/var/www/html/** ではなく、**/srv/myweb/** を使用したい場合があります。Red Hat Enterprise Linux では、**/srv** ディレクトリーには **var_t** タイプのラベルが付けられます。**/srv** で作成されるファイルおよびディレクトリーは、このタイプを継承します。また、**/myserver** などの最上位のディレクトリーに新規作成したオブジェクトには、**default_t** タイプのラベルが付けられます。SELinux は、Apache HTTP Server (**httpd**) がこの両方のタイプにアクセスできないようにします。アクセスを許可するには、SELinux では、**/srv/myweb/** のファイルが **httpd** からアクセス可能であることを認識する必要があります。

```
# semanage fcontext -a -t httpd_sys_content_t "/srv/myweb(/.*)?"
```

この **semanage** コマンドは、**/srv/myweb/** ディレクトリーおよびその下のすべてのファイルおよびディレクトリーのコンテキストを SELinux ファイルコンテキストの設定に追加します。**semanage** ユーティリティーはコンテキストを変更しません。root で **restorecon** ユーティリティーを使用して変更を適用します。

```
# restorecon -R -v /srv/myweb
```

コンテキストの誤り

matchpathcon ユーティリティーは、ファイルパスのコンテキストを確認し、そのパスのデフォルトラベルと比較します。以下の例では、ラベルが間違っているファイルを含むディレクトリーにおける **matchpathcon** の使用方法を説明します。

```
$ matchpathcon -V /var/www/html/*
/var/www/html/index.html has context unconfined_u:object_r:user_home_t:s0, should be
system_u:object_r:httpd_sys_content_t:s0
/var/www/html/page1.html has context unconfined_u:object_r:user_home_t:s0, should be
system_u:object_r:httpd_sys_content_t:s0
```

この例では、**index.html** ファイルおよび **page1.html** ファイルに、**user_home_t** タイプのラベルが付けられています。このタイプは、ユーザーのホームディレクトリーのファイルに使用されます。**mv** コマンドを使用してファイルをホームディレクトリーから移動すると、ファイルに **user_home_t** タイプのラベルが付けられることがあります。このタイプは、ホームディレクトリー内しか存在しません。**restorecon** ユーティリティーを使用して、対象のファイルを正しいタイプに戻します。

```
# restorecon -v /var/www/html/index.html
restorecon reset /var/www/html/index.html context unconfined_u:object_r:user_home_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
```

ディレクトリー下の全ファイルのコンテキストを復元するには、**-R** オプションを使用します。

```
# restorecon -R -v /var/www/html/
restorecon reset /var/www/html/page1.html context unconfined_u:object_r:samba_share_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
restorecon reset /var/www/html/index.html context unconfined_u:object_r:samba_share_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
```

標準以外の方法で設定された制限のあるアプリケーション

サービスはさまざまな方法で実行できます。この場合は、サービスの実行方法を指定する必要があります。これは、SELinux ポリシーの一部をランタイム時に変更できるようにする SELinux ブール値を使用して実行できます。これにより、SELinux ポリシーの再読み込みや再コンパイルを行わずに、サービスが NFS ボリュームにアクセスするのを許可するなどの変更が可能になります。また、デフォルト以外のポート番号でサービスを実行するには、**semanage** コマンドを使用してポリシー設定を更新する必要があります。

たとえば、Apache HTTP Server が MariaDB と接続するのを許可する場合は、**httpd_can_network_connect_db** のブール値を有効にします。

```
# setsebool -P httpd_can_network_connect_db on
```

-P オプションを使用すると、システムの再起動後も設定が永続化されることに注意してください。

特定のサービスでアクセスが拒否される場合は、**getsebool** ユーティリティーおよび **grep** ユーティリティーを使用して、アクセスを許可するブール値が利用できるかどうかを確認します。たとえば、**getsebool -a | grep ftp** コマンドを使用して FTP 関連のブール値を検索します。

```
$ getsebool -a | grep ftp
ftpd_anon_write --> off
ftpd_full_access --> off
ftpd_use_cifs --> off
ftpd_use_nfs --> off
```

```
ftpd_connect_db --> off
httpd_enable_ftp_server --> off
tftp_anon_write --> off
```

ブール値の一覧を取得し、ブール値が有効または無効かどうかを確認する場合は、**getsebool -a** コマンドを使用します。ブール値の一覧とその意味を取得し、有効かどうかを調べるには、**selinux-policy-devel** パッケージをインストールして、root で **semanage boolean -l** コマンドを実行します。

