Red Hat Enterprise Linux 7

SELinux ユーザーおよび管理者のガイド

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

Last Updated: 2023-01-29

Red Hat Enterprise Linux 7 SELinux ユーザーおよび管理者のガイド

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、SELinux と 制限のあるサービスの管理 の 2 つの部分で構成されています。前者は、SELinux 機能の基本と原則を説明します。後者は、さまざまなサービスを設定して設定する実用的なタスクに重点を置いています。

目次

パート I. SELINUX

本書では、Security Enhanced Linux(SELinux) が機能するための基本と原則について説明します。

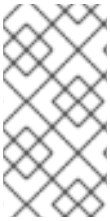
第1章 はじめに

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

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

SELinux は、Mandatory Access Control (MAC) を実装しています。それぞれのプロセスおよびシステムリソースには、*SELinux コンテキスト* と呼ばれる特別なセキュリティーラベルがあります。SELinux コンテキストは *SELinux label* 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一貫した方法が提供されますが、他の識別方法にある曖昧さも削除されます。たとえば、ファイルはバインドマウントを使用するシステムで複数の有効なパス名を持つことができます。

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。



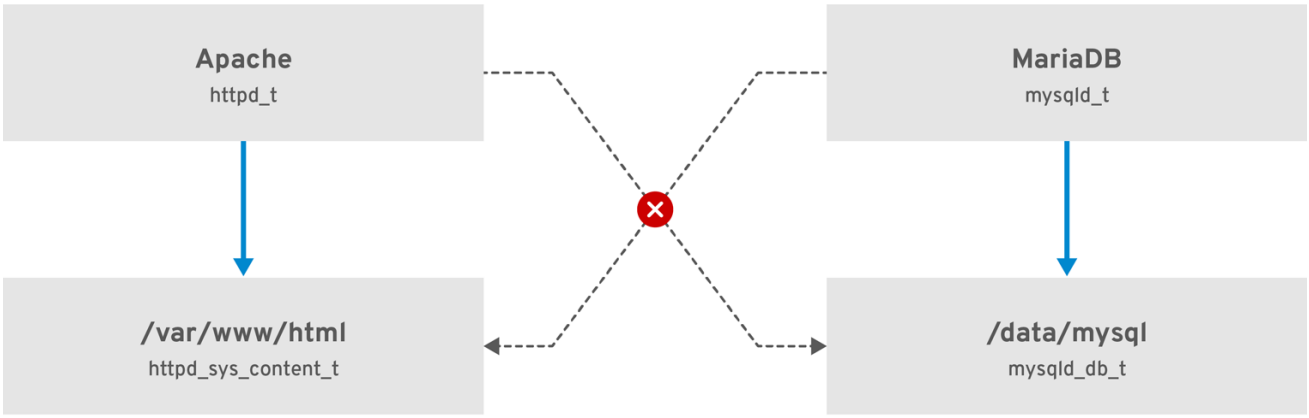
注記

SELinux ポリシールールは、DAC ルールの後にチェックされること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 のタイプは通常、末尾に **_t** を付けます。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のタイプコンテキストは、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 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、SELinux は、mysqld_t タイプを持つプロセスが、httpd_sys_content_t というラベルが付いた /var/www/html/ ディレクトリーにアクセスすることも適切に拒否します。



[D]

関連情報

詳細は、以下のドキュメントを参照してください。

- **selinux (8)** の man ページと、 **apropos selinux** コマンドで表示される man ページ。
- 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る man ページ。詳細は、「サービスの man ページ」 を参照してください。
- The SELinux Coloring Book
- SELinux Wiki FAQ

1.1. 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。

1.2. 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。詳細は、「制限のあるユーザーおよび制限のないユーザー」 を参照してください。
- プロセスとデータの分離が向上します。プロセスは独自のドメインで実行され、プロセスが他のプロセスによって使用されているファイルにアクセスできないようにします。また、プロセスが他のプロセスにアクセスできないようにします。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] デフォルトの SELinux ポリシーは、BIND **named** デーモン自体と他のプロセスによってゾーンファイルの更新を阻止します。
- NetworkWorld.com の記事 A seatbelt for server software: SELinux blocks real-world exploits を参照してください。[2]SELinux に関する背景情報と、SELinux が阻止したさまざまな不正使用に関する情報です。

1.3. SELINUX ARCHITECTURE

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。

1.4. SELINUX のステータスおよびモード

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

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。

Permissive モードでは、システムは、SELinux が読み込んだセキュリティーポリシーを実行しているかのように動作します。これには、オブジェクトのラベル付けや、アクセスを拒否したエントリーをログに出力するなどの動作が含まれますが、いずれの操作も拒否される訳ではありません。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 ポリシーの開発には役に立ちます。

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。

Enforcing モードと Permissive モードを切り替えるには、**setenforce** ユーティリティーを使用します。**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
```

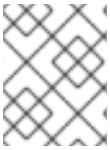
```
~]# setenforce 0
~]# getenforce
Permissive
```

```
~]# 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
```

詳細は、「Permissive ドメイン」 を参照してください。



注記

永続的な状態およびモードの変更は、「SELinux のステータスおよびモードの永続的変更」 で説明されています。

1.5. 関連情報

Red Hat Identity Management(IdM) は、SELinux ユーザーマップを定義する集中型ソリューションを提供します。詳細は、Defining SELinux User Maps in the Linux Domain Identity, Authentication, and Policy Guide を参照してください。

---

[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。

[2] Marti, Don."A seatbelt for server software: SELinux blocks real-world exploits".2008 年 2 月 24 日公開。2009 年 8 月 27 日にアクセス: http://www.networkworld.com/article/2283723/lan-wan/a-seatbelt-for-server-software--selinux-blocks-real-world-exploits.html

第2章 SELINUX コンテキスト

プロセスとファイルには、SELinux ユーザー、ロール、タイプ、オプションでレベルなどの追加情報が含まれる SELinux コンテキストでラベルが付けられます。SELinux の実行時には、すべての情報はアクセス制御の決定に使用されます。Red Hat Enterprise Linux では、SELinux はロールベースアクセス制御 (RBAC)、Type Enforcement(TE)、およびオプションで Multi-Level Security(MLS) の組み合わせを提供します。

以下は、SELinux コンテキストを示しています。SELinux コンテキストは、SELinux を実行する Linux オペレーティングシステムの、プロセス、Linux ユーザー、およびファイルで使用されます。以下のコマンドを使用して、ファイルおよびディレクトリーの SELinux コンテキストを表示します。

```
~]$ ls -Z file1
-rwxrw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0      file1
```

SELinux コンテキストは、**SELinux user:role:type:level** 構文に従います。各フィールドは、以下のとおりです。

SELinux ユーザー

SELinux ユーザー ID は、特定のロールセットと特定の MLS/MCS 範囲に対して承認されるポリシーに既知の ID です。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。マップされた SELinux ユーザー ID は、開始できるロールとレベルを定義するために、そのセッションのプロセスに対して SELinux コンテキストで使用されます。root で以下のコマンドを入力して、SELinux と Linux ユーザーアカウントのマッピングの一覧を表示します (policycoreutils-pythonパッケージがインストールされている必要があります)。

```
~]# 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       *
system_u             system_u             s0-s0:c0.c1023       *
```

出力は、システムごとにわずかに異なる場合があります

- **Login Name** 列には、Linux ユーザーが一覧表示されます。
- **SELinux User** 列には、Linux ユーザーがマッピングされている SELinux ユーザーが一覧表示されます。プロセスの場合、SELinux ユーザーはアクセス可能なロールとレベルを制限します。
- **MLS/MCS Range** 列は、MLS(Multi-Level Security) および Multi-Category Security(MCS) で使用されるレベルです。
- **Service** 列は、Linux ユーザーがシステムにログインする予定の正しい SELinux コンテキストを決定します。デフォルトでは、任意のサービスを表すアスタリスク (*) 文字が使用されます。

role

SELinux の一部は、ロールベースのアクセス制御 (RBAC) セキュリティーモデルです。ロールは RBAC の属性です。SELinux ユーザーはロールに対して許可され、ロールはドメインに対して承認されます。このロールは、ドメインと SELinux ユーザーの仲介として機能します。入力可能なロー

ルによって入力できるドメインが決まります。最終的には、アクセス可能なオブジェクトタイプを制御します。これにより、権限昇格攻撃への脆弱性が軽減されます。

type

タイプは、Type Enforcement の属性です。タイプは、プロセスのドメインとファイルのタイプを定義します。SELinux ポリシールールは、タイプにアクセスするドメインであるか、別のドメインにアクセスするドメインであるかにかかわらず、タイプが相互にアクセスできる方法を定義します。アクセスは、それを許可する特定の SELinux ポリシールールが存在する場合にのみ許可されます。

level

このレベルは MLS と MCS の属性です。MLS の範囲はレベルのペアで、レベルが異なる場合は **lowlevel-highlevel** と記述され、レベルが同一である場合は **lowlevel** と記述されます (**s0-s0** は **s0**と同じ)。各レベルは機密カテゴリーのペアで、カテゴリーはオプションになります。カテゴリーがある場合、レベルは **sensitivity:category-set** として記述されます。カテゴリーがない場合は、**sensitivity** と記述されます。

カテゴリーセットが連続したシリーズの場合は、省略できます。たとえば、**c0.c3** は **c0,c1,c2,c3** と同じです。**/etc/selinux/targeted/setrans.conf** ファイルはレベル (**s0:c0**) を人間が判読できる形式 (**CompanyConfidential**) にマップします。Red Hat Enterprise Linux では、Targeted ポリシーが MCS を強制し、MCS では 1 つの機密レベル (**s0**) があります。Red Hat Enterprise Linux の MCS は、1024 の異なるカテゴリー (**c0** から **c1023** まで) に対応しています。**s0-s0:c0.c1023** は機密レベル **s0** で、すべてのカテゴリーの権限が承認されます。

MLS は Bell-La Padula Mandatory Access Model を強制し、LSPP(Labeled Security Protection Profile) 環境で使用されます。MLS の制限を使用するには、selinux-policy-mls パッケージをインストールし、MLS がデフォルトの SELinux ポリシーになるように設定します。Red Hat Enterprise Linux に同梱される MLS ポリシーは、評価された設定に含まれなかった多くのプログラムドメインを省略するため、デスクトップワークステーションの MLS は使用できません (X Window System へのサポートなし)。ただし、アップストリームの SELinux 参照ポリシー から、すべてのプログラムドメインを含む MLS ポリシーを構築することができます。MLS 設定の詳細は、「Multi-Level Security (MLS)」 を参照してください。

2.1. ドメインの移行

あるドメインのプロセスは、新規ドメインの **entrypoint** タイプを持つアプリケーションを実行することにより、別のドメインに移行します。**entrypoint** パーミッションは、SELinux ポリシーで使用され、どのアプリケーションがドメインに入るために使用できるかを制御します。以下の例は、ドメイン移行を示しています。

手順2.1 ドメイン移行の例

1. ユーザーがパスワードを変更したい。これを実行するには、**passwd** ユーティリティーを実行します。**/usr/bin/passwd** 実行可能ファイルには、**passwd_exec_t** タイプのラベルが付けられます。

```
~]$ ls -Z /usr/bin/passwd
-rwsr-xr-x  root root system_u:object_r:passwd_exec_t:s0 /usr/bin/passwd
```

passwd ユーティリティーは、**shadow_t** タイプのラベルが付けられた **/etc/shadow** にアクセスします。

```
~]$ ls -Z /etc/shadow
-r--------. root root system_u:object_r:shadow_t:s0    /etc/shadow
```

2. SELinux ポリシールールでは、**passwd_t** ドメインで実行しているプロセスが、**shadow_t** タイプのラベルが付けられたファイルに読み取りと書き込みが許可されていることを示しています。**shadow_t** タイプは、パスワードの変更に必要なファイルにのみ適用されます。これには、**/etc/gshadow**、**/etc/shadow**、およびバックアップファイルが含まれます。

3. SELinux ポリシールールは、**passwd_t** ドメインの **entrypoint** パーミッションが **passwd_exec_t** タイプに設定されていることを示しています。

4. ユーザーが **passwd** ユーティリティーを実行すると、ユーザーのシェルプロセスは **passwd_t** ドメインに移行します。SELinux では、デフォルトのアクションは否定することで、**passwd_t** ドメインで実行しているアプリケーションが **shadow_t** タイプでラベル付けされたファイルにアクセスできるようにするルールが存在するため、**passwd** アプリケーションは **/etc/shadow** にアクセスし、ユーザーのパスワードを更新します。

この例は網羅的なものではなく、ドメイン移行を説明する基本的な例として使用されます。**passwd_t** ドメインで実行しているサブジェクトが **shadow_t** ファイルタイプでラベル付けされたオブジェクトにアクセスできるようにする実際のルールがありますが、サブジェクトが新しいドメインに移行する前に、その他の SELinux ポリシールールを満たす必要があります。この例では、Type Enforcement により以下が確保されます。

- **passwd_t** ドメインを入力できるのは、**passwd_exec_t** タイプのラベルが付けられたアプリケーションを実行する場合のみです。これは、**lib_t** タイプなどの承認された共有ライブラリーからのみ実行でき、他のアプリケーションは実行できません。

- **passwd_t** などの承認されたドメインのみが、**shadow_t** タイプのラベルが付けられたファイルに書き込みできます。他のプロセスがスーパーユーザー権限で実行されている場合でも、これらのプロセスは **passwd_t** ドメインで実行されていないため、**shadow_t** タイプのラベルが付けられたファイルに書き込みできません。

- 承認されたドメインのみが **passwd_t** ドメインに移行できます。たとえば、**sendmail_t** ドメインで実行する **sendmail** プロセスには **passwd** を実行する適切な理由がないため、**passwd_t** ドメインに移行することはできません。

- **passwd_t** ドメインで実行しているプロセスは、**etc_t** または **shadow_t** タイプのラベルが付けられたファイルなど、承認されたタイプに対する読み書きのみが可能です。これにより、**passwd** アプリケーションが任意のファイルの読み書きを行うのを防ぎます。

2.2. プロセスの SELINUX コンテキスト

ps -eZ コマンドを使用して、プロセスの SELinux コンテキストを表示します。以下に例を示します。

手順2.2 passwd ユーティリティーの SELinux コンテキストの表示

1. Applications → System Tools → Terminal などのターミナルを開きます。

2. **passwd** ユーティリティーを実行します。新しいパスワードは入力しないでください。

```
~]$ passwd
Changing password for user user_name.
Changing password for user_name.
(current) UNIX password:
```

3. 新しいタブまたは別の端末を開き、以下のコマンドを入力します。出力は以下の例のようになります。

```
~]$ ps -eZ | grep passwd
unconfined_u:unconfined_r:passwd_t:s0-s0:c0.c1023 13212 pts/1 00:00:00 passwd
```

4. 最初のタブ/ターミナルで **Ctrl+C** を押して **passwd** ユーティリティーをキャンセルします。

この例では、(**passwd_exec_t** タイプのラベルが付けられた) **passwd** ユーティリティーが実行されると、ユーザーのシェルプロセスは **passwd_t** ドメインに移行します。タイプはプロセスのドメインとファイルのタイプを定義することに注意してください。

実行中のすべてのプロセスの SELinux コンテキストを表示するには、**ps** ユーティリティーを再度実行します。以下は出力の省略された例であり、お使いのシステムでは異なる場合があります。

```
]$ ps -eZ
system_u:system_r:dhcpc_t:s0              1869 ?  00:00:00 dhclient
system_u:system_r:sshd_t:s0-s0:c0.c1023  1882 ?  00:00:00 sshd
system_u:system_r:gpm_t:s0               1964 ?  00:00:00 gpm
system_u:system_r:crond_t:s0-s0:c0.c1023 1973 ?  00:00:00 crond
system_u:system_r:kerneloops_t:s0        1983 ?  00:00:05 kerneloops
system_u:system_r:crond_t:s0-s0:c0.c1023 1991 ?  00:00:00 atd
```

system_r ロールは、デーモンなどのシステムプロセスに使用されます。Enforcement と入力してから、各ドメインを分けます。

2.3. ユーザーの SELINUX コンテキスト

以下のコマンドを使用して、Linux ユーザーに関連付けられている SELinux コンテキストを表示します。

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

Red Hat Enterprise Linux では、Linux ユーザーはデフォルトで制限なしで実行されます。この SELinux コンテキストは、Linux ユーザーが SELinux **unconfined_u** ユーザーにマップされ、**unconfined_r** ロールとして実行され、**unconfined_t** ドメインで実行されていることを示しています。**s0-s0** は MLS 範囲で、この場合は **s0** と同じです。ユーザーがアクセスできるカテゴリーは**c0.c1023** で定義されます。これは、すべてのカテゴリー (**c0** から **c1023**) です。

第3章 TARGETED ポリシー

Targeted ポリシーは、Red Hat Enterprise Linux で使用されるデフォルトの SELinux ポリシーです。Targeted ポリシーを使用する場合、対象となるプロセスは制限のあるドメインで実行され、対象とならないプロセスは制限のないドメインで実行されます。たとえば、デフォルトでは、ログイン中のユーザーは **unconfined_t** ドメインで実行され、init によって起動されたシステムプロセスは **unconfined_service_t** ドメインで実行されます。これらのドメインはどちらも制限されません。

実行可能および書き込み可能なメモリーチェックは、制限のあるドメインと制限のないドメインの両方に適用できます。ただし、デフォルトでは、制限のないドメインで実行しているサブジェクトは、書き込み可能なメモリーを割り当てて実行できます。これらのメモリーチェックを有効にするには、ブール値を設定します。これにより、SELinux ポリシーをランタイム時に変更できます。ブール値設定については、後で説明します。

3.1. 制限のあるプロセス

sshd や **httpd** などのネットワーク上でリッスンするほぼすべてのサービスは、Red Hat Enterprise Linux に制限されます。また、root ユーザーで実行され、**passwd** ユーティリティーなどのユーザーのタスクを実行するほとんどのプロセスには制限があります。プロセスが制限されると、**httpd_t** ドメインで実行している **httpd** プロセスのように、独自のドメインで実行されます。制限のあるプロセスが攻撃者によって侵害された場合、SELinux ポリシーの設定に応じて、攻撃者のリソースへのアクセスと、攻撃者が行う可能性のある損害は制限されます。

以下の手順に従って、SELinux が有効で、システムが以下の例を実行する準備が完了していることを確認します。

手順3.1 SELinux の状態を確認する方法

1. SELinux が有効になり、Enforcing モードで実行されていること、および Targeted ポリシーが使用されていることを確認します。正しい出力は以下のようになります。

```
~]$ 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
Max kernel policy version:      30
```

   SELinux モードの変更に関する詳細は、「SELinux のステータスおよびモードの永続的変更」を参照してください。

2. root として、**/var/www/html/** ディレクトリーにファイルを作成します。

```
~]# touch /var/www/html/testfile
```

3. 以下のコマンドを入力して、新規作成されたファイルの SELinux コンテキストを表示します。

```
~]$ ls -Z /var/www/html/testfile
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/testfile
```

デフォルトでは、Linux ユーザーは Red Hat Enterprise Linux で制限なしで実行されます。そのため、**testfile** ファイルには SELinux **unconfined_u** ユーザーのラベルが付けられます。RBAC は、ファイルではなくプロセスに使用されます。ロールには意味がありません。**object_r** ロールはファイル (永続ストレージおよびネットワークファイルシステム上) に使用される汎用ロールです。/**proc** ディレクトリー下では、プロセスに関連するファイルは **system_r** ロールを使用する場合があります。**httpd_sys_content_t** タイプを使用すると、**httpd** プロセスがこのファイルにアクセスできるようになります。

以下の例は、SELinux が Samba で使用することが意図されているファイルなど、正しくラベル付けされていないファイルを読み込めるように、SELinux が Apache HTTP Server(**httpd**) を防ぐ方法を示しています。これはサンプルであり、本番環境では使用しないでください。httpd パッケージと wget パッケージがインストールされ、SELinux Targeted ポリシーが使用され、SELinux が Enforcing モードで動作していることを前提とします。

手順3.2 競合するプロセスの例

1. root で **httpd** デーモンを起動します。

```
~]# systemctl start httpd.service
```

サービスが実行中であることを確認します。出力には以下の情報が含まれている必要があります (タイムスタンプのみは異なります)。

```
~]$ systemctl status httpd.service
httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled)
   Active: active (running) since Mon 2013-08-05 14:00:55 CEST; 8s ago
```

2. Linux ユーザーが書き込み権限を持つディレクトリーに移動し、以下のコマンドを入力します。デフォルト設定が変更されない限り、このコマンドは成功します。

```
~]$ wget http://localhost/testfile
--2009-11-06 17:43:01--  http://localhost/testfile
Resolving localhost... 127.0.0.1
Connecting to localhost|127.0.0.1|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 0 [text/plain]
Saving to: `testfile'

[ <=>                                 ] 0     --.-K/s   in 0s

2009-11-06 17:43:01 (0.00 B/s) - `testfile' saved [0/0]
```

3. **chcon** コマンドによってファイルが再ラベル付けされます。ただし、このようなラベルの変更は、ファイルシステムの再ラベル付け時に維持されません。ファイルシステムの再ラベル付け後も存続する永続的な変更については、**semanage** ユーティリティーを使用します。これについては後述します。root で以下のコマンドを実行して、タイプを Samba が使用するタイプに変更します。

```
~]# chcon -t samba_share_t /var/www/html/testfile
```

以下のコマンドを入力して、変更を表示します。

```
~]$ ls -Z /var/www/html/testfile
-rw-r--r--  root root unconfined_u:object_r:samba_share_t:s0 /var/www/html/testfile
```

4. 現在の DAC パーミッションでは、**httpd** プロセスが **testfile** にアクセスできるようになります。ユーザーの書き込み権限があるディレクトリーに移動し、以下のコマンドを入力します。デフォルト設定が変更されない限り、このコマンドは失敗します。

```
~]$ wget http://localhost/testfile
--2009-11-06 14:11:23--  http://localhost/testfile
Resolving localhost... 127.0.0.1
Connecting to localhost|127.0.0.1|:80... connected.
HTTP request sent, awaiting response... 403 Forbidden
2009-11-06 14:11:23 ERROR 403: Forbidden.
```

5. root で **testfile** を削除します。

```
~]# rm -i /var/www/html/testfile
```

6. **httpd** を実行する必要がない場合は、root で以下のコマンドを実行して停止します。

```
~]# systemctl stop httpd.service
```

この例は、SELinux により追加された追加のセキュリティーを示しています。DAC ルールでは、手順 2 で **httpd** プロセスが **testfile** へアクセスすることが許可されましたが、**httpd** プロセスがアクセス権を持たないタイプでファイルがラベル付けされているため、SELinux ではアクセスが拒否されました。

auditd デーモンが実行している場合は、以下のようなエラーが **/var/log/audit/audit.log** に記録されます。

```
type=AVC msg=audit(1220706212.937:70): avc:  denied  { getattr } for  pid=1904 comm="httpd"
path="/var/www/html/testfile" dev=sda5 ino=247576 scontext=unconfined_u:system_r:httpd_t:s0
tcontext=unconfined_u:object_r:samba_share_t:s0  tclass=file

type=SYSCALL msg=audit(1220706212.937:70): arch=40000003 syscall=196 success=no exit=-13
a0=b9e21da0 a1=bf9581dc a2=555ff4 a3=2008171 items=0 ppid=1902 pid=1904 auid=500 uid=48
gid=48 euid=48 suid=48 fsuid=48 egid=48 sgid=48 fsgid=48 tty=(none) ses=1 comm="httpd"
exe="/usr/sbin/httpd" subj=unconfined_u:system_r:httpd_t:s0 key=(null)
```

また、以下のようなエラーが **/var/log/httpd/error_log** に記録されます。

```
[Wed May 06 23:00:54 2009] [error] [client 127.0.0.1] (13)Permission denied: access to /testfile
denied
```

3.2. 制限のないプロセス

制限のないプロセスは、制限のないドメインで実行されます。たとえば、**init** によって実行される制限のないサービスは **unconfined_service_t** ドメインで実行され、カーネルによって実行される制限のないサービスは **kernel_t** ドメインで実行され、制限のない Linux ユーザーによって実行される制限のないサービスは、**unconfined_t** ドメインで実行されます。制限のないプロセスの場合、SELinux ポリシールールが適用されますが、制限のないドメインで実行しているプロセスを許可するポリシールールが存在します。制限のないドメインで実行されているプロセスは、DAC ルールだけを使用するようにフォールバックされます。制限のないプロセスが侵害された場合、SELinux は攻撃者がシステムリソー

スやデータにアクセスすることを防ぎませんが、もちろん、DAC ルールは引き続き使用されます。SELinux は、DAC ルールに加えたセキュリティー強化です。DAC ルールを置き換えるものではありません。

SELinux が有効になり、システムが以下のサンプルを実行する準備が整っていることを確認するには、「制限のあるプロセス」 で説明されている 手順3.1「SELinux の状態を確認する方法」 を実行します。

以下の例は、制限のない実行時に Apache HTTP Server(**httpd**) が Samba が使用するデータにアクセスする方法を示しています。Red Hat Enterprise Linux では、**httpd** プロセスはデフォルトで制限のある **httpd_t** ドメインで実行されます。これはサンプルであり、本番環境では使用しないでください。httpd、wget、dbus、および audit パッケージがインストールされ、SELinux Targeted ポリシーが使用され、SELinux が Enforcing モードで実行されていることを前提としています。

手順3.3 制限のないプロセスの例

1. **chcon** コマンドによってファイルが再ラベル付けされます。ただし、このようなラベルの変更は、ファイルシステムの再ラベル付け時に維持されません。ファイルシステムの再ラベル付け後も存続する永続的な変更については、**semanage** ユーティリティーを使用します。これについては後述します。root ユーザーとして以下のコマンドを実行し、タイプを Samba が使用するタイプに変更します。

```
~]# chcon -t samba_share_t /var/www/html/testfile
```

変更を表示します。

```
~]$ ls -Z /var/www/html/testfile
-rw-r--r--  root root unconfined_u:object_r:samba_share_t:s0 /var/www/html/testfile
```

2. 以下のコマンドを実行して、**httpd** プロセスが実行されていないことを確認します。

```
~]$ systemctl status httpd.service
httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled)
   Active: inactive (dead)
```

出力が異なる場合は、root で以下のコマンドを実行して、**httpd** プロセスを停止します。

```
~]# systemctl stop httpd.service
```

3. **httpd** プロセスが制限なしで実行されるようにするには、root で以下のコマンドを実行して、**/usr/sbin/httpd** ファイルのタイプを、制限のあるドメインに移行しないタイプに変更します。

```
~]# chcon -t bin_t /usr/sbin/httpd
```

4. **/usr/sbin/httpd** に **bin_t** タイプのラベルが付けられていることを確認します。

```
~]$ ls -Z /usr/sbin/httpd
-rwxr-xr-x. root root system_u:object_r:bin_t:s0       /usr/sbin/httpd
```

5. root で、**httpd** プロセスを開始して、正常に起動したことを確認します。

```
~]# systemctl start httpd.service
```

```
~]# systemctl status httpd.service
httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled)
   Active: active (running) since Thu 2013-08-15 11:17:01 CEST; 5s ago
```

6. 以下のコマンドを入力して、**unconfined_service_t** ドメインで実行している **httpd** を表示します。

```
~]$ ps -eZ | grep httpd
system_u:system_r:unconfined_service_t:s0 11884 ? 00:00:00 httpd
system_u:system_r:unconfined_service_t:s0 11885 ? 00:00:00 httpd
system_u:system_r:unconfined_service_t:s0 11886 ? 00:00:00 httpd
system_u:system_r:unconfined_service_t:s0 11887 ? 00:00:00 httpd
system_u:system_r:unconfined_service_t:s0 11888 ? 00:00:00 httpd
system_u:system_r:unconfined_service_t:s0 11889 ? 00:00:00 httpd
```

7. Linux ユーザーが書き込み権限を持つディレクトリーに移動し、以下のコマンドを入力します。デフォルト設定が変更されない限り、このコマンドは成功します。

```
~]$ wget http://localhost/testfile
--2009-05-07 01:41:10--  http://localhost/testfile
Resolving localhost... 127.0.0.1
Connecting to localhost|127.0.0.1|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 0 [text/plain]
Saving to: `testfile'

[ <=>                                 ]--.-K/s   in 0s

2009-05-07 01:41:10 (0.00 B/s) - `testfile' saved [0/0]
```

httpd プロセスは **samba_share_t** タイプのラベル付けされたファイルにアクセスできませんが、**httpd** は制限のない **unconfined_service_t** ドメインで実行され、DAC ルールの使用にフォールバックします。したがって、**wget** コマンドは成功します。**httpd** が制限のある **httpd_t** ドメインで実行されていた場合、**wget** コマンドは失敗していました。

8. **restorecon** ユーティリティーは、ファイルのデフォルトの SELinux コンテキストを復元します。root で次のコマンドを実行して、**/usr/sbin/httpd** のデフォルトの SELinux コンテキストを復元します。

```
~]# restorecon -v /usr/sbin/httpd
restorecon reset /usr/sbin/httpd context system_u:object_r:unconfined_exec_t:s0-
>system_u:object_r:httpd_exec_t:s0
```

/usr/sbin/httpd に **httpd_exec_t** タイプのラベルが付けられていることを確認します。

```
~]$ ls -Z /usr/sbin/httpd
-rwxr-xr-x  root root system_u:object_r:httpd_exec_t:s0 /usr/sbin/httpd
```

9. root で次のコマンドを実行して **httpd** を再起動します。再起動したら、制限のある **httpd_t** ドメインで **httpd** が実行されていることを確認します。

```
~]# systemctl restart httpd.service
```

```
~]$ ps -eZ | grep httpd
system_u:system_r:httpd_t:s0    8883 ?        00:00:00 httpd
system_u:system_r:httpd_t:s0    8884 ?        00:00:00 httpd
system_u:system_r:httpd_t:s0    8885 ?        00:00:00 httpd
system_u:system_r:httpd_t:s0    8886 ?        00:00:00 httpd
system_u:system_r:httpd_t:s0    8887 ?        00:00:00 httpd
system_u:system_r:httpd_t:s0    8888 ?        00:00:00 httpd
system_u:system_r:httpd_t:s0    8889 ?        00:00:00 httpd
```

10. root で **testfile** を削除します。

```
~]# rm -i /var/www/html/testfile
rm: remove regular empty file `/var/www/html/testfile'? y
```

11. **httpd** を実行する必要がない場合は、root で以下のコマンドを実行して **httpd** を停止します。

```
~]# systemctl stop httpd.service
```

これらのセクションの例では、セキュリティー侵害を受けた制限のあるプロセス (SELinux で保護) からデータを保護する方法、およびセキュリティー侵害を受けた制限のないプロセス (SELinux によって保護されていない) のデータがいかに攻撃者にとってアクセスしやすいかについて示します。

3.3. 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。この Linux ユーザーマッピングは、root で **semanage login -l** コマンドを実行して表示されます。

```
~]# 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       *
system_u             system_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、これは、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
```

以下の手順では、新しい Linux ユーザーをシステムに追加し、そのユーザーを SELinux **unconfined_u** ユーザーにマッピングする方法を説明します。これは、デフォルトで Red Hat Enterprise Linux でそうであるように、root ユーザーが制限なしで実行されていることを前提としています。

手順3.4 新しい Linux ユーザーの SELinux unconfined_u ユーザーへのマッピング

1. root で以下のコマンドを入力し、**newuser** という名前の Linux ユーザーを作成します。

```
~]# useradd newuser
```

2. Linux **newuser** ユーザーにパスワードを割り当てるには、以下を実行します。root で以下のコマンドを実行します。

```
~]# passwd newuser
Changing password for user newuser.
New UNIX password: Enter a password
Retype new UNIX password: Enter the same password again
passwd: all authentication tokens updated successfully.
```

3. 現在のセッションからログアウトし、Linux **newuser** 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。以下のコマンドを入力して、Linux ユーザーのコンテキストを表示します。

```
[newuser@localhost ~]$ id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

注記

システムで **newuser** ユーザーが不要になった場合は、Linux **newuser** のセッションからログアウトし、アカウントにログインして root で **userdel -r newuser** コマンドを実行します。これにより、**newuser** が自分のホームディレクトリーとともに削除されます。

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
   sysadm_u
   system_u
   xguest_u
   root
   guest_u
   staff_u
   user_u
   unconfined_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 Linux 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 Linux ユーザーは、その対象である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。以下の表では、Red Hat Enterprise Linux で、Linux ユーザー向けの基本的な制限のあるドメインの例を確認できます。

表3.1 SELinux ユーザー機能

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

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

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

- デフォルトで、**staff_t**、**user_t**、**guest_t**、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。これらのユーザーが、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を実行できないようにするには、書き込みアクセス権のあるディレクトリーで **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。

  ユーザーがホームディレクトリーおよび **/tmp** でアプリケーションを実行するのを許可および禁止する方法については、「アプリケーションを実行するユーザーに対するブール値」 を参照してください。

- **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。

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 であることに注意してください。これは、Linux ユーザーに関連付けることはできません。また、**unconfined_u** および **root** は制限のないユーザーです。このような理由により、これらのユーザーは、前述の SELinux ユーザー機能の表には含まれていません。

前述の SELinux ユーザーとともに、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。詳細は、「タイプ」 を参照してください。
- **dbadm_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。詳細は、「タイプ」 および 「タイプ」 を参照してください。
- **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、以下のコマンドを実行します。

```
~]$ seinfo -r
```

前述したように、**seinfo** コマンドは、setools-console パッケージにより提供されます。これは、デフォルトではインストールされません。

3.3.1. sudo 移行および SELinux ロール

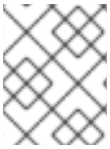
特定のケースでは、制限のあるユーザーが root 権限を必要とする管理タスクを実行する必要があります。これを実行するには、このような制限のあるユーザーが、**sudo** コマンドを使用して、*制限のある管理者* SELinux ロールを取得する必要があります。**sudo** コマンドは、信頼できるユーザーに管理アクセスを付与するために使用されます。信頼されるユーザーが、管理コマンドの前に **sudo** を付けると、この **ユーザー自身** のパスワードが要求されます。ユーザーが認証され、コマンドが許可されると、管理コマンドは root 権限で実行されているかのように実行されます。

表3.1「SELinux ユーザー機能」 に示されているように、**sudo** をデフォルトで使用できるのは、**staff_u** および **sysadm_u** SELinux の制限のあるユーザーのみになります。このようなユーザーが **sudo** でコマンドを実行すると、**/etc/sudoers** 設定ファイルで指定されたルールに基づいて、または、**/etc/sudoers.d/** ディレクトリーのそれぞれのファイル (このようなファイルが存在する場合) に基づいて、ロールを変更することができます。

sudo の詳細は、Red Hat Enterprise Linux 7 System Administrator's Guide の 『Gaining Privileges』 セクションを参照してください。

手順3.5 sudo 移行の設定

この手順では、新たに作成された *SELinux_user_u* の制限のあるユーザーを *default_role_r* から *administrator_r* 管理者ロールに移行するように **sudo** を設定する方法を説明します。



注記

既存の SELinux ユーザーに制限のある管理者ロールを設定するには、最初の 2 つの手順を省略します。

1. 新しい SELinux ユーザーを作成し、このユーザーにデフォルトの SELinux ロールと、補助制限のある管理者ロールを指定します。

```
~]# semanage user -a -r s0-s0:c0.c1023 -R "default_role_r administrator_r" SELinux_user_u
```

2. デフォルトの SElinux ポリシーコンテキストファイルを設定します。たとえば、SELinux ユーザー **staff_u** と同じ SELinux ルールを設定するには、 **staff_u** コンテキストファイルをコピーします。

```
~]# cp /etc/selinux/targeted/contexts/users/staff_u
/etc/selinux/targeted/contexts/users/SELinux_user_u
```

3. 新たに作成した SELinux ユーザーを既存の Linux ユーザーにマッピングします。

```
semanage login -a -s SELinux_user_u -rs0:c0.c1023 linux_user
```

4. **/etc/sudoers.d/** ディレクトリーに Linux ユーザーと同じ名前で新しい設定ファイルを作成し、以下の文字列を追加します。

```
~]# echo "linux_user ALL=(ALL) TYPE=administrator_t ROLE=administrator_r /bin/bash " >
/etc/sudoers.d/linux_user
```

5. **restorecon** ユーティリティーを使用して、 *linux_user* ホームディレクトリーのラベルを変更します。

```
~]# restorecon -FR -v /home/linux_user
```

6. 新たに作成した Linux ユーザーとしてシステムにログインし、ユーザーにデフォルトの SELinux ロールのラベルがあることを確認します。

```
~]$ id -Z
SELinux_user_u:default_role_r:SELinux_user_t:s0:c0.c1023
```

7. **sudo** を実行して、ユーザーの SELinux コンテキストを、 **/etc/sudoers.d/*linux_user*** で指定した補助 SELinux ロールに変更します。**sudo** で使用する **-i** オプションは、対話式シェルが実行される原因となっていました。

```
~]$ sudo -i
~]# id -Z
SELinux_user_u:administrator_r:administrator_t:s0:c0.c1023
```

default_role_r や *administrator_r* などのプレースホルダーをさらに理解するには、以下の例を参照してください。

例3.1 sudo 移行の設定

この例では、デフォルトで割り当てられたロール **staff_r** と、**confined_u** のロールを **staff_r** から **webadm_r** に変更するように設定された **sudo** を使用して、新しい SELinux ユーザー **confined_u** を作成します。

- root ユーザーとして、**sysadm_r** ロールまたは **unconfined_r** ロールで以下のコマンドをすべて入力します。

```
~]# semanage user -a -r s0-s0:c0.c1023 -R "staff_r webadm_r" confined_u
~]# cp /etc/selinux/targeted/contexts/users/staff_u
/etc/selinux/targeted/contexts/users/confined_u
~]# semanage login -a -s confined_u -rs0:c0.c1023 linux_user
```

```
~]# restorecon -FR -v /home/linux_user
~]# echo "linux_user ALL=(ALL) ROLE=webadm_r TYPE=webadm_t /bin/bash " >
/etc/sudoers.d/linux_user
```

- 新たに作成した Linux ユーザーとしてシステムにログインし、ユーザーにデフォルトの SELinux ロールのラベルがあることを確認します。

```
~]$ id -Z
confined_u:staff_r:staff_t:s0:c0.c1023
~]$ sudo -i
~]# id -Z
confined_u:webadm_r:webadm_t:s0:c0.c1023
```

第4章 SELINUX の使用

以下のセクションでは、Red Hat Enterprise Linux の主な SELinux パッケージの概要、パッケージのインストールと更新、使用するログファイル、主要な SELinux 設定ファイル、SELinux の有効化および無効化、SELinux モードの設定、ブール値の設定、ファイルとディレクトリーラベルの一時的および永続的な変更、**mount** コマンドでファイルシステムのラベルの上書き、NFS ボリュームのマウント、SELinux コンテキストの保存方法、ファイルとディレクトリーのコピー/アーカイブ時に SELinux コンテキストを保持する方法を説明します。

4.1. SELINUX パッケージ

Red Hat Enterprise Linux のフルインストールでは、インストール中に手動で除外されていない限り、SELinux パッケージがデフォルトでインストールされます。テキストモードで最小インストールを実行する場合、policycoreutils-python および policycoreutils-gui パッケージはデフォルトではインストールされません。また、デフォルトでは SELinux は Enforcing モードで実行され、SELinux Targeted ポリシーが使用されます。以下の SELinux パッケージがデフォルトでシステムにインストールされている。

- policycoreutils は、SELinux の運用および管理用に **restorecon**、**secon**、**setfiles**、**semodule**、**load_policy**、および **setsebool** などのユーティリティーを提供します。
- selinux-policy は、 基本的なディレクトリー構造、 **selinux-policy.conf** ファイル、RPM マクロを提供します。
- selinux-policy-targeted は、SELinux の Targeted ポリシーを提供します。
- libselinux - SELinux アプリケーションの API を提供します。
- libselinux-utils は **avcstat**、**getenforce**、**getsebool**、**matchpathcon**、**selinuxconlist**、**selinuxdefcon**、**selinuxenabled**、および **setenforce** ユーティリティーを提供します。
- libselinux-python は、SELinux アプリケーションを開発するための Python バインディングを提供します。

以下のパッケージはデフォルトではインストールされていませんが、オプションで **yum install *<package-name>*** コマンドを実行してインストールできます。

