Red Hat Enterprise Linux 9

SELinux の使用

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

Last Updated: 2022-08-08

Red Hat Enterprise Linux 9 SELinux の使用

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

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、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

オープンソースをより包摂的に

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。これは大規模な取り組みであるため、これらの変更は今後の複数のリリースで段階的に実施されます。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** 形式になっていていることを確認してください。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 SELINUX の使用

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、基本的に次の質問に答えます。**May <subject> do <action> to <object>?**。以下に例を示します。**Web サーバーが、ユーザーのホームディレクトリーにあるファイルにアクセスできる可能性がありますか?**

1.1. SELINUX の概要

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。

Security Enhanced Linux (SELinux) は Mandatory Access Control (MAC) を実装します。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。

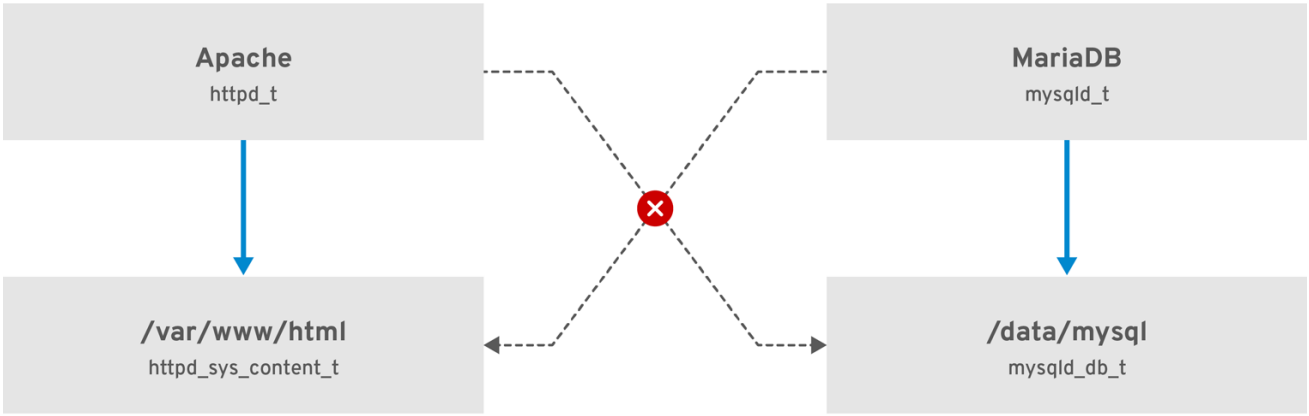
注記

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

- **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 **user_u** ユーザーにマッピングすると、**sudo** や **su** などのユーザー ID (setuid) アプリケーションを設定しない限り、Linux ユーザーを実行できなくなります。
- 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

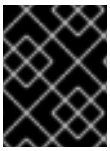
SELinux により傍受され、読み込んだセキュリティーポリシーのコンテキストに従って検討されます。読み込んだポリシーが操作を許可すると、その操作は継続します。許可しないと、その操作はブロックされ、プロセスがエラーを受け取ります。

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。

RHEL 9 では、システムサービスは **systemd** デーモンにより制御されています。**systemd** はすべてのサービスを開始および停止し、ユーザーとプロセスは **systemctl** ユーティリティーを使用して **systemd** と通信します。**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。

プロセスが別のプロセスに **D-Bus** メッセージを送信しているときに、この 2 つのプロセスの **D-Bus** 通信が SELinux ポリシーで許可されていない場合は、システムが **USER_AVC** 拒否メッセージを出力し、D-Bus 通信がタイムアウトになります。2 つのプロセス間の D-Bus 通信は双方向に機能することに注意してください。



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。

RHEL 9 は、SELinux を使用するための以下のパッケージを提供します。

- 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
```

警告

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。

SELinux では、誤ったラベル付けやラベル付けされていないファイルが問題を引き起こすことを防ぐため、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。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。

再ラベル付けのためにシステムを再起動する前に、**enforcing=0** カーネルオプションを使用するなどして、システムが Permissive モードで起動することを確認します。これにより、**selinux-autorelabel** サービスを起動する前に、**systemd** が必要とするラベルのないファイルがシステムにある場合に、システムが起動に失敗することを防ぎます。詳細は、RHBZ#2021835 を参照してください。

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

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

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. 以前は無効にしていたシステムで 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。

警告

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。

SELinux では、誤ったラベル付けやラベル付けされていないファイルが問題を引き起こすことを防ぐため、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。

再ラベル付けのためにシステムを再起動する前に、**enforcing=0** カーネルオプションを使用するなどして、システムが Permissive モードで起動することを確認します。これにより、**selinux-autorelabel** サービスを起動する前に、**systemd** が必要とするラベルのないファイルがシステムにある場合に、システムが起動に失敗することを防ぎます。詳細は、RHBZ#2021835 を参照してください。

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. 次の再起動時に、ファイルが再ラベル付けされていることを確認します。

```
# fixfiles -F onboot
```

これにより、**-F** オプションを含む **/.autorelabel** ファイルが作成されます。

警告

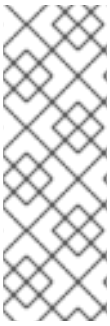
fixfiles -F onboot コマンドを入力する前に、必ず Permissive モードに切り替えてください。これにより、ラベルのないファイルがシステムに含まれている場合に、システムが起動に失敗することを防ぎます。詳細は、RHBZ#2021835 を参照してください。

5. 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

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。

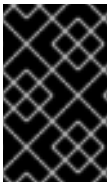
- **unconfined_service_t** ドメインでアプリケーションを実行します。
- アプリケーションに新しいポリシーを記述します。詳細は、カスタム SELinux ポリシーの作成 のセクションを参照してください。

関連情報

- SELinux states and modes section covers temporary changes in modes.

2.5. SELINUX の無効化

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。



重要

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。

前提条件

- **grubby** パッケージがインストールされている。

```
$ rpm -q grubby
grubby-version
```

手順

SELinux を永続的に無効にするには、以下を実行します。

1. ブートローダーを設定して、カーネルコマンドラインに**selinux=0** を追加します。

```
$ sudo grubby --update-kernel ALL --args selinux=0
```

2. システムを再起動します。

```
$ reboot
```

検証

- 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。

  ```
  # dnf 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       *
```

限られたユーザーは、現在の SELinux ポリシーで明示的に定義されている SELinux ルールにより制限されます。制限のないユーザーには、SELinux による制限が最小限にとどまります。

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。これは、デフォルトではインストールされません。

制限のない Linux ユーザーが、**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引き続き受けます。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

SELinux ポリシーは、各 Linux ユーザーを SELinux ユーザーにマッピングします。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。

ポリシーのブール値を調整することで、SELinux ポリシーの制限ユーザーの権限を、特定のニーズに合わせてカスタマイズできます。**semanage boolean -l** コマンドを使用すると、このブール値の現在の状態を確認できます。

表3.1 SELinux ユーザーのロール

User	デフォルトロール	追加のロール
unconfined_u	**unconfined_r**	**system_r**
guest_u	**guest_r**	
xguest_u	**xguest_r**	
user_u	**user_r**	
staff_u	**staff_r**	**sysadm_r**
		unconfined_r
		system_r
sysadm_u	**sysadm_r**	
root	**staff_r**	**sysadm_r**
		unconfined_r
		system_r
system_u	**system_r**	

system_u は、システムプロセスおよびオブジェクトの特別なユーザー ID であり、**system_r** は関連付けられたロールであることに注意してください。管理者は、この **system_u** ユーザーと **system_r** ロールを 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。このため、この SELinux ユーザーに関連付けられたロールは、以下の表の SELinux ロールの種類とアクセスには含まれていません。

各 SELinux ロールは SELinux のタイプに対応しており、特定のアクセス権限を提供します。

表3.2 SELinux ロールの種類とアクセス

Role	タイプ	X Window System を使用したログイン	su および sudo	ホームディレクトリーおよび /tmp (デフォルト) での実行	ネットワーキング
unconfined_r	**unconfined_t**	はい	はい	はい	はい
guest_r	**guest_t**	いいえ	いいえ	はい	いいえ
xguest_r	**xguest_t**	はい	いいえ	はい	Web ブラウザーのみ (Firefox、GNOME Web)
user_r	**user_t**	はい	いいえ	はい	はい
staff_r	**staff_t**	はい	**sudo** のみ	はい	はい
auditadm_r	**auditadm_t**		はい	はい	はい
secadm_r	**secadm_t**		はい	はい	はい
sysadm_r	**sysadm_t**	**xdm_sysadm_login** のブール値が **on** の場合のみ	はい	はい	はい

- **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
```