ポート番号

ポリシー設定によっては、サービスは特定のポート番号でのみ実行できます。ポリシーを変更せずにサービスが実行するポートを変更しようとすると、サービスが起動できなくなる可能性があります。たとえば、root で **semanage port -l | grep http** コマンドを 実行して、 **http** 関連のポートを一覧表示します。

```
# semanage port -l | grep http
http_cache_port_t              tcp      3128, 8080, 8118
http_cache_port_t              udp      3130
http_port_t                    tcp      80, 443, 488, 8008, 8009, 8443
pegasus_http_port_t            tcp      5988
pegasus_https_port_t           tcp      5989
```

http_port_t ポートタイプは、Apache HTTP Server がリッスン可能なポートを定義します。この場合の TCP ポートは 80、443、488、8008、8009、および 8443 になります。**httpd** がポート 9876 でリッスンする (**Listen 9876**) ように、管理者が **httpd.conf** を設定していれば、これを反映するようにポリシーが更新されていないと、以下のコマンドに失敗します。

```
# systemctl start httpd.service
Job for httpd.service failed. See 'systemctl status httpd.service' and 'journalctl -xn' for details.

# systemctl status httpd.service
httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled)
   Active: failed (Result: exit-code) since Thu 2013-08-15 09:57:05 CEST; 59s ago
  Process: 16874 ExecStop=/usr/sbin/httpd $OPTIONS -k graceful-stop (code=exited,
status=0/SUCCESS)
  Process: 16870 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited,
status=1/FAILURE)
```

以下のような SELinux 拒否メッセージのログは、**/var/log/audit/audit.log** に記録されます。

```
type=AVC msg=audit(1225948455.061:294): avc:  denied  { name_bind } for  pid=4997
comm="httpd" src=9876 scontext=unconfined_u:system_r:httpd_t:s0
tcontext=system_u:object_r:port_t:s0 tclass=tcp_socket
```

httpd が **http_port_t** ポートタイプに追加されていないポートをリッスンできるようにするには、**semanage port** コマンドを使用して別のラベルをポートに割り当てます。

```
# semanage port -a -t http_port_t -p tcp 9876
```

-a オプションは新規レコードを追加します。**-t** オプションはタイプを定義し、**-p** オプションはプロトコルを定義します。最後の引数は、追加するポート番号です。

進化または破損したアプリケーション、および侵害されたシステム (稀に発生する難しいケース)

アプリケーションにバグが含まれ、SELinux がアクセスを拒否する可能性があります。また、SELinux ルールは進化しています。アプリケーションが特定の方法で実行しているのを SELinux が認識しておらず、アプリケーションが期待どおりに動作していても、アクセスを拒否してしまうような場合もあります。たとえば、PostgreSQL の新規バージョンがリリースされると、現在のポリシーが考慮されないアクションが実行される可能性があり、アクセスが許可される場合でもアクセスが拒否されます。

このような状況では、アクセスが拒否された後に **audit2allow** ユーティリティーを使用して、アクセスを許可するカスタムポリシーモジュールを作成します。SELinux ポリシーで足りないルールは Red Hat Bugzilla から報告できます。Red Hat Enterprise Linux 8 の場合は、製品で **Red Hat Enterprise Linux 8** を選択し、**selinux-policy** コンポーネントを選択してバグを作成します。このバグレポートに、**audit2allow -w -a** コマンドおよび **audit2allow -a** コマンドの出力を追加します。

アプリケーションが主要なセキュリティー特権を要求すると、そのアプリケーションが危険にさらされたことを示す警告が発生することがあります。侵入検出ツールを使用して、このような疑わしい動作を検証します。

また、Red Hat カスタマーポータル の **Solution Engine** では、同じ問題または非常に類似する問題に関する記事が提供されます。ここでは、問題の解決策と考えられる方法が示されています。関連する製品とバージョンを選択し、**selinux**、**avc** などの SELinux 関連のキーワードと、ブロックされたサービスまたはアプリケーションの名前 (**selinux samba** など) を使用します。

5.4. AUDIT ログの SELINUX 拒否

Linux Audit システムは、デフォルトで **/var/log/audit/audit.log** ファイルにログエントリーを保存します。SELinux 関連の記録のみを一覧表示するには、メッセージタイプパラメーターの **AVC** および **AVC_USER** (並びに必要に応じたパラメーター) を付けて **ausearch** コマンドを実行します。

```
# ausearch -m AVC,USER_AVC,SELINUX_ERR,USER_SELINUX_ERR
```