- selinux-policy-devel は、カスタムの SELinux ポリシーおよびポリシーモジュールを作成するユーティリティーを提供します。
- selinux-policy-doc では、さまざまなサービスで SELinux を完全に設定する方法を説明する man ページを提供します。
- selinux-policy-mls は、MLS(Multi-Level Security)SELinux ポリシーを提供します。
- setroubleshoot-server は、SELinux がアクセスが拒否されたときに生成される拒否メッセージを、**sealert** ユーティリティーで表示できる詳細の説明 (このパッケージでも提供) に変換します。
- setools-console は、Tresys Technology SETools ディストリビューション を提供します。これは、ポリシー、監査ログの監視とレポート、ファイルコンテキスト管理を分析およびクエリーするためのユーティリティーおよびライブラリーです。 setools パッケージは、SETools のメタパッケージです。setools-gui パッケージは、**apol** ユーティリティーおよび **seaudit** ユーティリティーを提供します。setools-console パッケージは、**sechecker**、**sediff**、**seinfo**、**sesearch** および **findcon** コマンドラインユーティリティー

を提供します。これらのユーティリティーの詳細は、Tresys Technology SETools ページを参照してください。setools および setools-gui パッケージは、Red Hat Network Optional チャンネルが有効にされている場合にのみ使用できることに注意してください。詳細は、Scope of Coverage Details を参照してください。

- mcstrans は、**s0-s0:c0.c1023** などのレベルを、**SystemLow-SystemHigh** などの読みやすい形式に変換します。
- policycoreutils-python は、SELinux の操作および管理用に **semanage**、**audit2allow**、**audit2why**、**chcat** などのユーティリティーを提供します。
- policycoreutils-gui は、SELinux を管理するためのグラフィカルユーティリティーである **system-config-selinux** を提供します。

4.2. どのログファイルが使用されるか

Red Hat Enterprise Linux では、デフォルトのパッケージ選択から削除されていない限り、dbus パッケージおよび audit パッケージがデフォルトでインストールされます。yum を使用して setroubleshoot-server がインストールされている必要があります (**yum install setroubleshoot-server** コマンドを使用します)。

auditd デーモンが実行している場合は、以下のような SELinux 拒否メッセージが、デフォルトで **/var/log/audit/audit.log** に書き込まれます。

```
type=AVC msg=audit(1223024155.684:49): avc:  denied  { getattr } for  pid=2000 comm="httpd"
path="/var/www/html/file1" dev=dm-0 ino=399185 scontext=unconfined_u:system_r:httpd_t:s0
tcontext=system_u:object_r:samba_share_t:s0 tclass=file
```

さらに、以下のようなメッセージは **/var/log/message** ファイルに書き込まれます。

```
May 7 18:55:56 localhost setroubleshoot: SELinux is preventing httpd (httpd_t) "getattr" to
/var/www/html/file1 (samba_share_t). For complete SELinux messages. run sealert -l de7e30d6-
5488-466d-a606-92c9f40d316d
```

Red Hat Enterprise Linux 7 では、**setroubleshootd** はサービスとして継続的に実行されなくなりました。ただし、AVC メッセージの分析には依然として使用されます。2 つの新しいプログラムは、必要に応じて **setroubleshoot** を開始するメソッドとして機能します。

- **sedispatch** ユーティリティーは、**監査** サブシステムの一部として実行します。AVC 拒否メッセージが返されると、**sedispatch** は **dbus** を使用してメッセージを送信します。これらのメッセージは、すでに実行中の場合は **setroubleshootd** に直接表示されます。これが実行されていない場合、**sedispatch** は自動的に起動します。
- **seapplet** ユーティリティーはシステムツールバーで実行され、**setroubleshootd** の dbus メッセージをリッスンします。通知バブルを起動し、ユーザーが AVC メッセージを確認できるようにします。

手順4.1 デーモンの自動起動

1. **auditd** デーモンおよび **rsyslog** デーモンが、システムの起動時に自動的に起動するように設定するには、root で次のコマンドを実行します。

```
~]# systemctl enable auditd.service
```

```
~]# systemctl enable rsyslog.service
```

2. デーモンが有効になっていることを確認するには、シェルプロンプトで以下のコマンドを入力します。

```
~]$ systemctl is-enabled auditd
enabled
```

```
~]$ systemctl is-enabled rsyslog
enabled
```

または、**systemctl status *service-name*.service** コマンドを使用して、コマンド出力で **enabled** キーワードを検索します。以下に例を示します。

```
~]$ systemctl status auditd.service | grep enabled
auditd.service - Security Auditing Service
   Loaded: loaded (/usr/lib/systemd/system/auditd.service; enabled)
```

systemd デーモンによるシステムサービスの管理方法の詳細は、System Administrator's Guide の Managing System Services の章を参照してください。

4.3. 主要設定ファイル

/etc/selinux/config ファイルは、主な SELinux 設定ファイルです。これは、SELinux が有効/無効であるか、SELinux モードと SELinux ポリシーを使用するかどうかを制御します。

```
# 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
```

SELINUX=

SELINUX オプションは、SELinux の無効化、有効化、およびモード (Enforcing または Permissive) が実行されるかどうかを設定します。

- **SELINUX=enforcing** を使用すると、SELinux ポリシーが適用され、SELinux ポリシールールが SELinux ポリシーに基づいてアクセスを拒否します。拒否メッセージがログに記録されます。
- **SELINUX=permissive** を使用すると、SELinux ポリシーは強制されません。SELinux はアクセスを拒否しませんが、SELinux を Enforcing モードで実行した場合に拒否されたであろうアクションの拒否がログに記録されます。
- **SELINUX=disabled** を使用すると、SELinux は無効になり、SELinux モジュールは Linux カーネルに登録されず、DAC ルールだけが使用されます。

SELINUXTYPE=

SELINUXTYPE オプションは、使用する SELinux ポリシーを設定します。Targeted ポリシーはデフォルトのポリシーです。MLS ポリシーを使用する場合は、このオプションを変更します。MLS ポリシーを有効にする方法は、「SELinux での MLS の有効化」 を参照してください。

4.4. 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」 で説明したように、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
Max kernel policy version:      30
```

注記

SELinux を Permissive モードで実行すると、ユーザーにファイルにラベルを誤って付けることができます。SELinux が無効になっている間に作成されるファイルにはラベル付けされません。この動作により、ファイルが誤ってラベル付けされないか、または全くラベル付けされないため、Enforcing モードへの変更時に問題が発生します。ラベルが誤って設定されていたり、ファイルにラベルが付いていないために問題が発生するのを防ぐために、Disabled 状態から Permissive モードまたは Enforcing モードに変更すると、ファイルシステムのラベルが自動的に再設定されます。

4.4.1. SELinux の有効化

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。

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、Red Hat は以下の手順に従うことを推奨しています。

1. SELinux を Permissive モードで有効にします。詳細は、「Permissive モードに設定する場合:」を参照してください。
2. システムを再起動します。
3. SELinux 拒否メッセージを確認します。詳細は、「拒否の検索と表示」を参照してください。
4. 拒否がない場合は、Enforcing モードに切り替えます。詳細は、「Enforcing モードに設定する場合:」 を参照してください。

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。詳細は、「制限のないプロセス」 を参照してください。
- アプリケーションに新しいポリシーを記述します。詳細は、ナレッジベースの記事 カスタム SELinux ポリシーの作成 を参照してください。

4.4.1.1. Permissive モードに設定する場合:

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。

モードを Permissive へ永続的に変更するには、以下の手順に従ってください。

手順4.2 Permissive モードへの変更

1. 以下のように **/etc/selinux/config** ファイルを編集します。

```
# 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
```

2. システムを再起動します。

```
~]# reboot
```

4.4.1.2. Enforcing モードに設定する場合:

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。Red Hat Enterprise Linux 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。

SELinux が無効の場合は、以下の手順に従って、再度 Enforcing モードに変更してください。

手順4.3 Enforcing モードへの変更

この手順では、selinux-policy-targeted、selinux-policy、libselinux、libselinux-python、libselinux-utils、policycoreutils、および policycoreutils-python パッケージがインストールされていることを前提としています。パッケージがインストールされていることを確認するには、以下のコマンドを使用します。

```
rpm -q package_name
```

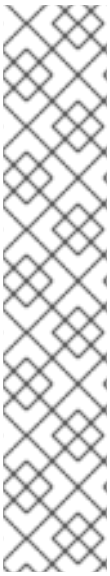
1. 以下のように **/etc/selinux/config** ファイルを編集します。

```
# 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
```

2. 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。



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 -ts today
```

または、setroubleshoot-server パッケージがインストールされている場合は、root で以下のコマンドを入力します。

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

SELinux が一部のアクションを拒否する場合は、*11章 トラブルシューティング*でトラブルシューティングを参照してください。

モードの一時的な変更は、「SELinux のステータスおよびモード」 で説明されています。

4.4.2. 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」 に記載されている 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 モードに設定する場合:」 を参照してください。

SELinux を永続的に無効にするには、以下の手順を行います。

手順4.4 SELinux の無効化

1. **/etc/selinux/config** ファイルに **SELINUX=disabled** を設定します。

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

2. システムを再起動します。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
```

4.5. システムの起動時での 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

このパラメーターを設定すると、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たことを認識し、**/.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
```

システムのラベリングに多数のエラーが含まれる場合は、自動再ラベルが正常に実行されるために Permissive モードで起動する必要がある場合があります。

checkreqprot などの追加の SELinux 関連のカーネルブートパラメーターは、**/usr/share/doc/kernel-doc-*<KERNEL_VER>*/Documentation/kernel-parameters.txt** ファイルを参照してください。このドキュメントは、kernel-doc パッケージでインストールされます。*<KERNEL_VER>* 文字列をインストール済みカーネルのバージョン番号に置き換えます。以下に例を示します。

```
~]# yum install kernel-doc
~]$ less /usr/share/doc/kernel-doc-3.10.0/Documentation/kernel-parameters.txt
```

4.6. ブール値

ブール値を使用すると、SELinux ポリシーの記述に関する知識がなくても、SELinux ポリシーの一部をランタイム時に変更できます。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。

4.6.1. ブール値の一覧表示

ブール値の一覧の場合、それぞれが何であるかの説明と、そのものがオンまたはオフであるかどうかを確認するには、Linux root ユーザーとして **semanage boolean -l** コマンドを実行します。以下の例は、簡潔化のために出力が短くしているので、すべてのブール値を一覧表示していません。

```
~]# semanage boolean -l
SELinux boolean                State  Default Description


smartmon_3ware                 (off  ,  off)  Determine whether smartmon can...
mpd_enable_homedirs            (off  ,  off)  Determine whether mpd can traverse...
```

注記

より詳細な説明を取得するには、selinux-policy-devel パッケージをインストールします。

SELinux boolean 列には、ブール値名が一覧表示されます。**Description** コラムには、ブール値がオンまたはオフであるかと、その実行内容が記載されています。

getsebool -a コマンドはブール値 (on または off) を一覧表示しますが、それぞれの説明は表示されません。以下の例は、すべてのブール値を一覧表示していません。

```
~]$ getsebool -a
cvs_read_shadow --> off
daemons_dump_core --> on
```

getsebool *boolean-name* コマンドを実行して、*boolean-name* ブール値のステータスのみを一覧表示します。

```
~]$ getsebool cvs_read_shadow
cvs_read_shadow --> off
```

スペースで区切られたリストを使用して、複数のブール値をリストします。

```
~]$ getsebool cvs_read_shadow daemons_dump_core
cvs_read_shadow --> off
daemons_dump_core --> on
```

4.6.2. ブール値の設定

setsebool *boolean_name* on/off 形式の **setsebool** ユーティリティーを実行し、ブール値を有効または無効にします。

以下の例は、**httpd_can_network_connect_db** ブール値の設定を示しています。

手順4.5 ブール値の設定

1. デフォルトでは、**httpd_can_network_connect_db** ブール値はオフになっています。Apache HTTP Server スクリプトおよびモジュールがデータベースサーバーに接続できません。

   ```
   ~]$ getsebool httpd_can_network_connect_db
   httpd_can_network_connect_db --> off
   ```

2. Apache HTTP Server スクリプトおよびモジュールがデータベースサーバーに一時的に接続できるようにするには、root で以下のコマンドを入力します。

   ```
   ~]# setsebool httpd_can_network_connect_db on
   ```

3. **getsebool** ユーティリティーを使用して、ブール値が有効であることを確認します。

   ```
   ~]$ getsebool httpd_can_network_connect_db
   httpd_can_network_connect_db --> on
   ```

   これにより、Apache HTTP Server スクリプトおよびモジュールはデータベースサーバーに接続できます。

4. この変更は、再起動後は維持されません。再起動後も変更を永続化するには、root で **setsebool -P *boolean-name* on** コマンドを実行します。[3]

   ```
   ~]# setsebool -P httpd_can_network_connect_db on
   ```

4.6.3. シェル自動完了

getsebool ユーティリティー、**setsebool** ユーティリティー、および **semanage** ユーティリティーを使用すると、シェルの自動完了を使用できます。**getsebool** と **setsebool** のオートコンプリートを使用して、コマンドラインパラメーターとブール値の両方を完了します。コマンドラインパラメーターのみを一覧表示するには、コマンド名の後にハイフン文字 ("-") を追加して、**Tab** キーを押します。

```
~]# setsebool -[Tab]
-P
```

ブール値を完了するには、ブール値名の記述を開始して **Tab** キーを押します。

```
~]$ getsebool samba_[Tab]
samba_create_home_dirs   samba_export_all_ro      samba_run_unconfined
samba_domain_controller  samba_export_all_rw      samba_share_fusefs
samba_enable_home_dirs   samba_portmapper         samba_share_nfs
```

```
~]# setsebool -P virt_use_[Tab]
virt_use_comm     virt_use_nfs      virt_use_sanlock
virt_use_execmem  virt_use_rawip    virt_use_usb
virt_use_fusefs   virt_use_samba    virt_use_xserver
```

semanage ユーティリティーは、複数のコマンドライン引数で1つずつ完了したもので使用されます。**semanage** コマンドの最初の引数はオプションです。これは、管理対象の SELinux ポリシーの一部を指定します。

```
~]# semanage [Tab]
boolean     export      import      login       node        port
dontaudit   fcontext    interface   module      permissive  user
```

次に、1つ以上のコマンドラインパラメーターに従います。

```
~]# semanage fcontext -[Tab]
-a          -D          --equal      --help       -m          -o
--add       --delete    -f           -l           --modify    -S
-C          --deleteall  --ftype     --list       -n          -t
-d          -e          -h           --locallist  --noheading  --type
```

最後に、ブール値、SELinux ユーザー、ドメインなどの特定の SELinux エントリーの名前を入力します。エントリーの入力を開始して、**Tab** キーを押します。

```
~]# semanage fcontext -a -t samba<tab>
samba_etc_t                  samba_secrets_t
sambagui_exec_t              samba_share_t
samba_initrc_exec_t          samba_unconfined_script_exec_t
samba_log_t                  samba_unit_file_t
samba_net_exec_t
```

コマンドラインパラメーターは、以下のコマンドでチェーンできます。

```
~]# semanage port -a -t http_port_t -p tcp 81
```

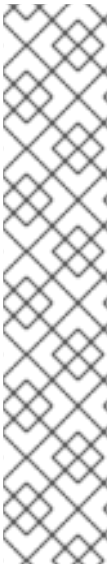
4.7. SELINUX コンテキスト - ファイルのラベル付け

SELinux を実行しているシステムでは、セキュリティー関連の情報を表す方法で、すべてのプロセスとファイルにラベルが付けられます。この情報は SELinux コンテキストと呼ばれます。ファイルの場合は、ls -Z コマンドを使用して表示されます。

```
~]$ ls -Z file1
-rw-rw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0 file1
```

この例では、SELinux はユーザー (unconfined_u)、ロール (object_r)、タイプ (user_home_t)、およ

びレベル (s0) を提供します。この情報は、アクセス制御の決定を行うために使用されます。DAC システムでは、アクセスは Linux ユーザーおよびグループ ID に基づいて制御されます。SELinux ポリシールールは、DAC ルールの後にチェックされます。DAC ルールがアクセスを拒否する場合は、SELinux ポリシールールは使用されません。



注記

デフォルトでは、新規作成されたファイルおよびディレクトリーは親ディレクトリーの SELinux タイプを継承します。たとえば、**etc_t** タイプのラベルが付けられた **/etc** ディレクトリーに新しいファイルを作成する場合、新しいファイルは同じタイプを継承します。

```
~]$ ls -dZ - /etc
drwxr-xr-x. root root system_u:object_r:etc_t:s0       /etc
```

```
~]# touch /etc/file1
```

```
~]# ls -lZ /etc/file1
-rw-r--r--. root root unconfined_u:object_r:etc_t:s0   /etc/file1
```

SELinux は、**chcon**、**semanage fcontext**、**restorecon**、および **matchpathcon** など、ファイルシステムのラベリングを管理するための複数のコマンドを提供します。

4.7.1. 一時的な変更: chcon

chcon コマンドは、ファイルの SELinux コンテキストを変更します。ただし、**chcon** コマンドを使用して行った変更は、ファイルシステムのラベル変更や **restorecon** コマンドの実行後も維持されません。SELinux ポリシーは、ユーザーが任意のファイルの SELinux コンテキストを変更できるかどうかを制御します。**chcon** を使用すると、ユーザーは変更する SELinux コンテキストのすべてまたは一部を提供します。ファイルタイプは、SELinux がアクセスを拒否する一般的な原因です。

クイックリファレンス

- **chcon -t *type file-name*** コマンドを実行して、ファイルタイプを変更します。*type* は SELinux タイプです。たとえば、**httpd_sys_content_t** で、*file-name* はファイルまたはディレクトリー名になります。

```
~]$ chcon -t httpd_sys_content_t file-name
```

- **chcon -R -t *type directory-name*** コマンドを実行して、ディレクトリーのタイプとそのコンテンツを変更します。*type* は SELinux タイプです。たとえば、**httpd_sys_content_t** で、*directory-name* はディレクトリー名になります。

```
~]$ chcon -R -t httpd_sys_content_t directory-name
```

手順4.6 ファイルまたはディレクトリーのタイプの変更

次の手順は、タイプの変更を示しており、SELinux コンテキストの他の属性は示していません。このセクションの例では、**file1** がディレクトリーである場合など、ディレクトリーに対して同じように動作します。

1. ホームディレクトリーに移動します。

2. 新しいファイルを作成して、その SELinux コンテキストを表示します。

```
~]$ touch file1
```

```
~]$ ls -Z file1
-rw-rw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0 file1
```

この例では、**file1** の SELinux コンテキストには、SELinux の**unconfined_u** ユーザー、**object_r** ロール、**user_home_t** タイプ、および **s0** レベルが含まれます。SELinux コンテキストの各部分の説明は、*2章SELinux コンテキスト* を参照してください。

3. 次のコマンドを実行して、タイプを **samba_share_t** に変更します。**-t** オプションはタイプを変更します。次に、変更を表示します。

```
~]$ chcon -t samba_share_t file1
```

```
~]$ ls -Z file1
-rw-rw-r--  user1 group1 unconfined_u:object_r:samba_share_t:s0 file1
```

4. 以下のコマンドを使用して、**file1** ファイルの SELinux コンテキストを復元します。変更を確認するには、**-v** オプションを使用します。

```
~]$ restorecon -v file1
restorecon reset file1 context unconfined_u:object_r:samba_share_t:s0-
>system_u:object_r:user_home_t:s0
```

この例では、前のタイプ **samba_share_t** は正しいタイプ **user_home_t** タイプに復元されます。Targeted ポリシー (Red Hat Enterprise Linux のデフォルトの SELinux ポリシー) を使用する場合、**restorecon** コマンドは **/etc/selinux/targeted/contexts/files/** ディレクトリーのファイルを読み取り、どの SELinux コンテキストファイルが存在するかを確認します。

手順4.7 ディレクトリーとそのコンテンツタイプの変更

以下の例は、新しいディレクトリーを作成し、そのディレクトリーのファイルタイプを Apache HTTP Server が使用するタイプに変更します。この例での設定は、Apache HTTP サーバーが、(**/var/www/html/** の代わりに) 別のドキュメント root を使用する場合に使用します。

1. root ユーザーとして、このディレクトリー内に新しい **web/** ディレクトリーを作成し、3 つの空のファイル (**file1**、**file2**、および **file3**) を作成します。**web/** ディレクトリーおよびそのファイルには、**default_t** タイプのラベルが付けられます。

```
~]# mkdir /web
```

```
~]# touch /web/file{1,2,3}
```

```
~]# ls -dZ /web
drwxr-xr-x  root root unconfined_u:object_r:default_t:s0 /web
```

```
~]# ls -lZ /web
-rw-r--r--  root root unconfined_u:object_r:default_t:s0 file1
-rw-r--r--  root root unconfined_u:object_r:default_t:s0 file2
-rw-r--r--  root root unconfined_u:object_r:default_t:s0 file3
```

2. root で次のコマンドを実行して、**web/** ディレクトリー (とそのコンテンツ) のタイプを **httpd_sys_content_t** に変更します。

```
~]# chcon -R -t httpd_sys_content_t /web/
```

```
~]# ls -dZ /web/
drwxr-xr-x  root root unconfined_u:object_r:httpd_sys_content_t:s0 /web/
```

```
~]# ls -lZ /web/
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file1
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file2
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file3
```

3. デフォルトの SELinux コンテキストを復元するには、root で **restorecon** ユーティリティーを使用します。

```
~]# restorecon -R -v /web/
restorecon reset /web context unconfined_u:object_r:httpd_sys_content_t:s0-
>system_u:object_r:default_t:s0
restorecon reset /web/file2 context unconfined_u:object_r:httpd_sys_content_t:s0-
>system_u:object_r:default_t:s0
restorecon reset /web/file3 context unconfined_u:object_r:httpd_sys_content_t:s0-
>system_u:object_r:default_t:s0
restorecon reset /web/file1 context unconfined_u:object_r:httpd_sys_content_t:s0-
>system_u:object_r:default_t:s0
```

chcon の詳細は、chcon(1) man ページを参照してください。

注記

Type Enforcement は、SELinux の Targeted ポリシーで使用される主要なパーミッション制御です。ほとんどの場合、SELinux のユーザーとロールは無視できます。

4.7.2. 永続的な変更 - semanage fcontext

semanage fcontext コマンドは、ファイルの SELinux コンテキストを変更するのに使用します。新しく作成したファイルおよびディレクトリーのコンテキストを表示するには、root で以下のコマンドを実行します。

```
~]# semanage fcontext -C -l
```

semanage fcontext が加えた変更は、以下のユーティリティーで使用されます。**setfiles** ユーティリティーは、ファイルシステムに再ラベル付けされ、**restorecon** ユーティリティーがデフォルトの SELinux コンテキストを復元するときに使用されます。つまり、ファイルシステムが再ラベル付けされても、**semanage fcontext** による変更は永続します。SELinux ポリシーは、ユーザーが任意のファイルの SELinux コンテキストを変更できるかどうかを制御します。

クイックリファレンス

ファイルシステムの再ラベル付け後に SELinux コンテキストを変更するには、以下のコマンドを実行します。

1. 次のコマンドを入力します。ファイルまたはディレクトリーのフルパスを使用することを忘れないでください。

```
~]# semanage fcontext -a options file-name|directory-name
```

2. **restorecon** ユーティリティーを使用して、コンテキストの変更を適用します。

```
~]# restorecon -v file-name|directory-name
```

semanage fcontext での正規表現の使用

semanage fcontext が適切に機能するようにするためには、完全修飾パスまたは Perl-compatible regular expressions (PCRE) を使用できます。使用されている唯一のPCRE フラグは **PCRE2_DOTALL** です。これにより、**.** ワイルドカードは、改行を含むあらゆるものにマッチします。パスを表す文字列はバイトとして処理されます。つまり、ASCII 以外の文字は、1つのワイルドカードで一致しません。

semanage fcontext を使用して指定したファイルコンテキストの定義は、定義方法が逆順で評価されることに注意してください。定義方法は、stem の長さに関係なく、直近のエントリーが最初に評価されます。**file_contexts.local** に保存されているローカルファイルコンテキストの変更の優先度は、ポリシーモジュールで指定されているものよりも高くなります。つまり、指定したファイルパスが **file_contexts.local** で一致するものが見つかるたびに、その他のファイルコンテキスト定義は考慮されません。

重要

semanage fcontext コマンドを使用して指定したファイルコンテキスト定義は、他のすべてのファイルコンテキスト定義を効果的に上書きします。したがって、ファイルシステムの他の部分に意図せず影響を与えないように、すべての正規表現は可能な限り具体的にする必要があります。

file-context 定義とフラグで使用される正規表現のタイプの詳細は、man ページの **semanage-fcontext(8)** を参照してください。

手順4.8 ファイルまたはディレクトリーのタイプの変更

以下の例は、ファイルのタイプを変更する方法を示していますが、SELinux コンテキストの他の属性はありません。この例は、**file1** がディレクトリーであった場合など、ディレクトリーに対しても同じように機能します。

1. root ユーザーで、**/etc** ディレクトリーに新しいファイルを作成します。デフォルトでは、**/etc** に新規に作成されたファイルには **etc_t** タイプのラベルが付けられます。

```
~]# touch /etc/file1
```

```
~]$ ls -Z /etc/file1
-rw-r--r--  root root unconfined_u:object_r:etc_t:s0       /etc/file1
```

ディレクトリーの情報を一覧表示するには、次のコマンドを使用します。

```
~]$ ls -dZ directory_name
```

2. root で次のコマンドを実行して、**file1** タイプを **samba_share_t** に変更します。**-a** オプションは新しいレコードを追加し、**-t** オプションはタイプ (**samba_share_t**) を定義します。このコマンドを実行すると、タイプを直接変更しません。**file1** には、まだ **etc_t** タイプのラベルが付けられています。

```
~]# semanage fcontext -a -t samba_share_t /etc/file1
```

```
~]# ls -Z /etc/file1
-rw-r--r--  root root unconfined_u:object_r:etc_t:s0       /etc/file1
```

```
~]$ semanage fcontext -C -l
/etc/file1    unconfined_u:object_r:samba_share_t:s0
```

3. **root で restorecon ユーティリティーを使用してタイプを変更します。semanage により、/etc/file1 の file_contexts.local にエントリーが追加されたため、restorecon によりタイプが samba_share_t に変わります。**

```
~]# restorecon -v /etc/file1
restorecon reset /etc/file1 context unconfined_u:object_r:etc_t:s0-
>system_u:object_r:samba_share_t:s0
```

手順4.9 ディレクトリーとそのコンテンツタイプの変更

以下の例は、新しいディレクトリーを作成し、そのディレクトリーのファイルタイプを Apache HTTP サーバーが使用するタイプに変更します。この例の設定は、Apache HTTP サーバーで、/var/www/html/ の代わりに別のドキュメント root を使用する場合に使用します。

1. **root ユーザーとして、このディレクトリー内に新しい web/ ディレクトリーを作成し、3 つの空のファイル (file1、file2、および file3) を作成します。web/ ディレクトリーおよびそのファイルには、default_t タイプのラベルが付けられます。**

```
~]# mkdir /web
```

```
~]# touch /web/file{1,2,3}
```

```
~]# ls -dZ /web
drwxr-xr-x  root root unconfined_u:object_r:default_t:s0 /web
```

```
~]# ls -lZ /web
-rw-r--r--  root root unconfined_u:object_r:default_t:s0 file1
-rw-r--r--  root root unconfined_u:object_r:default_t:s0 file2
-rw-r--r--  root root unconfined_u:object_r:default_t:s0 file3
```

2. **root で次のコマンドを実行して、web/ ディレクトリーのタイプとその中のファイルを httpd_sys_content_t に変更します。-a オプションは新しいレコードを追加し、-t オプションはタイプを定義します (httpd_sys_content_t)。"/web(/.*)?" 正規表現を使用すると、semanage は、web/ およびその中のファイルに変更を適用します。このコマンドを実行しても、タイプを直接変更しません。web/ ファイルおよびこのファイルには、default_t タイプのラベルが付けられます。**

```
~]# semanage fcontext -a -t httpd_sys_content_t "/web(/.*)?"
```

```
~]$ ls -dZ /web
drwxr-xr-x  root root unconfined_u:object_r:default_t:s0 /web
```

```
~]$ ls -lZ /web
-rw-r--r--  root root unconfined_u:object_r:default_t:s0 file1
-rw-r--r--  root root unconfined_u:object_r:default_t:s0 file2
-rw-r--r--  root root unconfined_u:object_r:default_t:s0 file3
```

semanage fcontext -a -t httpd_sys_content_t "/web(/.*)?" コマンドは、以下のエントリーを /etc/selinux/targeted/contexts/files/file_contexts.local に追加します。

```
/web(/.*)?    system_u:object_r:httpd_sys_content_t:s0
```

3. **root で restorecon ユーティリティーを使用して、web/ のタイプと、その中のすべてのファイルを変更します。-R は再帰用です。これは、web/ の下にあるすべてのファイルおよびディレクトリーに httpd_sys_content_t タイプのラベルが付けられることを意味します。semanage により、/web(/.*)? の file.contexts.local にエントリーが追加されたため、restorecon によりタイプが httpd_sys_content_t に変わります。**

```
~]# restorecon -R -v /web
restorecon reset /web context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
restorecon reset /web/file2 context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
restorecon reset /web/file3 context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
restorecon reset /web/file1 context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
```

デフォルトでは、新規作成されたファイルおよびディレクトリーは親ディレクトリーの SELinux タイプを継承する点に注意してください。

手順4.10 追加したコンテキストの削除

以下の例では、SELinux コンテキストの追加および削除を示しています。/web(/.*)? のように、コンテキストが正規表現の一部である場合は、正規表現を引用符で囲んでください。

```
~]# semanage fcontext -d "/web(/.*)?"
```

1. **コンテキストを削除するには、root で次のコマンドを実行します。file_contexts.local の最初の部分は *file-name|directory-name* です。**

```
~]# semanage fcontext -d file-name|directory-name
```

以下は、file_contexts.local のコンテキストの例になります。

```
/test    system_u:object_r:httpd_sys_content_t:s0
```

最初のパートは test です。restorecon の実行後、またはファイルシステムの再ラベル付け後に、test/ ディレクトリーにhttpd_sys_content_t のラベルが付けられないようにするには、root で次のコマンドを実行して、file_contexts.local からコンテキストを削除します。

```
~]# semanage fcontext -d /test
```

2. **root で、restorecon ユーティリティーを使用して、デフォルトの SELinux コンテキストを復元します。**

semanage の詳細は、semanage(8) および semanage-fcontext(8) の man ページを参照してください。

重要

semanage fcontext -a で SELinux コンテキストを変更する場合は、ファイルシステムの再ラベル付け後や restorecon コマンドの実行後にファイルに間違ったラベル付けが行われないように、ファイルまたはディレクトリーのフルパスを使用します。

4.7.3. ファイルコンテキストの決定方法

ファイルコンテキストの判断は、システムセキュリティーポリシー (.fcファイル) で指定されているファイルコンテキスト定義に基づいて行われます。semanage は、システムポリシーに基づいて、file_contexts.homedirs ファイルと file_contexts ファイルを生成します。

システム管理者は、semanage fcontext コマンドを使用して、ファイルコンテキストの定義をカスタマイズできます。このようなカスタマイズは、file_contexts.local ファイルに保存されます。

matchpathcon または restorecon などのラベリングユーティリティーが、指定されたパスに適したラベルを決定する場合は、最初にローカル変更 (file_contexts.local) を検索します。一致するパターンが見つからない場合は、file_contexts.homedirsファイルが検索され、最後にfile_contextsファイルが検索されます。ただし、指定したファイルパスに一致するものが見つかると、検索が終了します。このユーティリティーは、追加の file-context 定義を検索します。つまり、ホームディレクトリー関連のファイルコンテキストの優先度は、その他のファイルコンテキストの優先度よりも高く、ローカルのカスタマイズによりシステムポリシーが上書きされます。

system policy で指定されるファイルコンテキストの定義 (file_contexts.homedirs ファイルおよび file_contexts ファイルのコンテンツ) は、評価前に stem (ワイルドカードの前のパスの接頭辞) の長さでソートされます。つまり、最も具体的なパスが選択されます。ただし、semanage fcontext を使用して指定したファイルコンテキスト定義は、定義された方法とは逆の順序で評価されます。つまり、直近のエントリーは、stem の長さに関係なく最初に評価されます。

詳細は、以下を参照してください。

- chcon を使用してファイルコンテキストを変更する方法は、「一時的な変更: chcon」 を参照してください。
- semanage fcontext を使用したファイルコンテキストの定義の変更および追加は、「永続的な変更 - semanage fcontext」 を参照してください。
- システムポリシーオペレーションを介してファイルコンテキスト定義を変更して追加する場合は、「SELinux ラベルの維持」 または 「SELinux ポリシーモジュールの優先付けと無効化」 を参照してください。

4.8. FILE_T タイプおよび DEFAULT_T タイプ

拡張属性 (EA) をサポートするファイルシステムを使用する場合、file_t タイプは EA 値がまだ割り当てられていないファイルのデフォルトタイプになります。このタイプはこの目的にのみ使用され、正しくラベル付けされたファイルシステムには存在しません。SELinux を実行しているシステム上のすべてのファイルには適切な SELinux コンテキストがあり、file_t タイプは file-context 設定では使用されません。[4].

default_t タイプは、file-context 設定のパターンと一致しないファイルで使用されます。そのため、このようなファイルはディスクにコンテキストを持たないファイルから区別でき、通常は制限のあるドメインからアクセスできなくなります。たとえば、mydirectory/ などの新しい最上位のディレクトリーを

作成した場合には、このディレクトリーには default_t タイプのラベルが付けられます。サービスがこのディレクトリーにアクセスする必要がある場合は、この場所の file-contexts 設定を更新する必要があります。file-context 設定にコンテキストを追加する方法は、「永続的な変更 - semanage fcontext」 を参照してください。

4.9. ファイルシステムのマウント

デフォルトでは、拡張属性に対応するファイルシステムをマウントすると、各ファイルのセキュリティーコンテキストは、ファイルの security.selinux 拡張属性から取得されます。拡張属性に対応していないファイルシステムのファイルには、ファイルシステムタイプに基づいて、ポリシー設定から単一のデフォルトのセキュリティーコンテキストが割り当てられます。

mount -o context コマンドを使用して、既存の拡張属性を上書きするか、拡張属性に対応していないファイルシステムに別のデフォルトコンテキストを指定します。これは、複数のシステムで使用されているリムーバブルメディアなど、正しい属性を提供するファイルシステムを信頼しない場合に役立ちます。**mount -o context** コマンドは、ファイル割り当てテーブル (FAT) ボリュームや NFS ボリュームなど、拡張属性に対応していないファイルシステムのラベリングにも使用できます。**context** オプションで指定したコンテキストはディスクに書き込まれません。元のコンテキストは保持され、ファイルシステムに拡張属性がある場合は、**context** なしでマウントすると確認できます。

ファイルシステムのラベリングの詳細は、James Morris の記事 "Filesystem Labeling in SELinux" : http://www.linuxjournal.com/article/7426 を参照してください。

4.9.1. コンテキストマウント

指定したコンテキストでファイルシステムをマウントし、既存のコンテキストが存在する場合は上書きするか、拡張属性に対応していないファイルシステムに別のデフォルトコンテキストを指定するには、root ユーザーで、必要なファイルシステムをマウントする際に **mount -o context=*SELinux_user:role:type:level*** コマンドを使用します。コンテキストの変更はディスクには書き込まれません。デフォルトで、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。一般的なポリシーでは、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。追加のマウントオプションがないと、Apache HTTP サーバーなどの他のサービスを使用した NFS ボリュームの共有を防ぐことができます。以下の例は、Apache HTTP Server を使用して共有できるように NFS ボリュームをマウントしています。

```
~]# mount server:/export /local/mount/point -o \ context="system_u:object_r:httpd_sys_content_t:s0"
```

このファイルシステムで新規作成されたファイルおよびディレクトリーは、**-o context** で指定した SELinux コンテキストがあるようです。ただし、これらの変更はディスクには書き込まれないため、このオプションで指定したコンテキストはマウント間で維持されません。したがって、必要なコンテキストを維持するには、すべてのマウント時に指定したコンテキストでこのオプションを使用する必要があります。コンテキストマウントを永続化する方法の詳細は、「コンテキストマウントの永続化」 を参照してください。

Type Enforcement は、SELinux の Targeted ポリシーで使用される主要なパーミッション制御です。大半の部分では、SELinux のユーザーとロールを無視できます。したがって、**-o context** コンテキストで SELinux コンテキストを上書きする場合は、SELinux の **system_u** ユーザーおよび **object_r** ロールを使用し、タイプに集中します。MLS ポリシーまたは複数カテゴリーのセキュリティーを使用していない場合は、**s0** レベルを使用します。

注記

ファイルシステムが **context** オプションでマウントされると、ユーザーおよびプロセスによるコンテキストの変更が禁止されます。たとえば、**context** オプションでマウントされたファイルシステムで **chcon** コマンドを実行すると、**Operation not supported** のエラーが発生します。

4.9.2. デフォルトコンテキストの変更

「file_t タイプおよび default_t タイプ」 で説明されているように、拡張属性をサポートするファイルシステムでは、ディスク上の SELinux コンテキストのないファイルにアクセスすると、SELinux ポリシーで定義されているデフォルトのコンテキストがあるかのように処理されます。一般的なポリシーでは、このデフォルトのコンテキストは **file_t** タイプを使用します。別のデフォルトコンテキストを使用する場合は、**defcontext** オプションでファイルシステムをマウントします。

以下の例では、**/dev/sda2** 上で新しく作成したファイルシステムを、新しく作成した **test/** ディレクトリーにマウントします。**/etc/selinux/targeted/contexts/files/** には、**test/** ディレクトリーのコンテキストを定義するルールがないことを前提とします。

```
~]# mount /dev/sda2 /test/ -o defcontext="system_u:object_r:samba_share_t:s0"
```

この例では、以下のように設定されています。

- **defcontext** オプションでは、**system_u:object_r:samba_share_t:s0** が "the default security context for unlabeled files" であると定義されています。[5].
- マウント時に、ファイルシステムの root ディレクトリー (**test/**) は、**defcontext** で指定されたコンテキストでラベル付けされているかのように扱われます (このラベルはディスクには保存されません)。これは、**test/** で作成されるファイルのラベリングに影響します。新しいファイルは **samba_share_t** タイプを継承します。これらのラベルはディスクに保存されます。
- ファイルシステムが **defcontext** でマウントされているときに **test/** で作成されたファイルは、そのラベルが保持されます。

4.9.3. NFS ボリュームのマウント

デフォルトで、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。一般的なポリシーでは、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。ポリシー設定によっては、Apache HTTP Server や MariaDB などのサービスは、**nfs_t** タイプのラベルが付けられたファイルを読み取ることができません。これにより、このタイプのラベルが付けられたファイルシステムがマウントされ、他のサービスによって読み込まれたりエクスポートしたりできなくなる可能性があります。

NFS ボリュームをマウントして、そのファイルシステムを別のサービスで読み取りまたはエクスポートする場合は、マウント時に **context** オプションを使用して **nfs_t** タイプを上書きします。以下のコンテキストオプションを使用して NFS ボリュームをマウントし、Apache HTTP サーバーを使用して共有できるようにします。

```
~]# mount server:/export /local/mount/point -o context="system_u:object_r:httpd_sys_content_t:s0"
```

ただし、これらの変更はディスクには書き込まれないため、このオプションで指定したコンテキストは、マウント間で維持されません。したがって、必要なコンテキストを維持するには、すべてのマウント時に指定したコンテキストでこのオプションを使用する必要があります。コンテキストマウントを永続化する方法の詳細は、「コンテキストマウントの永続化」 を参照してください。

context オプションを使用してファイルシステムをマウントする代わりに、ブール値を有効にして、nfs_t タイプのラベルが付けられたファイルシステムへのサービスアクセスを許可できます。nfs_t タイプへのサービスアクセスを許可するブール値の設定方法については、パートII「制限のあるサービスの管理」 を参照してください。

4.9.4. 複数の NFS マウント

同じ NFS エクスポートから複数のマウントをマウントする場合に、異なるコンテキストで各マウントの SELinux コンテキストを上書きしようとすると、以降のマウントコマンドに失敗します。以下の例では、NFS サーバーに、web/ と database/ の 2 つのサブディレクトリーを持つ 1 つのエクスポート export/ があります。以下のコマンドは、1 つの NFS エクスポートから 2 つのマウントを試み、各マウントに対してコンテキストを上書きしようとします。

```
~]# mount server:/export/web /local/web -o context="system_u:object_r:httpd_sys_content_t:s0"
```

```
~]# mount server:/export/database /local/database -o context="system_u:object_r:mysqld_db_t:s0"
```

2 番目のマウントコマンドが失敗し、次のログが /var/log/messages に記録されます。

```
kernel: SELinux: mount invalid.  Same superblock, different security settings for (dev 0:15, type nfs)
```