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。これは、デフォルトではインストールされません。

関連情報

- **seinfo(1)**、**semanage-login(8)** および **xguest_selinux(8)** の man ページ

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されるように、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

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

- **pam_selinux(8)** の man ページ

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されるように、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

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

- **pam_selinux(8)** の man ページ

3.5. 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。

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。

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

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
   ```

3.6. 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。

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。

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. オプション:**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. オプション: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

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.
```

3.7. 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。

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。

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. オプション: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

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.
```

3.8. 関連情報

- **unconfined_selinux(8)**、**user_selinux(8)**、**staff_selinux(8)** および **sysadm_selinux(8)** の man ページ
- SELinux で制限されたユーザーでシステムを設定する方法

第4章 非標準設定でのアプリケーションとサービスの SELINUX 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。

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/test_www/html
/var/www/html       system_u:object_r:httpd_sys_content_t:s0
/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

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。

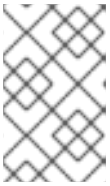
```
# semanage boolean -l | grep 'nfs\|cifs' | grep httpd
httpd_use_cifs                 (off  ,  off)  Allow httpd to access cifs file systems
httpd_use_nfs                  (off  ,  off)  Allow httpd to access nfs file systems
```

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

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

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

4.3. 関連情報

- SELinux 関連の問題のトラブルシューティング

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。
  プロセスが終了したら、フルパス監査を無効にします。

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

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

- **sealert(8)** の man ページ

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 9 の場合、 **Red Hat Enterprise Linux 9** の製品に対してバグを作成し、**selinux-policy** コンポーネントを選択します。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

- **auditd(8)** および **ausearch(8)** の man ページ

5.5. 関連情報

- CLI での基本的な SELinux トラブルシューティング
- 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 エラー」

第6章 MLS (MULTI-LEVEL SECURITY) の使用

Multi-Level Security (MLS) ポリシーは、米国のコミュニティーが最初に設計したクリアランスの **レベル** を使用します。MLS は、軍隊など、厳格に管理された環境での情報管理をベースにした、非常に限定的なセキュリティー要件を満たしています。

MLS の使用は複雑で、一般的なユースケースのシナリオには適切にマッピングされません。

6.1. MULTI-LEVEL SECURITY (MLS)

Multi-Level Security (MLS) テクノロジーは、以下の情報セキュリティーレベルを使用して、データを階層に分類します。

- [lowest] Unclassified
- [low] Confidential
- [high] Secret
- [highest] Top secret

デフォルトでは、MLS SELinux ポリシーは機密レベルを 16 個使用します。

- **s0** は、最も機密レベルが低く、
- **s15** は、最も機密レベルが高くなります。

MLS は特定の用語を使用して機密レベルに対応します。

- ユーザーおよびプロセスは **サブジェクト** と呼ばれ、その機密レベルは **クリアランス** と呼ばれます。
- システムのファイル、デバイス、およびその他のパッシブコンポーネントは **オブジェクト** と呼ばれ、その機密レベルは **区分** と呼ばれます。

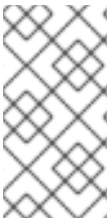
SELinux は **Bell-La Padula Model** (BLP) モデルを使用して、MLS を実装します。このモデルは、各サブジェクトおよびオブジェクトに割り当てられたラベルに基づいてシステム内で情報をフローする方法を指定します。

BLP の基本的な原則は「**No read up (NRU)、no write down (NWD)**」で、自分の機密レベルと同じかそれ以下のファイルは読み取りだけでき、データは、低いレベルから高いレベルにしか流せず、反対方向には流せません。

MLS SELinux ポリシー (RHEL 上の MLS の実装) では、**書き込み等価を使用した Bell-La Padula** と呼ばれる修正済みの原則を適用します。つまり、ユーザーは自分の機密レベル以下のファイルを読み取ることができますが、書き込みは自分のレベルだけにしか不可能です。これにより、クリアランスが低いユーザーがコンテンツを Top secret ファイルに書き込めないようにします。

たとえば、デフォルトではクリアランスレベル **s2** を持つユーザーには以下が適用されます。

- 機密レベルが **s0**、**s1**、および **s2** のファイルを読み取ることができます。
- 機密レベルが **s3** 以上のファイルを読み取ることはできません。
- 機密レベルが **s2** のファイルを変更できます。
- 機密レベルが **s2** 以外のファイルは変更できません。



注記

セキュリティー管理者は、システムの SELinux ポリシーを変更することで、この動作を調整できます。たとえば、ユーザーがより低いレベルでファイルを変更できるようにすることで、ファイルの機密レベルをユーザーのクリアランスレベルまで引き上げることができます。

実際には、ユーザーは通常、**s1-s2** などの範囲のクリアランスレベルに割り当てられます。ユーザーは、自分の最大レベルよりも低い機密レベルのファイルの読み取りと、その範囲内の任意のファイルへの書き込みが可能です。

たとえば、デフォルトではクリアランス範囲が **s1-s2** のユーザーには以下が適用されます。

- 機密レベル **s0** と **s1** のファイルを読み取ることができます。
- 機密レベル **s2** 以上のファイルを読み取ることはできません。
- 機密レベル **s1** のファイルを変更できます。
- 機密レベルが **s1** 以外のファイルは変更できません。
- 自分のクリアランスレベルを **s2** に変更できます。

MLS 環境における非特権ユーザーのセキュリティーコンテキストは次のとおりです。