Audit ログファイルの SELinux 拒否エントリーは次のようになります。

```
type=AVC msg=audit(1395177286.929:1638): avc:  denied  { read } for  pid=6591 comm="httpd"
name="webpages" dev="0:37" ino=2112 scontext=system_u:system_r:httpd_t:s0
tcontext=system_u:object_r:nfs_t:s0 tclass=dir
```

このエントリーで最も重要な部分は以下の通りです。

- **avc: denied** - SELinux によって実行され、アクセスベクターキャッシュ (AVC) で記録されるアクション
- **{ read }** - 拒否の動作
- **pid=6591** - 拒否されたアクションの実行を試みたサブジェクトのプロセス ID
- **comm="httpd"** - 分析しているプロセスを呼び出すのに使用されたコマンドの名前
- **httpd_t** - プロセスの SELinux タイプ
- **nfs_t** - プロセスのアクションに影響するオブジェクトの SELinux タイプ
- **tclass=dir** - ターゲットオブジェクトクラス

このログエントリーは、以下のように解釈できます。

SELinux が、nfs_t タイプのディレクトリーから読み込む PID 6591 および httpd_t タイプの httpd プロセスを拒否します。

Apache HTTP Server が Samba スイートのタイプでラベル付けされたディレクトリーにアクセスしようとすると、以下の SELinux 拒否メッセージが発生します。

```
type=AVC msg=audit(1226874073.147:96): avc:  denied  { getattr } for  pid=2465 comm="httpd"
path="/var/www/html/file1" dev=dm-0 ino=284133 scontext=unconfined_u:system_r:httpd_t:s0
tcontext=unconfined_u:object_r:samba_share_t:s0 tclass=file
```

- **{ getattr }** - **getattr** エントリーは、ターゲットファイルのステータス情報をソースプロセスが読み取ろうとしているのを示します。これは、ファイルを読み取る前に発生します。プロセスがファイルにアクセスし、適切なラベルがないため、SELinux はこのアクションを拒否します。一般的に表示されるパーミッションには、**getattr**、**read**、**write** などが含まれます。
- **path="/var/www/html/file1"** - アクセスを試みたオブジェクト (ターゲット) へのパス。
- **scontext="unconfined_u:system_r:httpd_t:s0"** - 拒否されたアクションを試みたプロセス (ソース) の SELinux コンテキスト。この場合、Apache HTTP Server は **httpd_t** タイプで実行している SELinux コンテキストです。
- **tcontext="unconfined_u:object_r:samba_share_t:s0"** - プロセスがアクセスを試みたオブジェクト (ターゲット) の SELinux コンテキストです。この例では、これが **file1** の SELinux コンテキストです。

この SELinux 拒否は、以下のように解釈できます。

SELinux は、samba_share_t タイプの /var/www/html/file1 ファイルにアクセスする PID 2465 の httpd プロセスを拒否し、その他に許可するような設定がない場合は、httpd_t ドメインで実行しているプロセスにアクセスできません。

関連情報

- 詳細は、man ページの **auditd(8)** および **ausearch(8)** を参照してください。

5.5. 関連情報

- カスタマーポータルの「Basic SELinux Troubleshooting in CLI」
- Fedora People のプレゼンテーション - 「What is SELinux trying to tell me?The 4 key causes of SELinux errors」

第6章 カスタム SELINUX ポリシーの作成

本セクションでは、SELinux が制限するアプリケーションの実行を可能にするカスタムポリシーを作成および使用する方法を説明します。

6.1. カスタム SELINUX ポリシーおよび関連ツール

SELinux セキュリティーポリシーは、SELinux ルールのコレクションです。ポリシーは、SELinux の中核となるコンポーネントで、SELinux ユーザー空間ツールによりカーネルに読み込まれます。カーネルは、SELinux ポリシーを使用して、システム上のアクセス要求を評価します。デフォルトでは、SELinux は、読み込んだポリシーで指定されたルールに対応するリクエストを除き、すべてのリクエストを拒否します。

各 SELinux ポリシールールは、プロセスとシステムリソースとの対話を説明します。

```
ALLOW apache_process apache_log:FILE READ;
```

このルールの例は、「**Apache プロセスが、そのログファイル を読む こと**」のように読み取ることができます。このルールでは、**apache_process** と **apache_log** が **ラベル** になります。SELinux セキュリティーポリシーは、プロセスにラベルを割り当て、システムリソースに関係を定義します。これにより、ポリシーはオペレーティングシステムエンティティーを SELinux レイヤーにマッピングします。