1 つの NFS エクスポートから複数のマウントをマウントするには、各マウントに異なるコンテキストがある場合は、-o nosharecache,context オプションを使用します。以下の例では、1 つの NFS エクスポートから複数のマウントをマウントし、各マウントに異なるコンテキストを使用しています (各マウントに 1 つのサービスアクセスを許可します)。

```
~]# mount server:/export/web /local/web -o
nosharecache,context="system_u:object_r:httpd_sys_content_t:s0"
```

```
~]# mount server:/export/database /local/database -o \
nosharecache,context="system_u:object_r:mysqld_db_t:s0"
```

この例では、server:/export/web が /local/web/ ディレクトリーにローカルにマウントされます。すべてのファイルに httpd_sys_content_t タイプのラベルが付けられ、Apache HTTP Server へのアクセスが許可されます。server:/export/database は /local/database/ にローカルにマウントされ、すべてのファイルに mysqld_db_t タイプのラベルが付けられ、MariaDB へのアクセスが許可されます。これらのタイプの変更はディスクには書き込まれません。

重要

nosharecache オプションを使用すると、/export/web/ を複数回マウントするなど、異なるコンテキストで、エクスポートの同じサブディレクトリーを複数回マウントできます。異なるコンテキストのエクスポートから同じサブディレクトリーを複数回マウントしないでください。マウントと重複するマウントが作成され、ファイルには 2 つの異なるコンテキストでアクセスできます。

4.9.5. コンテキストマウントの永続化

再マウントおよび再起動後もコンテキストのマウントを永続化するには、/etc/fstab ファイルまたはオートマウントマップでファイルシステムのエントリーを追加し、マウントオプションとして必要なコンテキストを使用します。以下の例では、NFS コンテキストマウントの /etc/fstab にエントリーを追加します。

■

```
server:/export /local/mount/ nfs context="system_u:object_r:httpd_sys_content_t:s0" 0 0
```

4.10. SELINUX ラベルの維持

以下のセクションでは、ファイルおよびディレクトリーのコピー、移動、およびアーカイブ時に SELinux コンテキストの動作を説明します。また、コピーおよびアーカイブ時にコンテキストを保存する方法も説明します。

4.10.1. ファイルおよびディレクトリーのコピー

ファイルまたはディレクトリーのコピー時に、新しいファイルまたはディレクトリーが存在しない場合は作成されます。この新しいファイルまたはディレクトリーのコンテキストは、元のコンテキストの保存にオプションが使用されていない限り、デフォルトのラベル付けルールに基づいています。たとえば、ユーザーのホームディレクトリーに作成されたファイルには、**user_home_t** タイプのラベルが付けられます。

```
~]$ touch file1
```

```
~]$ ls -Z file1
-rw-rw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0 file1
```

このようなファイルを、**/etc** などの別のディレクトリーにコピーすると、**/etc** のデフォルトのラベル付けルールに従って、新しいファイルが作成されます。追加オプションなしでファイルをコピーすると、元のコンテキストが保持されない場合があります。

```
~]$ ls -Z file1
-rw-rw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0 file1
```

```
~]# cp file1 /etc/
```

```
~]$ ls -Z /etc/file1
-rw-r--r--  root root unconfined_u:object_r:etc_t:s0   /etc/file1
```

file1 を **/etc** にコピーする際に、**/etc/file1** が存在しない場合は、**/etc/file1** が新規に作成されます。上記の例で示すように、**/etc/file1** には、default-labeling ルールに基づいて**etc_t** タイプのラベルが付けられます。

既存ファイルにファイルをコピーすると、**--preserve=context** など、元のファイルのコンテキストを保持するためにユーザーが**cp**オプションを指定しない限り、既存ファイルのコンテキストが保持されます。SELinux ポリシーは、コピー中にコンテキストが保持されないことがあります。

手順4.11 SELinux コンテキストを保持せずにコピーする

この手順では、**cp** コマンドでファイルをコピーする場合 (オプションが指定されていない場合)、タイプはターゲットの親ディレクトリーから継承されます。

1. ユーザーのホームディレクトリーにファイルを作成します。ファイルには、**user_home_t** タイプのラベルが付けられます。

   ```
   ~]$ touch file1
   ```

```
~]$ ls -Z file1
-rw-rw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0 file1
```

2. **/var/www/html/ ディレクトリーには、以下のコマンドのように httpd_sys_content_t タイプのラベルが付けられます。**

```
~]$ ls -dZ /var/www/html/
drwxr-xr-x  root root system_u:object_r:httpd_sys_content_t:s0 /var/www/html/
```

3. **file1 が /var/www/html/ にコピーされると、httpd_sys_content_t タイプを継承します。**

```
~]# cp file1 /var/www/html/
```

```
~]$ ls -Z /var/www/html/file1
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file1
```

手順4.12 コピー時の SELinux コンテキストの保持

この手順では、--preserve=context オプションを使用して、コピー時にコンテキストを保持する方法を説明します。

1. **ユーザーのホームディレクトリーにファイルを作成します。ファイルには、user_home_t タイプのラベルが付けられます。**

```
~]$ touch file1
```

```
~]$ ls -Z file1
-rw-rw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0 file1
```

2. **/var/www/html/ ディレクトリーには、以下のコマンドのように httpd_sys_content_t タイプのラベルが付けられます。**

```
~]$ ls -dZ /var/www/html/
drwxr-xr-x  root root system_u:object_r:httpd_sys_content_t:s0 /var/www/html/
```

3. **--preserve=context オプションを使用すると、コピー操作時に SELinux コンテキストが保持されます。以下に示すように、ファイルが /var/www/html/ にコピーされると、user_home_t タイプの file1 が保持されます。**

```
~]# cp --preserve=context file1 /var/www/html/
```

```
~]$ ls -Z /var/www/html/file1
-rw-r--r--  root root unconfined_u:object_r:user_home_t:s0 /var/www/html/file1
```

手順4.13 コンテキストのコピーと変更

この手順では、--context オプションを使用して宛先コピーのコンテキストを変更する方法を説明します。以下の例は、ユーザーのホームディレクトリーで実行されます。

1. **ユーザーのホームディレクトリーにファイルを作成します。ファイルには、user_home_t タイプのラベルが付けられます。**

```
~]$ touch file1
```

```
~]$ ls -Z file1
-rw-rw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0 file1
```

2. **--context オプションを使用して SELinux コンテキストを定義します。**

```
~]$ cp --context=system_u:object_r:samba_share_t:s0 file1 file2
```

3. **--context がない場合、file2 には unconfined_u:object_r:user_home_t コンテキストでラベルが付けられます。**

```
~]$ ls -Z file1 file2
-rw-rw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0 file1
-rw-rw-r--  user1 group1 system_u:object_r:samba_share_t:s0 file2
```

手順4.14 既存ファイルを介したファイルのコピー

この手順では、ファイルが既存のファイルにコピーされた場合に、コンテキストの保存にオプションを使用しない限り、既存のファイルのコンテキストが保持されることを示しています。

1. **root として、/etc ディレクトリーに新しいファイル file1 を作成します。以下に示すように、ファイルには etc_t タイプのラベルが付けられています。**

```
~]# touch /etc/file1
```

```
~]$ ls -Z /etc/file1
-rw-r--r--  root root unconfined_u:object_r:etc_t:s0   /etc/file1
```

2. **別のファイル file2 を /tmp ディレクトリーに作成します。以下に示すように、ファイルには user_tmp_t タイプのラベルが付けられます。**

```
~]$ touch /tmp/file2
```

```
~$ ls -Z /tmp/file2
-rw-r--r--  root root unconfined_u:object_r:user_tmp_t:s0 /tmp/file2
```

3. **file1 を file2 で上書きします。**

```
~]# cp /tmp/file2 /etc/file1
```

4. **コピーした後、次のコマンドを実行すると、/etc/file1 を置き換えた /tmp/file2 の user_tmp_t タイプではなく、etc_t タイプのラベルが付けられた file1 が表示されます。**

```
~]$ ls -Z /etc/file1
-rw-r--r--  root root unconfined_u:object_r:etc_t:s0   /etc/file1
```

重要

ファイルおよびディレクトリーを移動せずにコピーします。これにより、正しい SELinux コンテキストでラベル付けされることを確認できます。SELinux コンテキストが間違っていると、プロセスがこのようなファイルおよびディレクトリーにアクセスできなくなる可能性があります。

4.10.2. ファイルとディレクトリーの移動

ファイルおよびディレクトリーは、移動時に現在の SELinux コンテキストを保持します。多くの場合、これは移動先の場所として正しくありません。以下の例は、ユーザーのホームディレクトリーから、Apache HTTP サーバーが使用する **/var/www/html/** ディレクトリーにファイルを移動する方法を示しています。ファイルは移動されるため、正しい SELinux コンテキストは継承されません。

手順4.15 ファイルとディレクトリーの移動

1. ホームディレクトリーに移動し、そこでファイルを作成します。ファイルには、**user_home_t** タイプのラベルが付けられます。

```
~]$ touch file1
```

```
~]$ ls -Z file1
-rw-rw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0 file1
```

2. 以下のコマンドを入力して、**/var/www/html/** ディレクトリーの SELinux コンテキストを表示します。

```
~]$ ls -dZ /var/www/html/
drwxr-xr-x  root root system_u:object_r:httpd_sys_content_t:s0 /var/www/html/
```

デフォルトでは、**/var/www/html/** には **httpd_sys_content_t** タイプのラベルが付けられます。**/var/www/html/** に作成された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。

3. root で **file1** を **/var/www/html/** に移動します。このファイルは移動したので、現在の **user_home_t** タイプを保持します。

```
~]# mv file1 /var/www/html/
```

```
~]# ls -Z /var/www/html/file1
-rw-rw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0 /var/www/html/file1
```

デフォルトでは、Apache HTTP Server は **user_home_t** タイプのラベルが付けられたファイルを読み取ることができません。Web ページを含むすべてのファイルに **user_home_t** タイプのラベルが付けられている場合、または Apache HTTP Server が読み取れない別のタイプの場合、Mozilla Firefox などの Web ブラウザーを使用してそれらにアクセスしようとすると、パーミッションが拒否されます。

重要

mv コマンドでファイルやディレクトリーを移動すると、SELinux コンテキストが正しくなくなり、Apache HTTP サーバーや Samba などのプロセスがそのようなファイルやディレクトリーにアクセスできなくなる可能性があります。

4.10.3. デフォルトの SELinux コンテキストの確認

matchpathcon ユーティリティーを使用して、ファイルおよびディレクトリーに正しい SELinux コンテキストがあるかどうかを確認します。このユーティリティーは、system policy にクエリーを行い、ファイルパスに関連付けられたデフォルトのセキュリティーコンテキストを提供します。[6] 以下の例では、matchpathcon を使用して、/var/www/html/ディレクトリーのファイルに正しくラベル付けされていることを確認します。

手順4.16 matchpathconを使用したデフォルトの SELinux Conxtext の確認

1. **root ユーザーとして、/var/www/html/ ディレクトリーに 3 つのファイル (file1、file2、および file3) を作成します。これらのファイルは、/var/www/html/ から httpd_sys_content_t タイプを継承します。**

```
~]# touch /var/www/html/file{1,2,3}
```

```
~]# ls -Z /var/www/html/
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file1
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file2
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file3
```

2. **root で、file1 タイプを samba_share_t に変更します。Apache HTTP Server は、samba_share_t タイプのラベルが付けられたファイルまたはディレクトリーを読み取ることができないことに注意してください。**

```
~]# chcon -t samba_share_t /var/www/html/file1
```

3. **matchpathcon -V オプションは、現在の SELinux コンテキストを SELinux ポリシー内の正しいデフォルトコンテキストと比較します。以下のコマンドを実行して、/var/www/html/ ディレクトリー内の全ファイルをチェックします。**

```
~]$ matchpathcon -V /var/www/html/*
/var/www/html/file1 has context unconfined_u:object_r:samba_share_t:s0, should be
system_u:object_r:httpd_sys_content_t:s0
/var/www/html/file2 verified.
/var/www/html/file3 verified.
```

matchpathcon コマンドの出力では、file1 には samba_share_t タイプのラベルが付けられていますが、httpd_sys_content_t タイプのラベルが付けられている必要があります。

```
/var/www/html/file1 has context unconfined_u:object_r:samba_share_t:s0, should be
system_u:object_r:httpd_sys_content_t:s0
```

ラベルの問題を解決し、root で Apache HTTP Server が file1 にアクセスできるようにするには、restorecon ユーティリティーを使用します。

```
~]# restorecon -v /var/www/html/file1
restorecon reset /var/www/html/file1 context unconfined_u:object_r:samba_share_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
```

4.10.4. tarによるファイルのアーカイブ

tar ユーティリティーは、デフォルトで拡張属性を保持しません。SELinux コンテキストは拡張属性に保存されるため、ファイルのアーカイブ時にコンテキストが失われる可能性があります。tar --selinux コマンドを使用して、コンテキストを保持するアーカイブを作成し、アーカイブからファイルを復元します。tar アーカイブに拡張属性のないファイルが含まれる場合、または拡張属性がシステムのデフォルトと一致するようにするには、restorecon ユーティリティーを使用します。

```
~]$ tar -xvf archive.tar | restorecon -f -
```

ディレクトリーによっては、restorecon を実行するためには、root ユーザーでなければならない場合があることに注意してください。

以下の例は、SELinux コンテキストを保持する tar アーカイブの作成を示しています。

手順4.17 tar アーカイブの作成

1. **/var/www/html/ ディレクトリーに移動し、その SELinux コンテキストを表示します。**

```
~]$ cd /var/www/html/
```

```
html]$ ls -dZ /var/www/html/
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 .
```

2. **root で、/var/www/html/ に 3 つのファイル (file1、file2、および file3) を作成します。これらのファイルは、/var/www/html/ から httpd_sys_content_t タイプを継承します。**

```
html]# touch file{1,2,3}
```

```
html]$ ls -Z /var/www/html/
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file1
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file2
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file3
```

3. **root で、以下のコマンドを実行して、test.tar という名前のtar アーカイブを作成します。SELinux コンテキストを維持するには、--selinux を使用します。**

```
html]# tar --selinux -cf test.tar file{1,2,3}
```

4. **root で test/ という名前の新規ディレクトリーを作成し、すべてのユーザーによる完全なアクセスを許可します。**

```
~]# mkdir /test
```

```
~]# chmod 777 /test/
```

5. **test.tar ファイルを test/ にコピーします。**

```
~]$ cp /var/www/html/test.tar /test/
```

6. **test/ ディレクトリーに移動します。このディレクトリーに移動し、以下のコマンドを実行して tar アーカイブを展開します。--selinux オプションを再度指定すると、SELinux コンテキストが default_t に変更されます。**

```
~]$ cd /test/
```

```
test]$ tar --selinux -xvf test.tar
```

7. **SELinux コンテキストを表示します。httpd_sys_content_t タイプは default_t に変更されず、保持されました。これは、 --selinux が使用されていなかったとしたら、変更されていたでしょう。**

```
test]$ ls -lZ /test/
-rw-r--r--  user1 group1 unconfined_u:object_r:httpd_sys_content_t:s0 file1
-rw-r--r--  user1 group1 unconfined_u:object_r:httpd_sys_content_t:s0 file2
-rw-r--r--  user1 group1 unconfined_u:object_r:httpd_sys_content_t:s0 file3
-rw-r--r--  user1 group1 unconfined_u:object_r:default_t:s0 test.tar
```

8. **test/ ディレクトリーが必要なくなった場合は、root で以下のコマンドを実行して、そのディレクトリー内のすべてのファイルも削除します。**

```
~]# rm -ri /test/
```

拡張属性をすべて保持する --xattrs オプションなどのtar の詳細は、tar(1) man ページを参照してください。

4.10.5. starを使用したファイルのアーカイブ

star ユーティリティーは、デフォルトで拡張属性を保持しません。SELinux コンテキストは拡張属性に保存されるため、ファイルのアーカイブ時にコンテキストが失われる可能性があります。star -xattr -H=exustar コマンドを使用して、コンテキストを保持するアーカイブを作成します。star パッケージはデフォルトでインストールされません。star をインストールするには、root ユーザーとして yum install star を実行します。

以下の例は、SELinux コンテキストを保持する star アーカイブの作成を示しています。

手順4.18 star アーカイブの作成

1. **root で、/var/www/html/ に 3 つのファイル (file1、file2、および file3) を作成します。これらのファイルは、/var/www/html/ から httpd_sys_content_t タイプを継承します。**

```
~]# touch /var/www/html/file{1,2,3}
```

```
~]# ls -Z /var/www/html/
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file1
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file2
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 file3
```

2. **/var/www/html/ ディレクトリーに変更します。このディレクトリーに、root で以下のコマンドを実行して、test.star という名前の star アーカイブを作成します。**

```
~]$ cd /var/www/html
```

```
html]# star -xattr -H=exustar -c -f=test.star file{1,2,3}
star: 1 blocks + 0 bytes (total of 10240 bytes = 10.00k).
```

3. root で test/ という名前の新規ディレクトリーを作成し、すべてのユーザーによる完全なアクセスを許可します。

```
~]# mkdir /test
```

```
~]# chmod 777 /test/
```

4. 次のコマンドを実行して、test.star ファイルを test/ にコピーします。

```
~]$ cp /var/www/html/test.star /test/
```

5. test/ に移動します。このディレクトリーに移動したら、以下のコマンドを実行して star アーカイブを展開します。

```
~]$ cd /test/
```

```
test]$ star -x -f=test.star
star: 1 blocks + 0 bytes (total of 10240 bytes = 10.00k).
```

6. SELinux コンテキストを表示します。httpd_sys_content_t タイプは default_t に変更されず、保持されています。これは、-xattr -H=exustar オプションが使用されていなかったとしたら、変更されていたでしょう。

```
~]$ ls -lZ /test/
-rw-r--r--  user1 group1 unconfined_u:object_r:httpd_sys_content_t:s0 file1
-rw-r--r--  user1 group1 unconfined_u:object_r:httpd_sys_content_t:s0 file2
-rw-r--r--  user1 group1 unconfined_u:object_r:httpd_sys_content_t:s0 file3
-rw-r--r--  user1 group1 unconfined_u:object_r:default_t:s0 test.star
```

7. test/ ディレクトリーが必要なくなった場合は、root で以下のコマンドを実行して、そのディレクトリー内のすべてのファイルも削除します。

```
~]# rm -ri /test/
```

8. star が不要になった場合は、root でパッケージを削除します。

```
~]# yum remove star
```

star の詳細は、star(1) man ページを参照してください。

4.11. 情報収集ツール

以下のユーティリティーは、アクセスベクターキャッシュ統計やクラス、タイプ、ブール値などの適切にフォーマットされた情報を提供するコマンドラインツールです。

avcstat

このコマンドにより、システムの起動後のアクセスベクターキャッシュの統計情報の出力が短縮されます。時間間隔を秒単位で指定すると、統計をリアルタイムで表示できます。これにより、初期出力以降に更新された統計が提供されます。使用される統計ファイルは /sys/fs/selinux/avc/cache_stats であるため、-f /path/to/file で別のキャッシュファイルを指定できます。

```
~]# avcstat
  lookups       hits     misses     allocs   reclaims      frees
 47517410   47504630      12780      12780      12176      12275
```

seinfo

このユーティリティーは、クラス、タイプ、ブール値、許可ルールなどの数など、ポリシーの内訳を説明するのに役立ちます。seinfo は、policy.conf ファイル、バイナリーポリシーファイル、モジュール形式のポリシーパッケージ一覧、またはポリシーリストファイルを入力として使用するコマンドラインユーティリティーです。seinfo ユーティリティーを使用するには、setools-console パッケージがインストールされている必要があります。

seinfo の出力はバイナリーファイルとソースファイルで異なります。たとえば、ポリシーソースファイルでは、{ }括弧を使用して複数のルール要素を 1 行にグループ化します。同様の効果は、単一の属性が 1 つまたは複数のタイプに展開される属性で行われます。これらは展開され、バイナリーポリシーファイルでは関連性がなくなったため、検索結果での戻り値はゼロになります。ただし、括弧を使用した 1 行ルールが複数行になったため、ルールの数が大幅に増えています。

一部の項目はバイナリーポリシーに記載されていません。たとえば、neverallow ルールは実行時ではなくポリシーのコンパイル時にのみ確認され、初期のセキュリティー識別子 (SID) は起動時にカーネルがポリシーを読み込む前に必要であるため、バイナリーポリシーの一部ではありません。

```
~]# seinfo

Statistics for policy file: /sys/fs/selinux/policy
Policy Version & Type: v.28 (binary, mls)

   Classes:            77    Permissions:       229
   Sensitivities:       1    Categories:       1024
   Types:            3001    Attributes:        244
   Users:               9    Roles:              13
   Booleans:          158    Cond. Expr.:       193
   Allow:          262796    Neverallow:          0
   Auditallow:         44    Dontaudit:      156710
   Type_trans:      10760    Type_change:        38
   Type_member:        44    Role allow:         20
   Role_trans:        237    Range_trans:      2546
   Constraints:        62    Validatetrans:       0
   Initial SIDs:       27    Fs_use:             22
   Genfscon:           82    Portcon:           373
   Netifcon:            0    Nodecon:             0
   Permissives:        22    Polcap:              2
```

seinfo ユーティリティーは、ドメイン属性でタイプ数を一覧表示することもできます。これにより、さまざまな制限のあるプロセスの数を見積もります。

```
~]# seinfo -adomain -x | wc -l
550
```

すべてのドメインタイプが制限されているわけではありません。制限のないドメインの数を確認するには、unconfined_domain 属性を使用します。

```
~]# seinfo -aunconfined_domain_type -x | wc -l
52
```

Permissive ドメインは、**--permissive** オプションでカウントできます。

```
~]# seinfo --permissive -x | wc -l
31
```

上記のコマンドで追加の **| wc -l** コマンドを削除すると、一覧が表示されます。

sesearch

sesearch ユーティリティーを使用して、ポリシー内の特定のルールを検索できます。ポリシーソースファイルまたはバイナリーファイルのいずれかを検索できます。以下に例を示します。

```
~]$ sesearch --role_allow -t httpd_sys_content_t
Found 20 role allow rules:
   allow system_r sysadm_r;
   allow sysadm_r system_r;
   allow sysadm_r staff_r;
   allow sysadm_r user_r;
   allow system_r git_shell_r;
   allow system_r guest_r;
   allow logadm_r system_r;
   allow system_r logadm_r;
   allow system_r nx_server_r;
   allow system_r staff_r;
   allow staff_r logadm_r;
   allow staff_r sysadm_r;
   allow staff_r unconfined_r;
   allow staff_r webadm_r;
   allow unconfined_r system_r;
   allow system_r unconfined_r;
   allow system_r user_r;
   allow webadm_r system_r;
   allow system_r webadm_r;
   allow system_r xguest_r;
```

sesearch ユーティリティーは、allow ルールの数を提供できます。

```
~]# sesearch --allow | wc -l
262798
```

また、dontauditルールの数も以下のようになります。

```
~]# sesearch --dontaudit | wc -l
156712
```

4.12. SELINUX ポリシーモジュールの優先付けと無効化

/etc/selinux/ の SELinux モジュールストレージでは、SELinux モジュールで優先度を使用できます。root で次のコマンドを実行すると、異なる優先度の 2 つのモジュールディレクトリーが表示されます。

```
~]# ls /etc/selinux/targeted/active/modules
100  400  disabled
```

semodule ユーティリティーで使用されるデフォルトの優先度は 400 ですが、selinux-policy パッケージで使用される優先度は 100 であるため、優先度 100 でインストールされたほとんどの SELinux モジュールを見つけることができます。

優先度を上げると、同じ名前で修正されたモジュールで、既存のモジュールを上書きできます。同じ名前と異なる優先度を持つモジュールが多い場合にポリシーを構築すると、最も高い優先度を持つモジュールのみが使用されます。

> **例4.1 SELinux ポリシーモジュールの優先度の使用**
>
> 変更したファイルコンテキストを持つ新しいモジュールを準備します。**semodule -i** コマンドでモジュールをインストールし、モジュールの優先度を 400 に設定します。以下の例では **sandbox.pp** を使用します。
>
> ```
> ~]# semodule -X 400 -i sandbox.pp
> ~]# semodule --list-modules=full | grep sandbox
> 400 sandbox          pp
> 100 sandbox          pp
> ```
>
> デフォルトモジュールに戻すには、root で **semodule -r** コマンドを実行します。
>
> ```
> ~]# semodule -X 400 -r sandbox
> libsemanage.semanage_direct_remove_key: sandbox module at priority 100 is now active.
> ```

システムポリシーモジュールの無効化

system policy を無効にするには、root で以下のコマンドを実行します。

```
semodule -d MODULE_NAME
```

警告

semodule -r コマンドを使用して **system policy** モジュールを削除すると、システムのストレージから削除されるため、再度読み込むことができません。すべての**system policy** モジュールを復元するために、selinux-policy-targeted パッケージが不必要に再インストールされないようにするには、代わりに**semodule -d** を使用します。

4.13. MULTI-LEVEL SECURITY (MLS)

マルチレベルセキュリティーテクノロジーは、Bell-La Padula Mandatory Access Model を強制するセキュリティースキームを指します。MLS では、ユーザーとプロセスは *サブジェクト* と呼ばれ、ファイル、デバイス、およびシステムの他のパッシブコンポーネントは *オブジェクト* と呼ばれます。サブジェクトとオブジェクトの両方がセキュリティーレベルでラベル付けされ、これはサブジェクトのクリアランスとオブジェクトの分類を必要とします。各セキュリティーレベルは、*sensitivity* と *category* で設定されます。たとえば、内部リリーススケジュールは、機密扱いの内部文書カテゴリーに保管されます。

図4.1「クリアランスのレベル」 は、US 防衛コミュニティーが当初設計したクリアランス (機密情報取

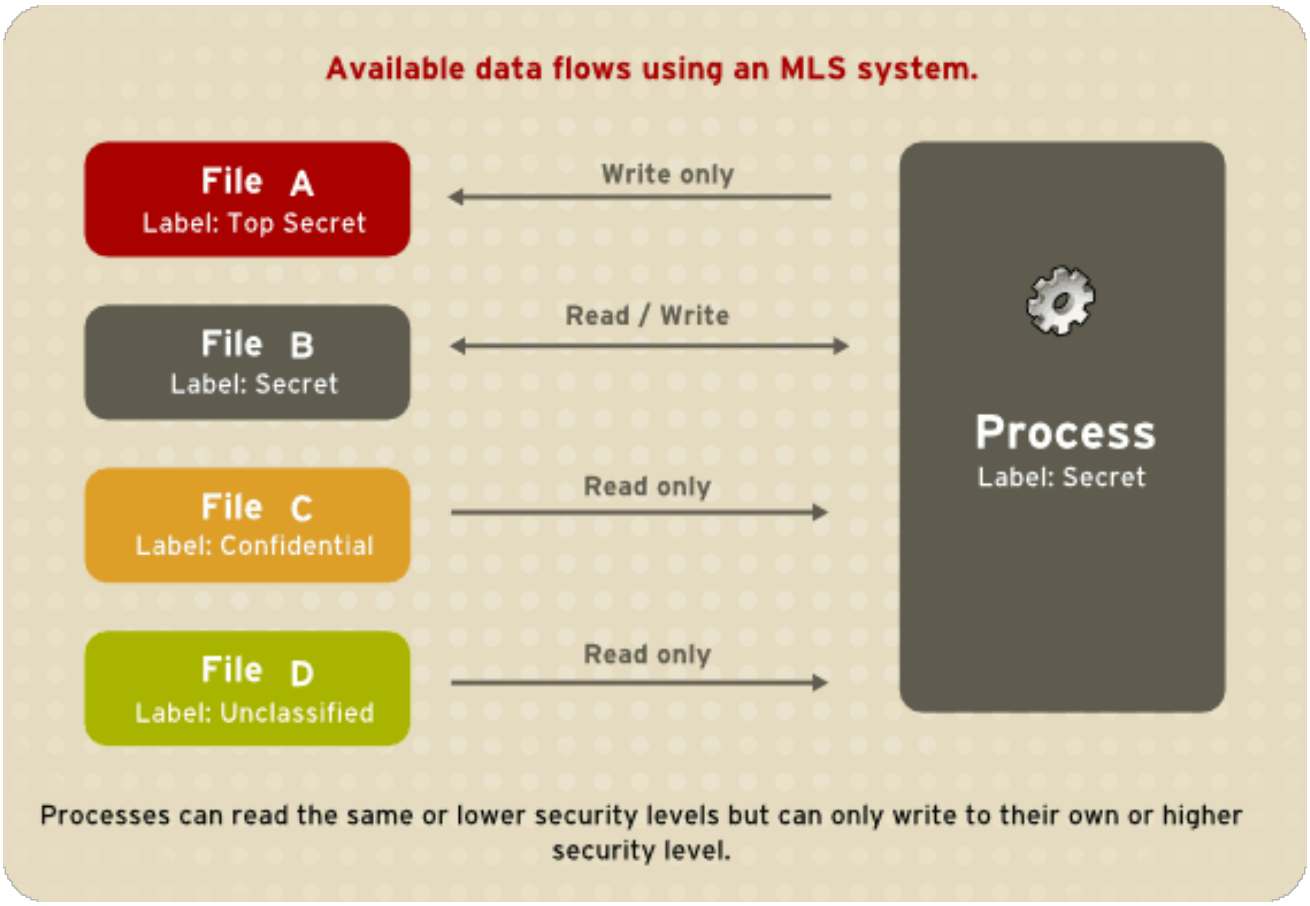
扱許可) レベルを示しています。上記の内部スケジュールの例に関連して、コンフィデンシャルクリアランスを取得したユーザーのみが、機密カテゴリーのドキュメントを表示できます。ただし、コンフィデンシャルクリアランスのみを有するユーザーは、より高いレベルまたはクリアランスを必要とする文書を表示することはできません。読み取りアクセスは、より低いレベルのクリアランスを有する文書に対してのみ許可され、より高いレベルのクリアランスを有する文書への書き込みアクセスは許可されます。

図4.1 クリアランスのレベル

[D]

図4.2「MLS を使用した許可されたデータフロー」 "Secret" セキュリティーレベルで実行しているサブジェクトと、セキュリティーレベルが異なるさまざまなオブジェクトとの間で許可されているデータフローをすべて表示します。簡単に言うと、Bell-LaPadula モデルでは、*no read up* と *no write down* の 2 つのプロパティーが適用されます。

図4.2 MLS を使用した許可されたデータフロー



[D]

4.13.1. MLS およびシステム権限

MLS アクセスルールは、常に従来のアクセス権限 (ファイル権限) と組み合わせられます。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、セキュリティーレベルのユーザーによるアクセスもブロックされます。SELinux ポリシールールは、DAC ルールの 後に チェックされることを覚えておくことが重要になります。より高いセキュリティークリアランスを設定しても、ファイルシステムを任意にブラウズするパーミッションは自動的には付与されません。

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。コンピューターのすべての情報にアクセスできる場合もありますが、管理者権限を設定するのとは異なります。

4.13.2. SELinux での MLS の有効化

注記

X Window System を実行しているシステムでは、MLS ポリシーを使用することは推奨されていません。

以下の手順に従って、システムで SELinux MLS ポリシーを有効にします。

手順4.19 SELinux MLS ポリシーの有効化

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

```
~]# yum install selinux-policy-mls
```

2. MLS ポリシーを有効にする前に、ファイルシステムの各ファイルに、MLS ラベルで再ラベル付けする必要があります。ファイルシステムに再ラベル付けすると、制限されたドメインのアクセスが拒否される可能性があります。これにより、システムが正しく起動しなくなる可能性があります。これを防ぐには、**/etc/selinux/config** ファイルで **SELINUX=permissive** を設定します。また、**SELINUXTYPE=mls** を設定して、MLS ポリシーを有効にします。設定ファイルは以下のようになります。

```
# 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=mls
```

3. SELinux が Permissive モードで実行していることを確認します。

```
~]# setenforce 0
```

```
~]$ getenforce
Permissive
```

4. **fixfiles** スクリプトを使用して、**-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。

```
~]# fixfiles -F onboot
```

5. システムを再起動します。次回の起動時に、MLS ポリシーに従って、すべてのファイルシステムに再ラベル付けされます。ラベルプロセスは、適切な SELinux コンテキストを使用して、すべてのファイルにラベルを付けます。

```
*** Warning -- SELinux mls policy relabel is required.
*** Relabeling could take a very long time, depending on file
*** system size and speed of hard drives.
***********
```

一番下の行にある * (アスタリスク) 文字は、ラベル付けされている 1000 ファイルを表します。上記の例では、11 の* 文字が、ラベルが付けられた 11000 ファイルを表しています。すべてのファイルにラベルを付けるのにかかる時間は、システムのファイル数と、ハードディスクドライブの速度により異なります。最新のシステムでは、このプロセスに 10 分程度かかる場合があります。ラベリングプロセスが終了すると、システムが自動的に再起動します。

6. Permissive モードでは SELinux ポリシーは強制されませんが、Enforcing モードで実行された場合に拒否されたであろうアクションの拒否は引き続きログに記録されます。Enforcing モードに切り替える前に、root で次のコマンドを実行して、システムの最後の起動時に SELinux がアクションを拒否しなかったことを確認します。最後のシステムの起動時に SELinux がアク

ションを拒否しなかった場合に、このコマンドを実行しても出力は返されません。システムの起動時に SELinux がアクセスを拒否された場合のトラブルシューティングは、11章 トラブルシューティングを参照してください。

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

7. **/var/log/messages ファイルに拒否メッセージがないか、すでに拒否を解決している場合は、/etc/selinux/config ファイルに SELINUX=enforcing 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=mls
```

8. **システムを再起動し、SELinux が Enforcing モードで実行していることを確認します。**

```
~]$ getenforce
Enforcing
```

MLS ポリシーが有効になっていることも確認します。

```
~]# sestatus |grep mls
Policy from config file:       mls
```

4.13.3. 特定の MLS 範囲を持つユーザーの作成

以下の手順に従って、特定の MLS 範囲で新規の Linux ユーザーを作成します。

手順4.20 特定の MLS 範囲を持つユーザーの作成

1. **useradd コマンドを使用して新しい Linux ユーザーを追加し、その新しい Linux ユーザーを既存の SELinux ユーザー (この例では staff_u) にマッピングします。**

```
~]# useradd -Z staff_u john
```

2. **新しく作成した Linux ユーザーにパスワードを割り当てます。**

```
prompt~]# passwd john
```

3. **root で次のコマンドを実行し、SELinux ユーザーと Linux ユーザー間のマッピングを表示します。出力は、以下のようになります。**

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

__default__          user_u               s0-s0                *
```

```
john              staff_u           s0-s15:c0.c1023     *
root              root              s0-s15:c0.c1023     *
staff             staff_u           s0-s15:c0.c1023     *
sysadm            staff_u           s0-s15:c0.c1023     *
system_u          system_u          s0-s15:c0.c1023     *
```

4. **ユーザー john に指定範囲を定義します。**

```
~]# semanage login --modify --range s2:c100 john
```

5. **SELinux ユーザーと Linux ユーザー間のマッピングを再度表示します。ユーザー john には、指定した MLS レンジが定義されていることに注意してください。**

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

__default__         user_u              s0-s0               *
john                staff_u             s2:c100             *
root                root                s0-s15:c0.c1023     *
staff               staff_u             s0-s15:c0.c1023     *
sysadm              staff_u             s0-s15:c0.c1023     *
system_u            system_u            s0-s15:c0.c1023     *
```

6. **必要に応じて、john のホームディレクトリーのラベルを修正するには、次のコマンドを入力します。**

```
~]# chcon -R -l s2:c100 /home/john
```

4.13.4. Polyinstantiated ディレクトリーの設定

/tmp ディレクトリーおよび /var/tmp/ ディレクトリーは、通常、すべてのプログラム、サービス、およびユーザーが一時的に保存するために使用されます。ただし、このような設定を行うと、競合状態攻撃や、ファイル名に基づく情報漏えいに対して脆弱になります。SELinux は、*polyinstantiated* ディレクトリーの形式でソリューションを提供します。これは、事実上、/tmp と /var/tmp/ の両方がインスタンス化され、各ユーザーに対して非公開になっていることを意味します。ディレクトリーのインスタンス化を有効にすると、各ユーザーの /tmp ディレクトリーおよび /var/tmp/ ディレクトリーが、/tmp-inst および /var/tmp/tmp-inst に自動的にマウントされます。

ディレクトリーのポリインスタンス化を有効にするには、以下の手順を行います。

手順4.21 ポリインスタンス化ディレクトリーの有効化

1. **/etc/security/namespace.conf ファイルの最後の 3 行のコメントを解除して、/tmp、/var/tmp/、およびユーザーのホームディレクトリーのインスタンス化を有効にします。**

```
~]$ tail -n 3 /etc/security/namespace.conf
/tmp     /tmp-inst/           level      root,adm
/var/tmp /var/tmp/tmp-inst/   level      root,adm
$HOME    $HOME/$USER.inst/    level
```

2. **/etc/pam.d/login ファイルーで、pam_namespace.so モジュールが session 用に設定されていることを確認します。**

```
~]$ grep namespace /etc/pam.d/login
session    required     pam_namespace.so
```

3. システムを再起動します。

4.14. ファイル名の推移

file name transition 機能を使用すると、ポリシーの移行ルールを作成する際に、ポリシーの作成者がファイル名を指定できます。以下のようなルールを作成できます。**A_t** というラベルの付いたプロセスが、**B_t** というラベルの付いたディレクトリーに、指定されたオブジェクトクラスを作成し、指定されたオブジェクトクラスの名前が **objectname** である場合、ラベル **C_t** を取得します。このメカニズムにより、システム上のプロセスをより詳細に制御できます。

ファイル名を変更しない場合は、オブジェクトにラベルを付ける方法が 3 つあります。

- デフォルトでは、オブジェクトは親ディレクトリーからラベルを継承します。たとえば、**etc_t** のラベルが付いたディレクトリーにファイルを作成すると、そのファイルには **etc_t** のラベルも付きます。ただし、この方法は、異なるラベルのディレクトリー内に複数のファイルがあることが望ましい場合には役に立ちません。

- ポリシーの作成者は、以下のようなルールをポリシーに作成できます。タイプ **A_t** のプロセスが **B_t** というラベルが付いたディレクトリーに指定されたオブジェクトクラスを作成すると、オブジェクトは新しい **C_t** ラベルを取得します。この方法は、1つのプログラムが、各オブジェクトに個別のラベルが必要な同じディレクトリーに複数のオブジェクトを作成する場合に問題になります。また、作成されるオブジェクトの名前が指定されていないため、このルールでは部分的な制御しか行われません。

- 特定のアプリケーションでは、SELinux が、このようなアプリケーションがシステムに、特定のパスのラベルを尋ねられるように認識しています。その後、このようなアプリケーションは、カーネルに、必要なラベルを持つオブジェクトを作成するように要求します。SELinux 対応のアプリケーションの例としては、rpm パッケージマネージャー、restorecon ユーティリティー、または udev デバイスマネージャーが挙げられます。ただし、SELinux 対応のファイルまたはディレクトリーを作成するアプリケーションをすべて指示することはできません。作成後、多くの場合、オブジェクトに正しいラベルを付け直す必要があります。制限されているドメインがオブジェクトを使用しようとすると、AVC メッセージが返されます。

ファイル名の変更機能により、間違ったラベル付けに関連する問題が低減し、システムの安全性が向上します。ポリシーの作成者は、特定のアプリケーションが、指定されたディレクトリーで、指定された名前を持つファイルのみを作成できることを適切に表明できます。ルールではファイルパスではなくファイル名が考慮されます。ファイルパスのベース名です。ファイル名の変更では、**strcmp()** 機能による完全一致が使用されることに注意してください。正規表現またはワイルドカード文字の使用は考慮されません。

注記

ファイルパスはカーネルによって異なる場合があり、ファイル名の変更ではパスを使用してラベルを判断しません。そのため、この機能は初期ファイル作成にのみ影響し、作成済みオブジェクトの間違ったラベルを修正することはありません。

例4.2 ファイル名の推移で記述されるポリシールールの例

以下の例は、ファイル名が遷移するポリシールールを示しています。

```
filetrans_pattern(unconfined_t, admin_home_t, ssh_home_t, dir, ".ssh")
```

このルールは、unconfined_t タイプのプロセスが admin_home_t のラベルが付いたディレクトリーに ~/.ssh/ ディレクトリーを作成すると、~/.ssh/ ディレクトリーはラベル ssh_home_t を取得します。

ファイル名の遷移が含まれるポリシールールの例を以下に示します。