```
user_u:user_r:user_t:s1
```

詳細は以下のようになります。

user_u
SELinux ユーザーです。

user_r
SELinux ロールです。

user_t
SELinux のタイプです。

s1
MLS 機密レベルの範囲です。

システムは、MLS アクセス制御ルールと従来のファイルアクセスパーミッションを常に組み合わせます。たとえば、セキュリティーレベルが「Secret」のユーザーが、DAC (Discretionary Access Control) を使用して他のユーザーによるファイルへのアクセスをブロックした場合に、「Top Secret」のユーザーはそのファイルにアクセスできなくなります。セキュリティーのクリアランスが高くても、ユーザーはファイルシステム全体を自動的に閲覧できません。

トップレベルの許可があるユーザーは、マルチレベルのシステムで自動的に管理者権限を取得しません。システム上のすべての機密情報にアクセスできる場合もありますが、管理者権限を持っている場合とも異なります。

さらに、管理者権限があっても機密情報にアクセスできるわけではありません。たとえば、**root** としてログインした場合でも、Top secret 情報を読み込めません。

カテゴリーを使用して MLS システム内でアクセスをさらに調整できます。Multi-Category Security (MCS) を使用すると、プロジェクトや部門などのカテゴリーを定義でき、ユーザーは割り当てられたカテゴリーのファイルにしかアクセスできません。詳細は、Using Multi-Category Security (MCS) for

data confidentiality を参照してください。

6.2. MLS の SELINUX ロール

SELinux ポリシーは、各 Linux ユーザーを SELinux ユーザーにマッピングします。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。

重要

MLS ポリシーには、制限なしのユーザー、タイプおよびロールなど、**Unconfined** モジュールは含まれません。そのため、**root** など制限のないユーザーは、すべてのオブジェクトにアクセスして、ターゲットポリシーで実行可能なアクションをすべて実行できるわけではありません。

ポリシーのブール値を調整することで、SELinux ポリシーの制限ユーザーの権限を、特定のニーズに合わせてカスタマイズできます。**semanage boolean -l** コマンドを使用すると、このブール値の現在の状態を確認できます。

表6.1 MLS での SELinux ユーザーのロール

User	デフォルトロール	追加のロール
guest_u	**guest_r**	
xguest_u	**xguest_r**	
user_u	**user_r**	
staff_u	**staff_r**	**auditadm_r**
		secadm_r
		sysadm_r
		staff_r
sysadm_u	**sysadm_r**	
root	**staff_r**	**auditadm_r**
		secadm_r
		sysadm_r
		system_r
system_u	**system_r**	

system_u は、システムプロセスおよびオブジェクトの特別なユーザー ID であり、**system_r** は関連付

けられたロールであることに注意してください。管理者は、この **system_u** ユーザーと **system_r** ロールを 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。このため、この SELinux ユーザーに関連付けられたロールは、以下の表の SELinux ロールの種類とアクセスには含まれていません。

各 SELinux ロールは SELinux のタイプに対応しており、特定のアクセス権限を提供します。

表6.2 MLS での SELinux ロールのタイプおよびアクセス

Role	タイプ	X Window System を使用したログイン	**su** および **sudo**	ホームディレクトリーおよび **/tmp** (デフォルト) での実行	ネットワーキング
guest_r	**guest_t**	いいえ	いいえ	はい	いいえ
xguest_r	**xguest_t**	はい	いいえ	はい	Web ブラウザーのみ (Firefox、GNOME Web)
user_r	**user_t**	はい	いいえ	はい	はい
staff_r	**staff_t**	はい	**sudo** のみ	はい	はい
auditadm_r	**auditadm_t**		はい	はい	はい
secadm_r	**secadm_t**		はい	はい	はい
sysadm_r	**sysadm_t**	**xdm_sysadm_login** のブール値が **on** の場合のみ	はい	はい	はい

- デフォルトでは、**sysadm_r** ロールには **secadm_r** ロールの権限があります。つまり、**sysadm_r** ロールを持つユーザーは、セキュリティーポリシーを管理できることを意味します。ユースケースが一致しない場合は、ポリシーの **sysadm_secadm** を無効にすることで、2 つのロールを分離できます。詳細は、Separating system administration from security administration in MLS を参照してください。

- ログイン以外のロールの **dbadm_r**、**logadm_r**、および **webadm_r** は、管理タスクのサブセットに使用できます。デフォルトでは、これらのロールは SELinux ユーザーに関連付けられていません。

6.3. SELINUX ポリシーの MLS への切り替え

SELinux ポリシーをターゲットの MLS (Multi-Level Security) から Multi-Level Security (MLS) に切り替えるには、以下の手順に従います。

重要

Red Hat は、X Window System を実行しているシステムで MLS ポリシーを使用することは推奨していません。さらに、MLS ラベルでファイルシステムのラベルを変更すると、制限のあるドメインにアクセスできなくなる可能性があるため、システムが正常に起動しなくなる可能性があります。したがって、ファイルに再ラベルする前に、SELinux を Permissive モードに切り替えます。多くのシステムでは、MLS に移動後に SELinux の拒否が多く表示され、そのほとんどは修正するだけでは限りません。

手順

1. **selinux-policy-mls** パッケージをインストールします。

```
# dnf install selinux-policy-mls
```

2. 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
```

3. SELinux モードを Enforcing から Permissive に変更し、ターゲットポリシーから MLS に切り替えます。

```
SELINUX=permissive
SELINUXTYPE=mls
```

変更を保存し、エディターを終了します。

4. MLS ポリシーを有効にする前に、MLS ラベルでファイルシステム上の各ファイルに再度ラベル付けする必要があります。

```
# fixfiles -F onboot
System will relabel on next boot
```

5. システムを再起動します。

```
# reboot
```

6. SELinux 拒否を確認します。

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

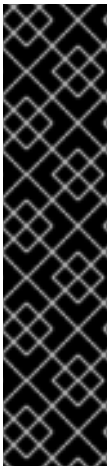
上記のコマンドはすべてのシナリオに対応していないため、SELinux 拒否の識別、分析、修正のガイダンスは「SELinux 関連の問題のトラブルシューティング」を参照してください。

7. システムに SELinux に関連する問題がないことを確認したら、**/etc/selinux/config** で該当するオプションを変更して、SELinux を Enforcing モードに切り替えます。

```
SELINUX=enforcing
```

8. システムを再起動します。

```
# reboot
```



重要

システムが起動しなかったり、MLS に切り替えた後にログインできない場合は、**enforcing=0** パラメーターをカーネルコマンドラインに追加します。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。

また、MLS では、**sysadm_r** SELinux ロールにマッピングされた **root** ユーザーとしての SSH ログインは**staff_r** で **root** としてログインするのとは異なります。MLS で初めてシステムを起動する前に、SELinux ブール値 **ssh_sysadm_login** を **1** に設定して、SSH ログインを **sysadm_r** として許可することを検討してください。後で すでに MLS に存在する **ssh_sysadm_login** を有効にするには、既に MLS にいる場合、**root** として **staff_r** ログインし、**newrole -r sysadm_r** コマンドを使用して **sysadm_r** で **root** に切り替えてから、ブール値を **1**に設定します。

検証

1. SELinux が Enforcing モードで実行されていることを確認します。

```
# getenforce
Enforcing
```

2. SELinux のステータスが **mls** の値を返すことを確認します。

```
# sestatus | grep mls
Loaded policy name:             mls
```

関連情報

- **fixfiles(8)**、**setsebool(8)**、および **ssh_selinux(8)** の man ページ。

6.4. MLS でのユーザークリアランスの確立

SELinux ポリシーを MLS に切り替えた後に、制限のある SELinux ユーザーにマッピングすることで、セキュリティークリアランスレベルをユーザーに割り当てる必要があります。デフォルトでは、セキュリティークリアランスが指定されているユーザーには以下が適用されます。

- 機密レベルが高いオブジェクトを読み取ることはできません。
- 機密レベルが異なるオブジェクトに書き込むことはできません。

前提条件

- SELinux ポリシーが **mls** に設定されている。
- SELinux モードが **Enforcing** に設定されている。
- **policycoreutils-python-utils** パッケージがインストールされている。
- SELinux の制限のあるユーザーに割り当てられているユーザー:
  - 非特権ユーザーの場合、**user_u** (以下の手順では **example_user**) に割り当てられます。
  - 特権ユーザーの場合、**staff_u** (以下の手順では **staff**) に割り当てられます。

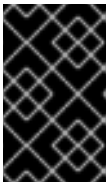
注記

MLS ポリシーがアクティブな時に、ユーザーが作成されていることを確認します。他の SELinux ポリシーで作成されたユーザーは MLS で使用できません。

手順

1. オプション:SELinux ポリシーにエラーを追加しないようにするには、**Permissive** SELinux モードに切り替えてください。切り替えることでトラブルシューティングが容易になります。

```
# setenforce 0
```



重要

Permissive モードでは、SELinux はアクティブなポリシーを有効せず、アクセスベクターキャッシュ (AVC) メッセージをログに記録するだけです。このログは、トラブルシューティングやデバッグに使用できます。

2. SELinux ユーザー **staff_u** のクリアランスの範囲を定義します。たとえば、このコマンドは、クリアランスの範囲を **s1** から **s15** に、デフォルトのクリアランスレベルを **s1** に設定します。

```
# semanage user -m -L s1 -r s1-s15 _staff_u
```

3. ユーザーのホームディレクトリー用の SELinux ファイルコンテキスト設定エントリーを生成します。

```
# genhomedircon
```

4. ファイルセキュリティーコンテキストをデフォルトに復元します。

```
# restorecon -R -F -v /home/
Relabeled /home/staff from staff_u:object_r:user_home_dir_t:s0 to
staff_u:object_r:user_home_dir_t:s1
Relabeled /home/staff/.bash_logout from staff_u:object_r:user_home_t:s0 to
staff_u:object_r:user_home_t:s1
Relabeled /home/staff/.bash_profile from staff_u:object_r:user_home_t:s0 to
staff_u:object_r:user_home_t:s1
Relabeled /home/staff/.bashrc from staff_u:object_r:user_home_t:s0 to
staff_u:object_r:user_home_t:s1
```