SELinux ラベルは、**ext2** などのファイルシステムの拡張属性として保存されます。**getfattr** ユーティリティーまたは **ls -Z** コマンドを使用すると、これを一覧表示できます。以下に例を示します。

```
$ ls -Z /etc/passwd
system_u:object_r:passwd_file_t:s0 /etc/passwd
```

ここでの **system_u** は SELinux ユーザーで、 **object_r** は SELinux ロールの例になります。**passwd_file_t** は SELinux ドメインです。

selinux-policy パッケージが提供するデフォルトの SELinux ポリシーには、Red Hat Enterprise Linux 8 の一部であるアプリケーションおよびデーモンのルールが含まれ、リポジトリーのパッケージにより提供されます。このディストリビューションポリシーのルールで説明されていないアプリケーションは、SELinux によって制限されません。これを変更するには、追加の定義およびルールが含まれるポリシーモジュールを使用してポリシーを変更する必要があります。

Red Hat Enterprise Linux 8 では、インストールした SELinux ポリシーに問い合わせ、**sepolicy** ツールを使用して新しいポリシーモジュールを生成できるようになります。**sepolicy** がポリシーモジュールとともに生成するスクリプトには、**restorecon** ユーティリティーを使用するコマンドが常に含まれます。このユーティリティーは、ファイルシステムの選択した部分で問題のラベル付けを行う基本的なツールです。

関連情報

- 詳細は、man ページの **sepolicy(8)** および **getfattr(1)** を参照してください。

6.2. カスタムアプリケーション用の SELINUX ポリシーの作成および実施

この手順例では、SELinux で単純なデーモンを指定する手順を説明します。デーモンをカスタムアプリケーションに置き換え、そのアプリケーションとセキュリティーポリシーの要件に従ってサンプルルールを変更します。

前提条件

- **policycoreutils-devel** パッケージとその依存関係がシステムにインストールされている。

手順

1. この手順例では、**/var/log/messages** ファイルを開いて書き込みを行う簡単なデーモンを準備します。

   a. 新しいファイルを作成して、選択したテキストエディターで開きます。

   ```
   $ vi mydaemon.c
   ```

   b. 以下のコードを挿入します。

   ```
   #include <unistd.h>
   #include <stdio.h>

   FILE *f;

   int main(void)
   {
   while(1) {
   f = fopen("/var/log/messages","w");
           sleep(5);
           fclose(f);
       }
   }
   ```

   c. ファイルをコンパイルします。

   ```
   $ gcc -o mydaemon mydaemon.c
   ```

   d. デーモンの **systemd** ユニットファイルを作成します。

   ```
   $ vi mydaemon.service
   [Unit]
   Description=Simple testing daemon

   [Service]
   Type=simple
   ExecStart=/usr/local/bin/mydaemon

   [Install]
   WantedBy=multi-user.target
   ```

   e. デーモンをインストールして起動します。

   ```
   # cp mydaemon /usr/local/bin/
   # cp mydaemon.service /usr/lib/systemd/system
   # systemctl start mydaemon
   # systemctl status mydaemon
   ● mydaemon.service - Simple testing daemon
     Loaded: loaded (/usr/lib/systemd/system/mydaemon.service; disabled; vendor preset:
   ```

```
disabled)
   Active: active (running) since Sat 2020-05-23 16:56:01 CEST; 19s ago
 Main PID: 4117 (mydaemon)
    Tasks: 1
   Memory: 148.0K
   CGroup: /system.slice/mydaemon.service
           └─4117 /usr/local/bin/mydaemon

May 23 16:56:01 localhost.localdomain systemd[1]: Started Simple testing daemon.
```

f. 新しいデーモンが SELinux によって制限されていないことを確認します。

```
$ ps -efZ | grep mydaemon
system_u:system_r:unconfined_service_t:s0 root 4117    1  0 16:56 ?        00:00:00
/usr/local/bin/mydaemon
```

2. デーモンのカスタムポリシーを生成します。

```
$ sepolicy generate --init /usr/local/bin/mydaemon
Created the following files:
/home/example.user/mysepol/mydaemon.te # Type Enforcement file
/home/example.user/mysepol/mydaemon.if # Interface file
/home/example.user/mysepol/mydaemon.fc # File Contexts file
/home/example.user/mysepol/mydaemon_selinux.spec # Spec file
/home/example.user/mysepol/mydaemon.sh # Setup Script
```