```
filetrans_pattern(staff_t, user_home_dir_t, httpd_user_content_t, dir, "public_html")
filetrans_pattern(thumb_t, user_home_dir_t, thumb_home_t, file, "missfont.log")
filetrans_pattern(kernel_t, device_t, xserver_misc_device_t, chr_file, "nvidia0")
filetrans_pattern(puppet_t, etc_t, krb5_conf_t, file, "krb5.conf")
```

注記

ファイル名の変更機能は主にポリシー作成者に影響を及ぼしますが、ユーザーは、ほぼ常に含まれるディレクトリーのデフォルトラベルで作成されるファイルオブジェクトの代わりに、ポリシーで指定されているように、一部のファイルオブジェクトには異なるラベルがあることに注意してください。

4.15. PTRACE() の無効化

ptrace() システムコールを使用すると、あるプロセスで別のプロセスの実行を監視および制御し、そのメモリーとレジスターを変更できます。この呼び出しは、strace ユーティリティーの使用など、デバッグ中の開発者が主に使用します。ptrace() が必要ない場合は無効にして、システムセキュリティーを向上させることができます。これは、deny_ptrace ブール値を有効にすると実行できます。これにより、unconfined_tのドメインで実行しているプロセスであっても、他のプロセスでptrace() を使用できなくなります。

deny_ptrace ブール値は、既定では無効になっています。これを有効にするには、root で setsebool -P deny_ptrace on を実行します。

```
~]# setsebool -P deny_ptrace on
```

このブール値が有効かどうかを確認するには、次のコマンドを使用します。

```
~]$ getsebool deny_ptrace
deny_ptrace --> on
```

このブール値を無効にするには、root でsetsebool -P deny_ptrace off を実行します。

```
~]# setsebool -P deny_ptrace off
```

注記

setsebool -P は永続的な変更を行います。システムを再起動しても変更を持続させない場合は、-P オプションを使用しないでください。

このブール値は、Red Hat Enterprise Linux に含まれるパッケージにのみ影響します。したがって、サードパーティーのパッケージでは、依然としてptrace() システムコールを使用できます。ptrace() の使用が許可されているドメインをすべて一覧表示するには、次のコマンドを実行します。setools-console パッケージは、sesearch ユーティリティーを提供し、パッケージはデフォルトではインストールされないことに注意してください。

```
~]# sesearch -A -p ptrace,sys_ptrace -C | grep -v deny_ptrace | cut -d ' ' -f 5
```

4.16. サムネイルの保護

サムネイルアイコンを使用すると、USB デバイスや CD などのリムーバブルメディアを使用して攻撃者がロックされたマシンに侵入できる可能性があります。システムがリムーバブルメディアを検出すると、Nautilus ファイルマネージャーがサムネイルドライバーコードを実行し、マシンがロックされていても、適切なファイルブラウザーにサムネイルアイコンを表示します。サムネイルの実行ファイルが脆弱であれば、攻撃者はサムネイルのドライバーコードを使用してパスワードを入力しなくてもロック画面を回避できるため、この動作は安全ではありません。

したがって、このような攻撃を防ぐために、新しい SELinux ポリシーが使用されています。このポリシーにより、画面のロック時に、すべてのサムネールドライバーがロックされます。サムネイル保護は、制限のあるユーザーと制限のないユーザーの両方で有効になります。このポリシーは、次のアプリケーションに影響します。

- /usr/bin/evince-thumbnailer
- /usr/bin/ffmpegthumbnailer
- /usr/bin/gnome-exe-thumbnailer.sh
- /usr/bin/gnome-nds-thumbnailer
- /usr/bin/gnome-xcf-thumbnailer
- /usr/bin/gsf-office-thumbnailer
- /usr/bin/raw-thumbnailer
- /usr/bin/shotwell-video-thumbnailer
- /usr/bin/totem-video-thumbnailer
- /usr/bin/whaaw-thumbnailer
- /usr/lib/tumbler-1/tumblerd
- /usr/lib64/tumbler-1/tumblerd

---

[3] 一時的にデフォルトの動作に戻すには、Linux の root ユーザーとして **setsebool httpd_can_network_connect_db off** コマンドを実行します。再起動後も維持される変更の場合は、**setsebool -P httpd_can_network_connect_db off** コマンドを実行します。

[4] **/etc/selinux/targeted/contexts/files/** ディレクトリーのファイルは、ファイルおよびディレクトリーのコンテキストを定義します。このディレクトリーのファイルは、**restorecon** ユーティリティーおよび **setfiles** ユーティリティーにより読み込まれ、ファイルおよびディレクトリーをデフォルトコンテキストに復元します。

[5] Morris, James."Filesystem Labeling in SELinux".2004 年 10 月 1 日公開。2008 年 10 月 14 日にアクセス: http://www.linuxjournal.com/article/7426

[6] **matchpathcon** の詳細は、matchpathcon(8) man ページを参照してください。

第5章 SEPOLICY スイート

sepolicy ユーティリティーは、インストールした SELinux ポリシーをクエリーする一連の機能を提供します。このような機能は、新しく追加されたものか、以前は sepolgen または setrans などの別のユーティリティーで提供されていました。このスイートにより、移行レポート、man ページ、または新しいポリシーモジュールを生成できるようになり、ユーザーのアクセスが容易になり、SELinux ポリシーをより深く理解できるようになります。

policycoreutils-devel パッケージは、sepolicy を提供します。sepolicy をインストールする場合は、root で以下のコマンドを実行します。

```
~]# yum install policycoreutils-devel
```

sepolicy スイートは、コマンドラインパラメーターとして呼び出される以下の機能を提供します。

表5.1 sepolicy の機能

機能	説明
ブール値	ブール値の説明を表示するには、SELinux ポリシーをクエリーします。
通信	SELinux ポリシーをクエリーして、ドメインが互いに通信できるかどうかを確認します。
generate	SELinux ポリシーモジュールテンプレートの生成
gui	SELinux ポリシーのグラフィカルユーザーインターフェイス
interface	SELinux ポリシーのインターフェイスの一覧を表示する
man ページ	SELinux の man ページの生成
network	SELinux ポリシーのネットワーク情報をクエリーする
遷移	SELinux ポリシーのクエリーと、プロセスの移行レポートの生成

5.1. SEPOLICY PYTHON のバインディング

以前のバージョンの Red Hat Enterprise Linux では、setools に sesearch ユーティリティーおよび seinfo ユーティリティーが同梱されていました。sesearch ユーティリティーは SELinux ポリシー内のルールを検索するのに使用しますが、seinfo ユーティリティーは、ポリシー内の他のさまざまなコンポーネントにクエリーできるようにします。

Red Hat Enterprise Linux 7 では、sepolicy スイートでこのユーティリティーの機能を使用できるように、sesearch および seinfo 用の Python バインディングが追加されました。以下の例を参照してください。

```
> python
>>> import sepolicy
>>> sepolicy.info(sepolicy.ATTRIBUTE)
```

```
Returns a dictionary of all information about SELinux Attributes
>>>sepolicy.search([sepolicy.ALLOW])
Returns a dictionary of all allow rules in the policy.
```

5.2. SELINUX ポリシーモジュールの生成: SEPOLICY GENERATE

以前のバージョンの Red Hat Enterprise Linux では、SELinux ポリシーを生成するのに **sepolgen** またはユーティリティー **selinux-polgengui** ユーティリティーが使用されていました。このツールは、**sepolicy** スイートに統合されています。Red Hat Enterprise Linux 7 では、**sepolicy generate** コマンドを使用して最初の SELinux ポリシーモジュールテンプレートを生成します。

sepolgen とは異なり、root ユーザーとして **sepolicy generate** を実行する必要はありません。また、このユーティリティーは、RPM 仕様ファイルを作成します。これを使用して、ポリシーパッケージファイル (***NAME*.pp**) とインターフェイスファイル (***NAME*.if**) を正しい場所にインストールする RPM パッケージを構築し、SELinux ポリシーをカーネルにインストールして、ラベリングを修正します。セットアップスクリプトは、引き続き SELinux ポリシーをインストールし、ラベリングを設定します。また、インストールしたポリシーに基づく man ページは、**sepolicy manpage** コマンドを使用して生成されます。[7] 最後に、**sepolicy generate** が SELinux ポリシーと man ページを構築してコンパイルし、RPM パッケージを作成します。これにより、別のシステムにインストールできるようになります。

sepolicy generate を実行すると、以下のファイルが生成されます。

***NAME*.te** – Type Enforcement ファイル

このファイルは、特定のドメインに対するすべてのタイプとルールを定義します。

***NAME*.if** - インターフェイスファイル

このファイルは、システムのデフォルトのファイルコンテキストを定義します。これは、**NAME.te** ファイルで作成されるファイルタイプを取り、ファイルのパスをタイプに関連付けます。**restorecon** および **rpm** などのユーティリティーは、このパスを使用してラベルを書き込みます。

***NAME*_selinux.spec** – RPM SPEC ファイル

このファイルは、SELinux ポリシーをインストールし、ラベリングを設定する RPM SPEC ファイルです。このファイルでは、インターフェイスファイルと、ポリシーを説明する man ページもインストールされます。**sepolicy manpage -d *NAME*** コマンドを使用すると、man ページを生成できます。

***NAME*.sh** - ヘルパーシェルスクリプト

このスクリプトは、システムでラベリングをコンパイル、インストール、および修正するのに役立ちます。また、インストールしたポリシーに基づいて man ページを生成し、他のシステムにインストールするのに適した RPM パッケージをコンパイルして構築します。

SELinux ポリシーモジュールを生成できる場合、**sepolicy generate** はソースドメインからターゲットドメインに生成されたすべてのパスを出力します。**sepolicy generate** の詳細は、sepolicy-generate(8) man ページを参照してください。

5.3. ドメイントランジションの概要: SEPOLICY TRANSITION

以前は、**setrans** ユーティリティーを使用して、2 つのドメインまたはプロセスタイプ間の移行が可能であり、これらのドメインまたはプロセス間の移行に使用されるすべての中間タイプを出力していまし

た。Red Hat Enterprise Linux 7 では、**setrans** が **sepolicy** スイートに同梱され、代わりに **sepolicy transition** コマンドが使用されるようになりました。

sepolicy transition コマンドは、SELinux ポリシーにクエリーし、プロセスの移行レポートを作成します。**sepolicy transition** コマンドには、ソースドメイン (**-s** オプションで指定) とターゲットドメイン (**-t** オプションで指定) の 2 つのコマンドライン引数が必要です。ソースドメインのみを入力した場合は、**sepolicy transition** により、ソースドメインが移行可能なドメインがすべて一覧表示されます。以下の出力には、すべてのエントリーが含まれているわけではありません。「@」 の文字は、「execute」 を意味します。

```
~]$ sepolicy transition -s httpd_t
httpd_t @ httpd_suexec_exec_t --> httpd_suexec_t
httpd_t @ mailman_cgi_exec_t --> mailman_cgi_t
httpd_t @ abrt_retrace_worker_exec_t --> abrt_retrace_worker_t
httpd_t @ dirsrvadmin_unconfined_script_exec_t --> dirsrvadmin_unconfined_script_t
httpd_t @ httpd_unconfined_script_exec_t --> httpd_unconfined_script_t
```

ターゲットドメインが指定されている場合、**sepolicy transition** はソースドメインからターゲットドメインへのすべての移行パスについて SELinux ポリシーを検査し、これらのパスを一覧表示します。以下の出力は完了していません。

```
~]$ sepolicy transition -s httpd_t -t system_mail_t
httpd_t @ exim_exec_t --> system_mail_t
httpd_t @ courier_exec_t --> system_mail_t
httpd_t @ sendmail_exec_t --> system_mail_t
httpd_t ... httpd_suexec_t @ sendmail_exec_t --> system_mail_t
httpd_t ... httpd_suexec_t @ exim_exec_t --> system_mail_t
httpd_t ... httpd_suexec_t @ courier_exec_t --> system_mail_t
httpd_t ... httpd_suexec_t ... httpd_mojomojo_script_t @ sendmail_exec_t --> system_mail_t
```

sepolicy transition の詳細は、sepolicy-transition(8) man ページを参照してください。

5.4. 手動ページの生成: SEPOLICY MANPAGE

sepolicy manpage コマンドは、ドメインを処理する SELinux ポリシーに基づいて man ページを生成します。その結果、このようなドキュメントは常に最新になります。自動生成される man ページの名前は、それぞれプロセスドメイン名と、**_selinux** の接尾辞 (**httpd_selinux** など) で設定されます。

man ページには、限られたドメインに関する SELinux ポリシーのさまざまな部分に関する情報を提供するセクションがいくつかあります。

- **Entrypoints** セクションには、ドメイン移行時に実行する必要がある実行ファイルがすべて含まれています。
- **Process Types** セクションでは、ターゲットドメインと同じ接頭辞で始まるプロセスタイプをすべて表示します。
- **Booleans** セクションは、ドメインに関連付けられたブール値の一覧を表示します。
- **Port Types** セクションにはドメインと同じ接頭辞に一致するポートタイプが含まれ、これらのポートタイプに割り当てられるデフォルトのポート番号を説明します。
- **Managed Files** のセクションでは、ドメインへの書き込みが許可されるタイプと、これらのタイプに関連付けられたデフォルトのパスを説明します。

- File Contexts セクションには、ドメインに関連付けられたすべてのファイルタイプと、これらのファイルタイプとシステム上のデフォルトのパスラベリングの使用方法が説明されています。
- Sharing Files セクションでは、public_content_t などのドメイン共有タイプの使用方法を説明します。

sepolicy manpage の詳細は、sepolicy-manpage(8) man ページを参照してください。

---

[7] sepolicy manpage の詳細は 「手動ページの生成: sepolicy manpage」 を参照してください。

第6章 ユーザーの制限

Red Hat Enterprise Linux では、ユーザーはデフォルトで SELinux **unconfined_u** ユーザーにマッピングされています。**unconfined_u** が実行するプロセスはすべて、**unconfined_t** ドメイン内にあります。つまり、ユーザーは、標準の Linux DAC ポリシーの制限内で、システム全体にアクセスできます。ただし、Red Hat Enterprise Linux では、限られた SELinux ユーザーの多くが利用できます。つまり、ユーザーは限られた機能セットに限定できます。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、(ユーザーによっては) SELinux ユーザーに設定された制限を継承できなくなります。

- X Window System を実行する
- ネットワークを使用する
- setuid アプリケーションを実行します (SELinux ポリシーで許可されていない場合)。
- または、**su** コマンドおよび **sudo** コマンドを実行します。

たとえば、SELinux **user_u** ユーザーが実行しているプロセスは、**user_t** ドメイン内にあります。このようなプロセスはネットワークに接続できますが、**su** コマンドまたは **sudo** コマンドは実行できません。これにより、ユーザーからシステムを保護できます。制限のあるユーザーとその能力の詳細は、「制限のあるユーザーおよび制限のないユーザー」、表3.1「SELinux ユーザー機能」 を参照してください。

6.1. LINUX および SELINUX のユーザーマッピング

root で、以下のコマンドを実行して、Linux ユーザーと SELinux ユーザーのマッピングを表示します。

```
~]# 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       *
system_u             system_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、続いて、これは SELinux **unconfined_u** ユーザーにマッピングされます。**useradd** コマンドで Linux ユーザーを作成し、オプションを指定しないと、SELinux **unconfined_u** ユーザーにマッピングされます。以下は、default-mapping を定義します。

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

6.2. 新規の LINUX ユーザーの制限: USERADD

SELinux **unconfined_u** ユーザーにマッピングされた Linux ユーザーは、**unconfined_t** ドメインで実行します。これは、**unconfined_u** にマッピングされた Linux ユーザーとしてログインしながら**id -Z** を実行すると確認できます。

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

Linux ユーザーが **unconfined_t** ドメインで実行している場合は、SELinux ポリシールールが適用されますが、**unconfined_t** ドメインで実行している Linux ユーザーのほぼすべてのアクセスを許可するポ

リシールールが存在します。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。セキュリティー上の利点は、Linux ユーザーが制限されていなくても、アプリケーションは制限されたままになるため、アプリケーションの欠陥の悪用はポリシーにより制限される可能性があることです。

注記

これにより、システムがユーザーから保護されることはありません。その代わりに、ユーザーとシステムは、アプリケーションの欠陥による潜在的な不具合から保護されています。

useradd コマンドで Linux ユーザーを作成する場合は、**-Z** オプションを使用して、マッピング先の SELinux ユーザーを指定します。以下の例では、新しい Linux ユーザー **useruuser** を作成し、そのユーザーを SELinux **user_u** ユーザーにマップしています。SELinux **user_u** ユーザーにマッピングされた Linux ユーザーは、**user_t** ドメインで実行します。このドメインでは、SELinux ポリシーで setuid アプリケーションが許可 (**passwd** など) されていない限り、Linux ユーザーは setuid アプリケーションを実行できず、**su** コマンドまたは **sudo** コマンドを実行できないため、これらのコマンドで root ユーザーになることができません。

手順6.1新しい Linux ユーザーを user_u SELinux ユーザーに限定する

1. root で、SELinux **user_u** ユーザーにマッピングする新しい Linux ユーザー (**useruuser**) を作成します。

```
~]# useradd -Z user_u useruuser
```

2. **useruuser** と **user_u** との間のマッピングを表示するには、root で以下のコマンドを実行します。

```
~]# 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       *
system_u             system_u             s0-s0:c0.c1023       *
useruuser            user_u               s0                   *
```

3. root で、Linux **useruuser** ユーザーにパスワードを割り当てます。

```
~]# passwd useruuser
Changing password for user useruuser.
New password: Enter a password
Retype new password: Enter the same password again
passwd: all authentication tokens updated successfully.
```

4. 現行セッションからログアウトし、Linux **useruuser** ユーザーとしてログインします。ログインすると、**pam_selinux** により、Linux ユーザーが SELinux ユーザー (この例では**user_u**) にマップされ、作成される SELinux コンテキストが設定されます。Linux ユーザーのシェルはこのコンテキストで起動します。以下のコマンドを入力して、Linux ユーザーのコンテキストを表示します。

```
~]$ id -Z
user_u:user_r:user_t:s0
```

5. Linux **useruuser** のセッションをログアウトし、アカウントで再度ログインします。Linux **useruuser** を使用しない場合は、root で次のコマンドを実行し、ホームディレクトリーとともに削除します。

```
~]# userdel -Z -r useruuser
```

6.3. 既存の LINUX ユーザーの制限: SEMANAGE ログイン

Linux ユーザーが SELinux **unconfined_u** ユーザー (デフォルトの動作) にマッピングされており、マップされている SELinux ユーザー を変更する場合は、**semanage login** コマンドを使用します。以下の例では、**newuser** という名前の新しい Linux ユーザーを作成し、その Linux ユーザーを SELinux**user_u** ユーザーにマップします。

手順6.2 SELinux ユーザーへの Linux ユーザーのマッピング

1. root で、新しい Linux ユーザー (**newuser**) を作成します。このユーザーはデフォルトのマッピングを使用するため、**semanage login -l**の出力には表示されません。

```
~]# useradd newuser
```

```
~]# 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       *
system_u             system_u             s0-s0:c0.c1023       *
```

2. Linux **newuser** ユーザーを SELinux**user_u** ユーザーにマッピングするには、root で以下のコマンドを実行します。

```
~]# semanage login -a -s user_u newuser
```

-a オプションは新しいレコードを追加し、**-s** オプションは、Linux ユーザーをマッピングする SELinux ユーザーを指定します。最後の引数 **newuser** は、指定した SELinux ユーザーにマッピングする Linux ユーザーです。

3. Linux **newuser** ユーザーと **user_u** との間のマッピングを表示するには、**semanage** ユーティリティーを再度使用します。

```
~]# semanage login -l

Login Name           SELinux User         MLS/MCS Range        Service

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

4. root で、Linux **newuser** ユーザーにパスワードを割り当てます。

```
~]# passwd newuser
Changing password for user newuser.
New password: Enter a password
Retype new password: Enter the same password again
passwd: all authentication tokens updated successfully.
```

5. 現在のセッションからログアウトし、Linux **newuser** ユーザーとしてログインします。次のコマンドを実行すると、**newuser** の SELinux コンテキストが表示されます。

```
~]$ id -Z
user_u:user_r:user_t:s0
```

6. Linux **newuser** のセッションをログアウトし、アカウントで再度ログインします。Linux **newuser** を使用しない場合は、root で次のコマンドを実行し、ホームディレクトリーとともに削除します。

```
~]# userdel -r newuser
```

root で、Linux **newuser** ユーザーおよび **user_u** 間のマッピングを削除します。

```
~]# semanage login -d newuser
```

```
~]# 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       *
system_u             system_u             s0-s0:c0.c1023       *
```

6.4. デフォルトマッピングの変更

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、続いて、これは SELinux **unconfined_u** ユーザーにマッピングされます。新規の Linux ユーザーと、SELinux ユーザーに特別にマッピングされていない Linux ユーザーをデフォルトで制限する場合は、**semanage login** コマンドを使用してデフォルトのマッピングを変更します。

たとえば、root で次のコマンドを入力すると、デフォルトのマッピングが **unconfined_u** から **user_u** に変更されます。

```
~]# semanage login -m -S targeted -s "user_u" -r s0 __default__
```

__default__ ログインが **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-s0:c0.c1023      *
root                 unconfined_u       s0-s0:c0.c1023      *
system_u             system_u           s0-s0:c0.c1023      *
```

新しい Linux ユーザーを作成し、その SELinux ユーザーを指定しない場合、または既存の Linux ユーザーがログインし、**semanage login -l** の出力からの入力と一致しない場合は、**__default__** ログインと同様に、**user_u** にマッピングされます。

デフォルトの動作に戻すには、root で次のコマンドを実行します。次に示すように、SELinux **unconfined_u** ユーザーに **__default__** ログインをマッピングします。

```
~]# semanage login -m -S targeted -s "unconfined_u" -r s0-s0:c0.c1023 __default__
```

6.5. XGUEST: キオスクモード

xguest パッケージは、Kiosk ユーザーアカウントを提供します。このアカウントは、図書館、銀行、空港、情報キオスク、コーヒーショップなど、人々が立ち寄って使用するマシンを保護するために使用されます。キオスクのユーザーアカウントは非常に制限されています。基本的に、ユーザーはログインして Firefox を使用してインターネット Web サイトを閲覧することしかできません。ゲストユーザーは **xguest_u** に割り当てられています。表3.1「SELinux ユーザー機能」 を参照してください。ファイルの作成や設定の変更など、このアカウントでログインした変更は、ログアウトすると失われます。

キオスクアカウントを設定するには、次のコマンドを実行します。

1. root で、xguest パッケージをインストールします。必要に応じて依存関係をインストールします。

   ```
   ~]# yum install xguest
   ```

2. キオスクアカウントをさまざまな人が使用できるようにするために、アカウントはパスワードで保護されていません。そのため、アカウントは SELinux が Enforcing モードで実行されている場合にのみ保護できます。このアカウントでログインする前に、**getenforce** ユーティリティーを使用して、SELinux が Enforcing モードで実行していることを確認します。

   ```
   ~]$ getenforce
   Enforcing
   ```

   実行していない場合は、「SELinux のステータスおよびモードの永続的変更」 で Enforcing モードへの変更に関する詳細を参照してください。SELinux が Permissive モードまたは Disabled の場合は、このアカウントでログインできません。

3. GNOME ディスプレイマネージャー (GDM) を使用しないと、このアカウントにログインできません。xguest パッケージがインストールされると、**Guest** アカウントが GDM ログイン画面に追加されます。

6.6. アプリケーションを実行するユーザーに対するブール値

Linux ユーザーが、ホームディレクトリーや、書き込みアクセス権を持つ **/tmp** ディレクトリーでアプリケーション (ユーザーの権限を継承) を実行できないようにすると、欠陥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、ユーザーが所有するファイルを変更できなくなります。

ブール値はこの動作を変更するために使用でき、root として実行する必要がある **setsebool** ユーティリティーで設定されます。**setsebool -P** は永続的な変更を行います。システムを再起動しても変更を持続させない場合は、**-P** オプションを使用しないでください。

guest_t

guest_t ドメイン内の Linux ユーザーによる、ホームディレクトリーおよび/tmp でのアプリケーションの実行を阻止するには、次のコマンドを実行します。

```
~]# setsebool -P guest_exec_content off
```

xguest_t

xguest_t ドメイン内の Linux ユーザーによる、ホームディレクトリーおよび/tmp でのアプリケーションの実行を阻止するには、次のコマンドを実行します。

```
~]# setsebool -P xguest_exec_content off
```

user_t

user_t ドメイン内の Linux ユーザーによる、ホームディレクトリーおよび/tmp でのアプリケーションの実行を阻止するには、次のコマンドを実行します。

```
~]# setsebool -P user_exec_content off
```

staff_t

staff_t ドメイン内の Linux ユーザーによる、ホームディレクトリーおよび/tmp でのアプリケーションの実行を阻止するには、次のコマンドを実行します。

```
~]# setsebool -P staff_exec_content off
```

staff_exec_content のブール値を有効にし、staff_t ドメインの Linux ユーザーを許可して、ホームディレクトリーおよび /tmp でアプリケーションを実行するには、以下のコマンドを実行します。

```
~]# setsebool -P staff_exec_content on
```

第7章 SANDBOX を使用したプログラムの保護

sandbox セキュリティーユーティリティーは、システム管理者が、厳しい制限のある SELinux ドメイン内でアプリケーションを実行できるように、一連の SELinux ポリシーを追加します。新しいファイルを開いたり、ネットワークにアクセスするパーミッションの制限を定義できます。これにより、システムのダメージを防ぎ、信頼できないソフトウェアの処理特性を安全にテストできます。

7.1. サンドボックスを使用したアプリケーションの実行

sandbox ユーティリティーを使用する前に、policycoreutils-sandbox パッケージがインストールされている必要があります。

```
~]# yum install policycoreutils-sandbox
```

アプリケーションを制限する基本的な構文は、以下のとおりです。

```
~]$ sandbox [options] application_under_test
```

sandbox でグラフィカルアプリケーションを実行する場合は、**-X** オプションを使用します。以下に例を示します。

```
~]$ sandbox -X evince
```

この**-X**は、sandbox に、アプリケーションに限定されたセカンダリー X Server (この場合は**evince**) を設定してから、必要なリソースをコピーし、ユーザーの **home** ディレクトリーまたは **/tmp** ディレクトリーに閉じた仮想環境を作成するように指示します。

あるセッションから次のセッションへデータを保存するには、以下のコマンドを実行します。

```
~]$ sandbox -H sandbox/home -T sandbox/tmp -X firefox
```

/home には **sandbox/home** が使用され、**/tmp** には **sandbox/tmp** が使用されることに注意してください。異なるアプリケーションは、異なる制限された環境に置かれます。アプリケーションは全画面モードで実行されるため、その他の機能にアクセスできなくなります。前述のように、**sandbox_x_file_t** のラベルが付いたファイルを除き、ファイルを開いたり、作成したりすることはできません。

sandbox 内では、ネットワークへのアクセスも最初は不可能です。アクセスを許可するには、**sandbox_web_t** ラベルを使用します。たとえば、Firefox を起動する場合:

```
~]$ sandbox -X -t sandbox_web_t firefox
```

警告

sandbox_net_t ラベルは、すべてのネットワークポートへの無制限の双方向ネットワークアクセスを許可します。**sandbox_web_t** では、Web ブラウジングにのみ必要なポートへの接続が許可されます。

sandbox_net_t の使用は、必要な場合にのみ注意して行う必要があります。

詳細および利用可能なオプションの完全な一覧は、**sandbox** (8) man ページを参照してください。

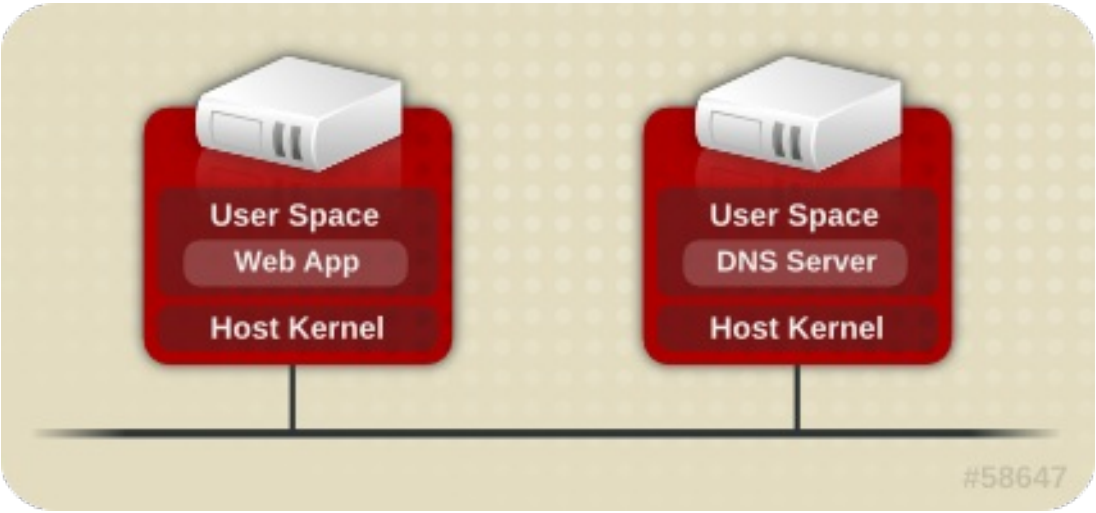
第8章 SVIRT

sVirt は、SELinux と仮想化を統合する Red Hat Enterprise Linux に含まれる技術で、仮想マシンの使用時にセキュリティーを向上させるために、MAC (Mandatory Access Control) を適用します。このような技術を統合する主な理由は、セキュリティーを向上させ、ホストや別の仮想マシン向けの攻撃ベクトルとして使用される可能性があるハイパーバイザーのバグに対してシステムを強化するためです。

本章では、sVirt が Red Hat Enterprise Linux の仮想化技術と統合する方法を説明します。

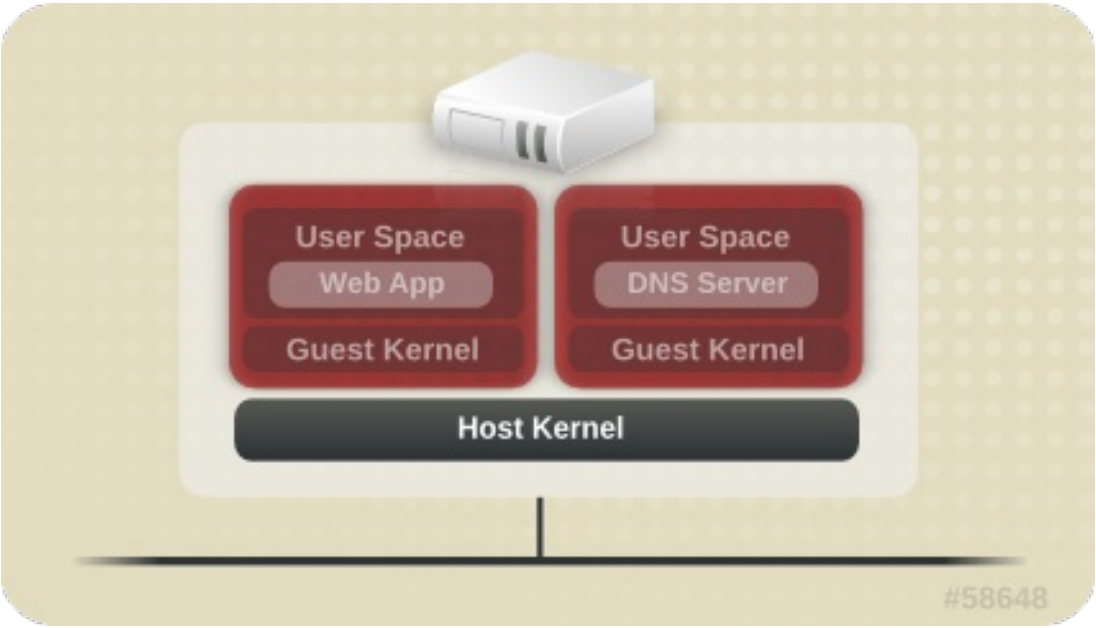
非仮想化環境

非仮想化環境では、ホストは物理的に相互分離しており、各ホストには Web サーバーや DNS サーバーなどのサービスで設定される自己完結型の環境があります。これらのサービスは、独自のユーザースペース、ホストカーネル、物理ホストと直接通信して、ネットワークに直接サービスを提供します。以下のイメージは、仮想化されていない環境を表しています。



仮想化環境

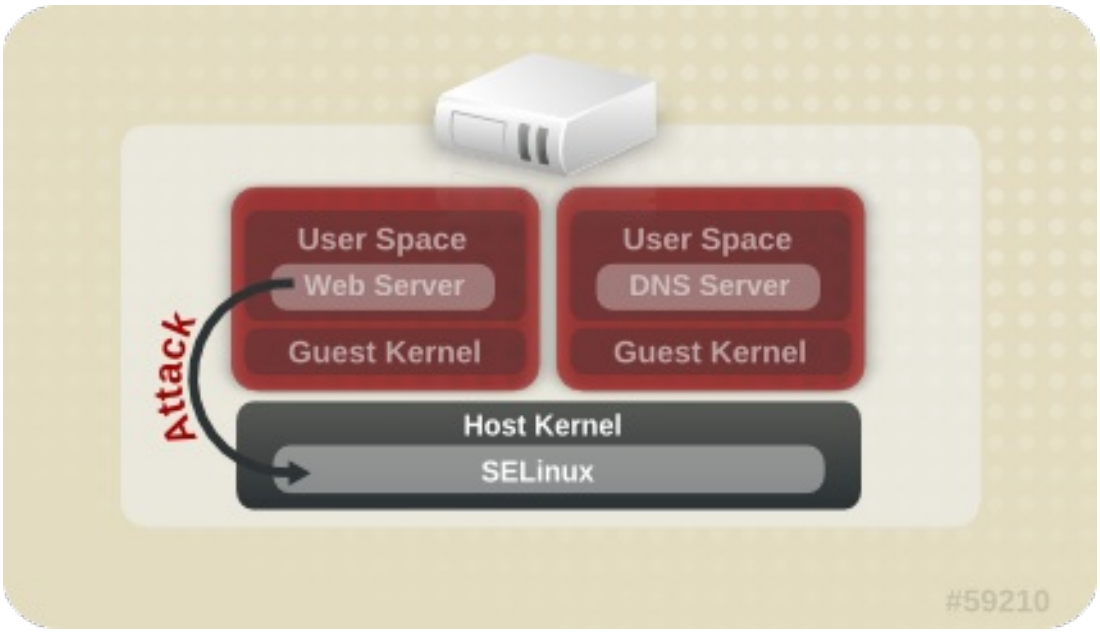
仮想化環境では、複数のオペレーティングシステムを 1 つのホストカーネルと物理ホスト内に (ゲスト) として収納できます。以下のイメージは、仮想化環境を表しています。



8.1. セキュリティーおよび仮想化

サービスが仮想化されていない場合は、マシンは物理的に分離されています。通常、エクスプロイトは影響を受けるマシン内に封じ込められ、ネットワーク攻撃は明らかな例外となります。仮想化環境でサービスをグループ化すると、システムに追加の脆弱性が発生します。ハイパーバイザーに、ゲストインスタンスによって悪用される可能性のあるセキュリティー上の欠陥がある場合、このゲストはホストだけでなく、そのホストで実行されている他のゲストも攻撃できる可能性があります。これは理論上のものではありません。攻撃は、ハイパーバイザーにすでに存在しています。このような攻撃はゲストインスタンス以外にも及ぶ可能性があり、その他のゲストを攻撃にさらす可能性があります。

sVirt は、ゲストを隔離し、悪用された場合にさらに攻撃を開始する能力を制限するための取り組みです。これは、次のイメージで示されています。攻撃は仮想マシンを壊し、別のホストインスタンスに拡張することはできません。



SELinux では、MAC (Mandatory Access Control) の実装で、仮想インスタンスにプラグ可能なセキュリティーフレームワークを導入しました。sVirt フレームワークでは、ゲストとそのリソースに一意のラベルを付けることができます。ラベルを付けたら、異なるゲスト間のアクセスを拒否できるルールを適用できます。

8.2. SVIRT のラベル付け

SELinux の保護下にあるその他のサービスと同様、sVirt はプロセスベースのメカニズムと制限を使用して、ゲストインスタンスに対してセキュリティーの層を追加します。通常の使用では、sVirt がバックグラウンドで動作していることに気付くことさえありません。本セクションでは、sVirt のラベリング機能を説明します。

以下の出力に示すように、sVirt を使用すると、各仮想マシン (VM) プロセスにラベルが付けられ、動的に生成されたレベルで実行されます。各プロセスは、レベルが異なる他の仮想マシンから分離されています。

```
~]# ps -eZ | grep qemu

system_u:system_r:svirt_t:s0:c87,c520 27950 ?  00:00:17 qemu-kvm
system_u:system_r:svirt_t:s0:c639,c757 27989 ? 00:00:06 qemu-system-x86
```

以下の出力に示すように、実際のディスクイメージには、プロセスに一致するように自動的にラベルが付けられます。

```
~]# ls -lZ /var/lib/libvirt/images/*

system_u:object_r:svirt_image_t:s0:c87,c520   image1
```

以下の表は、sVirt の使用時に割り当てることができるさまざまなラベルの概要を示しています。

表8.1 sVirt ラベル

タイプ	SELinux コンテキスト	説明
仮想マシンプロセス	system_u:system_r:svirt_t:MCS1	MCS1 は、ランダムに選択された MCS フィールドです。現在は、約 500,000 のラベルがサポートされています。
仮想マシンのイメージ	system_u:object_r:svirt_image_t:MCS1	同じ MCS フィールドで **svirt_t** とラベル付けされたプロセスのみが、このイメージファイルおよびデバイスの読み取り/書き込みが可能です。
仮想マシンの共有読み取り/書き込みコンテンツ	system_u:object_r:svirt_image_t:s0	**svirt_t** のラベルが付いたすべてのプロセスは、svirt_image_t:s0 ファイルおよびデバイスに書き込むことができます。
仮想マシンのイメージ	system_u:object_r:virt_content_t:s0	イメージが存在する場合に使用されるシステムのデフォルトラベル。**svirt_t** 仮想プロセスは、このラベルの付いたファイル/デバイスの読み取りはできません。

sVirt を使用する場合は、静的ラベリングを実行することもできます。静的ラベルにより管理者は、仮想マシン用に MCS/MLS フィールドなどの特定のラベルを選択できます。静的なラベルが付いた仮想マシンを実行する管理者は、イメージファイルに正しいラベルを設定する責任を担います。仮想マシンは常にそのラベルで起動し、sVirt システムは静的なラベルの付いた仮想マシンのコンテンツを決して変更しません。これにより、MLS 環境で sVirt コンポーネントを実行できます。要件に応じて、システムに応じて異なる感度レベルで複数の仮想マシンを実行することもできます。

第9章 セキュアな LINUX コンテナー

Linux コンテナー (LXC) は、システムで同じサービスの複数のコピーを同時に実行できるようにする低レベルの仮想化機能です。完全仮想化と比較して、コンテナーは新しいシステム全体を起動する必要がなく、使用メモリーが少なく、読み取り専用の方法でベースオペレーティングシステムを使用できます。たとえば、LXC では、システムデータを共有しながら、各自のデータを使用して、複数の Web サーバーを同時に実行したり、root ユーザーとして実行したりできます。ただし、コンテナー内で特権プロセスを実行すると、コンテナーの外部で実行しているその他のプロセスや、その他のコンテナーで実行しているプロセスに影響する可能性があります。セキュアな Linux コンテナーは SELinux コンテキストを使用するため、そのコンテナー内で実行しているプロセスが互いに、またはホストと相互作用しないようにします。

Docker アプリケーションは、Red Hat Enterprise Linux で Linux コンテナーを管理する主なユーティリティーです。もしくは、libvirt パッケージが提供する **virsh** コマンドラインユーティリティーを使用することもできます。

Linux コンテナーの詳細は、Getting Started with Containers を参照してください。

第10章 SELINUX SYSTEMD のアクセス制御

Red Hat Enterprise Linux 7 では、システムサービスは **systemd** デーモンにより制御されます。Red Hat Enterprise Linux の以前のリリースでは、デーモンを起動する方法が 2 つありました。

- システムの起動時に、System V **init** デーモンは**init.rc** スクリプトを起動し、このスクリプトにより必要なデーモンを起動します。たとえば、システムの起動時に起動した Apache サーバーには、以下の SELinux ラベルが貼られています。

  ```
  system_u:system_r:httpd_t:s0
  ```