5. ユーザーにクリアランスレベルを割り当てます。

```
# semanage login -m -r s1 example_user
```

ここで、**s1** は、ユーザーに割り当てられたクリアランスレベルに置き換えます。

6. ユーザーのホームディレクトリーのラベルをユーザーのクリアランスレベルに付け直します。

```
# chcon -R -l s1 /home/example_user
```

7. オプション:SELinux モードを **Permissive** に切り替えた場合は、すべてが想定通りに動作することを確認したら、SELinux モードを **Enforcing** モードに戻します。

```
# setenforce 1
```

検証手順

1. ユーザーが正しい SELinux ユーザーにマッピングされ、クリアランスレベルが正しく割り当てら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-s0                *
example_user    user_u               s1                   *
...
```

2. MLS 内のユーザーとしてログインします。

3. ユーザーのセキュリティーレベルが正しく機能していることを確認します。

重要

検証に使用するファイルには、設定が間違っており、ユーザーが実際に認証なしにファイルにアクセスできてしまう場合に備え、機密情報を含めないようにしてください。

   a. ユーザーが機密レベルの高いファイルを読み取れないことを確認します。

   b. ユーザーが同じ機密レベルのファイルに書き込めることを確認します。

   c. ユーザーが機密レベルの低いファイルを読み取れることを確認します。

関連情報

- 「SELinux ポリシーの MLS への切り替え」
- 「新規ユーザーを SELinux で制限されたユーザーとして追加」
- *2章SELinux のステータスおよびモードの変更*
- *5章SELinux 関連の問題のトラブルシューティング*
- ナレッジベースの記事「Basic SELinux Troubleshooting in CLI」

6.5. MLS で定義されたセキュリティー範囲内におけるユーザーのクリアランスレベルの変更

Multi-Level Security (MLS) のユーザーとして、管理者が割り当てた範囲内で現在のクリアランスレベルを変更できます。範囲の上限を超えたり、範囲の下限を超えてレベルを下げることはできません。これにより、たとえば、機密レベルを最高のクリアランスレベルまで上げることなく、機密性の低いファイルを変更できます。

たとえば、**s1-s3** の範囲に割り当てられたユーザーには以下が適用されます。

- レベル **s1**、**s2**、および **s3** に移行できます。
- **s1-s2** および **s2-s3** の範囲に切り換えることができます。

- **s0-s3** または**s1-s4** の範囲に切り替えることはできません。

注記

別のレベルに切り替えると、異なるクリアランスで新しいシェルが開きます。つまり、下げる場合と同じ方法で元のクリアランスレベルに戻すことはできません。ただし、**exit** を入力すると、いつでも前のシェルに戻ることができます。

前提条件

- SELinux ポリシーが **mls** に設定されている。
- SELinux モードが **enforcing** に設定されている。
- MLS クリアランスレベルの範囲に割り当てられたユーザーとしてログインできます。

手順

1. セキュアな端末からユーザーとしてログインします。

注記

セキュアな端末は、、**/etc/selinux/mls/contexts/securetty_types** ファイルで定義されています。デフォルトでは、コンソールはセキュアな端末ですが、SSH はセキュアではありません。

2. 現在のユーザーのセキュリティーコンテキストを確認します。

```
$ id -Z
user_u:user_r:user_t:s0-s2
```

この例では、ユーザーは **user_u** SELinux ユーザー、**user_r** ロール、**user_t** タイプ、**s0-s2** の MLS セキュリティーファイルに割り当てられます。

3. 現在のユーザーのセキュリティーコンテキストを確認します。

```
$ id -Z
user_u:user_r:user_t:s1-s2
```

4. ユーザーのクリアランス範囲内で別のセキュリティークリアランス範囲に切り替えます。

```
$ newrole -l s1
```

最大が割り当てられた範囲以下の範囲に切り替えることができます。単一レベルの範囲を入力すると、割り当てられた範囲の下限が変更されます。たとえば、範囲が **s0-s2** のユーザーとして **newrole -l s1** を入力することは、**newrole -l s1-s2** を入力することに相当します。

検証

1. 現在のユーザーのセキュリティーコンテキストを確認します。

```
$ id -Z
user_u:user_r:user_t:s1-s2
```

2. 現在のシェルを終了し、元の範囲で前のシェルに戻ります。

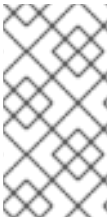
```
$ exit
```

関連情報

- using-multi-level-security-mls_using-selinux.xml
- **newrole(1)** の man page
- **securetty_types(5)** man page

6.6. MLS におけるファイル機密レベルの引き上げ

デフォルトでは、MLS (Multi-Level Security) ユーザーはファイル機密レベルを高くすることはできません。ただし、セキュリティー管理者 (**secadm_r**) は、システムの SELinux ポリシーにローカルモジュール **mlsfilewrite** を追加することで、デフォルトの動作を変更し、ユーザーによるファイルの機密レベルの引き上げを許可できます。次に、ポリシーモジュールで定義された SELinux タイプに割り当てられたユーザーは、ファイルを変更することで、ファイルの分類レベルを引き上げることができます。ユーザーがファイルを変更すると、ユーザーの現在のセキュリティー範囲の下限値までファイルの機密性レベルが引き上げられます。



注記

セキュリティー管理者は、**secadm_r** ロールに割り当てられたユーザーとしてログインすると、 **chcon -l s0 /path/to/file** コマンドを使用してファイルのセキュリティーレベルを変更できます。詳細は、「MLS でのファイル機密レベルの変更」 を参照してください。

前提条件

- SELinux ポリシーが **mls** に設定されている。
- SELinux モードが **Enforcing** に設定されている。
- **policycoreutils-python-utils** パッケージがインストールされている。
- **mlsfilewrite** ローカルモジュールが SELinux MLS ポリシーにインストールされている。
- MLS のユーザーとしてログインしている。つまり:
  - 定義済みのセキュリティー範囲に割り当てられている。以下の例は、セキュリティー範囲が **s0-s2** のユーザーを示しています。
  - **mlsfilewrite** モジュールで定義されているのと同じ SELinux タイプに割り当てられている。この例では、**(typeattributeset mlsfilewrite (user_t))** モジュールが必要です。

手順

1. オプション:現在のユーザーのセキュリティーコンテキストを表示します。

```
$ id -Z
user_u:user_r:user_t:s0-s2
```

2. ユーザーの MLS クリアランス範囲の下位レベルを、ファイルに割り当てるレベルに変更します。

```
$ newrole -l s1-s2
```

3. オプション:現在のユーザーのセキュリティーコンテキストを表示します。

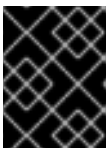
```
$ id -Z
user_u:user_r:user_t:s1-s2
```

4. オプション:ファイルのセキュリティーコンテキストを表示します。

```
$ ls -Z /path/to/file
user_u:object_r:user_home_t:s0 /path/to/file
```

5. ファイルを変更して、ファイルの機密レベルをユーザーのクリアランス範囲の下位レベルに変更します。

```
$ touch /path/to/file
```



重要

システムで **restorecon** コマンドを使用すると、分類レベルはデフォルト値に戻ります。

6. オプション:シェルを終了して、ユーザーの以前のセキュリティー範囲に戻ります。

```
$ exit
```

検証

- ファイルのセキュリティーコンテキストを表示します。

```
$ ls -Z /path/to/file
user_u:object_r:user_home_t:s1 /path/to/file
```

関連情報

- 「MLS ユーザーによる下位レベルのファイルの編集を許可」.

6.7. MLS でのファイル機密レベルの変更

MLS SELinux ポリシーでは、ユーザーは自分の機密レベルのファイルしか変更できません。これは、クリアランスレベルが低いユーザーに極秘情報が公開されないようにすること、またクリアランスレベルの低いユーザーが機密レベルの高いドキュメントを作成できないようにすることが目的です。ただし、管理者は、ファイルをより高いレベルで処理するなど、ファイル区分を手動で増やすことができます。

前提条件

- SELinux ポリシーが **mls** に設定されている。
- SELinux モードが Enforcing に設定されている。

- セキュリティー管理者権限が割り当てられている。以下のいずれかに割り当てます。
  - **secadm_r** ロール。
  - **sysadm_secadm** モジュールが有効になっている場合は、**sysadm_r** ロール。**sysadm_secadm** モジュールはデフォルトで有効になっています。
- **policycoreutils-python-utils** パッケージがインストールされている。
- クリアランスレベルが割り当てられているユーザー。詳細は、Establishing user clearance levels in MLS を参照してください。
  この例では、**User1** のクリアランスレベルは **s1** です。
- 区分レベルが割り当てられ、アクセスできるファイル。
  この例では、**/path/to/file** の区分レベルは **s1** です。

手順

1. ファイルの区分レベルを確認します。