3. 直前のコマンドで作成した設定スクリプトを使用して、新しいポリシーモジュールでシステムポリシーを再構築します。

```
# ./mydaemon.sh
Building and Loading Policy
+ make -f /usr/share/selinux/devel/Makefile mydaemon.pp
Compiling targeted mydaemon module
Creating targeted mydaemon.pp policy package
rm tmp/mydaemon.mod.fc tmp/mydaemon.mod
+ /usr/sbin/semodule -i mydaemon.pp
...
```

restorecon コマンドを使用して、設定スクリプトがファイルシステムの対応する部分の再ラベル付けを行うことに注意してください。

```
restorecon -v /usr/local/bin/mydaemon /usr/lib/systemd/system
```

4. デーモンを再起動して、SELinux が制限のあるデーモンを実行していることを確認します。

```
# systemctl restart mydaemon
$ ps -efZ | grep mydaemon
system_u:system_r:mydaemon_t:s0 root        8150       1  0 17:18 ?        00:00:00
/usr/local/bin/mydaemon
```

5. デーモンは SELinux によって制限されているため、SELinux はこのデーモンが **/var/log/messages** にアクセスできないようにします。対応する拒否メッセージを表示します。

```
# ausearch -m AVC -ts recent
...
type=AVC msg=audit(1590247112.719:5935): avc:  denied  { open } for  pid=8150
comm="mydaemon" path="/var/log/messages" dev="dm-0" ino=2430831
scontext=system_u:system_r:mydaemon_t:s0 tcontext=unconfined_u:object_r:var_log_t:s0
tclass=file permissive=1
...
```

6. **sealert** ツールを使用して追加情報を取得することもできます。

```
$ sealert
SELinux is preventing mydaemon from open access on the file /var/log/messages.

  Plugin catchall (100. confidence) suggests   *

If you believe that mydaemon should be allowed open access on the messages file by
default.
Then you should report this as a bug.
You can generate a local policy module to allow this access.
Do
allow this access for now by executing:
# ausearch -c 'mydaemon' --raw | audit2allow -M my-mydaemon
# semodule -X 300 -i my-mydaemon.pp

Additional Information:
Source Context                system_u:system_r:mydaemon_t:s0
Target Context                unconfined_u:object_r:var_log_t:s0
Target Objects                /var/log/messages [ file ]
Source                        mydaemon

...
```

7. **audit2allow ツールを使用して、変更を提案します。**

```
$ ausearch -m AVC -ts recent | audit2allow -R

require {
 type mydaemon_t;
}

#============= mydaemon_t ==============
logging_write_generic_logs(mydaemon_t)
```

8. **audit2allow が提案するルールは特定のケースでは正しくない可能性があるため、出力の一部のみを使用して対応するポリシーインターフェースを見つけます。**

```
$ grep -r "logging_write_generic_logs" /usr/share/selinux/devel/include/ | grep .if
/usr/share/selinux/devel/include/system/logging.if:interface(logging_write_generic_logs',
```

9. **インターフェースの定義を確認します。**

```
$ cat /usr/share/selinux/devel/include/system/logging.if
...
interface(logging_write_generic_logs',
```

```
        gen_require(`
                type var_log_t;
        ')

        files_search_var($1)
        allow $1 var_log_t:dir list_dir_perms;
        write_files_pattern($1, var_log_t, var_log_t)
')
...
```

10. **この場合は、推奨されるインターフェースを使用できます。タイプの適用ファイルに対応するルールを追加します。**

```
$ echo "logging_write_generic_logs(mydaemon_t)" >> mydaemon.te
```

または、インターフェースを使用する代わりに、このルールを追加することもできます。

```
$ echo "allow mydaemon_t var_log_t:file { open write getattr };" >> mydaemon.te
```

11. **ポリシーを再インストールします。**

```
# ./mydaemon.sh
Building and Loading Policy
+ make -f /usr/share/selinux/devel/Makefile mydaemon.pp
Compiling targeted mydaemon module
Creating targeted mydaemon.pp policy package
rm tmp/mydaemon.mod.fc tmp/mydaemon.mod
+ /usr/sbin/semodule -i mydaemon.pp
...
```

検証手順

1. **アプリケーションが SELinux によって制限されて実行されていることを確認します。以下に例を示します。**

```
$ ps -efZ | grep mydaemon
system_u:system_r:mydaemon_t:s0 root       8150       1  0 17:18 ?        00:00:00
/usr/local/bin/mydaemon
```