- 管理者が手動で**init.rc** スクリプトを起動すると、デーモンが実行します。たとえば、**service httpd restart** コマンドを Apache サーバーで実行すると、表示される SELinux ラベルは次のようになります。

  ```
  unconfined_u:system_r:httpd_t:s0
  ```

手動で開始すると、このプロセスでは、開始した SELinux ラベルのユーザー部分が採用され、上記の 2 つのシナリオでのラベリングに矛盾が発生します。**systemd** デーモンでは、遷移が大きく異なります。**systemd** は、**init_t** タイプを使用して、システムでデーモンを開始および停止するすべての呼び出しを処理するため、デーモンを手動で再起動すると、ラベルのユーザー部分を上書きできます。その結果、上記の両方のシナリオのラベルは、期待どおりに **system_u:system_r:httpd_t:s0** で、SELinux ポリシーを改善して、どのドメインがどのユニットを制御できるかを管理できます。

10.1. サービスへの SELINUX のアクセス権限

以前のバージョンの Red Hat Enterprise Linux では、管理者は System V Init スクリプトのラベルに基づいて、どのユーザーまたはアプリケーションがサービスを開始または停止できるかを制御できました。現在、**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、呼び出し元が管理しようとしてい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。

たとえば、管理者は、NetworkManager が **systemctl** を実行して D-Bus メッセージを**systemd** に送信できるようにする必要がありました。これにより、NetworkManager が要求するサービスが開始または停止します。実際、NetworkManager は、**systemctl** ができることをすべて実行できるようになっていました。また、特定のサービスのみを開始または停止できるように、限定管理者を設定することもできませんでした。

この問題を修正するために、**systemd** は SELinux Access Manager としても機能します。**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できます。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。ポリシーの変更には、以下のパーミッションを持つ **service** という名前の新しいクラスが関与します。

```
class service
{
      start
      stop
      status
```

```
        reload
        kill
        load
        enable
        disable
}
```

たとえば、ポリシーの作成者は、ドメインがサービスのステータスを取得したり、サービスを開始および停止したりできるようになりましたが、サービスを有効または無効にすることはできません。SELinux および **systemd** でのアクセス制御操作は、すべての場合で一致するわけではありません。**systemd** メソッドの呼び出しを SELinux アクセスチェックと揃えるために、マッピングが定義されています。表10.2「SELinux アクセスチェックでの systemd 一般システムコールのマッピング」は、システム全般のアクセスチェックをカバーするのに対し、表10.1「SELinux アクセスチェックでの systemd ユニットファイルメソッド呼び出しのマッピング」は、ユニットファイルのアクセスチェックをマップします。いずれかのテーブルに一致するものが見つからない場合は、**undefined** システムチェックが呼び出されます。

表10.1 SELinux アクセスチェックでの systemd ユニットファイルメソッド呼び出しのマッピング

systemd ユニットファイルメソッド	SELinux アクセスチェック
DisableUnitFiles	disable
EnableUnitFiles	enable
GetUnit	status
GetUnitByPID	status
GetUnitFileState	status
Kill	stop
KillUnit	stop
LinkUnitFiles	enable
ListUnits	status
LoadUnit	status
MaskUnitFiles	disable
PresetUnitFiles	enable
ReenableUnitFiles	enable
Reexecute	start
Reload	reload

systemd ユニットファイルメソッド	SELinux アクセスチェック
ReloadOrRestart	start
ReloadOrRestartUnit	start
ReloadOrTryRestart	start
ReloadOrTryRestartUnit	start
ReloadUnit	reload
ResetFailed	stop
ResetFailedUnit	stop
Restart	start
RestartUnit	start
開始	start
StartUnit	start
StartUnitReplace	start
停止	stop
StopUnit	stop
TryRestart	start
TryRestartUnit	start
UnmaskUnitFiles	enable

表10.2 SELinux アクセスチェックでの systemd 一般システムコールのマッピング

systemd 汎用システムコール	SELinux アクセスチェック
ClearJobs	reboot
FlushDevices	halt
Get	status

systemd 汎用システムコール	SELinux アクセスチェック
GetAll	status
GetJob	status
GetSeat	status
GetSession	status
GetSessionByPID	status
GetUser	status
Halt	halt
Introspect	status
KExec	reboot
KillSession	halt
KillUser	halt
ListJobs	status
ListSeats	status
ListSessions	status
ListUsers	status
LockSession	halt
PowerOff	halt
再起動	reboot
SetUserLinger	halt
TerminateSeat	halt
TerminateSession	halt
TerminateUser	halt

例10.1 システムサービスに関する SELinux ポリシー

sesearch ユーティリティーを使用すると、システムサービスのポリシールールを一覧表示できます。たとえば、**sesearch -A -s NetworkManager_t -c service** コマンドを呼び出すと以下が返されます。

```
allow NetworkManager_t dnsmasq_unit_file_t : service { start stop status reload kill load } ;
allow NetworkManager_t nscd_unit_file_t : service { start stop status reload kill load } ;
allow NetworkManager_t ntpd_unit_file_t : service { start stop status reload kill load } ;
allow NetworkManager_t pppd_unit_file_t : service { start stop status reload kill load } ;
allow NetworkManager_t polipo_unit_file_t : service { start stop status reload kill load } ;
```

10.2. SELINUX と JOURNALD

systemd では、**journald** デーモン (**systemd-journal** とも呼ばれます) は、ログデータを収集して保存するシステムサービスである **syslog** ユーティリティーの代替手段です。これは、カーネル、**libc syslog()** 関数を使用するユーザープロセス、システムサービスの標準出力およびエラー出力、またはネイティブの API を使用して受信したログ情報に基づいて、構造化されたジャーナルおよびインデックス化されたジャーナルを作成し、維持します。暗黙的に、各ログメッセージのメタデータフィールドを安全な方法で収集します。

systemd-journal サービスは、セキュリティーを向上させるために、SELinux で使用できます。SELinux は、設計された機能を実行することのみを許可することで、プロセスを制御します。ポリシー作成者のセキュリティー目標によっては、これよりも低くなることもあります。たとえば、SELinux は、侵害された **ntpd** プロセスが Network Time を処理する以外の操作を行わないようにします。ただし、**ntpd** プロセスは、**syslog** メッセージを送信するため、SELinux は、侵害されたプロセスが引き続きメッセージを送信できるようにします。侵害された**ntpd**は、**syslog** メッセージを他のデーモンに合わせてフォーマットし、管理者を誤認させる可能性があります。さらに悪い場合は、**syslog** ファイルを読み込んでシステム全体を危険にさらすユーティリティーです。

systemd-journal デーモンは、すべてのログメッセージを検証し、とりわけ SELinux ラベルを追加します。その後、ログメッセージで不整合を検出し、発生前にこのタイプの攻撃を防ぐことができます。**journalctl** ユーティリティーを使用すると、**systemd** ジャーナルのログをクエリーできます。コマンドライン引数が指定されていない場合は、このユーティリティーを実行すると、最も古いエントリーから始まり、ジャーナルの全コンテンツが一覧表示されます。システムコンポーネントのログを含む、システムで生成されたすべてのログを表示するには、root で **journalctl** を実行します。root 以外のユーザーで実行した場合は、現在ログインしているユーザーに関連するログのみが出力されます。

例10.2 **journalctl** でログの一覧表示

journalctl を使用すると、特定の SELinux ラベルに関連するログをすべて一覧表示できます。たとえば、次のコマンドは、**system_u:system_r:policykit_t:s0** ラベルに記録されているログをすべて一覧表示します。

```
~]# journalctl _SELINUX_CONTEXT=system_u:system_r:policykit_t:s0
Oct 21 10:22:42 localhost.localdomain polkitd[647]: Started polkitd version 0.112
Oct 21 10:22:44 localhost.localdomain polkitd[647]: Loading rules from directory /etc/polkit-1/rules.d
Oct 21 10:22:44 localhost.localdomain polkitd[647]: Loading rules from directory /usr/share/polkit-1/rules.d
Oct 21 10:22:44 localhost.localdomain polkitd[647]: Finished loading, compiling and executing 5 rules
Oct 21 10:22:44 localhost.localdomain polkitd[647]: Acquired the name
```

```
org.freedesktop.PolicyKit1 on the system bus Oct 21 10:23:10 localhost polkitd[647]:
Registered Authentication Agent for unix-session:c1 (system bus name :1.49, object path
/org/freedesktop/PolicyKit1/AuthenticationAgent, locale en_US.UTF-8) (disconnected from
bus)
Oct 21 10:23:35 localhost polkitd[647]: Unregistered Authentication Agent for unix-
session:c1 (system bus name :1.80 [/usr/bin/gnome-shell --mode=classic], object path
/org/freedesktop/PolicyKit1/AuthenticationAgent, locale en_US.utf8)
```

journalctl の詳細は、journalctl(1) man **ページを参照してください。**

第11章 トラブルシューティング

本章では、SELinux がアクセスを拒否した場合の動作、問題の上位 3 つの原因、正しいラベル付けに関する情報の入手先、SELinux の拒否分析、および **audit2allow** でカスタムポリシーモジュールを作成する方法について説明します。

11.1. アクセスが拒否された場合の動作

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。SELinux がアクセスを拒否すると、拒否メッセージがログに記録されます。この拒否は AVC 拒否としても知られており、実行しているデーモンに応じて別の場所にログが記録されます。

デーモン: auditd on

ログの場所: **/var/log/audit/audit.log**

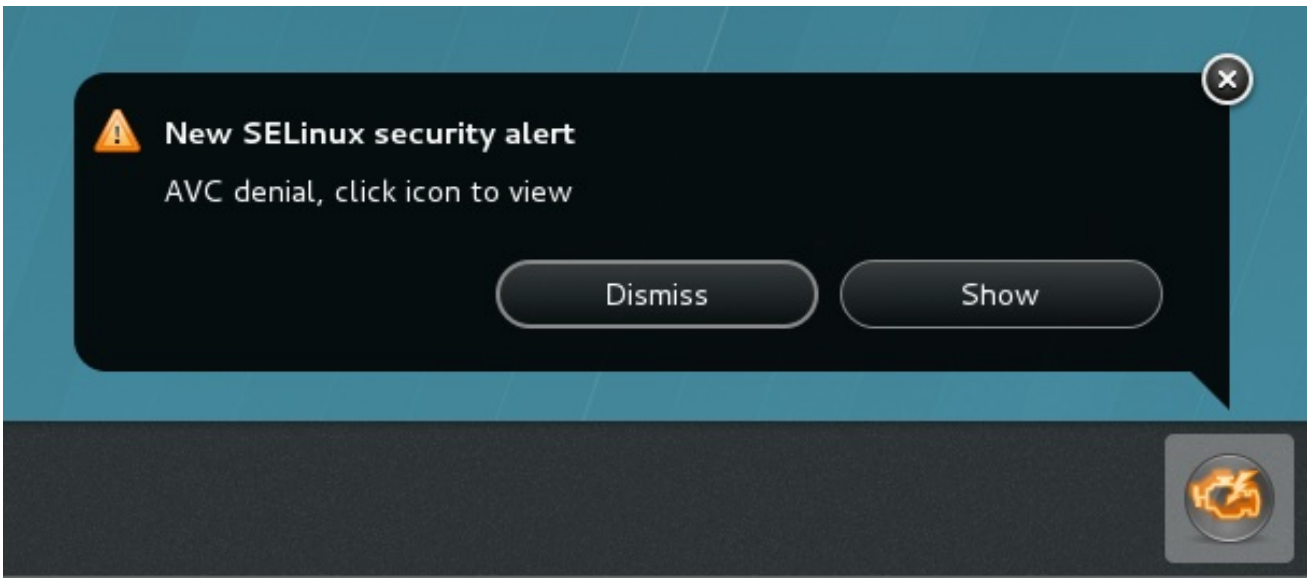
デーモン: auditd off、 rsyslogd on

ログの場所: **/var/log/messages**

デーモン: setroubleshootd、 rsyslogd、 auditd on

ログの場所: **/var/log/audit/audit.log**./**var/log/messages** にも送信される、読み取りが容易な拒否メッセージ

X Window System を実行中で、 setroubleshoot パッケージおよび setroubleshoot-server パッケージがインストールされており、**setroubleshootd** デーモンおよび **audd** デーモンを実行している場合は、SELinux がアクセスを拒否したときに警告が表示されます。



Show をクリックすると、SELinux がアクセスを拒否した理由の詳細な分析と、アクセスを許可するための可能な解決策が表示されます。X Window System を実行していない場合は、SELinux によってアクセスが拒否されたときはわかりにくくなります。たとえば、Web サイトをブラウズするユーザーには、次のようなエラーが表示されます。

> Forbidden
>
> You don't have permission to access *file name* on this server

このような状況で、DAC ルール (スタンダードの Linux パーミッション) でアクセスが許可されている場合は、**/var/log/messages** および **/var/log/audit/audit.log** で、それぞれ **"SELinux is preventing"** エラーおよび **"denied"** エラーを確認してください。これは、root ユーザーになり、以下のコマンドを実行して実行できます。

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

```
~]# grep "denied" /var/log/audit/audit.log
```

11.2. 問題の上位 3 つの原因

以下のセクションでは、問題の上位 3 つの原因 (ラベリングの問題、サービスのブール値とポートの設定、および SELinux ルールの進化) を説明します。

11.2.1. ラベル付けの問題

SELinux を実行しているシステムでは、すべてのプロセスとファイルに、セキュリティー関連の情報を含むラベルが付けられています。この情報は SELinux コンテキストと呼ばれます。このラベルが間違っていると、アクセスが拒否される可能性があります。ラベルが間違っていると、正しくないアプリケーションのプロセスにラベルが割り当てられる可能性があります。これにより、SELinux がアクセスを拒否し、プロセスが誤ったラベルのファイルを作成する可能性があります。

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、管理者は、Web サイトに **/var/www/html/** を使用する代わりに、**/srv/myweb/** を使用します。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。[8].**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実行して変更を適用します。

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

ファイルコンテキストの設定にコンテキストを追加する方法は、「永続的な変更 - semanage fcontext」 を参照してください。

11.2.1.1. 正しいコンテキストとは？

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。以下の例は、ラベルが間違っているファイルが含まれるディレクトリーで **matchpathcon** を使用した例を示しています。

```
~]$ 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の外に存在してはなりません。このようなファイルを正しいタイプに復元するには、**restorecon** ユーティリティーを使用します。

```
~]# 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
```

matchpathcon の詳細な例は、「デフォルトの SELinux コンテキストの確認」を参照してください。

11.2.2. 制限のあるサービスの実行方法

サービスはさまざまな方法で実行できます。これに対応するには、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 ポリシーの記述に関する知識がなくても、ランタイム時に SELinux ポリシーの一部を変更できるようにするブール値を使用して実現できます。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 サーバーが 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
```

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
```

ブール値の一覧と、ブール値の有無を確認するには、**getsebool -a** コマンドを実行します。ブール値の一覧、各ブール値の意味、有効/無効を説明するには、root で **semanage boolean -l** を実行します。ブール値の一覧表示と設定の詳細は、「ブール値」 を参照してください。

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 サーバーがリッスンできるポートを定義します。この場合は、TCP ポート 80、443、488、8008、8009、および 8443 です。**httpd** がポート 9876 (**Listen 9876**) でリッスンするように管理者が **httpd.conf** を設定しても、これを反映するようにポリシーが更新されていない場合、以下のコマンドは失敗します。

```
~]# 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** コマンドを実行して、ポリシー設定にポートを追加します。[9]:

```
~]# 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。

11.2.3. ルールの進化とアプリケーションの破損

アプリケーションが壊れ、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 の新しいバージョンがリリースされた場合、アクセスが許可されるはずなのに、現在のポリシーがこれまでに確認したことのないアクションを実行し、アクセスが拒否される可能性があります。

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。**audit2allow** の使用方法は、「アクセスの許可: audit2allow」 を参照してください。

11.3. 問題の修正

以下のセクションでは、問題のトラブルシューティングを行う方法を説明します。SELinux ルールよりも前に確認される Linux のパーミッションの確認、SELinux によるアクセス拒否が考えられる原因があるが、拒否はログに記録されない場合、サービス用の man ページ (ラベリングとブール値に関する情報を記載)、システム全体ではなく 1 つ プロセスが Permissive を実行できるようにするための Permissive ドメイン、拒否メッセージの検索および表示方法、拒否の分析、**audit2allow** を使用したカスタムポリシーモジュールの作成について説明します。

11.3.1. Linux の権限

アクセスが拒否された場合は、標準の Linux 権限を確認します。*1章 はじめに* で説明されているように、ほとんどのオペレーティングシステムは DAC (Discretionary Access Control) システムを使用してアクセスを制御し、ユーザーが所有するファイルのパーミッションを制御できるようにします。SELinux ポリシールールは、DAC ルールの後にチェックされます。DAC ルールがアクセスを拒否する場合は、SELinux ポリシールールは使用されません。

アクセスが拒否され、SELinux の拒否ログが記録されていない場合は、次のコマンドを使用して、標準の Linux 権限を表示します。

```
~]$ ls -l /var/www/html/index.html
-rw-r----- 1 root root 0 2009-05-07 11:06 index.html
```

この例では、**index.html** は root ユーザーおよびグループが所有します。root ユーザーには読み取り権限と書き込み権限 (**-rw**) があり、root グループのメンバーには読み取り権限 (**-r-**) があります。他の人にはアクセス権がありません (**---**)。既定では、このような権限では、**httpd** がこのファイルを読み込むことができません。この問題を解決するには、**chown** コマンドを使用して所有者とグループを変更します。このコマンドは、root で実行する必要があります。

```
~]# chown apache:apache /var/www/html/index.html
```

これは、デフォルト設定 (**httpd** は Linux Apache ユーザーとして実行) であることを前提としています。別のユーザーで **httpd** を実行している場合は、**apache:apache** をそのユーザーに置き換えます。

Linux のパーミッション管理の詳細は、Fedora Documentation Project "Permissions"のドラフトを参照してください。

11.3.2. サイレント拒否の考えられる原因

特定の状況では、SELinux がアクセスを拒否した場合に、AVC 拒否メッセージがログに記録されないことがあります。アプリケーションやシステムライブラリー関数は、多くの場合、タスクを実行するのに必要以上のアクセスをプローブします。無害なアプリケーションプローブを AVC 拒否で監査ログ記録につけることなく、最小の権限を維持するために、ポリシーは *dontaudit* ルールを使うことで、パーミッションを許可することなく、サイレントな AVC 拒否を行うことができます。このようなルールは、標準ポリシーでは一般的です。dontaudit の欠点は、SELinux がアクセスを拒否しても、拒否メッセージはログに記録されず、トラブルシューティングが難しくなることです。

dontaudit ルールを一時的に無効にし、すべての拒否ログを記録するには、root で次のコマンドを実行します。

```
~]# semodule -DB
```

-D オプションは、dontaudit ルールを無効にします。**-B** オプションはポリシーを再構築します。**semodule -DB** を実行したら、権限の問題が発生していたアプリケーションを実行してみて、アプリケーションに関連する SELinux の拒否がログに記録されているかどうかを確認します。拒否を許可するかどうかを決定する際には注意してください。拒否の中には、dontaudit ルールで無視および処理されるものもあります。疑わしい場合、またはガイドが必要な場合は、fedora-selinux-list などの SELinux のリストにある他の SELinux ユーザーおよび開発者に連絡してください。

ポリシーを再構築し、**dontaudit**ルールを有効にするには、root で次のコマンドを実行します。

```
~]# semodule -B
```

これにより、ポリシーが元の状態に復元されます。dontaudit ルールの一覧を表示するには、**sesearch --dontaudit** を実行します。**-s *domain*** オプションと **grep** コマンドを使用して、検索を絞り込みます。以下に例を示します。

```
~]$ sesearch --dontaudit -s smbd_t | grep squid
dontaudit smbd_t squid_port_t : tcp_socket name_bind ;
dontaudit smbd_t squid_port_t : udp_socket name_bind ;
```

拒否の分析の詳細については、「Raw 監査メッセージ」 および「sealert メッセージ」 を参照してください。

11.3.3. サービスの man ページ

サービスの man ページには、指定した状況で使用するファイルタイプや、サービスのアクセスを変更するためのブール値 (NFS ボリュームにアクセスする**httpd**など) などの貴重な情報が記載されています。この情報は、標準のマニュアルページ、または **sepolicy manpage** ユーティリティーを使用してすべてのサービスドメインの SELinux ポリシーから自動的に生成できるマニュアルページにある場合があります。このような man ページは、***service-name*_selinux** 形式で命名されます。このような man ページは、selinux-policy-doc パッケージに同梱されています。

たとえば、httpd_selinux(8) man ページには、特定の状況に使用するファイルタイプに関する情報と、スクリプトの共有、ファイルの共有、ユーザーのホームディレクトリー内のディレクトリーへのアクセスなどを許可するブール値が含まれます。サービスの SELinux 情報が記載されたその他の man ページには、以下が含まれます

- Samba - たとえば、samba_selinux(8) の man ページでは、**samba_enable_home_dirs** ブール値を有効にすると、Samba がユーザーのホームディレクトリーを共有できるようになることを説明しています。
- NFS - nfsd_selinux(8) の man ページでは、nfsd プロセスをできるだけ安全な方法で設定できるように、SELinux の nfsd ポリシーを説明しています。

man ページの情報は、正しいファイルタイプとブール値の設定に役立ちます。これにより、SELinux によるアクセスを拒否することを防ぐことができます。

sepolicy manpage の詳細は 「手動ページの生成: sepolicy manpage」 を参照してください。

11.3.4. Permissive ドメイン

SELinux が Permissive モードで実行している場合、SELinux はアクセスを拒否しませんが、Enforcing モードで実行しても拒否されるアクションのログは SELinux によって記録されます。以前は、1つのドメインを Permissive にすることはできませんでした (プロセスはドメインで実行されることに注意してください)。特定の状況では、これによりシステム全体が問題のトラブルシューティングを受け入れられるようになりました。

パーミッションドメインを使用すると、管理者はシステム全体を許容するのではなく、1つのプロセス (ドメイン) に対して Permissive を実行するように設定できます。SELinux によるチェックは依然として Permissive ドメインに対して実行されますが、カーネルは、SELinux がアクセスを拒否したであろう状況では、アクセスを許可し、AVC の拒否を報告します。

Permissive ドメインには、以下の用途があります。

- これを使用すると、1つのプロセス (ドメイン) を Permissive にして、システム全体をリスクにさらさずに問題のトラブルシューティングを行うことができます。
- これにより、管理者は新しいアプリケーションのポリシーを作成できます。以前は、最小限のポリシーを作成してから、マシン全体を Permissive モードにしてアプリケーションを実行できるようにし、SELinux の拒否ログが記録されるようにすることが推奨されていました。次に、ポリシーを作成するために **audit2allow** を使用することができます。これにより、システム全体が危険にさらされていました。Permissive ドメインでは、システム全体を危険にさらさずに、新しいポリシーのドメインのみが Permissive とマークされます。

11.3.4.1. ドメインを Permissive にする

ドメインを Permissive にするには、**semanage permissive -a *domain*** コマンドを実行します。*domain* は、Permissive にするドメインです。たとえば、root で次のコマンドを実行して、**httpd_t** ドメイン (Apache HTTP サーバーを実行しているドメイン) を Permissive (許容) にします。

```
~]# semanage permissive -a httpd_t
```

Permissive にしたドメインの一覧を表示するには、root で **semodule -l | grep permissive** を実行します。以下に例を示します。

```
~]# semodule -l | grep permissive
permissive_httpd_t    (null)
permissivedomains     (null)
```

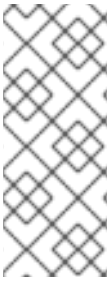
ドメインを Permissive にしない場合は、root で **semanage permissive -d *domain*** コマンドを実行します。以下に例を示します。

```
~]# semanage permissive -d httpd_t
```

11.3.4.2. Permissive ドメインの無効化

permissivedomains.pp には、システム上にある Permissive ドメイン宣言がすべて含まれています。すべての Permissive ドメインを無効にするには、root で次のコマンドを実行します。

```
~]# semodule -d permissivedomains
```



注記

module -d コマンドでポリシーモジュールが無効になると、**semodule -l** コマンドの出力に表示されなくなります。無効にしたものを含むすべてのポリシーモジュールを表示するには、root で次のコマンドを実行します。

```
~]# semodule --list-modules=full
```

11.3.4.3. Permissive ドメインの拒否

SYSCALL メッセージは、Permissive ドメインでは異なります。以下は、Apache HTTP サーバーによる AVC の拒否 (および関連するシステムコール) の例になります。

```
type=AVC msg=audit(1226882736.442:86): avc:  denied  { getattr } for  pid=2427 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

type=SYSCALL msg=audit(1226882736.442:86): arch=40000003 syscall=196 success=no exit=-13
a0=b9a1e198 a1=bfc2921c a2=54dff4 a3=2008171 items=0 ppid=2425 pid=2427 auid=502 uid=48
gid=48 euid=48 suid=48 fsuid=48 egid=48 sgid=48 fsgid=48 tty=(none) ses=4 comm="httpd"
exe="/usr/sbin/httpd" subj=unconfined_u:system_r:httpd_t:s0 key=(null)
```

既定では、**httpd_t** ドメインは Permissive ではありません。また、このようなアクションは拒否され、**SYSCALL** メッセージには **success=no** が含まれます。以下は、同じ状況での AVC による拒否の例になります。ただし、**semanage permissive -a httpd_t** コマンドを実行して、**httpd_t** ドメインを Permissive にします。

```
type=AVC msg=audit(1226882925.714:136): avc:  denied  { read } for  pid=2512 comm="httpd"
name="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

type=SYSCALL msg=audit(1226882925.714:136): arch=40000003 syscall=5 success=yes exit=11
a0=b962a1e8 a1=8000 a2=0 a3=8000 items=0 ppid=2511 pid=2512 auid=502 uid=48 gid=48
euid=48 suid=48 fsuid=48 egid=48 sgid=48 fsgid=48 tty=(none) ses=4 comm="httpd"
exe="/usr/sbin/httpd" subj=unconfined_u:system_r:httpd_t:s0 key=(null)
```

AVC 拒否が記録されましたが、**SYSCALL** メッセージの **success=yes** に示されるように、アクセスは拒否されませんでした。

Permissive ドメインの詳細は、Dan Walsh のブログエントリー "Permissive Domains" を参照してください。

11.3.5. 拒否の検索と表示

本セクションでは、setroubleshoot、setroubleshoot-server、dbus、および audit パッケージがインストールされ、**auditd**。**rsyslogd**、および **setroubleshootd** デーモンが実行していることを前提としています。これらのデーモンの起動方法は、「どのログファイルが使用されるか」 を参照してください。**ausearch**、**aureport**、および **sealert** などの SELinux AVC メッセージの検索および表示には、多数のユーティリティーを使用できます。

ausearch

audit パッケージは、さまざまな検索基準に基づいてイベントの **audit** デーモンログを照会できる **ausearch** ユーティリティーを提供します。[10] **ausearch** ユーティリティーは、**/var/log/audit/audit.log** にアクセスするため、root ユーザーとして実行する必要があります。

検索対象: すべての拒否

コマンド: ausearch -m avc,user_avc,selinux_err,user_selinux_err

検索対象: 今日の拒否

コマンド: ausearch -m avc -ts today

検索対象: 直近の 10 分間の拒否

コマンド: ausearch -m avc -ts recent

特定サービスの SELinux AVC メッセージを検索するには、-c *comm-name* オプションを使用します。*comm-name* は実行ファイルの名前 (Apache HTTP サーバーの場合は httpd、Samba の場合は smbd など) です。

```
~]# ausearch -m avc -c httpd
```

```
~]# ausearch -m avc -c smbd
```

ausearch コマンドを使用するたびに、読みやすくするために --interpret (-i) オプションを使用するか、スクリプト処理で --raw (-r) オプションを使用することが推奨されます。ausearch オプションの詳細は、ausearch(8) の man ページを参照してください。

aureport

audit パッケージは、aureport ユーティリティーを提供します。これにより、監査システムログのサマリーレポートを生成します。[11] aureport ユーティリティーは、/var/log/audit/audit.log にアクセスするため、root ユーザーとして実行する必要があります。SELinux の拒否メッセージと、その発生頻度を一覧表示するには、aureport -a を実行します。以下は、2 つの拒否を含む出力例になります。

```
~]# aureport -a

AVC Report
===============================================================
# date time comm subj syscall class permission obj event
===============================================================
1. 05/01/2009 21:41:39 httpd unconfined_u:system_r:httpd_t:s0 195 file getattr
system_u:object_r:samba_share_t:s0 denied 2
2. 05/03/2009 22:00:25 vsftpd unconfined_u:system_r:ftpd_t:s0 5 file read
unconfined_u:object_r:cifs_t:s0 denied 4
```

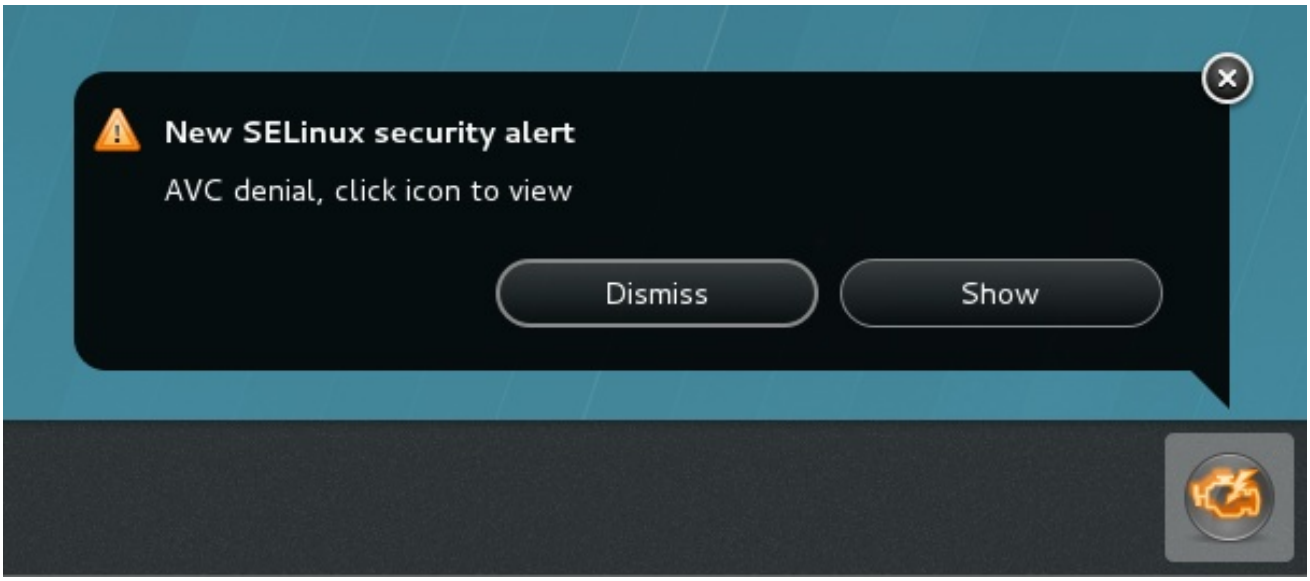
sealert

setroubleshoot-server パッケージは、sealert ユーティリティーを提供します。これは、setroubleshoot-server により変換された拒否メッセージを読み取ります。[12] 拒否は、/var/log/messages にあるように ID が割り当てられています。以下は、messages からの拒否例になります。

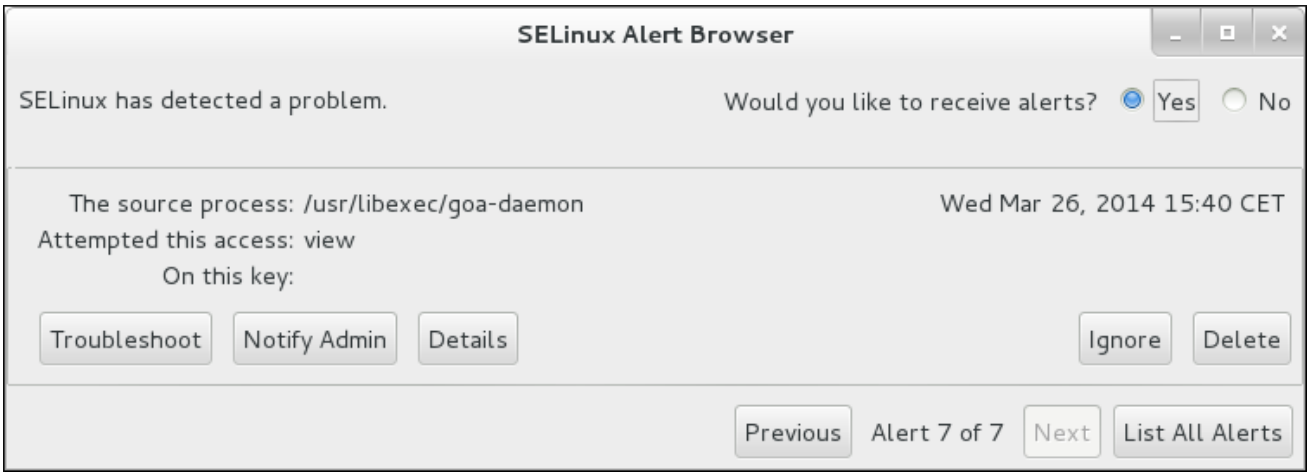
```
setroubleshoot: SELinux is preventing /usr/sbin/httpd from name_bind access on the tcp_socket. For
complete SELinux messages. run sealert -l 8c123656-5dda-4e5d-8791-9e3bd03786b7
```

この例では、拒否 ID は 8c123656-5dda-4e5d-8791-9e3bd03786b7 です。-l オプションは、識別子を引き数として取ります。sealert -l 8c123656-5dda-4e5d-8791-9e3bd03786b7 コマンドを実行すると、SELinux がアクセスを拒否した理由を詳細に分析し、アクセスを許可する解決策を示すことができます。

X Window System を実行中で、setroubleshoot パッケージおよび setroubleshoot-server パッケージがインストールされており、**setroubleshootd** デーモン、**dbus** デーモン、および **auditd** デーモンが実行されている場合は、SELinux がアクセスを拒否したときに警告が表示されます。



Show をクリックすると、**sealert** 画面が表示されます。これにより、トラブルシューティングが可能になります。



または、**sealert -b** コマンドを実行して、**sealert** GUI を起動します。すべての拒否メッセージの詳細な解析を表示するには、**sealert -l *** コマンドを実行します。

11.3.6. Raw 監査メッセージ

Raw 監査メッセージは **/var/log/audit/audit.log** にログ記録されます。以下は、Apache HTTP サーバー (**httpd_t** ドメインで実行している) が **/var/www/html/file1** ファイル (**samba_share_t** タイプのラベルが付いている) にアクセスしようとしたときに発生した AVC 拒否メッセージ (および関連するシステムコール) の例です。

```
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

type=SYSCALL msg=audit(1226874073.147:96): arch=40000003 syscall=196 success=no exit=-13
```

> a0=b98df198 a1=bfec85dc a2=54dff4 a3=2008171 items=0 ppid=2463 pid=2465 auid=502 uid=48 gid=48 euid=48 suid=48 fsuid=48 egid=48 sgid=48 fsgid=48 tty=(none) ses=6 comm="httpd" exe="/usr/sbin/httpd" subj=unconfined_u:system_r:httpd_t:s0 key=(null)

{ getattr }

中括弧内の項目は、拒否された権限を示しています。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ことを示します。これは、ファイルを読み取る前に発生します。アクセスしているファイルのラベルが間違っているため、このアクションは拒否されています。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。

comm="*httpd*"

プロセスを開始した実行ファイル。実行可能ファイルのフルパスは、システムコール (SYSCALL) メッセージの exe= セクションに記載されています。この例では exe="/usr/sbin/httpd" になります。

path="*/var/www/html/file1*"

プロセスが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実行しようとしたプロセスの 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の SELinux コンテキスト (ターゲット)。この例では、これが file1 の SELinux コンテキストです。samba_share_t タイプは、httpd_t ドメインで実行しているプロセスからアクセスできません。

特定の状況では、tcontext は、scontext と一致します (例: プロセスが、ユーザー ID など、実行中のプロセスの特性を変更するシステムサービスを実行しようとする場合)。また、プロセスが通常の制限よりも多くのリソース (メモリーなど) を使用しようとすると、tcontext がscontext に一致する可能性があります。その結果、そのプロセスがこれらの制限を破ることが許可されているかどうかを確認するためのセキュリティーチェックが行われます。

システムコール (SYSCALL) メッセージでは、以下の 2 つの項目が重要になります。

- **success=*no*: 拒否 (AVC) が強制されたかどうか。success=no はシステムコールが成功しなかった (SELinux がアクセスを拒否した) ことを示します。success=yes はシステムコールが成功したことを示します。これは、unconfined_service_t や kernel_t など、Permissive ドメインまたは制限のないドメインで見ることができます。**

- **exe="*/usr/sbin/httpd*": プロセスを開始した実行ファイルのフルパスです。この例ではexe="/usr/sbin/httpd" です。**

ファイルタイプは、SELinux によるアクセスを拒否する一般的な原因です。トラブルシューティングを開始するには、ソースコンテキスト (scontext) をターゲットコンテキスト (tcontext) と比較します。プロセス (scontext) がこのようなオブジェクト (tcontext) にアクセスする必要がありますか ?たとえば、Apache HTTP Server(httpd_t) は、httpd_sys_content_t、public_content_t など、httpd_selinux(8) man ページで指定されたタイプにのみアクセスする必要があります (特に設定されていない限り)。

11.3.7. sealert メッセージ

拒否は、/var/log/messages にあるように ID が割り当てられています。以下は、Apache HTTP サーバー (httpd_t ドメインで実行中) が /var/www/html/file1 ファイル (samba_share_t タイプのラベルが付いたファイル) にアクセスしようとしたときに発生した AVC 拒否 (messages に記録) の例です。

```
hostname setroubleshoot: SELinux is preventing httpd (httpd_t) "getattr" to /var/www/html/file1
(samba_share_t). For complete SELinux messages. run sealert -l 32eee32b-21ca-4846-a22f-
0ba050206786
```

推奨されるように、sealert -l 32eee32b-21ca-4846-a22f-0ba050206786 コマンドを実行して、メッセージ全体を表示します。このコマンドは、ローカルマシンでのみ機能し、sealert GUI と同じ情報を表示します。

```
~]$ sealert -l 32eee32b-21ca-4846-a22f-0ba050206786
SELinux is preventing httpd from getattr access on the file /var/www/html/file1.