```
# ls -lZ /path/to/file
-rw-r-----. 1 User1 User1 user_u:object_r:user_home_t:s1 0 12. Feb 10:43 /path/to/file
```

2. ファイルのデフォルトの区分レベルを変更します。

```
# semanage fcontext -a -r s2 /path/to/file
```

3. ファイルの SELinux コンテキストのラベルを強制的につけ直します。

```
# restorecon -F -v /path/to/file
Relabeled /path/to/file from user_u:object_r:user_home_t:s1 to
user_u:object_r:user_home_t:s2
```

検証

1. ファイルの区分レベルを確認します。

```
# ls -lZ /path/to/file
-rw-r-----. 1 User1 User1 user_u:object_r:user_home_t:s2 0 12. Feb 10:53 /path/to/file
```

2. オプション:クリアランスが低いユーザーがファイルを読み込めないことを確認します。

```
$ cat /path/to/file
cat: file: Permission denied
```

関連情報

- 「MLS でのユーザークリアランスの確立」

6.8. SEPARATING SYSTEM ADMINISTRATION FROM SECURITY ADMINISTRATION IN MLS

デフォルトでは、**sysadm_r** ロールには **secadm_r** ロールの権限があります。つまり、**sysadm_r** ロー

ルを持つユーザーは、セキュリティーポリシーを管理できることを意味します。セキュリティー認証の制御を強化する必要がある場合は、Linux ユーザーを **secadm_r** ロールに割り当て、SELinux ポリシーの**sysadm_secadm** モジュールを無効にすることで、システム管理をセキュリティー管理から分離することができます。

前提条件

- SELinux ポリシーが **mls** に設定されている。
- SELinux モードが **Enforcing** に設定されている。
- **policycoreutils-python-utils** パッケージがインストールされている。
- **secadm_r** ロールに割り当てられる Linux ユーザー。
  - ユーザーは、**staff_u** SELinux ユーザーに割り当てられます。
  - このユーザーのパスワードが定義されています。

警告

secadm ロールに割り当てられるユーザーでログインできることを確認してください。そうでない場合は、システムの SELinux ポリシーの将来の変更を防ぐことができます。

手順

1. ユーザー向けに、新しい **sudoers** ファイルを **/etc/sudoers.d** ディレクトリーに作成します。

```
# visudo -f /etc/sudoers.d/<sec_adm_user>
```

sudoers ファイルを整理しておくには、**<sec_adm_user>** を **secadm** ロールに割り当てられる Linux ユーザーに置き換えます。

2. **/etc/sudoers.d/<sec_adm_user>** ファイルに、以下の内容を追加します。

```
<sec_adm_user> ALL=(ALL) TYPE=secadm_t ROLE=secadm_r ALL
```

この行は、すべてのホスト上の **<secadmuser>** が、すべてのコマンドを実行することを許可し、デフォルトでユーザーを **secadm** SELinux タイプとロールにマップします。

3. <sec_adm_user> ユーザーでログインします。

注記

SELinux のコンテキスト (SELinux のユーザー、ロール、タイプで構成) が変更されていることを確認するために、**ssh**、コンソール、または **xdm** を使用してログインします。**su** および **sudo** などの他の方法では、SELinux コンテキスト全体を変更することはできません。

4. ユーザーのセキュリティーコンテキストを確認します。

```
$ id
uid=1000(<sec_adm_user>) gid=1000(<sec_adm_user>) groups=1000(<sec_adm_user>)
context=staff_u:staff_r:staff_t:s0-s15:c0.c1023
```

5. root ユーザーの対話型シェルを実行します。

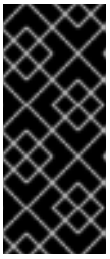
```
$ sudo -i
[sudo] password for <sec_adm_user>:
```

6. 現在のユーザーのセキュリティーコンテキストを確認します。

```
# id
uid=0(root) gid=0(root) groups=0(root) context=staff_u:secadm_r:secadm_t:s0-s15:c0.c1023
```

7. ポリシーから **sysadm_secadm** モジュールを無効にします。

```
# semodule -d sysadm_secadm
```



重要

semodule -r コマンドを使用してシステムポリシーモジュールを削除する代わりに、**semodule -d** コマンドを使用します。**semodule -r** コマンドは、システムのストレージからモジュールを削除します。これは、**selinux-policy-mls** パッケージを再インストールしないと、モジュールを再び読み込むことができないことを意味します。

検証

1. **secadm** ロールに割り当てられたユーザーとして、root ユーザーの対話型シェルで、セキュリティーポリシーデータにアクセスできることを確認します。

```
# seinfo -xt secadm_t

Types: 1
   type secadm_t, can_relabelto_shadow_passwords, (...) userdomain;
```

2. root シェルからログアウトします。

```
# logout
```

3. **<sec_adm_user>** ユーザーからログアウトします。

```
$ logout
Connection to localhost closed.
```

4. 現在のセキュリティーコンテキストを表示します。

```
# id
uid=0(root) gid=0(root) groups=0(root) context=root:sysadm_r:sysadm_t:s0-s15:c0.c1023
```

5. **sysadm_secadm** モジュールの有効化を試してください。コマンドは失敗するはずです。

```
# semodule -e sysadm_secadm
SELinux:  Could not load policy file /etc/selinux/mls/policy/policy.31:  Permission denied
/sbin/load_policy:  Can't load policy:  Permission denied
libsemanage.semanage_reload_policy: load_policy returned error code 2. (No such file or
directory).
SELinux:  Could not load policy file /etc/selinux/mls/policy/policy.31:  Permission denied
/sbin/load_policy:  Can't load policy:  Permission denied
libsemanage.semanage_reload_policy: load_policy returned error code 2. (No such file or
directory).
semodule:  Failed!
```

6. **sysadm_t** SELinux タイプに関する詳細の表示を試してください。コマンドは失敗するはずです。

```
# seinfo -xt sysadm_t
[Errno 13] Permission denied: '/sys/fs/selinux/policy'
```

6.9. MLS でのセキュアな端末の定義

SELinux ポリシーは、ユーザーが接続している端末のタイプをチェックし、特定の SELinux アプリケーション (**newrole** など) の実行を安全な端末からのみ許可します。セキュアではない端末からこれを試みると、エラーが発生します。**Error: you are not allowed to change levels on a non secure terminal;**.

/etc/selinux/mls/contexts/securetty_types ファイルは、MLS (Multi-Level Security) ポリシーのセキュアな端末を定義します。

ファイルのデフォルトコンテンツ:

```
console_device_t
sysadm_tty_device_t
user_tty_device_t
staff_tty_device_t
auditadm_tty_device_t
secureadm_tty_device_t
```

警告

端末タイプをセキュアな端末一覧に追加すると、システムがセキュリティーリスクにさらされる可能性があります。

前提条件

- SELinux ポリシーが **mls** に設定されている。
- すでにセキュアな端末から接続しているか、SELinux が Permissive モードである。

- セキュリティー管理者権限が割り当てられている。以下のいずれかに割り当てます。
  - **secadm_r** ロール。
  - **sysadm_secadm** モジュールが有効になっている場合は、**sysadm_r** ロール。**sysadm_secadm** モジュールはデフォルトで有効になっています。
- **policycoreutils-python-utils** パッケージがインストールされている。

手順

1. 現在の端末タイプを確認します。