2. **カスタムアプリケーションが SELinux 拒否を生じさせないことを確認します。**

```
# ausearch -m AVC -ts recent
<no matches>
```

関連情報

- **詳細は、 man ページの sepolgen(8)、ausearch(8)、audit2allow(1)、audit2why(1)、sealert(8)、および restorecon(8) を参照してください。**

6.3. 関連情報

- **詳細情報およびその他の例は、SELinux Policy Workshop を参照してください。**

第7章 コンテナーの SELINUX ポリシーの作成

RHEL 8は、udica パッケージを使用してコンテナーの SELinux ポリシーを生成するツールを提供します。udica を使用すると、最適なセキュリティーポリシーを作成して、ストレージ、デバイス、ネットワークなどのホストシステムリソースにコンテナーがアクセスする方法を制御できます。これにより、セキュリティー違反に対してコンテナーのデプロイメントを強化でき、規制コンプライアンスの実現や維持も簡単になります。

7.1. UDICA の SELINUX ポリシージェネレーターの概要

カスタムコンテナーの SELinux ポリシーの新規作成を簡素化するために、RHEL 8 には udica ユーティリティが用意されています。このツールを使用して、Linux の機能、マウントポイント、およびポート定義を含むコンテナーの JavaScript Object Notation (JSON) ファイルの検査に基づいてポリシーを作成できます。したがって、ツールは、検査の結果を使用して生成されたルールと、指定された SELinux Common Intermediate Language (CIL) ブロックから継承されたルールを組み合わせます。

udica を使用してコンテナーの SELinux ポリシーを生成するプロセスには、以下の 3 つの主要な部分があります。

1. JSON 形式のコンテナー仕様ファイルを解析する。
2. 最初の部分の結果に基づいて適切な許可ルールを見つける。
3. 最終的な SELinux ポリシーを生成する。

解析フェーズでは、udica が Linux 機能、ネットワークポート、およびマウントポイントを探します。

この結果に基づいて、udica はコンテナーに必要な Linux 機能を検出し、これらすべての機能を許可する SELinux ルールを作成します。コンテナーが特定のポートにバインドする場合は、udica が SELinux ユーザー空間ライブラリーを使用して、検査対象のコンテナーが使用するポートの正しい SELinux ラベルを取得します。

その後、udica は、どのディレクトリーがホストからコンテナーのファイルシステムのネームスペースにマウントされているかを検出します。

CIL のブロック継承機能により、udica は SELinux のテンプレートを作成でき、特定のアクションに焦点を当てた ルール を許可します。次に例を示します。

- ホームディレクトリーへのアクセスを許可する。
- ログファイルへのアクセスを許可する。
- Xserver との通信へのアクセスを許可する。

このようなテンプレートはブロックと呼ばれ、最終的な SELinux ポリシーはブロックをマージして作成されます。

関連情報

- udica を使用して SELinux ポリシーを生成するプロセスの詳細は、Red Hat ブログ記事「Generate SELinux policies for containers with udica」を参照してください。

7.2. カスタムコンテナーの SELINUX ポリシーを作成して使用

カスタムコンテナーの SELinux セキュリティポリシーを生成するには、以下の手順を行います。

前提条件

- **コンテナーを管理する podman ツールがインストールされている。そうでない場合は、yum install podman コマンドを使用します。**
- **カスタムの Linux コンテナー - この例では ubi8 です。**

手順

1. **udica パッケージをインストールします。**

```
# yum install -y udica
```

udica を含むコンテナーソフトウェアパッケージセットを提供する container-tools モジュールをインストールします。

```
# yum module install -y container-tools
```

2. **/home ディレクトリーを読み取り権限でマウントする ubi8 コンテナーと、読み取りおよび書き込みの権限で /var/spool ディレクトリーをマウントします。コンテナーはポート 21 を公開します。**

```
# podman run --env container=podman -v /home:/home:ro -v /var/spool:/var/spool:rw -p
21:21 -it ubi8 bash
```

*コンテナーは、SELinux のタイプが **container_t** で実行されることに注意してください。このタイプは、SELinux ポリシー内のすべてのコンテナーの汎用ドメインであり、シナリオに対して厳密すぎるか緩すぎる可能性があります。*