*****  Plugin restorecon (92.2 confidence) suggests   ************************

If you want to fix the label.
/var/www/html/file1 default label should be httpd_sys_content_t.
Then you can run restorecon.
Do
# /sbin/restorecon -v /var/www/html/file1

*****  Plugin public_content (7.83 confidence) suggests   *********************

If you want to treat file1 as public content
Then you need to change the label on file1 to public_content_t or public_content_rw_t.
Do
# semanage fcontext -a -t public_content_t '/var/www/html/file1'
# restorecon -v '/var/www/html/file1'

*****  Plugin catchall (1.41 confidence) suggests   **************************

If you believe that httpd should be allowed getattr access on the file1 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 'httpd' --raw | audit2allow -M my-httpd
# semodule -i my-httpd.pp


Additional Information:
Source Context                system_u:system_r:httpd_t:s0
Target Context                unconfined_u:object_r:samba_share_t:s0
Target Objects                /var/www/html/file1 [ file ]
Source                        httpd
Source Path                   httpd
Port                          <Unknown>
Host                          hostname.redhat.com
Source RPM Packages
Target RPM Packages
Policy RPM                    selinux-policy-3.13.1-166.el7.noarch
Selinux Enabled               True
Policy Type                   targeted
```

```
Enforcing Mode              Enforcing
Host Name                   hostname.redhat.com
Platform                    Linux hostname.redhat.com
                            3.10.0-693.el7.x86_64 #1 SMP Thu Jul 6 19:56:57
                            EDT 2017 x86_64 x86_64
Alert Count                 2
First Seen                  2017-07-20 02:52:11 EDT
Last Seen                   2017-07-20 02:52:11 EDT
Local ID                    32eee32b-21ca-4846-a22f-0ba050206786

Raw Audit Messages
type=AVC msg=audit(1500533531.140:295): avc:  denied  { getattr } for  pid=24934 comm="httpd"
path="/var/www/html/file1" dev="vda1" ino=31457414 scontext=system_u:system_r:httpd_t:s0
tcontext=unconfined_u:object_r:samba_share_t:s0 tclass=file


Hash: httpd,httpd_t,samba_share_t,file,getattr
```

概要

拒否されたアクションの概要これは、/var/log/messages の拒否と同じです。この例では、httpd プロセスが、samba_share_t タイプのラベルが付いたファイル (file1) へのアクセスを拒否されています。

詳細な説明

より詳細な説明この例では、file1 には samba_share_t タイプのラベルが付けられています。このタイプは、Samba を使用してエクスポートするファイルおよびディレクトリーに使用されます。この説明では、このようなアクセスが必要な場合は、Apache HTTP Server および Samba がアクセス可能なタイプにタイプを変更することが推奨されています。

アクセスの許可

アクセスを許可する方法を説明します。ファイルの再ラベル付け、ブール値の有効化、またはローカルポリシーモジュールの作成を行います。この場合、Apache HTTP Server と Samba の両方がアクセスできるタイプでファイルにラベルを付けることが提案されます。

Fix コマンド

アクセスを許可し、拒否を解決するための推奨されるコマンド。この例では、file1 タイプを public_content_t に変更するためのコマンドを提供します。これは Apache HTTP Server および Samba からアクセスできます。

追加情報

バグ報告に役に立つ情報 (ポリシーパッケージの名前やバージョン (selinux-policy-3.13.1-166.el7.noarch) ですが、拒否が発生した理由の解決には役立たない場合があります。

Raw 監査メッセージ

拒否に関連付けられた /var/log/audit/audit.log からの Raw 監査メッセージ。AVC 拒否に関する詳細は、「Raw 監査メッセージ」 を参照してください。

11.3.8. アクセスの許可: audit2allow

警告

実稼働環境では、このセクションの例を使用しないでください。これは、audit2allow ユーティリティーの使用を実証する目的でのみ使用されます。

audit2allow ユーティリティーは、拒否された操作のログから情報を収集し、SELinux ポリシー許可ルールを生成します。[13]「sealert メッセージ」に従って拒否メッセージを分析し、ラベル変更やブール値によるアクセスが許可されていない場合は、audit2allow を使用してローカルポリシーモジュールを作成します。SELinux がアクセスを拒否した場合に、audit2allow を実行すると、拒否したアクセスを許可する Type Enforcement ルールが生成されます。

SELinux 拒否が表示された場合、最初に audit2allow を使用してローカルポリシーモジュールを生成しないでください。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。詳細は、Four Key Causes of SELinux Errors のホワイトペーパーを参照してください。

以下の例は、audit2allow を使用してポリシーモジュールを作成する例を示しています。

1. **拒否メッセージと関連するシステムコールが/var/log/audit/audit.log ファイルにログ記録されます。**

```
type=AVC msg=audit(1226270358.848:238): avc:  denied  { write } for  pid=13349
comm="certwatch" name="cache" dev=dm-0 ino=218171
scontext=system_u:system_r:certwatch_t:s0 tcontext=system_u:object_r:var_t:s0 tclass=dir

type=SYSCALL msg=audit(1226270358.848:238): arch=40000003 syscall=39 success=no
exit=-13 a0=39a2bf a1=3ff a2=3a0354 a3=94703c8 items=0 ppid=13344 pid=13349
auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=(none)
ses=4294967295 comm="certwatch" exe="/usr/bin/certwatch"
subj=system_u:system_r:certwatch_t:s0 key=(null)
```

   この例では、certwatch は var_t タイプのラベルが付けられたディレクトリーへの書き込みアクセスを拒否します。「sealert メッセージ」に従って拒否メッセージを分析します。ラベル変更やブール値によるアクセスが許可されていない場合は、audit2allow を使用してローカルポリシーモジュールを作成します。

2. **以下のコマンドを入力して、アクセスが拒否された理由を人間が判読できるように説明します。audit2allow ユーティリティーは /var/log/audit/audit.log を読み取ります。そのような場合は、root ユーザーとして実行する必要があります。**

```
~]# audit2allow -w -a
type=AVC msg=audit(1226270358.848:238): avc:  denied  { write } for  pid=13349
comm="certwatch" name="cache" dev=dm-0 ino=218171
scontext=system_u:system_r:certwatch_t:s0 tcontext=system_u:object_r:var_t:s0 tclass=dir
 Was caused by:
  Missing type enforcement (TE) allow rule.

 You can use audit2allow to generate a loadable module to allow this access.
```

-a コマンドラインオプションを指定すると、すべての監査ログが読み込まれます。**-w** オプションは、人間が判読できる記述を生成します。このように、Type Enforcement ルールがないため、アクセスが拒否されました。

3. 次のコマンドを入力して、拒否されたアクセスを許可する Type Enforcement ルールを表示します。

```
~]# audit2allow -a


#============= certwatch_t ==============
allow certwatch_t var_t:dir write;
```

重要

欠落している Type Enforcement ルールは、通常 SELinux ポリシーのバグにより発生するため、Red Hat Bugzilla で報告する必要があります。Red Hat Enterprise Linux の場合、Red Hat Enterprise Linux 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 および audit2allow -a コマンドの出力を追加します。

4. **audit2allow -a が表示するルールを使用するには、root で以下のコマンドを実行して、カスタムモジュールを作成します。-M オプションは、-M で指定された名前で、現在の作業ディレクトリーに、Type Enforcement ファイル (.te) を作成します。**

```
~]# audit2allow -a -M mycertwatch
******************** IMPORTANT ***********************
To make this policy package active, execute:

semodule -i mycertwatch.pp
```

5. **また、audit2allow は、Type Enforcement ルールをポリシーパッケージ (.pp) にコンパイルします。**

```
~]# ls
mycertwatch.pp  mycertwatch.te
```

モジュールをインストールするには、root で以下のコマンドを実行します。

```
~]# semodule -i mycertwatch.pp
```

重要

audit2allow で作成したモジュールは、必要以上のアクセスを許可する場合があります。audit2allow で作成したポリシーは、アップストリームの SELinux 一覧に投稿して確認することが推奨されます。 ポリシーにバグがあると思われる場合は、Red Hat Bugzilla でバグを作成します。

複数のプロセスからの拒否メッセージが複数存在し、1 つのプロセスに対してのみカスタムポリシーを作成する場合は、grep ユーティリティーを使用して、audit2allow の入力内容を絞り込みます。以下の例は、grep を使用して、audit2allow 経由で certwatch に関連する拒否メッセージのみを送信することを示しています。

■

```
~]# grep certwatch /var/log/audit/audit.log | audit2allow -R -M mycertwatch2
****************** IMPORTANT ***********************
To make this policy package active, execute:

semodule -i mycertwatch2.pp
```

---

[8] **/etc/selinux/targeted/contexts/files/** のファイルは、ファイルおよびディレクトリーのコンテキストを定義します。このディレクトリーのファイルは、**restorecon** ユーティリティーおよび **setfiles** ユーティリティーにより読み込まれ、ファイルおよびディレクトリーをデフォルトコンテキストに復元します。

[9] **semanage port -a** は、**/etc/selinux/targeted/modules/active/ports.local** ファイルーにエントリーを追加します。デフォルトでは、このファイルは root のみが表示できることに注意してください。

[10] **ausearch** の詳細は、ausearch(8) の man ページを参照してください。

[11] **aureport** の詳細は、aureport(8) の man ページを参照してください。

[12] **sealert** の詳細は、sealert(8) man ページを参照してください。

[13] **audit2allow** の詳細は、audit2allow(1) man ページを参照してください。

第12章 追加情報

12.1. コントリビューター

- Dominick Grift – テクニカルエディター
- Murray McAllister – Red Hat 製品セキュリティー
- James Morris – テクニカルエディター
- エリック・パリス - テクニカルエディター
- Scott Radvan – Red Hat Customer Content Services
- Daniel Walsh – Red Hat セキュリティーエンジニアリング

12.2. その他リソース

Fedora

- メインページ: http://fedoraproject.org/wiki/SELinux.
- トラブルシューティング: http://fedoraproject.org/wiki/SELinux/Troubleshooting.
- Fedora SELinux FAQ: https://fedoraproject.org/wiki/SELinux_FAQ.

アメリカ国家安全保障局 (NSA: National Security Agency)

NSA は、SELinux の元の開発者でした。NSA の National Information Assurance Research Laboratory (NIARL) の研究者は、Linux カーネルの主要なサブシステムで柔軟な強制アクセス制御を設計および実装し、Flask アーキテクチャーによって提供される新しいオペレーティングシステムコンポーネント、つまりセキュリティーサーバーとアクセスベクトルキャッシュを実装しました。

- SELinux のメイン Web サイト: https://www.nsa.gov/what-we-do/research/selinux/。
- SELinux メーリングリスト: https://www.nsa.gov/what-we-do/research/selinux/mailing-list.shtml。
- SELinux ドキュメント: https://www.nsa.gov/what-we-do/research/selinux/documentation/index.shtml.
- SELinux の背景: https://www.nsa.gov/what-we-do/research/selinux/related-work/。

Tresys Technology

Tresys Technology は、以下のアップストリームを対象にしています。

- SELinux ユーザースペースライブラリーとツール.
- SELinux 参照ポリシー

SELinux GitHub リポジトリー

- SELinux プロジェクト: https://github.com/SELinuxProject
- Tresys Technology: https://github.com/TresysTechnology/

SELinux Project Wiki

- メインページ: http://selinuxproject.org/page/Main_Page
- ドキュメント、メーリングリスト、Web サイト、ツールへのリンクなどのユーザーリソース: http://selinuxproject.org/page/User_Resources.

The SELinux Notebook - The Foundations - 4th Edition

http://freecomputerbooks.com/books/The_SELinux_Notebook-4th_Edition.pdf

DigitalOcean: An Introduction to SELinux on CentOS 7

https://www.digitalocean.com/community/tutorial_series/an-introduction-to-selinux-on-centos-7

IRC

Freenode の場合:

- #selinux
- #fedora-selinux

パート II. 制限のあるサービスの管理

本書のこの部分では、実用的なタスクに重点を置いて、さまざまなサービスのセットアップおよび設定方法を説明します。サービスごとに、仕様とともに最も一般的なタイプとブール値がリストされています。また、これらのサービスの設定や、SELinux が動作を補完する方法のデモンストレーションを行う実際の例も含まれています。

SELinux が Enforcing モードの場合、Red Hat Enterprise Linux で使用されるデフォルトのポリシーが Targeted ポリシーになります。対象となるプロセスは制限のあるドメインで実行され、対象とされないプロセスは制限のないドメインで実行されます。対象のポリシーおよび制限のあるプロセスと制限のないプロセスに関する詳細は、3章*Targeted ポリシー*を参照してください。

第13章 APACHE HTTP サーバー

Apache HTTP サーバーは、現在の HTTP 標準とともにオープンソースの HTTP サーバーを提供します。[14]

Red Hat Enterprise Linux では、httpd パッケージは Apache HTTP Server を提供します。以下のコマンドを実行して、httpd パッケージがインストールされているかどうかを確認します。

```
~]$ rpm -q httpd
package httpd is not installed
```

インストールされておらず、Apache HTTP Server を使用する場合は、root で **yum** ユーティリティーを使用してインストールします。

```
~]# yum install httpd
```

13.1. APACHE HTTP サーバーおよび SELINUX

SELinux を有効にすると、Apache HTTP Server (**httpd**) はデフォルトで制限ありで実行されます。制限のあるプロセスは、独自のドメインで実行され、他の制限のあるプロセスから分離されます。制限のあるプロセスが攻撃者によって侵害された場合、SELinux ポリシーの設定に応じて、攻撃者のリソースへのアクセスと、攻撃者が行う可能性のある損害は制限されます。以下の例は、独自のドメインで実行している **httpd** プロセスを示しています。この例では、httpd、setroubleshoot、setroubleshoot-server、および policycoreutils-python パッケージがインストールされていることを前提としています。

1. **getenforce** コマンドを実行して、SELinux が Enforcing モードで実行されていることを確認します。

   ```
   ~]$ getenforce
   Enforcing
   ```

   SELinux が Enforcing モードで実行されていると、このコマンドは **Enforcing** を返します。

2. root で以下のコマンドを入力し、**httpd** を起動します。

   ```
   ~]# systemctl start httpd.service
   ```

   サービスが実行中であることを確認します。出力には以下の情報が含まれている必要があります (タイムスタンプのみは異なります)。

   ```
   ~]# systemctl status httpd.service
   httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled)
     Active: active (running) since Mon 2013-08-05 14:00:55 CEST; 8s ago
   ```

3. **httpd** プロセスを表示するには、以下のコマンドを実行します。

   ```
   ~]$ ps -eZ | grep httpd
   system_u:system_r:httpd_t:s0    19780 ?        00:00:00 httpd
   system_u:system_r:httpd_t:s0    19781 ?        00:00:00 httpd
   system_u:system_r:httpd_t:s0    19782 ?        00:00:00 httpd
   ```

```
system_u:system_r:httpd_t:s0    19783 ?        00:00:00 httpd
system_u:system_r:httpd_t:s0    19784 ?        00:00:00 httpd
system_u:system_r:httpd_t:s0    19785 ?        00:00:00 httpd
```

httpd プロセスに関連付けられている SELinux コンテキストは**system_u:system_r:httpd_t:s0** です。コンテキストの 2 番目の部分 **httpd_t** はタイプです。タイプは、プロセスのドメインとファイルのタイプを定義します。この場合、**httpd** プロセスは **httpd_t** ドメインで実行されています。

SELinux ポリシーは、制限のあるドメイン (**httpd_t**など) で実行されるプロセスが、ファイル、その他のプロセス、およびシステム一般とどのように対話するかを定義します。**httpd** によるアクセスを可能にするには、ファイルが正しくラベル付けされている必要があります。たとえば、**httpd** は **httpd_sys_content_t** タイプのラベルが付いたファイルを読み取ることができますが、Linux(DAC) パーミッションで書き込みが許可されている場合でも書き込みはできません。スクリプトのネットワークアクセスの許可、NFS および CIFS ボリュームへの **httpd** アクセスの許可、**httpd** による Common Gateway Interface (CGI) スクリプトの実行の許可など、特定の動作を許可するには、ブール値を有効にする必要があります。

/etc/httpd/conf/httpd.conf ファイルが TCP ポート 80、443、488、8008、8009、または 8443 以外のポートを **httpd** がリッスンするように設定されている場合は、**semanage port** コマンドを使用して、新しいポート番号を SELinux ポリシー設定に追加する必要があります。以下の例では、**httpd** 用の SELinux ポリシー設定で定義されていないポートをリッスンするように **httpd** を設定すると、**httpd** が起動できなくなります。また、この例では、ポリシーで定義されていない標準以外のポートを **httpd** が正常にリッスンできるように SELinux システムを設定する方法も示しています。この例では、httpd パッケージがインストールされていることを前提としています。この例では、各コマンドを root ユーザーとして実行します。

1. 以下のコマンドを実行して、**httpd** が実行されていないことを確認します。

```
~]# systemctl status httpd.service
httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled)
        Active: inactive (dead)
```

出力が異なる場合は、プロセスを停止します。

```
~]# systemctl stop httpd.service
```

2. **semanage** ユーティリティーを使用して、SELinux で **httpd** がリッスンできるポートを表示します。

```
~]# semanage port -l | grep -w http_port_t
http_port_t                    tcp      80, 443, 488, 8008, 8009, 8443
```

3. **/etc/httpd/conf/httpd.conf** ファイルを編集します。**Listen** オプションを設定し、**httpd** の SELinux ポリシー設定で設定されていないポートを一覧表示します。この例では、**httpd** がポート 12345 をリッスンするように設定されています。

```
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses (0.0.0.0)
#
#Listen 12.34.56.78:80
Listen 127.0.0.1:12345
```

4. 以下のコマンドを入力して、**httpd** を起動します。

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

以下のような SELinux 拒否メッセージが記録されます。

```
setroubleshoot: SELinux is preventing the httpd (httpd_t) from binding to port 12345. For
complete SELinux messages. run sealert -l f18bca99-db64-4c16-9719-1db89f0d8c77
```

5. SELinux で、**httpd** がポート 12345 をリッスンできるようにするには、以下のコマンドが必要です。

```
~]# semanage port -a -t http_port_t -p tcp 12345
```

6. **httpd** を再び起動し、新しいポートをリッスンしていることを確認します。

```
~]# systemctl start httpd.service
```

7. **httpd** が標準以外のポート (この例では TCP 12345) でリッスンできるように SELinux が設定されました。**httpd** はこのポートで正常に起動します。

8. **httpd** が TCP ポート 12345 でリッスンして通信していることを証明するには、以下のように、指定したポートへの telnet 接続を開き、HTTP GET コマンドを実行します。

```
~]# telnet localhost 12345
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 200 OK
Date: Wed, 02 Dec 2009 14:36:34 GMT
Server: Apache/2.2.13 (Red Hat)
Accept-Ranges: bytes
Content-Length: 3985
Content-Type: text/html; charset=UTF-8
[...continues...]
```

13.2. タイプ

高度なプロセス分離を提供するために、SELinux Targeted ポリシーで使用される主要なパーミッション制御方法は Type Enforcement です。すべてのファイルとプロセスにはタイプのラベルが付いています。タイプはプロセスの SELinux ドメインを定義し、ファイルの SELinux タイプを定義します。SELinux ポリシールールは、タイプにアクセスするドメインであるか、別のドメインにアクセスするドメインであるかにかかわらず、タイプが相互にアクセスする方法を定義します。アクセスは、それを許可する特定の SELinux ポリシールールが存在する場合にのみ許可されます。

以下の例では、**/var/www/html/** ディレクトリーに新しいファイルを作成し、親ディレクトリー (**/var/www/html/**) から**httpd_sys_content_t** タイプを継承したファイルを示しています。

1. 以下のコマンドを入力して、**/var/www/html/** の SELinux コンテキストを表示します。

```
~]$ ls -dZ /var/www/html
drwxr-xr-x  root root system_u:object_r:httpd_sys_content_t:s0 /var/www/html
```

これは、/var/www/html/ に httpd_sys_content_t タイプのラベルが付けられていることを示しています。

2. root で touch ユーティリティーを使用して新規ファイルを作成します。

```
~]# touch /var/www/html/file1
```

3. 以下のコマンドを入力して SELinux コンテキストを表示します。

```
~]$ ls -Z /var/www/html/file1
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file1
```

ls -Z コマンドは、httpd_sys_content_t タイプのラベルが付けられた file1 を表示します。SELinux を使用すると、httpd はこのタイプでラベル付けされたファイルを読み取ることができますが、Linux の権限で書き込みアクセスが許可されている場合でも書き込みはできません。SELinux ポリシーは、httpd_t ドメイン (httpd が実行されている場所) で実行中のプロセスが読み取りと書き込みを実行できるタイプを定義します。これは、プロセスが別のプロセスによって使用されるファイルにアクセスできないようにするのに役立ちます。

たとえば、httpd は、httpd_sys_content_t タイプ (Apache HTTP サーバー向け) のラベルが付いたファイルにアクセスできますが、デフォルトでは、samba_share_t タイプ (Samba 向け) のラベルが付いたファイルにはアクセスできません。また、ユーザーホームディレクトリーのファイルには、user_home_t タイプがラベル付けされています。デフォルトでは、httpd は、ユーザーホームディレクトリーのファイルを読み書きできません。

以下は、httpd で使用されるタイプの一覧です。異なるタイプを使用すると、柔軟なアクセスを設定できます。

httpd_sys_content_t

静的 Web サイトで使用される .html ファイルなど、静的な Web コンテンツにはこのタイプを使用します。このタイプでラベル付けされたファイルには、httpd および httpd によって実行されるスクリプトからアクセス (読み取りのみ) できます。既定では、このタイプのラベルが付いたファイルおよびディレクトリーは、httpd やその他のプロセスに書き込んだり、修正したりすることができません。初期設定では、/var/www/html/ ディレクトリーに作成またはコピーされるファイルには、httpd_sys_content_t タイプに応じたラベルが付けられています。

httpd_sys_script_exec_t

このタイプは、httpd を実行するスクリプトに使用します。このタイプは、/var/www/cgi-bin/ ディレクトリーの CGI (Common Gateway Interface) スクリプトで一般的に使用されます。初期設定では、SELinux ポリシーにより、httpd が CGI スクリプトを実行できません。これを可能にするには、httpd_sys_script_exec_t タイプでスクリプトにラベルを付け、httpd_enable_cgi ブール値を有効にします。httpd_sys_script_exec_t のラベルが付いたスクリプトは、httpd による実行時にhttpd_sys_script_t ドメインで実行されます。httpd_sys_script_t ドメインは、postgresql_t および mysqld_t などの他のシステムドメインにアクセスできます。

httpd_sys_rw_content_t

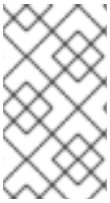
このタイプのラベルが付いたファイルは、httpd_sys_script_exec_t タイプのラベルが付いたスクリプトで作成できますが、その他のタイプのスクリプトでは変更できません。httpd_sys_rw_content_t タイプを使用して、httpd_sys_script_exec_t タイプのラベルが付いたスクリプトによる読み取りと書き込みの対象となるファイルにラベルを付ける必要があります。

httpd_sys_ra_content_t

このタイプのラベルが付いたファイルは、**httpd_sys_script_exec_t** タイプのラベルが付いたスクリプトにより追加できますが、その他のタイプのスクリプトにより変更することはできません。**httpd_sys_ra_content_t** タイプを使用して、**httpd_sys_script_exec_t** タイプのラベルが付いたスクリプトによる読み取りと追加を行うファイルに、ラベルを付ける必要があります。

httpd_unconfined_script_exec_t

このタイプでラベル付けされたスクリプトは、SELinux の保護なしで実行されます。その他のオプションをすべて使い切った後、複雑なスクリプトにはこのタイプのみを使用してください。**httpd** またはシステム全体の SELinux 保護を無効にするのではなく、このタイプを使用することが推奨されます。



注記

httpd で利用可能なタイプの詳細を確認するには、以下のコマンドを入力します。

```
~]$ grep httpd /etc/selinux/targeted/contexts/files/file_contexts
```

手順13.1 SELinux コンテキストの変更

ファイルおよびディレクトリーのタイプは、**chcon** コマンドで変更できます。**chcon** で行った変更は、ファイルシステムの再ラベル付けや **restorecon** コマンドでは存続しません。SELinux ポリシーは、ユーザーが任意のファイルの SELinux コンテキストを変更できるかどうかを制御します。以下の例では、**httpd** が使用する新しいディレクトリーと **index.html** ファイルを作成し、そのファイルおよびディレクトリーにラベルを付けて、それらに **httpd** がアクセスできるようにします。

1. **mkdir** ユーティリティーを root として使用して、**httpd** で使用されるファイルを保存するためのトップレベルのディレクトリー構造を作成します。

```
~]# mkdir -p /my/website
```

2. file-context 設定のパターンに一致しないファイルおよびディレクトリーには、**default_t** タイプのラベルが付けられます。このタイプは、制限のあるサービスからはアクセスできません。

```
~]$ ls -dZ /my
drwxr-xr-x  root root unconfined_u:object_r:default_t:s0 /my
```

3. root で以下のコマンドを入力し、**my/** ディレクトリーおよびサブディレクトリーのタイプを **httpd** にアクセスできるタイプに変更します。**my/website/** で作成されたファイルは、**default_t** タイプではなく、**httpd_sys_content_t** タイプを継承するようになったため、httpd からアクセスできるようになりました。

```
~]# chcon -R -t httpd_sys_content_t /my/
~]# touch /my/website/index.html
~]# ls -Z /my/website/index.html
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0 /my/website/index.html
```

chcon の詳細は、「一時的な変更: chcon」 を参照してください。

semanage fcontext コマンド (policycoreutils-python パッケージにより **semanage** が提供される) を使用して、再ラベル付けおよび **restorecon** コマンドに持続するラベル変更を行います。このコマンドは、file-context 設定に変更を加えます。次に、file-context 設定を読み取る **restorecon** を実行してラ

ベルの変更を適用します。以下の例では、**httpd** が使用する新しいディレクトリーと **index.html** ファイルを作成し、そのディレクトリーとファイルのラベルを永続的に変更して、**httpd** が使用できるようにします。

1. **mkdir** ユーティリティーを root として使用して、**httpd** で使用されるファイルを保存するためのトップレベルのディレクトリー構造を作成します。

```
~]# mkdir -p /my/website
```

2. root で以下のコマンドを入力し、ラベルの変更を file-context 設定に追加します。

```
~]# semanage fcontext -a -t httpd_sys_content_t "/my(/.*)?"
```

"/my(/.*)?" 式は、ラベルの変更が **my/** ディレクトリーと、その下のすべてのファイルおよびディレクトリーに適用されることを意味します。

3. root で **touch** ユーティリティーを使用して、新規ファイルを作成します。

```
~]# touch /my/website/index.html
```

4. root で以下のコマンドを入力し、ラベルの変更を適用します (**restorecon** は、手順 2 の **semanage** コマンドで変更された file-context 設定を読み取ります)。

```
~]# restorecon -R -v /my/
restorecon reset /my context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
restorecon reset /my/website context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
restorecon reset /my/website/index.html context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
```

semanage の詳細は 「永続的な変更 - semanage fcontext」を参照してください。

13.3. ブール値

SELinux は、サービスの実行に必要な最小アクセスレベルに基づいています。サービスはさまざまな方法で実行できます。そのため、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 ポリシーの記述に関する知識がなくても、ランタイム時に SELinux ポリシーの一部を変更できるようにするブール値を使用して実現できます。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。

ブール値の状態を変更するには、**setsebool** コマンドを使用します。たとえば、**httpd_anon_write** ブール値を有効にするには、root で以下のコマンドを入力します。

```
~]# setsebool -P httpd_anon_write on
```

同じ例を使用してブール値を無効にするには、以下のように、コマンドで **on** を **off** に変更します。

```
~]# setsebool -P httpd_anon_write off
```

注記

再起動後も、setsebool の変更を保持したくない場合は、-P オプションを使用しないでください。

以下は、以下は、httpd の実行方法に対応するために使用可能な一般的なブール値の説明です。

httpd_anon_write

このブール値を無効にすると、httpd は、public_content_rw_t タイプでラベル付けされたファイルへの読み取りアクセスのみを許可します。このブール値を有効にすると、httpd は、public_content_rw_t タイプでラベル付けされたファイル (公開ファイル転送サービスのファイルを含む公開ディレクトリーなど) に書き込むことができます。

httpd_mod_auth_ntlm_winbind

このブール値を有効にすると、httpd の mod_auth_ntlm_winbind モジュールを使用した NTLM および Winbind の認証メカニズムにアクセスできるようになります。

httpd_mod_auth_pam

このブール値を有効にすると、httpd の mod_auth_pam モジュールを使用した PAM 認証メカニズムにアクセスできるようになります。

httpd_sys_script_anon_write

このブール値は、公開ファイル転送サービスで使用される public_content_rw_t タイプでラベル付けされたファイルに、HTTP スクリプトによる書き込みアクセスを許可するかどうかを定義します。

httpd_builtin_scripting

このブール値は、httpd スクリプトへのアクセスを定義します。PHP コンテンツでは、このブール値を有効にする必要があることがよくあります。

httpd_can_network_connect

このブール値を無効にすると、HTTP スクリプトおよびモジュールは、ネットワークまたはリモートポートへの接続を開始できなくなります。このアクセスを許可するには、このブール値を有効にします。

httpd_can_network_connect_db

このブール値を無効にすると、HTTP スクリプトおよびモジュールは、データベースサーバーへの接続を開始できなくなります。このアクセスを許可するには、このブール値を有効にします。

httpd_can_network_relay

httpd を正引きまたはリバースプロキシーとして使用する場合は、このブール値を有効にします。

httpd_can_sendmail

このブール値を無効にすると、HTTP モジュールがメールを送信できなくなります。これにより、httpd でスパム攻撃による脆弱性の発生を防ぐことができます。このブール値を有効にして、HTTP モジュールがメールを送信できるようにします。

httpd_dbus_avahi

無効にすると、このブール値はD-Bus を介した avahi サービスへの httpd アクセスを拒否します。このアクセスを許可するには、このブール値を有効にします。

httpd_enable_cgi

無効にすると、このブール値は httpd が CGI スクリプトを実行できないようにします。httpd が CGI スクリプトを実行できるようにするには、このブール値を有効にします (CGI スクリプトには、httpd_sys_script_exec_t タイプのラベルが必要です)。

httpd_enable_ftp_server

このブール値を有効にすると、httpd は FTP ポートをリッスンし、FTP サーバーとして機能します。

httpd_enable_homedirs

このブール値を無効にすると、httpd はユーザーホームディレクトリーにアクセスできなくなります。このブール値を有効にすると、ユーザーホームディレクトリー (/home/*/ のコンテンツなど) へのhttpdアクセスを許可します。

httpd_execmem

このブール値を有効にすると、httpd は、実行可能ファイルと書き込み可能なメモリーアドレスを必要とするプログラムを実行できます。バッファーオーバーフローに対する保護が低減するため、セキュリティー上の理由から、このブール値を有効にすることは推奨できません。ただし、特定のモジュールおよびアプリケーション (Java アプリケーションや Mono アプリケーションなど) では、この権限が必要になります。

httpd_ssi_exec

このブール値は、Web ページのサーバー側のインクルード (SSI) 要素を実行できるかどうかを定義します。

httpd_tty_comm

このブール値は、httpd が制御端末へのアクセスを許可するかどうかを定義します。通常はこのアクセスは必要ありませんが、SSL 証明書ファイルの設定などの場合は、パスワードプロンプトを表示して処理するために端末アクセスが必要になります。

httpd_unified

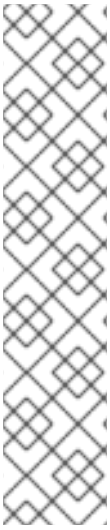
このブール値を有効にすると、httpd_tはすべてのhttpdタイプ (sys_content_t の実行、読み取り、または書き込み) に完全にアクセスできるようになります。無効にすると、読み取り専用、書き込み可能、または実行可能な Web コンテンツが切り離されます。このブール値を無効にすると、セキュリティーレベルが向上しますが、スクリプトやその他の Web コンテンツに、それぞれが持つ必要のあるファイルアクセス権に基づいて、個別にラベルを付ける必要があるという管理オーバーヘッドが追加されます。

httpd_use_cifs

このブール値を有効にすると、Samba を使用してマウントしたファイルシステムなど、cifs_tタイプでラベル付けされた CIFS ボリュームのファイルに、httpd がアクセスできるようになります。

httpd_use_nfs

このブール値を有効にすると、NFS を使用してマウントしたファイルシステムなど、nfs_t タイプでラベル付けされた NFS ボリュームのファイルに、httpd がアクセスできるようになります。



注記

SELinux ポリシーは継続的に開発されているため、上記のリストには、サービスに関連するすべてのブール値が常に含まれているとは限りません。これらを一覧表示するには、以下のコマンドを入力します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の説明を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

このコマンドが機能するには、**sepolicy** ユーティリティーを提供する追加の policycoreutils-devel パッケージが必要であることに注意してください。

13.4. 設定例

以下の例は、SELinux が Apache HTTP サーバーを補完する方法と、Apache HTTP サーバーの完全な機能を維持する方法の実例を紹介します。

13.4.1. 静的サイトの実行

静的 Web サイトを作成するには、その Web サイトの **.html** ファイルに **httpd_sys_content_t** タイプのラベルを付けます。デフォルトでは、Apache HTTP サーバーは **httpd_sys_content_t** タイプのラベルが付けられたファイルに書き込みできません。以下の例では、読み取り専用の Web サイトのファイルを保存する新しいディレクトリーを作成します。

1. root で **mkdir** ユーティリティーを使用して、最上位のディレクトリーを作成します。

```
~]# mkdir /mywebsite
```

2. root として **/mywebsite/index.html** ファイルを作成します。以下の内容を **/mywebsite/index.html** にコピーアンドペーストします。

```
<html>
<h2>index.html from /mywebsite/</h2>
</html>
```

3. Apache HTTP サーバーが、**/mywebsite/**、およびその下のファイルとサブディレクトリーに読み取り専用でアクセスできるようにするには、ディレクトリーに **httpd_sys_content_t** タイプのラベルを付けます。root で以下のコマンドを入力し、ラベルの変更を file-context 設定に追加します。

```
~]# semanage fcontext -a -t httpd_sys_content_t "/mywebsite(/.*)?"
```

4. **restorecon** ユーティリティーを root として使用し、ラベルを変更します。

```
~]# restorecon -R -v /mywebsite
restorecon reset /mywebsite context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
restorecon reset /mywebsite/index.html context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
```

5. この例では、root として /etc/httpd/conf/httpd.conf ファイルを編集します。既存の DocumentRoot オプションをコメントアウトします。DocumentRoot "/mywebsite" オプションを追加します。編集後、これらのオプションは以下のようになります。

```
#DocumentRoot "/var/www/html"
DocumentRoot "/mywebsite"
```

6. root で以下のコマンドを入力して、Apache HTTP Server のステータスを確認します。サーバーが停止している場合は、起動します。

```
~]# systemctl status httpd.service
httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled)
   Active: inactive (dead)
```

```
~]# systemctl start httpd.service
```

サーバーが実行している場合は、root で以下のコマンドを実行してサービスを再起動します (httpd.confに加えた変更も適用されます)。

```
~]# systemctl status httpd.service
httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled)
   Active: active (running) since Wed 2014-02-05 13:16:46 CET; 2s ago
```

```
~]# systemctl restart httpd.service
```

7. Web ブラウザーを使用して http://localhost/index.html に移動します。以下が表示されます。

```
index.html from /mywebsite/
```

13.4.2. NFS および CIFS ボリュームの共有

デフォルトで、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。一般的なポリシーでは、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、デフォルトでは、クライアント側でマウントされた 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のラベルが付けられます。一般的なポリシーでは、このデフォルトコンテキストはcifs_t タイプを使用します。

ポリシーの設定によっては、サービスが、nfs_t タイプまたは cifs_t タイプのラベルが付いたファイルを読み込めない場合があります。これにより、このタイプのラベルが付いたファイルシステムをマウントしてから、その他のサービスによる読み取りやエクスポートができなくなる可能性があります。ブール値を有効または無効にして、nfs_t タイプおよび cifs_t タイプへのアクセスを許可するサービスを制御できます。

httpd_use_nfs ブール値を有効にして、httpd が NFS ボリューム (nfs_t タイプのラベルが付いている) にアクセスし、共有できるようにします。

```
~]# setsebool -P httpd_use_nfs on
```

httpd_use_cifs ブール値を有効にして、httpd が CIFS ボリューム (cifs_t の種類のラベルが付いている) にアクセスし、共有できるようにします。

```
~]# setsebool -P httpd_use_cifs on
```

注記

再起動後も、setsebool の変更を保持したくない場合は、-P オプションを使用しないでください。

13.4.3. サービス間でのファイルの共有

Type Enforcement は、プロセスが別のプロセスで使用することを目的としたファイルにアクセスするのを防ぐのに役立ちます。たとえば、Samba は、デフォルトでは、httpd_sys_content_t のタイプでラベルが付いたファイル (Apache HTTP サーバーが使用するファイル) を読み込むことができません。ファイルに public_content_t または public_content_rw_t のタイプのラベルが付けられている場合は、Apache HTTP サーバー、FTP、rsync、および Samba との間でファイルを共有できます。

以下の例では、ディレクトリーとファイルを作成し、Apache HTTP Server、FTP、rsync、および Samba でディレクトリーとファイルを共有 (読み取り専用) できるようにします。

1. **root で mkdir ユーティリティーを使用して、複数のサービス間でファイルを共有するための新しいトップレベルディレクトリーを作成します。**

```
~]# mkdir /shares
```

2. **file-context 設定のパターンに一致しないファイルおよびディレクトリーには、default_t タイプのラベルが付けられます。このタイプは、制限のあるサービスからはアクセスできません。**

```
~]$ ls -dZ /shares
drwxr-xr-x  root root unconfined_u:object_r:default_t:s0 /shares
```

3. **root で、/shares/index.html ファイルを作成します。以下の内容を /shares/index.html にコピーアンドペーストします。**

```
<html>
<body>
<p>Hello</p>
</body>
</html>
```

4. **public_content_t タイプでの /shares/ のラベル付けでは、Apache HTTP サーバー、FTP、rsync、および Samba による読み取り専用アクセスが許可されます。root で以下のコマンドを入力し、ラベルの変更を file-context 設定に追加します。**

```
~]# semanage fcontext -a -t public_content_t "/shares(/.*)?"
```

5. **restorecon ユーティリティーを root で使用し、ラベルの変更を適用します。**

```
~]# restorecon -R -v /shares/
restorecon reset /shares context unconfined_u:object_r:default_t:s0-
>system_u:object_r:public_content_t:s0
restorecon reset /shares/index.html context unconfined_u:object_r:default_t:s0-
>system_u:object_r:public_content_t:s0
```

Samba 経由で/shares/を共有するには、以下を行います。

1. **samba パッケージ、samba-common パッケージ、および samba-client パッケージがインストールされていることを確認します (バージョン番号が異なる場合があります)。**

```
~]$ rpm -q samba samba-common samba-client
samba-3.4.0-0.41.el6.3.i686
samba-common-3.4.0-0.41.el6.3.i686
samba-client-3.4.0-0.41.el6.3.i686
```

これらのパッケージのいずれかがインストールされていない場合は、root で以下のコマンドを実行してインストールします。

```
~]# yum install package-name
```

2. **/etc/samba/smb.conf を root で編集します。このファイルの下部に以下のエントリーを追加し、Samba で/shares/ ディレクトリーを共有します。**

```
[shares]
comment = Documents for Apache HTTP Server, FTP, rsync, and Samba
path = /shares
public = yes
writable = no
```

3. **Samba ファイルシステムをマウントするには、Samba アカウントが必要です。root で次のコマンドを入力して、Samba アカウントを作成します。*username* は、既存の Linux ユーザーです。たとえば、smbpasswd -a testuser は、Linux testuser ユーザーの Samba アカウントを作成します。**

```
~]# smbpasswd -a testuser
New SMB password: Enter a password
Retype new SMB password: Enter the same password again
Added user testuser.
```

上記のコマンドを実行し、システムに存在しないアカウントのユーザー名を指定すると、Cannot locate Unix account for '*username*'! エラーが発生します。

4. **Samba サービスを再起動します。**

```
~]# systemctl start smb.service
```

5. **利用可能な共有を一覧表示するには、次のコマンドを入力します。*username* は、手順 3 で追加した Samba アカウントです。パスワードを求められたら、手順 3 で Samba アカウントに割り当てたパスワードを入力します (バージョン番号は異なる場合があります)。**

```
~]$ smbclient -U username -L localhost
Enter username's password:
Domain=[HOSTNAME] OS=[Unix] Server=[Samba 3.4.0-0.41.el6]

Sharename       Type      Comment
---------       ----      -------
shares          Disk      Documents for Apache HTTP Server, FTP, rsync, and Samba
IPC$            IPC       IPC Service (Samba Server Version 3.4.0-0.41.el6)
```

```
username        Disk      Home Directories
Domain=[HOSTNAME] OS=[Unix] Server=[Samba 3.4.0-0.41.el6]

Server               Comment
---------            -------

Workgroup            Master
---------            -------
```

6. **mkdir ユーティリティーを使用して、新しいディレクトリーを作成します。このディレクトリーは、shares の Samba 共有をマウントするために使用されます。**

```
~]# mkdir /test/
```

7. **root で次のコマンドを入力して、shares の Samba 共有を /test/ にマウントし、*username* を、手順 3 のユーザー名に置き換えます。**

```
~]# mount //localhost/shares /test/ -o user=username
```

手順 3 で設定した *username* のパスワードを入力します。

8. **Samba で共有されているファイルの内容を表示します。**

```
~]$ cat /test/index.html
<html>
<body>
<p>Hello</p>
</body>
</html>
```

Apache HTTP サーバー経由で /shares/ を共有するには、次のコマンドを実行します。

1. **httpd パッケージがインストールされていることを確認します (バージョン番号が異なる場合があります)。**

```
~]$ rpm -q httpd
httpd-2.2.11-6.i386
```

このパッケージがインストールされていない場合は、root で yum ユーティリティーを使用してインストールします。

```
~]# yum install httpd
```

2. **/var/www/html/ ディレクトリーに変更します。root で以下のコマンドを入力し、/shares/ ディレクトリーへのリンク (名前は shares) を作成します。**

```
html]# ln -s /shares/ shares
```

3. **Apache HTTP サーバーを起動します。**

```
~]# systemctl start httpd.service
```

4. Web ブラウザーを使用して **http://localhost/shares** に移動します。**/shares/index.html**が表示されます。

既定では、**index.html** ファイルが存在する場合は読み込まれます。**/shares/** に**index.html** がなく、代わりに **file1**、**file2**、および **file3** があった場合は、**http://localhost/shares** へのアクセス時にディレクトリーの一覧が表示されます。

1. **index.html** ファイルを削除します。