```
# ls -Z `tty`
root:object_r:user_devpts_t:s0 /dev/pts/0
```

この出力例では、**user_devpts_t** が現在の端末タイプです。

2. **/etc/selinux/mls/contexts/securetty_types** ファイルの新しい行に、関連する SELinux タイプを追加します。
3. オプション:SELinux を Enforcing モードに切り替えます。

```
# setenforce 1
```

検証

- **/etc/selinux/mls/contexts/securetty_types** ファイルに追加した、以前はセキュアでなかった端末からログインします。

関連情報

- **securetty_types(5)** man page

6.10. MLS ユーザーによる下位レベルのファイルの編集を許可

デフォルトでは、MLS ユーザーは、クリアランス範囲の下限値を下回る機密レベルのファイルに書き込むことはできません。シナリオで、ユーザーによる下位レベルのファイルの編集を許可する必要がある場合は、ローカル SELinux モジュールを作成することで実行できます。ただし、ファイルへの書き込みにより、機密レベルがユーザーの現在の範囲の下限まで上昇します増加します。

前提条件

- SELinux ポリシーが **mls** に設定されている。
- SELinux モードが **Enforcing** に設定されている。
- **policycoreutils-python-utils** パッケージがインストールされている。
- 検証用の **setools-console** パッケージおよび **audit** パッケージ。

手順

1. オプション:トラブルシューティングを実施しやすくするために、Permissive モードに切り替えます。

```
# setenforce 0
```

2. ~/**local_mlsfilewrite.cil** などのテキストエディターで新しい **.cil** ファイルを開き、以下のカスタムルールを挿入します。

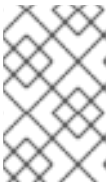
```
(typeattributeset mlsfilewrite (_staff_t_))
```

staff_t は、別の SELinux タイプに置き換えることができます。ここで SELinux タイプを指定することで、下位レベルのファイルを編集できる SELinux ロールを制御できます。

ローカルモジュールをより適切に整理するには、ローカル SELinux ポリシーモジュール名で接頭辞 **local_** を使用します。

3. ポリシーモジュールをインストールします。

```
# semodule -i ~/local_mlsfilewrite.cil
```



注記

ローカルポリシーモジュールを削除するには、**semodule -r ~/local_mlsfilewrite** を使用します。モジュール名を参照する場合は、接頭辞 **.cil** なしで参照する必要がある点に注意してください。

4. オプション:以前に permissive モードに戻した場合は、enforcing モードに戻ります。

```
# setenforce 1
```

検証

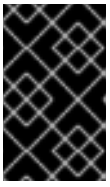
1. インストールされている SELinux モジュールの一覧でローカルモジュールを検索します。

```
# semodule -lfull | grep "local_mls"
400 local_mlsfilewrite  cil
```

ローカルモジュールの優先度は **400** であるため、**semodule -lfull | grep -v ^100** コマンドを使用して一覧表示することもできます。

2. カスタムルールで定義されているタイプ (**staff_t** など) に割り当てられたユーザーとしてログインします。

3. 機密レベルが低いファイルに書き込みを試みます。これにより、ファイルの区分レベルがユーザーのクリアランスレベルまで上昇します。



重要

検証に使用するファイルには、設定が間違っており、ユーザーが実際に認証なしにファイルにアクセスできてしまう場合に備え、機密情報を含めないようにしてください。

第7章 データの機密性に MCS (MULTI-CATEGORY SECURITY) を使用する

MCS を使用すると、データのカテゴリーを分類し、特定のプロセスやユーザーに特定のカテゴリーへのアクセスを許可することで、システムのデータの機密性を高めることができます。

7.1. MULTI-CATEGORY SECURITY (MCS)

MCS (Multi-Category Security) は、プロセスおよびファイルに割り当てられたカテゴリーを使用するアクセス制御メカニズムです。その後、同じカテゴリーに割り当てられているプロセスのみがファイルにアクセスできます。MCS の目的は、システムでデータの機密性を維持することです。

MCS カテゴリーは、**c0** から **c1023** までの値で定義されますが、“Personnel”、“ProjectX”、または “ProjectX.Personnel” のように、カテゴリーごと、またはカテゴリーの組み合わせに対して、テキストラベルを定義することもできます。その後、**mcstrans** (MCS Translation Service) では、カテゴリー値を、システムの入出力内の適切なラベルに置き換え、カテゴリー値の代わりにこれらのラベルを使用できるようにします。

ユーザーは、カテゴリーに割り当てられているときに、割り当てられているカテゴリーのいずれかでファイルにラベルを付けることができます。

MCS は単純な原則に基づいて動作します。ファイルにアクセスするには、ファイルに割り当てられたすべてのカテゴリーにユーザーを割り当てる必要があります。MCS チェックは、通常の Linux DAC (Discretionary Access Control) ルールおよび SELinux TE (Type Enforcement) ルールの後に適用されるため、既存のセキュリティー設定をさらに制限する必要があります。

Multi-Level Security 内の MCS

MCS は、単独で非階層システムとして使用することも、マルチレベルセキュリティー (MLS) と組み合わせて、階層システム内の非階層レイヤーとして使用することもできます。

MLS 内の MCS の例を以下に示します。ファイルが以下のように分類されます。

表7.1 セキュリティーレベルとカテゴリーの組み合わせの例

セキュリティーレベル	カテゴリー			
	未指定	プロジェクト X	プロジェクト Y	プロジェクト Z
Unclassified	**s0**	**s0:c0**	**s0:c1**	**s0:c2**
Confidential	**s1**	**s1:c0**	**s1:c1**	**s1:c2**
Secret	**s2**	**s2:c0**	**s2:c1**	**s2:c2**
Top secret	**s3**	**s3:c0**	**s3:c1**	**s3:c2**

注記

範囲が **s0: c0.1023**のユーザーは、DAC や Type Enforcement ポリシールールなどの他のセキュリティーメカニズムでアクセスが禁止されていない限り、レベル**s0** のすべてのカテゴリーに割り当てられたすべてのファイルにアクセスできます。

ファイルまたはプロセスのセキュリティーコンテキストは、以下の組み合わせになります。

- SELinux ユーザー
- SELinux ロール
- SELinux タイプ
- MLS 機密レベル
- MCS カテゴリー

たとえば、MLS/MCS 環境で、機密レベル 1 およびカテゴリー 2 にアクセスできる非特権ユーザーには、以下の SELinux コンテキストを持つことができます。

```
user_u:user_r:user_t:s1:c2
```

関連情報

- Using Multi-Level Security (MLS) .

7.2. データの機密性にマルチカテゴリーセキュリティーを設定

デフォルトでは、MCS は **targeted** ポリシーおよび **mls** SELinux ポリシーでは有効ですが、ユーザーには設定されません。**targeted** ポリシーでは、MCS は以下の場合にのみ設定されます。

- OpenShift
- virt
- sandbox
- ネットワークラベリング
- コンテナー (**container-selinux**)

Type Enforcement に加え、MCS ルールで **user_t** SELinux タイプを制約するルールを使用して、ローカルの SELinux モジュールを作成することにより、ユーザーを分類するように MCS を設定できます。

警告

特定のファイルのカテゴリーを変更すると、一部のサービスが稼働しなくなる場合があります。専門家でない場合は、Red Hat の営業担当者に連絡し、コンサルティングサービスを依頼してください。

前提条件

- SELinux モードが **Enforcing** に設定されている。
- SELinux のポリシーは **targeted** または **mls** に設定されている。

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

手順

1. 新しいファイルを作成します (例: **local_mcs_user.cil**)。

```
# vim local_mcs_user.cil
```

2. 以下のルールを挿入します。

```
(typeattributeset mcs_constrained_type (user_t))
```

3. ポリシーモジュールをインストールします。