3. ***podman ps** コマンドを入力して、コンテナーの **ID** を取得します。*

```
# podman ps
CONTAINER ID  IMAGE                                 COMMAND  CREATED        STATUS
PORTS  NAMES
37a3635afb8f  registry.access.redhat.com/ubi8:latest  bash     15 minutes ago  Up 15
minutes ago          heuristic_lewin
```

4. *コンテナーの JSON ファイルを作成し、**udica** を使用して JSON ファイルの情報に基づいてポリシーモジュールを作成します。*

```
# podman inspect 37a3635afb8f > container.json
# udica -j container.json my_container
Policy my_container with container id 37a3635afb8f created!
[...]
```

または、次のようになります。

```
# podman inspect 37a3635afb8f | udica my_container
Policy my_container with container id 37a3635afb8f created!

Please load these modules using:
# semodule -i my_container.cil
```

```
/usr/share/udica/templates/{base_container.cil,net_container.cil,home_container.cil}

Restart the container with: "--security-opt label=type:my_container.process" parameter
```

5. 前の手順の **udica** の出力で提案されているように、ポリシーモジュールを読み込みます。

```
# semodule -i my_container.cil
/usr/share/udica/templates/{base_container.cil,net_container.cil,home_container.cil}
```

6. コンテナーを停止し、**--security-opt label=type:my_container.process** オプションを使用して再起動します。

```
# podman stop 37a3635afb8f
# podman run --security-opt label=type:my_container.process -v /home:/home:ro -v
/var/spool:/var/spool:rw -p 21:21 -it ubi8 bash
```

検証手順

1. コンテナーが、**my_container.process** タイプで実行されることを確認します。

```
# ps -efZ | grep my_container.process
unconfined_u:system_r:container_runtime_t:s0-s0:c0.c1023 root 2275 434  1 13:49 pts/1
00:00:00 podman run --security-opt label=type:my_container.process -v /home:/home:ro -v
/var/spool:/var/spool:rw -p 21:21 -it ubi8 bash
system_u:system_r:my_container.process:s0:c270,c963 root 2317 2305  0 13:49 pts/0
00:00:00 bash
```

2. SELinux が、マウントポイント **/home** および **/var/spool** へのアクセスを許可していることを確認します。

```
[root@37a3635afb8f /]# cd /home
[root@37a3635afb8f home]# ls
username
[root@37a3635afb8f ~]# cd /var/spool/
[root@37a3635afb8f spool]# touch test
[root@37a3635afb8f spool]#
```

3. SELinux がポート 21 へのバインドのみを許可していることを確認します。

```
[root@37a3635afb8f /]# yum install nmap-ncat
[root@37a3635afb8f /]# nc -lvp 21
Ncat: Version 7.60 ( https://nmap.org/ncat )
Ncat: Generating a temporary 1024-bit RSA key. Use --ssl-key and --ssl-cert to use a
permanent one.
Ncat: SHA-1 fingerprint: 6EEC 102E 6666 5F96 CC4F E5FA A1BE 4A5E 6C76 B6DC
Ncat: Listening on :::21
Ncat: Listening on 0.0.0.0:21

[root@37a3635afb8f /]# nc -lvp 80
Ncat: Version 7.60 ( https://nmap.org/ncat )
Ncat: Generating a temporary 1024-bit RSA key. Use --ssl-key and --ssl-cert to use a
permanent one.
Ncat: SHA-1 fingerprint: 6EEC 102E 6666 5F96 CC4F E5FA A1BE 4A5E 6C76 B6DC
Ncat: bind to :::80: Permission denied. QUITTING.
```

-

関連情報

- *詳細は、man ページの **udica(8)** および **podman(1)** を参照してください。*
- *RHEL でコンテナーを起動する方法とコンテナーイメージの使用方法は、『コンテナーの構築、実行、および管理』を参照してください。*

7.3. 関連情報

- ***udica** を使用したポリシー作成の詳細は、「udica - Generate SELinux policies for containers」を参照してください。*

第8章 複数のシステムへの同じSELINUX 設定のデプロイメント

本セクションでは、検証したSELinux 設定を複数のシステムにデプロイする際に推奨される方法を説明します。

- RHEL システムロールおよびAnsible の使用
- スクリプトでsemanage のexport コマンドおよびimport コマンドの使用

8.1. SELINUX システムロールの概要

RHEL システムロールは、複数のRHEL システムをリモートで管理する一貫した構成インターフェースを提供するAnsible ロールおよびモジュールの集合です。SELinux システムロールは、以下のアクションを有効にします。