```
~]# rm -i /shares/index.html
```

2. **touch** ユーティリティーを root で使用し、**/shares/** に 3 つのファイルを作成します。

```
~]# touch /shares/file{1,2,3}
~]# ls -Z /shares/
-rw-r--r--  root root system_u:object_r:public_content_t:s0 file1
-rw-r--r--  root root unconfined_u:object_r:public_content_t:s0 file2
-rw-r--r--  root root unconfined_u:object_r:public_content_t:s0 file3
```

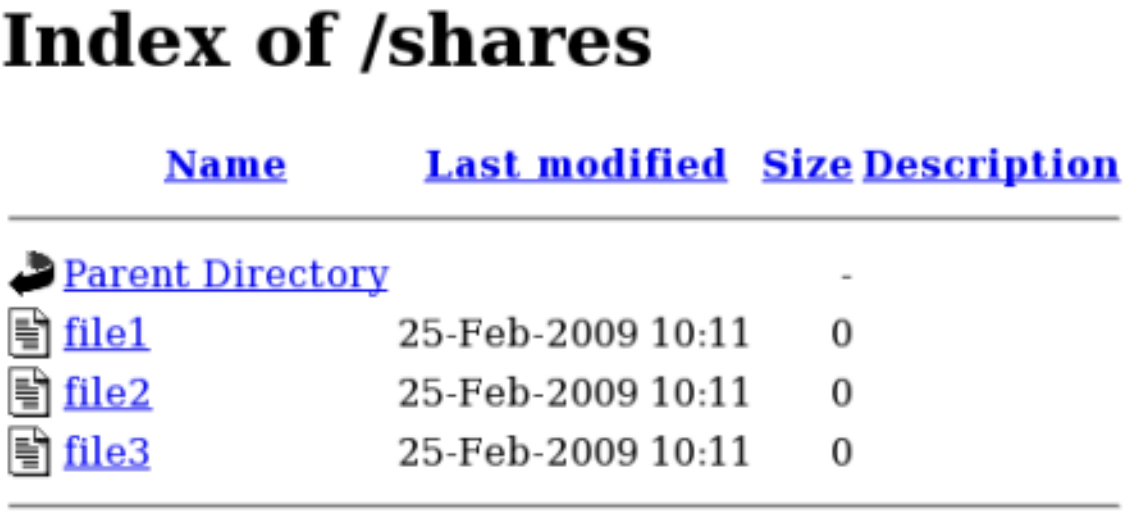
3. root で以下のコマンドを入力して、Apache HTTP サーバーのステータスを確認します。

```
~]# systemctl status httpd.service
httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled)
   Active: inactive (dead)
```

サーバーが停止している場合は、起動します。

```
~]# systemctl start httpd.service
```

4. Web ブラウザーを使用して **http://localhost/shares** に移動します。ディレクトリーの一覧が表示されます。



13.4.4. ポート番号の変更

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。**semanage** ユーティリティーを使用して、root ユーザーに、SELinux が**httpd** でリッスンできる

ポートを一覧表示します。

```
~]# semanage port -l | grep -w http_port_t
http_port_t                    tcp      80, 443, 488, 8008, 8009, 8443
```

標準設定では、**httpd** は TCP ポート 80、443、488、8008、8009、または 8443 でリッスンできます。**/etc/httpd/conf/httpd.conf** が、**http_port_t** に登録されていないポートで **httpd** がリッスンするように設定されていると、**httpd** が起動できません。

httpd を TCP ポート 80、443、488、8008、8009、または 8443 以外のポートで実行するように設定するには、次のコマンドを実行します。

1. **/etc/httpd/conf/httpd.conf** ファイルを root で編集し、**Listen** オプションに **httpd** の SELinux ポリシーで設定されていないポートの一覧が表示されるようにします。以下の例では、**httpd** が 10.0.0.1 IP アドレスおよび TCP ポート 12345 でリッスンするように設定します。

   ```
   # Change this to Listen on specific IP addresses as shown below to
   # prevent Apache from glomming onto all bound IP addresses (0.0.0.0)
   #
   #Listen 12.34.56.78:80
   Listen 10.0.0.1:12345
   ```

2. root ユーザーになり、次のコマンドを実行して、SELinux ポリシー設定にポートを追加します。

   ```
   ~]# semanage port -a -t http_port_t -p tcp 12345
   ```

3. ポートが追加されたことを確認します。

   ```
   ~]# semanage port -l | grep -w http_port_t
   http_port_t                    tcp      12345, 80, 443, 488, 8008, 8009, 8443
   ```

ポート 12345 で **httpd** を実行しなくなった場合は、root で **semanage** ユーティリティーを使用し、ポリシー設定からポートを削除します。

```
~]# semanage port -d -t http_port_t -p tcp 12345
```

---

[14] 詳細は、System Administrator's Guide の The Apache HTTP Sever セクションを参照してください。

第14章 SAMBA

Samba は、SMB (Server Message Block) プロトコルおよびCIFS (Common Internet File System) プロトコルのオープンソース実装で、さまざまなオペレーティングシステムでクライアント間でファイルサービスおよび印刷サービスを提供します。[15]

Red Hat Enterprise Linux では、samba パッケージにより Samba サーバーが提供されます。以下のコマンドを実行して、samba パッケージがインストールされているかどうかを確認します。

```
~]$ rpm -q samba
package samba is not installed
```

インストールされておらず、Samba を使用する場合は、**yum** ユーティリティーを root ユーザーでインストールします。

```
~]# yum install samba
```

14.1. SAMBA と SELINUX

SELinux が有効になると、Samba サーバー (**smbd**) はデフォルトで制限付きで実行します。制限のあるサービスは独自のドメインで実行し、その他の制限のあるサービスとは区別されます。以下の例は、独自の領域で実行している**smbd** プロセスを示しています。この例では、samba がインストールされていることを前提としています。

1. **getenforce** コマンドを実行して、SELinux が Enforcing モードで実行されていることを確認します。

   ```
   ~]$ getenforce
   Enforcing
   ```

   SELinux が Enforcing モードで実行されていると、このコマンドは **Enforcing** を返します。

2. root で次のコマンドを入力して、**smbd** を起動します。

   ```
   ~]# systemctl start smb.service
   ```

   サービスが実行中であることを確認します。出力には以下の情報が含まれている必要があります (タイムスタンプのみは異なります)。

   ```
   ~]# systemctl status smb.service
   smb.service - Samba SMB Daemon
     Loaded: loaded (/usr/lib/systemd/system/smb.service; disabled)
     Active: active (running) since Mon 2013-08-05 12:17:26 CEST; 2h 22min ago
   ```

3. **smbd** プロセスを表示するには、以下のコマンドを実行します。

   ```
   ~]$ ps -eZ | grep smb
   system_u:system_r:smbd_t:s0      9653 ?        00:00:00 smbd
   system_u:system_r:smbd_t:s0      9654?       00:00:00 smbd
   ```

   smbd プロセスに関連付けられた SELinux コンテキストは**system_u:system_r:smbd_t:s0** です。コンテキストの 2 番目の部分 **smbd_t** はタイプです。タイプは、プロセスのドメインと

ファイルのタイプを定義します。この例では、**smbd** プロセスが **smbd_t** ドメインで実行しています。

smbd がファイルにアクセスし、共有できるように、ファイルに正しくラベルを付ける必要があります。たとえば、**smbd** は、**samba_share_t** タイプのラベルが付いたファイルの読み取りと書き込みが可能ですが、デフォルトでは、**httpd_sys_content_t** タイプのラベルが付いたファイルにはアクセスできません。これは、Apache HTTP サーバーが使用することを目的としています。ブール値を有効にして、ホームディレクトリーや NFS ボリュームを Samba 経由でエクスポートできるようにしたり、Samba がドメインコントローラーとして機能できるようにする必要があります。

14.2. タイプ

高度なプロセス分離を提供するために、SELinux Targeted ポリシーで使用される主要なパーミッション制御方法は Type Enforcement です。すべてのファイルとプロセスにはタイプのラベルが付いています。タイプはプロセスの SELinux ドメインを定義し、ファイルの SELinux タイプを定義します。SELinux ポリシールールは、タイプにアクセスするドメインであるか、別のドメインにアクセスするドメインであるかにかかわらず、タイプが相互にアクセスする方法を定義します。アクセスは、それを許可する特定の SELinux ポリシールールが存在する場合にのみ許可されます。

samba_share_tのタイプでファイルにラベルを付け、Samba がファイルを共有できるようにします。作成したファイルのみにラベルを付け、**samba_share_t** のタイプでシステムファイルに再ラベル付けしないでください。ブール値を有効にすると、このようなファイルおよびディレクトリーを共有できます。**/etc/samba/smb.conf** ファイルおよび Linux のパーミッションが設定されていれば、**samba_share_t** タイプでラベル付けされたファイルに書き込むことができます。

samba_etc_tタイプは、**/etc/samba/**ディレクトリーの特定のファイル (**smb.conf** など) で使用されます。**samba_etc_t**の種類でファイルに手動でラベルを付けないでください。このディレクトリーのファイルにラベルが正しく設定されていない場合は、root ユーザーになり、**restorecon -R -v /etc/samba** を実行してファイルをデフォルトコンテキストに戻します。**/etc/samba/smb.conf** に**samba_etc_t** タイプを指定しないと、Samba の起動に失敗し、SELinux の拒否メッセージが記録される場合があります。以下は、**/etc/samba/smb.conf** に **httpd_sys_content_t** タイプのラベルが付いた拒否メッセージの例になります。

> setroubleshoot: SELinux is preventing smbd (smbd_t) "read" to ./smb.conf (httpd_sys_content_t). For complete SELinux messages. run sealert -l deb33473-1069-482b-bb50-e4cd05ab18af

14.3. ブール値

SELinux は、サービスの実行に必要な最小アクセスレベルに基づいています。サービスはさまざまな方法で実行できます。そのため、サービスの実行方法を指定する必要があります。以下のブール値を使用して SELinux を設定します。

smbd_anon_write

このブール値を有効にすると、**smbd** は、特殊なアクセス制限がない一般的なファイル用に予約されている領域など、公開ディレクトリーに書き込むことができます。

samba_create_home_dirs

ブール値を有効にすると、Samba は新しいホームディレクトリーを個別に作成できます。これは、PAM などのメカニズムにより行われることが多いです。

samba_domain_controller

これを有効にすると、ブール値により、Samba がドメインコントローラーとして機能し、useradd、groupadd、passwd などの関連コマンドを実行するパーミッションを得ることができます。

samba_enable_home_dirs

このブール値を有効にすると、Samba はユーザーのホームディレクトリーを共有できます。

samba_export_all_ro

ファイルまたはディレクトリーをエクスポートし、読み取り専用の権限を付与します。これにより、samba_share_t タイプのラベルが付いていないファイルおよびディレクトリーを、Samba で共有できます。samba_export_all_ro ブール値が有効であっても、samba_export_all_rw のブール値が無効になっていると、/etc/samba/smb.conf で書き込みアクセスが設定されていても、Samba 共有への書き込みアクセスは拒否されます。また、書き込みアクセスを許可している Linux パーミッションも拒否されます。

samba_export_all_rw

ファイルまたはディレクトリーをエクスポートして、読み取りおよび書き込みのパーミッションを許可します。これにより、samba_share_t タイプのラベルが付いていないファイルおよびディレクトリーを、Samba からエクスポートできます。書き込みアクセスを許可するには、/etc/samba/smb.conf および Linux のパーミッションを設定する必要があります。

samba_run_unconfined

ブール値を有効にすると、/var/lib/samba/scripts/ ディレクトリーで、Samba が定義されていないスクリプトを実行できるようになります。

samba_share_fusefs

Samba が fusefs ファイルシステムを共有するには、このブール値を有効にする必要があります。

samba_share_nfs

このブール値を無効にすると、smbd は Samba を介して NFS 共有に完全にアクセスできなくなります。このブール値を有効にすると、Samba は NFS ボリュームを共有できるようになります。

use_samba_home_dirs

Samba ホームディレクトリーにリモートサーバーを使用する場合は、このブール値を有効にします。

virt_use_samba

仮想マシンによる CIFS ファイルへのアクセスを許可します。

注記

SELinux ポリシーは継続的に開発されているため、上記のリストには、サービスに関連するすべてのブール値が常に含まれているとは限りません。これらを一覧表示するには、以下のコマンドを入力します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の説明を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

このコマンドが機能するには、**sepolicy** ユーティリティーを提供する追加の policycoreutils-devel パッケージが必要であることに注意してください。

14.4. 設定例

以下の例は、SELinux が Samba サーバーを補完する方法と、Samba サーバーの完全な機能を維持する方法の実例を紹介します。

14.4.1. 作成したディレクトリーの共有

以下の例では、新しいディレクトリーを作成し、Samba でそのディレクトリーを共有します。

1. samba パッケージ、samba-common パッケージ、および samba-client パッケージがインストールされていることを確認します。

```
~]$ rpm -q samba samba-common samba-client
package samba is not installed
package samba-common is not installed
package samba-client is not installed
```

これらのパッケージがいずれもインストールされていない場合は、root で **yum** ユーティリティーを使用してパッケージをインストールします。

```
~]# yum install package-name
```

2. **mkdir** ユーティリティーを root で使用し、Samba でファイルを共有するための新しいトップレベルディレクトリーを作成します。

```
~]# mkdir /myshare
```

3. **touch** ユーティリティーの root を使用して、空のファイルを作成します。このファイルは、後で Samba 共有が正しくマウントされたことを確認するために使用されます。

```
~]# touch /myshare/file1
```

4. **/etc/samba/smb.conf** ファイルおよび Linux のパーミッションが設定されていれば、**samba_share_t** タイプでラベル付けされたファイルの読み取りと書き込みが可能になります。root で以下のコマンドを入力し、ラベルの変更を file-context 設定に追加します。

```
~]# semanage fcontext -a -t samba_share_t "/myshare(/.*)?"
```

5. **restorecon ユーティリティーを root で使用し、ラベルの変更を適用します。**

```
~]# restorecon -R -v /myshare
restorecon reset /myshare context unconfined_u:object_r:default_t:s0-
>system_u:object_r:samba_share_t:s0
restorecon reset /myshare/file1 context unconfined_u:object_r:default_t:s0-
>system_u:object_r:samba_share_t:s0
```

6. **root で/etc/samba/smb.conf を編集します。このファイルの下部に以下を追加し、Samba で/myshare/ ディレクトリーを共有します。**

```
[myshare]
comment = My share
path = /myshare
public = yes
writable = no
```

7. **Samba ファイルシステムをマウントするには、Samba アカウントが必要です。root で次のコマンドを入力して、Samba アカウントを作成します。*username* は、既存の Linux ユーザーです。たとえば、smbpasswd -a testuser は、Linux testuser ユーザーの Samba アカウントを作成します。**

```
~]# smbpasswd -a testuser
New SMB password: Enter a password
Retype new SMB password: Enter the same password again
Added user testuser.
```

上記のコマンドを入力し、システムに存在しないアカウントのユーザー名を指定すると、Cannot locate Unix account for '*username*'! エラーが発生します。

8. **Samba サービスを再起動します。**

```
~]# systemctl start smb.service
```

9. **利用可能な共有を一覧表示するには、次のコマンドを入力します。*username* は、手順 7 で追加した Samba アカウントです。パスワードを求められたら、手順 7 で Samba アカウントに割り当てたパスワードを入力します (バージョン番号は異なる場合があります)。**

```
~]$ smbclient -U username -L localhost
Enter username's password:
Domain=[HOSTNAME] OS=[Unix] Server=[Samba 3.4.0-0.41.el6]

Sharename       Type      Comment
---------       ----      -------
myshare         Disk      My share
IPC$            IPC       IPC Service (Samba Server Version 3.4.0-0.41.el6)
username        Disk      Home Directories
Domain=[HOSTNAME] OS=[Unix] Server=[Samba 3.4.0-0.41.el6]

Server               Comment
---------            -------

Workgroup            Master
---------            -------
```

10. **mkdir ユーティリティーを root で実行して、新しいディレクトリーを作成します。このディレクトリーは、myshareの Samba 共有をマウントするために使用されます。**

```
~]# mkdir /test/
```

11. **root で次のコマンドを入力して、myshareの Samba 共有を/test/ にマウントし、*username* を、手順 7 のユーザー名に置き換えます。**

```
~]# mount //localhost/myshare /test/ -o user=username
```

手順 7 で設定した *username* のパスワードを入力します。

12. **手順 3 で作成したfile1 を表示する場合は、以下を実行します。**

```
~]$ ls /test/
file1
```

14.4.2. Web サイトの共有

/var/www/html/ ディレクトリー内の Web サイトを共有する場合など、samba_share_t タイプでファイルにラベルを付けることはできません。このような場合は、samba_export_all_ro ブール値を使用して (現在のラベルに関係なく) 任意のファイルまたはディレクトリーを共有して読み取り専用権限を許可するかsamba_export_all_rw ブール値を使用して (現在のラベルに関係なく) 任意のファイルまたはディレクトリーを共有して読み取りおよび書き込み権限を許可します。

以下の例では、/var/www/html/ で Web サイトのファイルを作成し、Samba でそのファイルを共有することで、読み取り権限と書き込み権限を付与します。この例では、httpd パッケージ、samba パッケージ、samba-common パッケージ、samba-client パッケージ、および wget パッケージがインストールされていることを前提としています。

1. **root で、/var/www/html/file1.html ファイルを作成します。以下の内容をこのファイルにコピーして貼り付けます。**

```
<html>
<h2>File being shared through the Apache HTTP Server and Samba.</h2>
</html>
```

2. **次のコマンドを実行すると、file1.html の SELinux コンテキストが表示されます。**

```
~]$ ls -Z /var/www/html/file1.html
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file1.html
```

ファイルには、httpd_sys_content_t のラベルが付けられています。デフォルトでは、Apache HTTP サーバーはこのタイプにアクセスできますが、Samba はアクセスできません。

3. **Apache HTTP サーバーを起動します。**

```
~]# systemctl start httpd.service
```

4. **ユーザーが書き込みアクセス権を持つディレクトリーに移動し、次のコマンドを実行します。デフォルト設定が変更されない限り、このコマンドは成功します。**

```
~]$ wget http://localhost/file1.html
Resolving localhost... 127.0.0.1
Connecting to localhost|127.0.0.1|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 84 [text/html]
Saving to: `file1.html.1'

100%[======================>] 84          --.-K/s   in 0s

`file1.html.1' saved [84/84]
```

5. **root で/etc/samba/smb.conf を編集します。このファイルの下部に以下を追加し、Samba で/var/www/html/ ディレクトリーを共有します。**

```
[website]
comment = Sharing a website
path = /var/www/html/
public = no
writable = no
```

6. **/var/www/html/ ディレクトリーには、httpd_sys_content_t タイプのラベルが付けられています。デフォルトでは、Samba は Linux の権限があっても、このタイプのラベルが付いたファイルおよびディレクトリーにはアクセスできません。Samba のアクセスを許可するには、samba_export_all_ro ブール値を有効にします。**

```
~]# setsebool -P samba_export_all_ro on
```

システムを再起動しても変更を持続させない場合は、-P オプションを使用しないでください。samba_export_all_ro ブール値を有効にすると、Samba は任意のタイプにアクセスできるようになることに注意してください。

7. **Samba サービスを再起動します。**

```
~]# systemctl start smb.service
```

---

[15] **詳細は、System Administrator's Guide の Samba のセクションを参照してください。**

第15章 ファイル転送プロトコル

ファイル転送プロトコル (FTP) は、今日インターネット上で見られる、最も古く、一般的に使用されているプロトコルです。この目的は、ユーザーがリモートホストに直接ログインしなくても、もしくはリモートシステムの使用法についての知識がなくとも、ネットワーク上のコンピューターホスト間で確実にファイルを転送することです。これにより、ユーザーは、標準の簡単なコマンドセットを使用してリモートシステム上のファイルにアクセスすることができます。

Very Secure FTP Daemon (**vsftpd**) は、高速で安定しており、最も重要なこととして安全であるようにゼロから設計されています。多数の接続を効率的かつ安全に処理できることが、Red Hat Enterprise Linux に同梱されるスタンドアロンの FTP が **vsftpd** のみである理由です。

Red Hat Enterprise Linux では、vsftpd パッケージに Very Secure FTP デーモンが同梱されています。次のコマンドを入力して、vsftpd がインストールされているかどうかを確認します。

```
~]$ rpm -q vsftpd
package vsftpd is not installed
```

FTP サーバーを使用する必要があるけれど、vsftpd パッケージがインストールされていない場合は、**yum** ユーティリティーを root ユーザーでインストールします。

```
~]# yum install vsftpd
```

15.1. タイプ

高度なプロセス分離を提供するために、SELinux Targeted ポリシーで使用される主要なパーミッション制御方法は Type Enforcement です。すべてのファイルとプロセスにはタイプのラベルが付いています。タイプはプロセスの SELinux ドメインを定義し、ファイルの SELinux タイプを定義します。SELinux ポリシールールは、タイプにアクセスするドメインであるか、別のドメインにアクセスするドメインであるかにかかわらず、タイプが相互にアクセスする方法を定義します。アクセスは、それを許可する特定の SELinux ポリシールールが存在する場合にのみ許可されます。

デフォルトでは、匿名ユーザーは FTP を使用してログインすると、**/var/ftp/** ディレクトリーのファイルへの読み取りアクセス権を持ちます。このディレクトリーには、**public_content_t** タイプのラベルが付けられており、書き込みアクセスが **/etc/vsftpd/vsftpd.conf** で設定されている場合でも、読み取りアクセスのみを許可します。**public_content_t** のタイプは、Apache HTTP サーバー、Samba、NFS などの他のサービスからアクセスできます。

次のいずれかのタイプを使用して、FTP でファイルを共有します。

public_content_t

public_content_t タイプで作成したラベルファイルおよびディレクトリーを、**vsftpd** で読み取り専用で共有します。Apache HTTP サーバー、Samba、NFS などのその他のサービスは、このタイプのラベルが付いたファイルにアクセスできます。**public_content_t** タイプが指定されているファイルは、書き込みが許可されていても書き込みできません。ライトアクセスが必要な場合は、**public_content_rw_t** タイプを使用します。

public_content_rw_t

public_content_rw_t タイプで作成したファイルおよびディレクトリーにラベルを付け、**vsftpd** で読み取りおよび書き込みの権限で共有します。Apache HTTP サーバー、Samba、NFS などのその他のサービスは、このタイプのラベルが付いたファイルにアクセスできます。各サービスのブール値は、このタイプのラベルが付いたファイルに書き込む前に有効にしておく必要があります。

15.2. ブール値

SELinux は、サービスの実行に必要な最小アクセスレベルに基づいています。サービスはさまざまな方法で実行できます。そのため、サービスの実行方法を指定する必要があります。以下のブール値を使用して SELinux を設定します。

ftpd_anon_write

このブール値を無効にすると、vsftpd は、public_content_rw_t タイプでラベル付けされたファイルおよびディレクトリーに書き込みできなくなります。このブール値を有効にすると、ユーザーは FTP を使用してファイルをアップロードできます。ファイルをアップロードするディレクトリーには、public_content_rw_t タイプにラベルを付け、それに応じて Linux の権限を設定する必要があります。

ftpd_full_access

このブール値が有効になると、アクセスの制御に Linux (DAC) の権限のみが使用され、認証されたユーザーは、public_content_t または public_content_rw_t タイプのラベルが付いていないファイルへの読み取りと書き込みが可能になります。

ftpd_use_cifs

ブール値を有効にすると、vsftpd は、cifs_t タイプでラベル付けされたファイルおよびディレクトリーにアクセスできるようになるため、このブール値を有効にすると、vsftpd を介して、Samba を使用してマウントしたファイルシステムを共有できます。

ftpd_use_nfs

ブール値を有効にすると、vsftpd はnfs_t タイプでラベル付けされたファイルとディレクトリーにアクセスできるようになるため、このブール値を使用すると、vsftpd 経由で NFS を使用してマウントしたファイルシステムを共有できます。

ftpd_connect_db

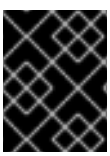
FTP デーモンによるデータベースへの接続の開始を許可します。

httpd_enable_ftp_server

httpd デーモンが FTP ポートをリッスンし、FTP サーバーとして機能できるようにします。

tftp_anon_write

ブール値を有効にすると、特別なアクセス制限がない共通ファイル用に予約されている領域など、公開ディレクトリーへの TFTP アクセスが許可されます。



重要

Red Hat Enterprise Linux 7.7 では、ftp_home_dir ブール値が提供されません。詳細は、Red Hat Enterprise Linux 7.3 Release Notesドキュメントを参照してください。

注記

SELinux ポリシーは継続的に開発されているため、上記のリストには、サービスに関連するすべてのブール値が常に含まれているとは限りません。これらを一覧表示するには、以下のコマンドを入力します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の説明を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

このコマンドが機能するには、**sepolicy** ユーティリティーを提供する追加の policycoreutils-devel パッケージが必要であることに注意してください。

第16章 ネットワークファイルシステム

ネットワークファイルシステム (NFS) を利用すると、リモートのホストがネットワーク経由でファイルシステムをマウントし、そのファイルシステムを、ローカルにマウントしているファイルシステムと同じように操作できるようになります。また、システム管理者は、リソースをネットワーク上の中央サーバーに統合することができるようになります。[16]

Red Hat Enterprise Linux では、NFS に完全に対応するには nfs-utils が必要です。次のコマンドを入力して、nfs-utils がインストールされているかどうかを確認します。

```
~]$ rpm -q nfs-utils
package nfs-utils is not installed
```

パッケージがインストールされておらず、NFS を使用する場合は、**yum** ユーティリティーを使って root でインストールします。

```
~]# yum install nfs-utils
```

16.1. NFS と SELINUX

SELinux の実行時に、NFS デーモンは、**nfsd** プロセスを除き、デフォルトで制限されています。nfsd プロセスは、制限のない **kernel_t** ドメインタイプでラベル付けされています。SELinux ポリシーでは、NFS がデフォルトでファイルを共有できるようになっています。また、クライアントとサーバー間で SELinux ラベルを渡すことに対応しています。これにより、NFS ボリュームにアクセスする制限されたドメインに対するセキュリティー制御が強化されます。たとえば、ホームディレクトリーを NFS ボリュームに設定している場合は、ホームディレクトリーにのみアクセスでき、ボリューム上のその他のディレクトリーにはアクセスできないように、限定ドメインを指定できます。同様に、セキュア仮想化などのアプリケーションは、NFS ボリュームでイメージファイルのラベルを設定できるため、仮想マシンの分離レベルが向上します。

ラベル付き NFS のサポートは、デフォルトでは無効になっています。有効にする方法は、「SELinux ラベル付き NFS のサポートの有効化」 を参照してください。

16.2. タイプ

高度なプロセス分離を提供するために、SELinux Targeted ポリシーで使用される主要なパーミッション制御方法は Type Enforcement です。すべてのファイルとプロセスにはタイプのラベルが付いています。タイプはプロセスの SELinux ドメインを定義し、ファイルの SELinux タイプを定義します。SELinux ポリシールールは、タイプにアクセスするドメインであるか、別のドメインにアクセスするドメインであるかにかかわらず、タイプが相互にアクセスする方法を定義します。アクセスは、それを許可する特定の SELinux ポリシールールが存在する場合にのみ許可されます。

デフォルトでは、クライアント側でマウントされた NFS ボリュームには、NFS のポリシーで定義されたデフォルトコンテキストを持つラベルが付けられます。一般的なポリシーでは、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。root ユーザーは、**mount** オプションを使用して、デフォルトのタイプを上書きできます。NFS では、以下のタイプが使用されます。異なるタイプを使用すると、柔軟なアクセスを設定できます。

var_lib_nfs_t

このタイプは、**/var/lib/nfs/** ディレクトリーにコピーまたは作成される既存ファイルおよび新規ファイルに使用されます。このタイプは、通常の操作で変更する必要はありません。変更をデフォルト設定に戻すには、root で**restorecon -R -v /var/lib/nfs** を実行します。

nfsd_exec_t

/usr/sbin/rpc.nfsd ファイルには、nfsd_exec_t のラベルが付けられています。また、他のシステム実行ファイルおよびライブラリーも NFS に関連しています。この種類のファイルにはラベルを付けないでください。nfsd_exec_t は、nfsd_t に移行します。

16.3. ブール値

SELinux は、サービスの実行に必要な最小アクセスレベルに基づいています。サービスはさまざまな方法で実行できます。そのため、サービスの実行方法を指定する必要があります。以下のブール値を使用して SELinux を設定します。

ftpd_use_nfs

このブール値を有効にすると、ftpd デーモンは NFS ボリュームにアクセスできるようになります。

cobbler_use_nfs

このブール値を有効にすると、cobblerd デーモンは NFS ボリュームにアクセスできるようになります。

git_system_use_nfs

このブール値を有効にすると、Git システムデーモンは、NFS ボリュームのシステム共有リポジトリーを読み取ることができます。

httpd_use_nfs

このブール値を有効にすると、httpd デーモンは、NFS ボリュームに保存されているファイルにアクセスできるようになります。

samba_share_nfs

このブール値を有効にすると、smbd デーモンが NFS ボリュームを共有できるようになります。このブール値を無効にすると、smbd は、Samba を使用して NFS 共有に完全にアクセスできなくなります。

sanlock_use_nfs

これを有効にすると、ブール値により、sanlock デーモンが NFS ボリュームを管理できるようになります。

sge_use_nfs

このブール値を有効にすると、sge スケジューラーは NFS ボリュームにアクセスできるようになります。

use_nfs_home_dirs

このブール値を有効にすると、NFS ホームディレクトリーのサポートが追加されます。

virt_use_nfs

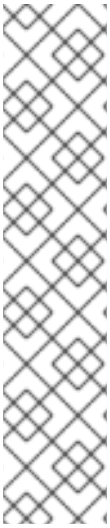
このブール値を有効にすると、信頼できる仮想ゲストが NFS ボリュームのファイルを管理できるようになります。

xen_use_nfs

このブール値を有効にすると、Xen は NFS ボリュームのファイルを管理できるようになります。

git_cgi_use_nfs

このブール値を有効にすると、CGI (Git Common Gateway Interface) は NFS ボリュームにアクセスできるようになります。



注記

SELinux ポリシーは継続的に開発されているため、上記のリストには、サービスに関連するすべてのブール値が常に含まれているとは限りません。これらを一覧表示するには、以下のコマンドを入力します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の説明を表示するには、以下のコマンドを実行します。

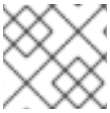
```
~]$ sepolicy booleans -b boolean_name
```

このコマンドが機能するには、**sepolicy** ユーティリティーを提供する追加の policycoreutils-devel パッケージが必要であることに注意してください。

16.4. 設定の例

16.4.1. SELinux ラベル付き NFS のサポートの有効化

以下の例は、SELinux でラベル付けされた NFS サポートを有効にする方法を示しています。この例では、nfs-utils パッケージがインストールされ、SELinux Targeted ポリシーが使用され、SELinux が Enforcing モードで動作していることを前提とします。



注記

手順 1 から 3 は、NFS サーバーの **nfs-srv** で実行する必要があります。

1. NFS サーバーが実行中の場合は、停止します。

```
[nfs-srv]# systemctl stop nfs
```

サーバーが停止していることを確認します。

```
[nfs-srv]# systemctl status nfs
nfs-server.service - NFS Server
   Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; disabled)
   Active: inactive (dead)
```

2. **/etc/sysconfig/nfs** ファイルを編集して、**RPCNFSDARGS** フラグを **"-V 4.2"** に設定します。

```
# Optional arguments passed to rpc.nfsd. See rpc.nfsd(8)
RPCNFSDARGS="-V 4.2"
```

3. サーバーを再起動して、サーバーが実行していることを確認します。この出力には以下の情報が含まれます。タイムスタンプのみが異なります。

```
[nfs-srv]# systemctl start nfs
```

```
[nfs-srv]# systemctl status nfs
nfs-server.service - NFS Server
   Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; disabled)
   Active: active (exited) since Wed 2013-08-28 14:07:11 CEST; 4s ago
```

4. **クライアントで、NFS サーバーをマウントします。**

```
[nfs-client]# mount -o v4.2 server:mntpoint localmountpoint
```

5. **すべての SELinux ラベルが、サーバーからクライアントに正常に渡されるようになりました。**

```
[nfs-srv]$ ls -Z file
-rw-rw-r--. user user unconfined_u:object_r:svirt_image_t:s0 file
[nfs-client]$ ls -Z file
-rw-rw-r--. user user unconfined_u:object_r:svirt_image_t:s0 file
```

注記

ラベル付けされている NFS サポート (ホームディレクトリーまたはその他のコンテンツ用) を有効にすると、そのコンテンツには、EXT ファイルシステムと同じラベルが付けられます。また、異なるバージョンの NFS を使用してシステムをマウントしたり、ラベル付きの NFS に対応していないサーバーをマウントしようとすると、エラーが返される可能性があります。

[16] **詳細は、Storage Administration Guide の ネットワークファイルシステム (NFS) を参照してください。**

第17章 BIND (BERKELEY INTERNET NAME DOMAIN)

BIND は、named デーモンを使用して名前解決サービスを実行します。BIND を使用すると、数値アドレスではなく、名前を使用してコンピューターのリソースやサービスを特定できます。

Red Hat Enterprise Linux では、bind により DNS サーバーが提供されます。以下のコマンドを実行して、bind パッケージがインストールされているかどうかを確認します。

```
~]$ rpm -q bind
package bind is not installed
```

インストールされていない場合は、root で yum ユーティリティーを使用し、インストールします。

```
~]# yum install bind
```

17.1. BIND および SELINUX

/var/named/slaves/ ディレクトリー、/var/named/dynamic/ ディレクトリー、および /var/named/data/ ディレクトリーに対するデフォルトのパーミッションにより、ゾーン転送と動的 DNS 更新を使用したゾーンファイルの更新が可能になります。/var/named/ のファイルには、マスターゾーンファイルに使用される named_zone_t タイプがラベル付けされています。

スレーブサーバーの場合は、スレーブゾーンを /var/named/slaves/ に配置するように /etc/named.conf ファイルを設定します。以下は、/var/named/slaves/ に testdomain.com のゾーンファイルを保存するスレーブ DNS サーバーの /etc/named.conf のドメインエントリーの例です。

```
zone "testdomain.com" {
   type slave;
   masters { IP-address; };
   file "/var/named/slaves/db.testdomain.com";
        };
```

ゾーンファイルに named_zone_t のラベルが付いている場合は、ゾーン転送を許可し、動的 DNS によるゾーンファイルの更新を許可するために、named_write_master_zones ブール値を有効にする必要があります。また、named ユーザーまたはグループに読み取り、書き込み、実行のアクセスを許可するには、親ディレクトリーのモードを変更する必要があります。

/var/named/ のゾーンファイルにnamed_cache_t タイプのラベルが付いている場合は、ファイルシステムの再ラベル付け、または実行中の restorecon -R /var/ により、ファイルシステムのタイプが named_zone_t に変更されます。

17.2. タイプ

高度なプロセス分離を提供するために、SELinux Targeted ポリシーで使用される主要なパーミッション制御方法は Type Enforcement です。すべてのファイルとプロセスにはタイプのラベルが付いています。タイプはプロセスの SELinux ドメインを定義し、ファイルの SELinux タイプを定義します。SELinux ポリシールールは、タイプにアクセスするドメインであるか、別のドメインにアクセスするドメインであるかにかかわらず、タイプが相互にアクセスする方法を定義します。アクセスは、それを許可する特定の SELinux ポリシールールが存在する場合にのみ許可されます。

BIND では、以下のタイプが使用されます。異なるタイプを使用すると、柔軟なアクセスを設定できます。

named_zone_t

マスターゾーンファイルに使用されます。その他のサービスでは、このタイプのファイルを変更できません。named デーモンは、named_write_master_zones ブール値が有効になっている場合に限り、このタイプのファイルを変更できます。

named_cache_t

デフォルトでは、named は、このタイプのラベルが付いたファイルに書き込むことができます。ブール値は追加されません。/var/named/slaves/ ディレクトリー、/var/named/dynamic/ ディレクトリー、および /var/named/data/ ディレクトリーでコピーまたは作成されたファイルには、named_cache_t のタイプに応じて自動的にラベルが付けられます。

named_var_run_t

/var/run/bind/ ディレクトリー、/var/run/named/ ディレクトリー、および /var/run/unbound/ ディレクトリーでコピーまたは作成されたファイルには、named_var_run_t のタイプに応じて自動的にラベルが付けられます。

named_conf_t

BIND 関連の設定ファイルは、通常 /etc ディレクトリーに保存され、自動的に named_conf_t のタイプのラベルが付けられます。

named_exec_t

BIND 関連の実行ファイルは、通常 /usr/sbin/ ディレクトリーに保存され、自動的に named_exec_t タイプのラベルが付けられます。

named_log_t

BIND 関連のログファイルは、通常 /var/log/ ディレクトリーに保存され、自動的に named_log_t のタイプのラベルが付けられます。

named_unit_file_t

/usr/lib/systemd/system/ ディレクトリー内の実行可能な BIND 関連ファイルには、named_unit_file_t のタイプに応じて自動的にラベルが付けられます。

17.3. ブール値

SELinux は、サービスの実行に必要な最小アクセスレベルに基づいています。サービスはさまざまな方法で実行できます。そのため、サービスの実行方法を指定する必要があります。以下のブール値を使用して SELinux を設定します。

named_write_master_zones

このブール値を無効にすると、named は、named_zone_t タイプでラベル付けされたゾーンファイルまたはディレクトリーに書き込みできなくなります。デーモンは通常、ゾーンファイルに書き込む必要はありません。ただし、必要な場合、またはセカンダリーサーバーがゾーンファイルに書き込む必要がある場合は、このブール値を有効にしてこのアクションを許可します

named_tcp_bind_http_port

このブール値を有効にすると、BIND による Apache ポートのバインドが可能になります。

注記

SELinux ポリシーは継続的に開発されているため、上記のリストには、サービスに関連するすべてのブール値が常に含まれているとは限りません。これらを一覧表示するには、以下のコマンドを入力します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の説明を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

このコマンドが機能するには、**sepolicy** ユーティリティーを提供する追加の policycoreutils-devel パッケージが必要であることに注意してください。

17.4. 設定の例

17.4.1. 動的 DNS

BIND を使用すると、ホストが DNS ファイルおよびゾーンファイルのレコードを動的に更新できます。ホストコンピューターの IP アドレスが頻繁に変更され、DNS レコードにリアルタイム修正が必要な場合に使用されます。

動的 DNS により更新するゾーンファイルには **/var/named/dynamic/** ディレクトリーを使用します。このディレクトリーで作成またはコピーしたファイルは、**named** による書き込みを許可する Linux 権限を継承します。このようなファイルには **named_cache_t** タイプのラベルが付いているため、SELinux では、**named** がそれらに書き込むことができます。

/var/named/dynamic/ のゾーンファイルに **named_zone_t** タイプのラベルが付いている場合は、マージ前に更新を最初にジャーナルに書き込む必要があるため、動的 DNS 更新が一定期間成功しないことがあります。ジャーナルのマージを試行する際に、ゾーンファイルに**named_zone_t** タイプのラベルが付けられている場合は、以下のようなエラーがログに記録されます。

```
named[PID]: dumping master file: rename: /var/named/dynamic/zone-name: permission denied
```

また、以下の SELinux の拒否メッセージがログに記録されます。

```
setroubleshoot: SELinux is preventing named (named_t) "unlink" to zone-name (named_zone_t)
```

このラベリングの問題を解決するには、root で **restorecon** ユーティリティーを使用します。

```
~]# restorecon -R -v /var/named/dynamic
```

第18章 同時バージョン管理システム

CVS (Concurrent Versioning System) は、リビジョン管理システムです。これは、複数のユーザーが通常アクセスする一元管理ファイルセットへの変更を監視して追跡するために使用されます。これは、プログラマーがソースコードリポジトリーを管理するために一般的に使用され、オープンソースの開発者により広く使用されています。

Red Hat Enterprise Linux では、cvs により CVS が提供されます。以下のコマンドを実行して、cvs パッケージがインストールされているかどうかを確認します。

```
~]$ rpm -q cvs
package cvs is not installed
```

パッケージがインストールされておらず、CVS を使用する場合は、**yum** ユーティリティーを使って root でインストールします。