```
# semodule -i local_mcs_user.cil
```

検証

- 各ユーザードメインに、すべてのコンポーネントの詳細を表示します。

```
# seinfo -xt user_t

Types: 1
type user_t, application_domain_type, nsswitch_domain, corenet_unlabeled_type, domain,
kernel_system_state_reader, mcs_constrained_type, netlabel_peer_type, privfd,
process_user_target, scsi_generic_read, scsi_generic_write, syslog_client_type,
pcmcia_typeattr_1, user_usertype, login_userdomain, userdomain, unpriv_userdomain,
userdom_home_reader_type, userdom_filetrans_type, xdmhomewriter, x_userdomain,
x_domain, dridomain, xdrawable_type, xcolormap_type;
```

関連情報

- Creating a local SELinux policy module
- コンテナーのコンテキストでの MCS の詳細については、ブログ記事 How SELinux separates containers using Multi-Level Security および Why you should be using Multi-Category Security for your Linux containers を参照してください。

7.3. MCS でのカテゴリーラベルの定義

setrans.conf ファイルを編集することで、MCS カテゴリーのラベル、または MCS カテゴリーと MLS レベルの組み合わせをシステムで管理および維持できます。このファイルでは、SELinux は、内部の機密レベルとカテゴリーレベル、および人間が判読できるラベルとの間でマッピングを維持しています。

注記

カテゴリーラベルは、ユーザーがカテゴリーを使いやすくするためだけのものです。MCS は、ラベルを定義するかどうかに関係なく同じように機能します。

前提条件

- SELinux モードが **Enforcing** に設定されている。

- SELinux のポリシーは **targeted** または **mls** に設定されている。
- **policycoreutils-python-utils** パッケージおよび **mcstrans** パッケージがインストールされている。

手順

1. テキストエディターで **/etc/selinux/<selinuxpolicy>/setrans.conf** ファイルを編集して、既存のカテゴリーを修正したり、カテゴリーを新たに作成したりできます。お使いの SELinux ポリシーに応じて、**<selinuxpolicy>** を **targeted** または **mls** に置き換えます。以下に例を示します。

```
# vi /etc/selinux/targeted/setrans.conf
```

2. ポリシーの **setrans.conf** ファイルで、構文 **s_<security level>_:c_<category number>_=<category.name>** を使用して、シナリオで必要なカテゴリーの組み合わせを定義します。以下に例を示します。

```
s0:c0=Marketing
s0:c1=Finance
s0:c2=Payroll
s0:c3=Personnel
```

   - カテゴリー番号は、**c0** から **c1023** まで使用できます。
   - **targeted** ポリシーでは、**s0** セキュリティーレベルを使用します。
   - **mls** ポリシーでは、機密レベルとカテゴリーの組み合わせにラベルを付けることができます。

3. オプション:**setrans.conf** ファイル内で、MLS 機密レベルにラベルを付けることもできます。
4. ファイルを保存し、終了します。
5. 変更を有効にするには、MCS 変換サービスを再起動します。

```
# systemctl restart mcstrans
```

検証

- 現在のカテゴリーを表示します。

```
# chcat -L
```

上の例の出力は、以下となります。

```
s0:c0                Marketing
s0:c1                Finance
s0:c2                Payroll
s0:c3                Personnel
s0
s0-s0:c0.c1023               SystemLow-SystemHigh
s0:c0.c1023                  SystemHigh
```

関連情報

- **setrans.conf(5)** の man ページ

7.4. MCS でのユーザーへのカテゴリーの割り当て

Linux ユーザーにカテゴリーを割り当てることで、ユーザー認証を定義できます。カテゴリーが割り当てられているユーザーは、ユーザーのカテゴリーのサブセットがあるファイルにアクセスして修正できます。また、ユーザーは自分が所有するファイルを、割り当てられたカテゴリーに割り当てることもできます。

Linux ユーザーは、関連する SELinux ユーザーに定義されたセキュリティー範囲外のカテゴリーに割り当てることはできません。

注記

カテゴリーアクセスは、ログイン時に割り当てられます。そのため、ユーザーは再度ログインするまで、新しく割り当てられたカテゴリーにアクセスできません。同様に、カテゴリーへのユーザーのアクセスを取り消した場合は、ユーザーが再度ログインしないと有効になりません。

前提条件

- SELinux モードが **Enforcing** に設定されている。
- SELinux のポリシーは **targeted** または **mls** に設定されている。
- **policycoreutils-python-utils** パッケージがインストールされている。
- Linux ユーザーは、SELinux の限定ユーザーに割り当てられます。
  - 特権のないユーザーは、**user_u** に割り当てられます。
  - 特権ユーザーは、**staff_u** に割り当てられます。

手順

1. SELinux ユーザーのセキュリティー範囲を定義します。

```
# semanage user -m -rs0:c0,c1-s0:c0.c9 <user_u>
```

setrans.conf ファイルで定義されているように、**c0** から **c1023** までのカテゴリー番号またはカテゴリーラベルを使用します。詳細は、Defining category labels in MCS を参照してください。

2. Linux ユーザーに MCS カテゴリーを割り当てます。指定できるのは、関連する SELinux ユーザーに定義された範囲内でのみです。

```
# semanage login -m -rs0:c1 <Linux.user1>
```

注記

chcat コマンドを使用して、Linux ユーザーにカテゴリーを追加または削除できます。以下の例では、**<category1>** を追加し、**<Linux.user1>** および **<Linux.user2>** から**<category2>** を削除します。

```
# chcat -l -- +<category1>,-<category2> <Linux.user1>,<Linux.user2>
```

-<category> 構文を使用する前に、コマンドラインで **--** を指定する必要があります。指定しないと、**chcat** コマンドは、カテゴリーの削除をコマンドオプションであると誤って解釈します。

検証

- Linux ユーザーに割り当てられているカテゴリーの一覧を表示します。

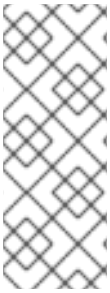
```
# chcat -L -l <Linux.user1>,<Linux.user2>
<Linux.user1>: <category1>,<category2>
<Linux.user2>: <category1>,<category2>
```

関連情報

- **chcat(8)** の man ページ

7.5. MCS でのファイルへのカテゴリーの割り当て

カテゴリーをユーザーに割り当てるには、管理者特権が必要です。ユーザーはカテゴリーをファイルに割り当てることができます。ファイルのカテゴリーを変更するには、そのファイルへのアクセス権が必要です。ユーザーは、割り当てられているカテゴリーにのみファイルを割り当てることができます。



注記

このシステムは、カテゴリーアクセスルールと従来のファイルアクセス権限を組み合わせています。たとえば、**bigfoot** のカテゴリーを持つユーザーが DAC (Discretionary Access Control) を使用して他のユーザーによるファイルへのアクセスをブロックすると、他の **bigfoot** ユーザーはそのファイルにアクセスできなくなります。利用可能なすべてのカテゴリーに割り当てられたユーザーは、ファイルシステム全体にアクセスできない場合があります。

前提条件

- SELinux モードが **Enforcing** に設定されている。
- SELinux のポリシーは **targeted** または **mls** に設定されている。
- **policycoreutils-python-utils** パッケージがインストールされている。
- 以下に該当する Linux ユーザーのアクセスおよびパーミッション
  - SELinux ユーザーに割り当てられている。
  - ファイルの割り当て先となるカテゴリーに割り当てられている。詳細は、Assigning categories to users in MCS を参照してください。

- カテゴリーに追加するファイルへのアクセスと権限。
- 検証目的: このカテゴリーに割り当てられていない Linux ユーザーへのアクセスとパーミッション

手順

- カテゴリーをファイルに追加します。

```
$ chcat -- +<category1>,+<category2> <path/to/file1>
```

setrans.conf ファイルで定義されているように、**c0** から **c1023** までのカテゴリー番号またはカテゴリーラベルを使用します。詳細は、Defining category labels in MCS を参照してください。

同じ構文を使用して、ファイルからカテゴリーを削除できます。

```
$ chcat -- -<category1>,-<category2> <path/to/file1>
```

注記

カテゴリーを削除する場合は、コマンドラインで **--** を指定してから **-<category>** 構文を使用する必要があります。このコマンドを実行しないと、**chcat** コマンドがカテゴリーの削除をコマンドオプションとして誤って解釈する可能性があります。

検証

1. ファイルのセキュリティーコンテキストを表示して、カテゴリーが正しいことを確認します。