- SELinux ブール値、ファイルコンテキスト、ポート、およびログインに関連するローカルポリシーの変更を消去します。
- SELinux ポリシーブール値、ファイルコンテキスト、ポート、およびログインの設定
- 指定されたファイルまたはディレクトリーでファイルコンテキストを復元します。

以下の表は、SELinux システムロールで利用可能な入力変数の概要を示しています。

表8.1 SELinux システムロール変数

ロール変数	説明	CLI の代替手段
selinux_policy	ターゲットプロセスまたは複数レベルのセキュリティー保護を保護するポリシーを選択します。	**/etc/selinux/config** の **SELINUXTYPE**
selinux_state	SELinux モードを切り替えます。**ansible-doc selinux** を参照してください。	**/etc/selinux/config** の **setenforce** and **SELINUX**
selinux_booleans	SELinux ブール値を有効または無効にします。**ansible-doc seboolean** を参照してください。	**setsebool**
selinux_fcontexts	SELinux ファイルコンテキストマッピングを追加または削除します。**ansible-doc sefcontext** を参照してください。	**semanage fcontext**
selinux_restore_dirs	ファイルシステムツリー内の SELinux ラベルを復元します。	**restorecon -R**
selinux_ports	ポートに SELinux ラベルを設定します。**ansible-doc seport** を参照してください。	**semanage port**

ロール変数	説明	CLI の代替手段
selinux_logins	ユーザーを SELinux ユーザーマッピングに設定します。**ansible-doc selogin** を参照してください。	**semanage login**

***rhel-system-roles** パッケージによりインストールされる **/usr/share/doc/rhel-system-roles/selinux/example-selinux-playbook.yml** のサンプル Playbook は、Enforcing モードでターゲットポリシーを設定する方法を示しています。Playbook は、複数のローカルポリシーの変更を適用し、**/tmp/test_dir/** ディレクトリーのファイルコンテキストを復元します。*

関連情報

- *SELinux ロール変数の詳細は、**rhel-system-roles** パッケージをインストールし、**/usr/share/doc/rhel-system-roles/selinux/** ディレクトリーの **README.md** または **README.html** ファイルを参照してください。*
- *RHEL システムロールの詳細は、「RHEL のシステムロールの概要」を参照してください。*

8.2. SELINUX システムロールを使用した複数のシステムに SELINUX 設定を適用

以下の手順に従って、検証した SELinux 設定を使用して Ansible Playbook を準備し、適用します。

前提条件

- *Red Hat Ansible Engine のサブスクリプションがシステムに割り当てられている。詳細は、「Red Hat Ansible Engine のダウンロードおよびインストールの方法」を参照してください。*

手順

1. *RHEL Ansible リポジトリーを有効にします。以下に例を示します。*

```
# subscription-manager repos --enable ansible-2-for-rhel-8-x86_64-rpms
```

2. *Ansible Engine をインストールします。*

```
# yum install ansible
```

3. *RHEL システムロールをインストールします。*

```
# yum install rhel-system-roles
```

4. *SELinux システムロールで Playbook を適用します。*
   *以下のコマンドは、**rhel-system-roles** パッケージに含まれる Playbook のサンプルを適用します。この Playbook をテンプレートとして使用できます。*

```
# ansible-playbook -i host1,host2,host3 /usr/share/doc/rhel-system-roles/selinux/example-selinux-playbook.yml
```

関連情報

- *詳細は、**rhel-system-roles** パッケージをインストールして、**/usr/share/doc/rhel-system-roles/selinux/** ディレクトリーおよび**/usr/share/ansible/roles/rhel-system-roles.selinux/** ディレクトリーを参照してください。*

8.3. SEMANAGE で別のシステムへの SELINUX 設定の転送

以下の手順に従って、RHEL 8 ベースのシステム間で、カスタムおよび検証された SELinux 設定を転送します。

前提条件

- ***policycoreutils-python-utils** パッケージがインストールされている。*

手順

1. *検証された SELinux 設定をエクスポートします。*

```
# semanage export -f ./my-selinux-settings.mod
```

2. *設定を含むファイルを新しいシステムにコピーします。*

```
# scp ./my-selinux-settings.mod new-system-hostname:
```

3. *新しいシステムにログインします。*

```
$ ssh root@new-system-hostname
```

4. *新しいシステムに設定をインポートします。*

```
new-system-hostname# semanage import -f ./my-selinux-settings.mod
```

関連情報

- *man ページの **semanage-export(8)** および **semanage-import(8)***