```
$ ls -lZ <path/to/file>
-rw-r--r--  <LinuxUser1> <Group1> root:object_r:user_home_t:_<sensitivity>_:_<category>_ <path/to/file>
```

ファイルの特定のセキュリティーコンテキストは異なる場合があります。

2. オプション:ファイルと同じカテゴリーに割り当てられていない Linux ユーザーとしてログインしたときに、ファイルにアクセスしようとします。

```
$ cat <path/to/file>
cat: <path/to/file>: Permission Denied
```

関連情報

- **semanage(8)** の man ページ
- **chcat(8)** の man ページ

第8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。

8.1. 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;
```

この例のルールは、以下のように読むことができます。**Apache 処理は、そのロギングファイル を 読み取り できます。**このルールでは、**apache_process** と **apache_log** が **labels** 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 9 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 9 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

- **sepolicy(8)** および **getfattr(1)** の man ページ

8.2. 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 -l "*"
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。Type Enforcement 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

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

- **sepolgen(8)、ausearch(8)、audit2allow(1)、audit2why(1)、sealert(8)、および restorecon(8) の man ページ**

8.3. CREATING A LOCAL SELINUX POLICY MODULE

アクティブな SELinux ポリシーに特定の SELinux ポリシーモジュールを追加することで、SELinux ポリシーに関する特定の問題を修正できます。この手順を使用して、Red Hat リリースノート で説明されている特定の既知の問題を修正するか、特定の Red Hat ソリューション を実装できます。

警告

Red Hat が提供するルールのみを使用します。Red Hat は、カスタムルールを使用した SELinux ポリシーモジュールの作成には対応していません。これは、製品サポートの対象範囲 外であるためです。専門家でない場合は、Red Hat の営業担当者に連絡し、コンサルティングサービスを依頼してください。

前提条件

- 検証用の **setools-console** パッケージおよび **audit** パッケージ。

手順

1. テキストエディターで新しい **.cil** を開きます。以下に例を示します。

```
# vim <local_module>.cil
```

ローカルモジュールをより適切に整理するには、ローカル SELinux ポリシーモジュール名で接頭辞 **local_** を使用します。

2. 既知の問題または Red Hat ソリューションからカスタムルールを挿入します。

重要

独自のルールを作成しないでください。特定の既知の問題または Red Hat ソリューションで提供されているルールのみを使用します。

たとえば、SELinux denies cups-lpd read access to cups.sock in RHELのソリューションを実装するには、以下の規則を挿入します。

注記

このサンプルソリューションは、RHBA-2021:4420 で RHEL 用に永続的に修正されました。したがって、このソリューションの特定の手順は更新済みの RHEL 8 システムおよび 9 システムに影響を与えず、構文のサンプルとしてのみ含まれています。

```
(allow cupsd_lpd_t cupsd_var_run_t (sock_file (read)))
```

2 つの SELinux ルール構文 CIL (Common Intermediate Language) および m4 のいずれかを使用できることに注意してください。たとえば、CIL の **(allow cupsd_lpd_t cupsd_var_run_t (sock_file (read)))** は、m4 の以下と同じになります。

```
module local_cupslpd-read-cupssock 1.0;
```

```
require {
    type cupsd_var_run_t;
    type cupsd_lpd_t;
    class sock_file read;
}

#============= cupsd_lpd_t ==============
allow cupsd_lpd_t cupsd_var_run_t:sock_file read;
```

3. ファイルを保存してから閉じます。

4. ポリシーモジュールをインストールします。

```
# semodule -i <local_module>.cil
```

注記

semodule -i を使用して作成したローカルポリシーモジュールを削除する場合は、.cil 接尾辞のないモジュール名を参照してください。ローカルポリシーモジュールを削除するには、semodule -r <local_module> を使用します。

5. ルールに関連するサービスを再起動します。

```
# systemctl restart <service-name>
```

検証

1. SELinux ポリシーにインストールされているローカルモジュールを一覧表示します。

```
# semodule -lfull | grep "local_"
400 local_module  cil
```

注記

ローカルモジュールの優先順位は 400 であるため、semodule -lfull | grep -v ^100 コマンドを使用して、その値を使用してリストからそれらをフィルター処理することもできます。

2. 関連する許可ルールを SELinux ポリシーで検索します。

```
# sesearch -A --source=<SOURCENAME> --target=<TARGETNAME> --
class=<CLASSNAME> --perm=<P1>,<P2>
```

<SOURCENAME> はソースの SELinux の種類、<TARGETNAME> はターゲットの SELinux の種類、<CLASSNAME> はセキュリティークラスまたはオブジェクトクラスの名前、そして <P1> と <P2> はルール固有の権限です。

たとえば、SELinux denies cups-lpd read access to cups.sock in RHELソリューションの場合は、以下を行います。

```
# sesearch -A --source=cupsd_lpd_t --target=cupsd_var_run_t --class=sock_file --
perm=read
allow cupsd_lpd_t cupsd_var_run_t:sock_file { append getattr open read write };
```

最後の行には、read 操作が含まれているはずです。

3. **関連するサービスが SELinux に制限されて実行されていることを確認します。**
   a. **関連するサービスに関連するプロセスを特定します。**

   ```
   $ systemctl status <service-name>
   ```

   b. **上記コマンドの出力で一覧表示されたプロセスの SELinux コンテキストを確認します。**

   ```
   $ ps -efZ | grep <process-name>
   ```

4. **サービスが SELinux の拒否を引き起こさないことを確認します。**

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

関連情報

- *5章SELinux 関連の問題のトラブルシューティング*

8.4. 関連情報

- **SELinux ポリシーワークショップ**

第9章 コンテナーの SELINUX ポリシーの作成

Red Hat Enterprise Linux 9 には、**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が同梱されています。**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。

9.1. 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 9 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

- Red Hat ブログ記事「Generate SELinux policies for containers with udica」

9.2. 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

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、**dnf install podman** 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。

```
# dnf 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。

```
# dnf 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. 新しいターミナルを開き、**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

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 /]# dnf install nmap-ncat
[root@37a3635afb8f /]# nc -lvp 21
...
Ncat: Listening on :::21
Ncat: Listening on 0.0.0.0:21
^C
[root@37a3635afb8f /]# nc -lvp 80
...
Ncat: bind to :::80: Permission denied. QUITTING.
```

関連情報

- **udica(8)** および **podman(1)** の man ページ
- コンテナーの構築、実行、および管理

9.3. 関連情報

- udica - コンテナーの SELinux ポリシーの生成

第10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**

10.1. 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 システムロールでは、以下の操作が可能になります。

- **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。**
- **SELinux モジュールの管理**

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10.1 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。	**/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。	**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。	**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。	**semanage port**
selinux_logins	ユーザーを SELinux ユーザーマッピングに設定します。	**semanage login**

ロール変数	説明	CLI の代替手段
selinux_modules	SELinux モジュールのインストール、有効化、無効化、または削除を行います。	**semodule**

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。

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。

関連情報

- RHEL システムロールの概要

10.2. SELINUX システムロールを使用した、複数のシステムでの SELINUX 設定の適用

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

- **1つまたは複数の 管理対象ノード へのアクセスとパーミッション (SELinux システムロールで設定するシステム)。**
- **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。**
  - **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。**

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. Playbook を準備します。ゼロから開始するか、**rhel-system-roles** パッケージの一部としてインストールされたサンプル Playbook を変更してください。

```
# cp /usr/share/doc/rhel-system-roles/selinux/example-selinux-playbook.yml my-selinux-playbook.yml
# vi my-selinux-playbook.yml
```

2. シナリオに合わせて Playbook の内容を変更します。たとえば、次の部分では、システムが SELinux モジュール **selinux-local-1.pp** をインストールして有効にします。

```
selinux_modules:
- { path: "selinux-local-1.pp", priority: "400" }
```

3. 変更を保存し、テキストエディターを終了します。

4. host1、host2 および host3 システムで Playbook を実行します。

```
# ansible-playbook -i host1,host2,host3 my-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。

10.3. 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 9 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** パッケージがシステムに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)**