```
~]# yum install cvs
```

18.1. CVS と SELINUX

cvs デーモンは、**cvs_t** のタイプにラベルを付けて実行します。Red Hat Enterprise Linux におけるデフォルトでは、CVS は特定のディレクトリーの読み取りと書き込みのみを許可されています。ラベル **cvs_data_t** は、**cvs** が読み取りおよび書き込みを行う領域を定義します。SELinux で CVS を使用する場合は、クライアントが CVS データ用に予約された領域に完全アクセスするために、正しいラベルの割り当てが必須となります。

18.2. タイプ

高度なプロセス分離を提供するために、SELinux Targeted ポリシーで使用される主要なパーミッション制御方法は Type Enforcement です。すべてのファイルとプロセスにはタイプのラベルが付いています。タイプはプロセスの SELinux ドメインを定義し、ファイルの SELinux タイプを定義します。SELinux ポリシールールは、タイプにアクセスするドメインであるか、別のドメインにアクセスするドメインであるかにかかわらず、タイプが相互にアクセスする方法を定義します。アクセスは、それを許可する特定の SELinux ポリシールールが存在する場合にのみ許可されます。

CVS では、以下のタイプが使用されます。異なるタイプを使用すると、柔軟なアクセスを設定できます。

cvs_data_t

このタイプは、CVS リポジトリーのデータに使用されます。CVS は、このタイプのデータがある場合に限り、そのデータへの完全アクセスを取得できます。

cvs_exec_t

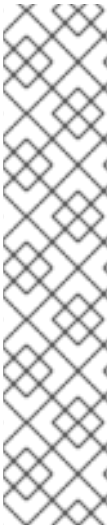
このタイプは、**/usr/bin/cvs** バイナリーに使用されます。

18.3. ブール値

SELinux は、サービスの実行に必要な最小アクセスレベルに基づいています。サービスはさまざまな方法で実行できます。そのため、サービスの実行方法を指定する必要があります。以下のブール値を使用して SELinux を設定します。

cvs_read_shadow

このブール値により、**cvs** デーモンはユーザー認証用の**/etc/shadow** ファイルにアクセスできるようになります。



注記

SELinux ポリシーは継続的に開発されているため、上記のリストには、サービスに関連するすべてのブール値が常に含まれているとは限りません。これらを一覧表示するには、以下のコマンドを入力します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の説明を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

このコマンドが機能するには、**sepolicy** ユーティリティーを提供する追加の policycoreutils-devel パッケージが必要であることに注意してください。

18.4. 設定の例

18.4.1. CDK の設定

この例では、簡単な CVS 設定と、リモートアクセスを可能にする SELinux 設定を説明します。ここでは、ホスト名が **192.168.1.1** の **cvs-srv** の CVS サーバーと、ホスト名が **cvs-client** で IP アドレスが **192.168.1.100** のクライアント の 2 つのホストが使用されます。両方のホストが同じサブネット (192.168.1.0/24) にあります。これは一例で、cvs パッケージと xinetd パッケージがインストールされ、SELinux targeted ポリシーが使用され、SELinux が実施モードで実行されていることを前提としています。

この例では、完全な DAC パーミッションを使用しても、SELinux はファイルラベルに基づくポリシールールを強制でき、CVS によるアクセスのラベルが付けられた特定の領域へのアクセスのみを許可することを示しています。

注記

手順 1-9 は、CVS サーバー **cvs-srv** 上で実行されているはずです。

1. この例では、cvs パッケージと xinetd パッケージが必要です。パッケージがインストールされていることを確認します。

   ```
   [cvs-srv]$ rpm -q cvs xinetd
   package cvs is not installed
   package xinetd is not installed
   ```

   インストールされていない場合は、root で **yum** ユーティリティーを使用してインストールします。

   ```
   [cvs-srv]# yum install cvs xinetd
   ```

2. root で次のコマンドを実行して、**CVS** という名前のグループを作成します。

```
[cvs-srv]# groupadd CVS
```

これは、system-config-users ユーティリティーを使用しても実行できます。

3. **ユーザー名が cvsuser のユーザーを作成し、作成したユーザーを CVS グループのメンバーにします。これは、system-config-users を使用して実行できます。**

4. **/etc/services を編集し、CVS サーバーに、以下のようなコメント解除されたエントリーがあることを確認します。**

```
cvspserver 2401/tcp   # CVS client/server operations
cvspserver 2401/udp   # CVS client/server operations
```

5. **ファイルシステムの root 領域に CVS リポジトリーを作成します。SELinux を使用する場合は、root ファイルシステムにリポジトリーを置いて、他のサブディレクトリーに影響を及ぼさずに再帰的なラベルを付けられるようにすることが最善の方法です。たとえば、root で、リポジトリーを格納する/cvs/ ディレクトリーを作成します。**

```
[root@cvs-srv]# mkdir /cvs
```

6. **すべてのユーザーに、/cvs/ ディレクトリーの完全なパーミッションを付与します。**

```
[root@cvs-srv]# chmod -R 777 /cvs
```

警告

これは単なる例で、このような権限は実稼働システムでは使用できません。

7. **/etc/xinetd.d/cvs ファイルを編集し、CVS セクションがコメント解除されており、/cvs/ ディレクトリーを使用するように設定されていることを確認します。ファイルは以下のようになります。**

```
service cvspserver
{
 disable = no
 port    = 2401
 socket_type  = stream
 protocol  = tcp
 wait    = no
 user    = root
 passenv   = PATH
 server    = /usr/bin/cvs
 env    = HOME=/cvs
 server_args  = -f --allow-root=/cvs pserver
# bind    = 127.0.0.1
```

8. **xinetd デーモンを起動します。**

```
[cvs-srv]# systemctl start xinetd.service
```

9. **system-config-firewall ユーティリティーを使用して、ポート 2401 で TCP 経由での着信接続を許可する規則を追加します。**

10. **クライアントで、cvsuser ユーザーとして以下のコマンドを実行します。**

```
[cvsuser@cvs-client]$ cvs -d /cvs init
```

11. **この時点で、CVS は設定されていますが、SELinux はログインとファイルアクセスを拒否します。これを実証するには、cvs-client で $CVSROOT 変数を設定し、リモートでログインを試みます。cvs-client では、以下の手順を行う必要があります。**

```
[cvsuser@cvs-client]$ export CVSROOT=:pserver:cvsuser@192.168.1.1:/cvs
[cvsuser@cvs-client]$
[cvsuser@cvs-client]$ cvs login
Logging in to :pserver:cvsuser@192.168.1.1:2401/cvs
CVS password: ********
cvs [login aborted]: unrecognized auth response from 192.168.100.1: cvs pserver: cannot
open /cvs/CVSROOT/config: Permission denied
```

SELinux がアクセスをブロックしました。SELinux にこのアクセスを許可させるために、次のステップは cvs-srv で実行されることになっています。

12. **/cvs/ディレクトリーのコンテキストを root 権限で変更し、/cvs/ディレクトリーの既存データーおよび新規データーに再帰的にラベル付けするcvs_data_tタイプを指定します。**

```
[root@cvs-srv]# semanage fcontext -a -t cvs_data_t '/cvs(/.*)?'
[root@cvs-srv]# restorecon -R -v /cvs
```

13. **クライアント cvs-client は、このリポジトリーにログインし、すべての CVS リソースにアクセスできるようになります。**

```
[cvsuser@cvs-client]$ export CVSROOT=:pserver:cvsuser@192.168.1.1:/cvs
[cvsuser@cvs-client]$
[cvsuser@cvs-client]$ cvs login
Logging in to :pserver:cvsuser@192.168.1.1:2401/cvs
CVS password: ********
[cvsuser@cvs-client]$
```

第19章 SQUID キャッシングプロキシー

Squid は、Web クライアント向けの高パフォーマンスのプロキシーキャッシュサーバーで、FTP、Gopher、および HTTP のデータオブジェクトをサポートします。これにより、帯域幅が削減され、頻繁に要求される Web ページのキャッシュと再利用により応答時間が改善されます。[17]

Red Hat Enterprise Linux では、squid パッケージに Squid キャッシングプロキシーが同梱されています。以下のコマンドを実行して、squid パッケージがインストールされているかどうかを確認します。

```
~]$ rpm -q squid
package squid is not installed
```

パッケージがインストールされておらず、squid を使用する場合は、**yum** ユーティリティーを使って root でインストールします。

```
~]# yum install squid
```

19.1. SQUID キャッシングプロキシーおよび SELINUX

SELinux が有効になると、Squid はデフォルトで制限されて実行します。制限のあるプロセスは、独自のドメインで実行され、他の制限のあるプロセスから分離されます。制限のあるプロセスが攻撃者によって侵害された場合、SELinux ポリシーの設定に応じて、攻撃者のリソースへのアクセスと、攻撃者が行う可能性のある損害は制限されます。以下の例は、独自のドメインで実行している Squid プロセスを示しています。この例では、squid パッケージがインストールされていることを前提としています。

1. **getenforce** コマンドを実行して、SELinux が Enforcing モードで実行されていることを確認します。

   ```
   ~]$ getenforce
   Enforcing
   ```

   SELinux が Enforcing モードで実行されていると、このコマンドは **Enforcing** を返します。

2. root で次のコマンドを実行して、**squid** デーモンを起動します。

   ```
   ~]# systemctl start squid.service
   ```

   サービスが実行中であることを確認します。出力には以下の情報が含まれている必要があります (タイムスタンプのみは異なります)。

   ```
   ~]# systemctl status squid.service
   squid.service - Squid caching proxy
     Loaded: loaded (/usr/lib/systemd/system/squid.service; disabled)
     Active: active (running) since Mon 2013-08-05 14:45:53 CEST; 2s ago
   ```

3. **squid**プロセスを表示するには、次のコマンドを実行します。

   ```
   ~]$ ps -eZ | grep squid
   system_u:system_r:squid_t:s0    27018 ?        00:00:00 squid
   system_u:system_r:squid_t:s0    27020 ?        00:00:00 log_file_daemon
   ```

   squid プロセスに関連付けられた SELinux コンテキストはsystem_u:system_r:squid_t:s0 で

す。コンテキストの 2 番目の部分 **squid_t** はタイプです。タイプは、プロセスのドメインとファイルのタイプを定義します。この例では、squid プロセスが **squid_t** ドメインで実行しています。

SELinux ポリシーは、制限のあるドメイン (**squid_t**など) で実行されるプロセスが、ファイル、その他のプロセス、およびシステム一般とどのように対話するかを定義します。squid によるアクセスを可能にするには、ファイルが正しくラベル付けされている必要があります。

/etc/squid/squid.conf ファイルが、デフォルトの TCP ポート 3128、3401、または 4827 以外のポートを **squid** がリッスンするように設定されている場合は、**semanage port** コマンドを使用して、必要なポート番号を SELinux ポリシー設定に追加する必要があります。以下の例では、SELinux ポリシー設定で最初に定義されていないポートをリッスンするように **squid** を設定すると、サーバーが起動しなくなります。また、この例では、ポリシーで定義されていない標準以外のポートをデーモンが正常にリッスンできるように SELinux システムを設定する方法も示しています。この例では、squid パッケージがインストールされていることを前提としています。この例では、各コマンドを root ユーザーとして実行します。

1. **squid** デーモンが実行していないことを確認します。

```
~]# systemctl status squid.service
squid.service - Squid caching proxy
   Loaded: loaded (/usr/lib/systemd/system/squid.service; disabled)
   Active: inactive (dead)
```

出力が異なる場合は、プロセスを停止します。

```
~]# systemctl stop squid.service
```

2. **squid** がリッスンできるポートを表示するには、以下のコマンドを実行します。

```
~]# semanage port -l | grep -w -i squid_port_t
squid_port_t                   tcp      3401, 4827
squid_port_t                   udp      3401, 4827
```

3. root で**/etc/squid/squid.conf** を編集します。**http_port** オプションを設定して、**squid** 用の SELinux ポリシー設定で設定されていないポートの一覧を表示するようにします。この例では、デーモンがポート 10000 をリッスンするように設定されています。

```
# Squid normally listens to port 3128
http_port 10000
```

4. **setsebool** コマンドを実行して、**squid_connect_any** のブール値が off に設定されていることを確認します。これにより、**squid** が特定のポートでのみ動作するようになります。

```
~]# setsebool -P squid_connect_any 0
```

5. **squid** デーモンを起動します。

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

以下のような SELinux 拒否メッセージが記録されます。

```
localhost setroubleshoot: SELinux is preventing the squid (squid_t) from binding to port
10000. For complete SELinux messages. run sealert -l 97136444-4497-4fff-a7a7-
c4d8442db982
```

6. SELinux で、**httpd** がポート 10000 をリッスンできるようにするには、以下のコマンドが必要です。

```
~]# semanage port -a -t squid_port_t -p tcp 10000
```

7. **squid** を再起動して、新しいポートでリッスンするようにします。

```
~]# systemctl start squid.service
```

8. 現在、SELinux は、標準以外のポート (この例では TCP 10000) で Squid がリッスンできるように設定されており、このポートで正常に起動します。

19.2. タイプ

高度なプロセス分離を提供するために、SELinux Targeted ポリシーで使用される主要なパーミッション制御方法は Type Enforcement です。すべてのファイルとプロセスにはタイプのラベルが付いています。タイプはプロセスの SELinux ドメインを定義し、ファイルの SELinux タイプを定義します。SELinux ポリシールールは、タイプにアクセスするドメインであるか、別のドメインにアクセスするドメインであるかにかかわらず、タイプが相互にアクセスする方法を定義します。アクセスは、それを許可する特定の SELinux ポリシールールが存在する場合にのみ許可されます。

Squid では、以下のタイプが使用されます。異なるタイプを使用すると、柔軟なアクセスを設定できます。

httpd_squid_script_exec_t

このタイプは、**cachemgr.cgi** などのユーティリティーに使用されます。このユーティリティーは、Squid とその設定に関するさまざまな統計情報を提供します。

squid_cache_t

/etc/squid/squid.conf の **cache_dir** ディレクティブで定義されているように、Squid によりキャッシュされるデータにこのタイプを使用します。初期設定では、**/var/cache/squid/** ディレクトリーおよび **/var/spool/squid/** ディレクトリーに作成またはコピーされるファイルには、**squid_cache_t** タイプに応じたラベルが付けられています。**/var/squidGuard/** ディレクトリーで作成またはコピーされる **squid** 用の squidGuard URL リダイレクタープラグインのファイルには、**squid_cache_t** タイプのラベルも付いています。Squid は、キャッシュされたデータに対して、このタイプのラベルが付いたファイルおよびディレクトリーのみを使用できます。

squid_conf_t

このタイプは、Squid が設定に使用するディレクトリーおよびファイルに使用されます。既存ファイル、または **/etc/squid/** ディレクトリーおよび **/usr/share/squid/** ディレクトリーで作成またはコピーされたファイルには、エラーメッセージやアイコンなどを含む、このタイプのラベルが付けられています。

squid_exec_t

このタイプは、**squid** のバイナリー (**/usr/sbin/squid**) に使用されます。

squid_log_t

このタイプはログに使用されます。既存のファイル、/var/log/squid/ または /var/log/squidGuard/ に作成またはコピーされたファイルには、この種類のラベルを付ける必要があります。

squid_initrc_exec_t

このタイプは、/etc/rc.d/init.d/squid にある squid を起動するために必要な初期化ファイルに使用されます。

squid_var_run_t

このタイプは、/var/run/ ディレクトリーのファイル、とりわけ /var/run/squid.pid という名前のプロセス ID (PID) で使用されます。この PID は、実行時に Squid により作成されます。

19.3. ブール値

SELinux は、サービスの実行に必要な最小アクセスレベルに基づいています。サービスはさまざまな方法で実行できます。そのため、サービスの実行方法を指定する必要があります。以下のブール値を使用して SELinux を設定します。

squid_connect_any

このブール値を有効にすると、Squid は任意のポートでリモートホストへの接続を開始できます。

squid_use_tproxy

これを有効にすると、ブール値により、Squid を透過プロキシーとして実行できます。

注記

SELinux ポリシーは継続的に開発されているため、上記のリストには、サービスに関連するすべてのブール値が常に含まれているとは限りません。これらを一覧表示するには、以下のコマンドを入力します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の説明を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

このコマンドが機能するには、**sepolicy** ユーティリティーを提供する追加の policycoreutils-devel パッケージが必要であることに注意してください。

19.4. 設定の例

19.4.1. 標準以外のポートへの Squid 接続

以下の例では、上記のブール値を適用し、デフォルトで特定のポートへのアクセスのみを許可することで、SELinux が Squid を補完する方法を実際に示しています。次に、この例では、ブール値を変更する方法を示し、アクセスが許可されることを示します。

これは単なる例で、SELinux が Squid の単純な設定にどのように影響するかを示していることに注意してください。Squid の包括的なドキュメントは、このドキュメントの範囲外です。詳細は、公式の Squid のドキュメント を参照してください。この例では、Squid ホストに 2 つのネットワークインター

フェイス (インターネットアクセス) があり、Squid がリッスンするデフォルトの TCP ポート (TCP 3128) を使用して内部インターフェイスへのアクセスを許可するようにファイアウォールが設定されていることを前提としています。

1. squidがインストールされていることを確認します。

```
~]$ rpm -q squid
package squid is not installed
```

このパッケージがインストールされていない場合は、root で yum ユーティリティーを使用してインストールします。

```
~]# yum install squid
```

2. 主な設定ファイル /etc/squid/squid.conf を編集し、cache_dir ディレクティブがコメント解除されており、以下のようになっていることを確認します。

```
cache_dir ufs /var/spool/squid 100 16 256
```

この行は、この例で使用する cache_dir ディレクティブのデフォルト設定を指定します。これは、Squid ストレージ形式 (ufs)、キャッシュが存在するシステムのディレクトリー (/var/spool/squid)、キャッシュに使用するディスク領域の量 (100)、および作成する第 1 レベルおよび第 2 レベルのキャッシュディレクトリーの数 (それぞれ 16 および 256) で設定されます。

3. 同じ設定ファイルで、http_access allow localnet ディレクティブのコメントが解除されていることを確認します。これにより、Red Hat Enterprise Linux への Squid のデフォルトインストールで自動的に設定される localnet ACL からのトラフィックが許可されます。これにより、既存の RFC1918 ネットワークのクライアントマシンがプロキシーを介してアクセスできるようになります。これは、この単純な例としては十分です。

4. 同じ設定ファイルで、visible_hostname ディレクティブがコメント解除され、マシンのホスト名に設定されていることを確認します。この値は、ホストの完全修飾ドメイン名 (FQDN) にする必要があります。

```
visible_hostname squid.example.com
```

5. root で、以下のコマンドを実行して、squid デーモンを起動します。squid を初めて起動する際に、このコマンドは、cache_dir ディレクティブで指定した方法でキャッシュディレクトリーを初期化し、デーモンを起動します。

```
~]# systemctl start squid.service
```

squid が正常に起動していることを確認します。この出力には、以下の情報が含まれます。タイムスタンプのみが異なります。

```
~]# systemctl status squid.service
squid.service - Squid caching proxy
   Loaded: loaded (/usr/lib/systemd/system/squid.service; disabled)
   Active: active (running) since Thu 2014-02-06 15:00:24 CET; 6s ago
```

6. squid_var_run_t 値からわかるように、squid プロセス ID (PID) が制限付きサービスとして起動していることを確認します。

```
~]# ls -lZ /var/run/squid.pid
-rw-r--r--. root squid unconfined_u:object_r:squid_var_run_t:s0 /var/run/squid.pid
```

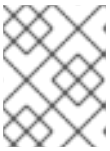
7. この時点で、以前に設定した **localnet** ACL に接続したクライアントマシンは、このホストの内部インターフェイスをプロキシーとして正常に使用できます。これは、すべての一般的な Web ブラウザーの設定で、またはシステム全体で設定できます。Squid はターゲットマシン (TCP 3128) のデフォルトポートでリッスンしていますが、ターゲットマシンは共通ポートを介したインターネット上の他のサービスへの発信接続のみを許可します。これは、SELinux 自体が定義するポリシーです。次の手順で示すように、SELinux は標準以外のポートへのアクセスを拒否します。

8. クライアントが、TCP ポート 10000 でリッスンしている Web サイトなどの Squid プロキシーを使用して、標準以外のポートを使用して要求を行うと、以下のような拒否がログに記録されます。

```
SELinux is preventing the squid daemon from connecting to network port 10000
```

9. これを可能にするには、**squid_connect_any** ブール値を変更する必要があります。これは、既定で無効になっているためです。

```
~]# setsebool -P squid_connect_any on
```



注記

再起動後も、**setsebool** の変更を保持したくない場合は、**-P** オプションを使用しないでください。

10. Squid がクライアントに代わってポートへの接続を開始できるようになったため、クライアントはインターネットの標準以外のポートにアクセスできるようになりました。

---

[17] 詳細は、Squid キャッシングプロキシー プロジェクトページを参照してください。

第20章 MARIADB (MYSQL の代替)

MariaDB データベースは、マルチユーザーのマルチスレッドの SQL データベースサーバーで、MariaDB サーバーデーモン (**mysqld**) と、多くのクライアントプログラムおよびライブラリーで設定されます。[18]

Red Hat Enterprise Linux では、mariadb-server パッケージにより、MariaDB が提供されます。以下のコマンドを実行して、mariadb-server パッケージがインストールされているかどうかを確認します。

```
~]$ rpm -q mariadb-server
package mariadb-server is not installed
```

インストールされていない場合は、root で **yum** ユーティリティーを使用してインストールします。

```
~]# yum install mariadb-server
```

20.1. MARIADB と SELINUX

MariaDB が有効になると、デフォルトで制限ありで実行されます。制限のあるプロセスは、独自のドメインで実行され、他の制限のあるプロセスから分離されます。制限のあるプロセスが攻撃者によって侵害された場合、SELinux ポリシーの設定に応じて、攻撃者のリソースへのアクセスと、攻撃者が行う可能性のある損害は制限されます。以下の例は、独自のドメインで実行している MariaDB プロセスを示しています。この例では、mariadb-server パッケージがインストールされていることを前提としています。

1. **getenforce** コマンドを実行して、SELinux が Enforcing モードで実行されていることを確認します。

   ```
   ~]$ getenforce
   Enforcing
   ```

   SELinux が Enforcing モードで実行されていると、このコマンドは **Enforcing** を返します。

2. root で次のコマンドを入力して、**mariadb** を起動します。

   ```
   ~]# systemctl start mariadb.service
   ```

   サービスが実行中であることを確認します。出力には以下の情報が含まれている必要があります (タイムスタンプのみは異なります)。

   ```
   ~]# systemctl status mariadb.service
   mariadb.service - MariaDB database server
      Loaded: loaded (/usr/lib/systemd/system/mariadb.service; disabled)
      Active: active (running) since Mon 2013-08-05 11:20:11 CEST; 3h 28min ago
   ```

3. **mysqld**プロセスを表示するには、次のコマンドを実行します。

   ```
   ~]$ ps -eZ | grep mysqld
   system_u:system_r:mysqld_safe_t:s0 12831 ?     00:00:00 mysqld_safe
   system_u:system_r:mysqld_t:s0    13014 ?       00:00:00 mysqld
   ```

mysqld プロセスに関連付けられた SELinux コンテキストはsystem_u:system_r:mysqld_t:s0 です。コンテキストの 2 番目の部分 mysqld_t はタイプです。タイプは、プロセスのドメインとファイルのタイプを定義します。この例では、mysqld プロセスが mysqld_t ドメインで実行しています。

20.2. タイプ

高度なプロセス分離を提供するために、SELinux Targeted ポリシーで使用される主要なパーミッション制御方法は Type Enforcement です。すべてのファイルとプロセスにはタイプのラベルが付いています。タイプはプロセスの SELinux ドメインを定義し、ファイルの SELinux タイプを定義します。SELinux ポリシールールは、タイプにアクセスするドメインであるか、別のドメインにアクセスするドメインであるかにかかわらず、タイプが相互にアクセスする方法を定義します。アクセスは、それを許可する特定の SELinux ポリシールールが存在する場合にのみ許可されます。

mysqld では、以下のタイプが使用されます。異なるタイプを使用すると、柔軟なアクセスを設定できます。

mysqld_db_t

このタイプは、MariaDB データベースの場所に使用されます。Red Hat Enterprise Linux では、データベースのデフォルトの場所は /var/lib/mysql/ ディレクトリーですが、これは変更できます。MariaDB データベースの場所を変更する場合は、このタイプのラベルを、新しい場所に作成する必要があります。デフォルトのデータベースの場所を変更する方法と、新しいセクションに適切なラベルを付ける方法は、「MariaDB によるデータベースの場所の変更」 の例を参照してください。

mysqld_etc_t

このタイプは、MariaDB のメイン設定ファイル/etc/my.cnfおよび /etc/mysql/ ディレクトリー内のその他の設定ファイルに使用されます。

mysqld_exec_t

このタイプは、/usr/libexec/mysqld にある mysqld バイナリーに使用されます。これは、Red Hat Enterprise Linux における MariaDB バイナリーのデフォルトの場所です。他のシステムは、このバイナリーを /usr/sbin/mysqld に置くことができます。このバイナリーには、このタイプのラベルを付ける必要があります。

mysqld_unit_file_t

このタイプは、Red Hat Enterprise Linux のデフォルトでは、/usr/lib/systemd/system/ ディレクトリーにある実行ファイル MariaDB 関連のファイルに使用されます。

mysqld_log_t

MariaDB のログには、適切な操作を行うためにこのタイプのラベルを付ける必要があります。mysql.* ワイルドカードに一致する /var/log/ ディレクトリー内のすべてのログファイルに、この種類のラベルを付ける必要があります。

mysqld_var_run_t

このタイプは、/var/run/mariadb/ ディレクトリーのファイル、特に、実行時に mysqld デーモンにより作成される /var/run/mariadb/mariadb.pid という名前のプロセス ID (PID) で使用されます。このタイプは、/var/lib/mysql/mysql.sock などの関連するソケットファイルにも使用されます。このようなファイルには、限定サービスとして適切な操作を行うために、正しくラベルを付ける必要があります。

20.3. ブール値

SELinux は、サービスの実行に必要な最小アクセスレベルに基づいています。サービスはさまざまな方法で実行できます。そのため、サービスの実行方法を指定する必要があります。以下のブール値を使用して SELinux を設定します。

selinuxuser_mysql_connect_enabled

このブール値を有効にすると、ローカルの MariaDB サーバーに接続できるようになります。

exim_can_connect_db

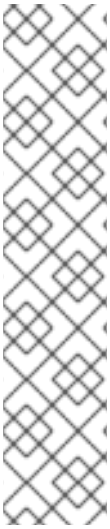
このブール値を有効にすると、exim メーラーはデータベースサーバーへの接続を開始できます。

ftpd_connect_db

このブール値を有効にすると、ftp デーモンがデータベースサーバーへの接続を開始できるようになります。

httpd_can_network_connect_db

Web サーバーがデータベースサーバーと通信するには、このブール値を有効にする必要があります。



注記

SELinux ポリシーは継続的に開発されているため、上記のリストには、サービスに関連するすべてのブール値が常に含まれているとは限りません。これらを一覧表示するには、以下のコマンドを入力します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の説明を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

このコマンドが機能するには、**sepolicy** ユーティリティーを提供する追加の policycoreutils-devel パッケージが必要であることに注意してください。

20.4. 設定の例

20.4.1. MariaDB によるデータベースの場所の変更

Red Hat Enterprise Linux を使用する場合、MariaDB がデータベースを保存するデフォルトの場所は **/var/lib/mysql/** です。ここでは、SELinux がデフォルトであると想定している場所であるため、**mysqld_db_t** タイプを使用して、この領域はすでに適切にラベル付けされています。

データベースの保存場所は、個々の環境の要件や設定により変更できますが、SELinux は、この新しい場所を認識しており、それに応じてラベルが付けられていることが重要です。この例では、MariaDB データベースの場所を変更し、その内容に基づいて SELinux の保護メカニズムを新しい領域に引き続き提供できるように、新しい場所にラベルを付ける方法を説明します。

これは単なる例で、SELinux が MariaDB に与える影響を示していることに注意してください。MariaDB の包括的なドキュメントは、このドキュメントの範囲外です。詳細は、公式の MariaDB のドキュメント を参照してください。この例では、mariadb-server パッケージおよび setroubleshoot-

server パッケージがインストールされていること、**auditd** サービスが実行中であること、および **/var/lib/mysql/** のデフォルトの場所に有効なデータベースがあることを前提としています。

1. **mysql** のデフォルトのデータベースの場所の SELinux コンテキストを表示します。

```
~]# ls -lZ /var/lib/mysql
drwx------. mysql mysql system_u:object_r:mysqld_db_t:s0 mysql
```

これは、データベースファイルの場所のデフォルトのコンテキスト要素である**mysqld_db_t** を示しています。このコンテキストを適切に機能させるために、この例で使用される新しいデータベースの場所に、手動で適用する必要があります。

2. 次のコマンドを入力し、**mysqld** root パスワードを入力して、利用可能なデータベースを表示します。

```
~]# mysqlshow -u root -p
Enter password: *******
+--------------------+
|     Databases      |
+--------------------+
| information_schema |
| mysql              |
| test               |
| wikidb             |
+--------------------+
```

3. **mariadb.service** サービスを停止します。

```
~]# systemctl stop mariadb.service
```

4. データベースの新しい場所に、新しいディレクトリーを作成します。この例では、**/mysql/** が使用されています。

```
~]# mkdir -p /mysql
```

5. 古い場所から新しい場所に、データベースファイルをコピーします。

```
~]# cp -R /var/lib/mysql/* /mysql/
```

6. mysql ユーザーおよびグループによるアクセスを許可するように、この場所の所有権を変更します。これにより、SELinux が引き続き監視する従来の Unix パーミッションが設定されます。

```
~]# chown -R mysql:mysql /mysql
```

7. 以下のコマンドを実行すると、作成したディレクトリーの初期コンテキストが表示されます。

```
~]# ls -lZ /mysql
drwxr-xr-x. mysql mysql unconfined_u:object_r:usr_t:s0   mysql
```

新しく作成されたこのディレクトリーのコンテキスト**usr_t** は、現在、MariaDB データベースファイルの場所として SELinux には適していません。コンテキストを変更すると、MariaDB はこのエリアで適切に機能できるようになります。

8. テキストエディターで、MariaDB のメイン設定ファイル/etc/my.cnfを開き、新しい場所を参照できるように **datadir** オプションを変更します。この例で入力する必要のある値は、**/mysql** です。

```
[mysqld]
datadir=/mysql
```

このファイルを保存して終了します。

9. **mariadb.service** を起動します。サービスの起動に失敗し、拒否メッセージが**/var/log/messages** ファイルに記録されます。

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

ただし、**audit** デーモンが **setroubleshoot** とともに実行されている場合は、拒否のログが **/var/log/audit/audit.log** に記録されます。

```
SELinux is preventing /usr/libexec/mysqld "write" access on /mysql. For complete SELinux messages. run sealert -l b3f01aff-7fa6-4ebe-ad46-abaef6f8ad71
```

この拒否の理由は、MariaDB データファイルに対して **/mysql/** が正しくラベル付けされていないことです。SELinux により、MariaDB が **usr_t** とラベル付けされたコンテンツにアクセスできなくなりました。この問題を解決するには、以下の手順を実行します。

10. 以下のコマンドを入力して、**/mysql/** のコンテキストマッピングを追加します。**semanage** ユーティリティーは、デフォルトではインストールされないことに注意してください。同梱されていない場合は、policycoreutils-python をインストールしてください。

```
~]# semanage fcontext -a -t mysqld_db_t "/mysql(/.*)?"
```

11. このマッピングは、**/etc/selinux/targeted/contexts/files/file_contexts.local** ファイルに書き込まれます。

```
~]# grep -i mysql /etc/selinux/targeted/contexts/files/file_contexts.local

/mysql(/.*)?    system_u:object_r:mysqld_db_t:s0
```

12. これで **restorecon** ユーティリティーを使用して、このコンテキストマッピングを実行中のシステムに適用します。

```
~]# restorecon -R -v /mysql
```

13. **/mysql/** の場所には MariaDB の正しいコンテキストでラベル付けされ、**mysqld** が起動します。

```
~]# systemctl start mariadb.service
```

14. **/mysql/** に対してコンテキストが変更されたことを確認します。

```
~]$ ls -lZ /mysql
drwxr-xr-x. mysql mysql system_u:object_r:mysqld_db_t:s0 mysql
```

15. **この場所が変更され、ラベルが付けられ、mysqld が正常に起動されました。この時点で、実行中のすべてのサービスをテストして、通常の操作を確認してください。**

[18] **詳細は、MariaDB プロジェクトページを参照してください。**

第21章 POSTGRESQL

PostgreSQL は、オブジェクトリレーショナルデータベース管理システム (DBMS) です。[19]

Red Hat Enterprise Linux では、postgresql-server パッケージが PostgreSQL を提供します。以下のコマンドを実行して、postgresql-server パッケージがインストールされているかどうかを確認します。

```
~]# rpm -q postgresql-server
```

インストールされていない場合は、root で **yum** ユーティリティーを使用してインストールします。

```
~]# yum install postgresql-server
```

21.1. POSTGRESQL および SELINUX

PostgreSQL が有効になると、デフォルトで制限ありで実行されます。制限のあるプロセスは、独自のドメインで実行され、他の制限のあるプロセスから分離されます。制限のあるプロセスが攻撃者によって侵害された場合、SELinux ポリシーの設定に応じて、攻撃者のリソースへのアクセスと、攻撃者が行う可能性のある損害は制限されます。以下の例は、独自のドメインで実行している PostgreSQL プロセスを示しています。この例では、postgresql-server がインストールされていることを前提としています。

1. **getenforce** コマンドを実行して、SELinux が Enforcing モードで実行されていることを確認します。

   ```
   ~]$ getenforce
   Enforcing
   ```

   SELinux が Enforcing モードで実行されていると、このコマンドは **Enforcing** を返します。

2. root で次のコマンドを入力して、**postgresql** を起動します。

   ```
   ~]# systemctl start postgresql.service
   ```

   サービスが実行中であることを確認します。出力には以下の情報が含まれている必要があります (タイムスタンプのみは異なります)。

   ```
   ~]# systemctl start postgresql.service
   postgresql.service - PostgreSQL database server
     Loaded: loaded (/usr/lib/systemd/system/postgresql.service; disabled)
     Active: active (running) since Mon 2013-08-05 14:57:49 CEST; 12s
   ```

3. **postgresql**プロセスを表示するには、次のコマンドを実行します。

   ```
   ~]$ ps -eZ | grep postgres
   system_u:system_r:postgresql_t:s0 395 ?    00:00:00 postmaster
   system_u:system_r:postgresql_t:s0 397 ?    00:00:00 postmaster
   system_u:system_r:postgresql_t:s0 399 ?    00:00:00 postmaster
   system_u:system_r:postgresql_t:s0 400 ?    00:00:00 postmaster
   system_u:system_r:postgresql_t:s0 401 ?    00:00:00 postmaster
   system_u:system_r:postgresql_t:s0 402 ?    00:00:00 postmaster
   ```

postgresql プロセスに関連付けられた SELinux コンテキスト はsystem_u:system_r:postgresql_t:s0 です。文脈の最後の部分である postgresql_t はタイプです。タイプは、プロセスのドメインとファイルのタイプを定義します。この例では、postgresql プロセスが postgresql_t ドメインで実行しています。

21.2. タイプ

高度なプロセス分離を提供するために、SELinux Targeted ポリシーで使用される主要なパーミッション制御方法は Type Enforcement です。すべてのファイルとプロセスにはタイプのラベルが付いています。タイプはプロセスの SELinux ドメインを定義し、ファイルの SELinux タイプを定義します。SELinux ポリシールールは、タイプにアクセスするドメインであるか、別のドメインにアクセスするドメインであるかにかかわらず、タイプが相互にアクセスする方法を定義します。アクセスは、それを許可する特定の SELinux ポリシールールが存在する場合にのみ許可されます。

postgresql では、以下のタイプが使用されます。異なるタイプを使用すると、柔軟なアクセスを設定できます。以下の一覧では、使用可能な場所全体を一致させるために、複数の正規表現を使用していることに注意してください。

postgresql_db_t

このタイプは複数の場所で使用されます。このタイプのラベルが付いた場所は、PostgreSQL のデータファイルに使用されます。

- /usr/lib/pgsql/test/regres
- /usr/share/jonas/pgsql
- /var/lib/pgsql/data
- /var/lib/postgres(ql)?

postgresql_etc_t

このタイプは、/etc/postgresql/ ディレクトリーの設定ファイルに使用されます。

postgresql_exec_t

このタイプは複数の場所で使用されます。このタイプのラベルが付いた場所は、PostgreSQL のバイナリーに使用されます。

- /usr/bin/initdb(.sepgsql)?
- /usr/bin/(se)?postgres
- /usr/lib(64)?/postgresql/bin/.*
- /usr/lib(64)?/pgsql/test/regress/pg_regress

systemd_unit_file_t

このタイプは、/usr/lib/systemd/system/ ディレクトリーに置かれている PostgreSQL 関連の実行ファイルに使用されます。

postgresql_log_t

このタイプは複数の場所で使用されます。このタイプのラベルが付いた場所は、ログファイルに使用されます。

- /var/lib/pgsql/logfile

- /var/lib/pgsql/pgstartup.log
- /var/lib/sepgsql/pgstartup.log
- /var/log/postgresql
- /var/log/postgres.log.*
- /var/log/rhdb/rhdb
- /var/log/sepostgresql.log.*

postgresql_var_run_t

このタイプは、/var/run/postgresql/ ディレクトリーのプロセス ID (PID) など、PostgreSQL のランタイムファイルに使用されます。

21.3. ブール値

SELinux は、サービスの実行に必要な最小アクセスレベルに基づいています。サービスはさまざまな方法で実行できます。そのため、サービスの実行方法を指定する必要があります。以下のブール値を使用して SELinux を設定します。

selinuxuser_postgresql_connect_enabled

ブール値を有効にすると、(PostgreSQL により定義される) ユーザードメインがデータベースサーバーへの接続を確立できるようになります。

注記

SELinux ポリシーは継続的に開発されているため、上記のリストには、サービスに関連するすべてのブール値が常に含まれているとは限りません。これらを一覧表示するには、以下のコマンドを入力します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の説明を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

このコマンドが機能するには、**sepolicy** ユーティリティーを提供する追加の policycoreutils-devel パッケージが必要であることに注意してください。

21.4. 設定の例

21.4.1. PostgreSQL データベースの場所の変更

Red Hat Enterprise Linux を使用する場合、PostgreSQL がデータベースを保存するデフォルトの場所は /var/lib/pgsql/data/ です。ここでは、SELinux がデフォルトであると想定している場所であるため、**postgresql_db_t** タイプを使用して、この領域はすでに適切にラベル付けされています。

データベースの場所は、個々の環境の要件や設定により変更できますが、SELinux は、この新しい場所を認識しており、それに応じてラベルが付けられていることが重要です。この例では、PostgreSQL

データベースの場所を変更し、その内容に基づいて SELinux の保護メカニズムを新しい領域に引き続き提供できるように、新しい場所にラベルを付ける方法を説明します。

これは単なる例で、SELinux が PostgreSQL に与える影響を示していることに注意してください。PostgreSQL の包括的なドキュメントは、このドキュメントの範囲外です。詳細は、公式の PostgreSQL のドキュメント を参照してください。この例では、postgresql-server パッケージがインストールされていることを前提としています。

1. **postgresql** のデフォルトのデータベースの場所の SELinux コンテキストを表示します。

```
~]# ls -lZ /var/lib/pgsql
drwx------. postgres postgres system_u:object_r:postgresql_db_t:s0 data
```

これは、データベースファイルの場所のデフォルトのコンテキスト要素である**postgresql_db_t** を示しています。このコンテキストを適切に機能させるために、この例で使用される新しいデータベースの場所に、手動で適用する必要があります。

2. データベースの新しい場所に、新しいディレクトリーを作成します。この例では、**/opt/postgresql/data/** が使用されています。別の場所を使用する場合は、以下の手順のテキストを、使用している場所に置き換えます。

```
~]# mkdir -p /opt/postgresql/data
```

3. 新しい場所のディレクトリーの一覧表示を実行します。新規ディレクトリーの最初のコンテキストは**usr_t** であることに注意してください。このコンテキストは、SELinux が PostgreSQL にその保護メカニズムを提供するためには不十分です。コンテキストを変更すると、新しいエリアで適切に機能できるようになります。

```
~]# ls -lZ /opt/postgresql/
drwxr-xr-x. root root unconfined_u:object_r:usr_t:s0   data
```

4. postgres ユーザーおよびグループによるアクセスを許可するために、新しい場所の所有権を変更します。これにより、SELinux が引き続き監視する従来の Unix パーミッションが設定されます。

```
~]# chown -R postgres:postgres /opt/postgresql
```

5. テキストエディターで **/etc/systemd/system/postgresql.service** ファイルを開き、**PGDATA** 変数および **PGLOG** 変数を変更して、新しい場所を指定します。

```
~]# vi /etc/systemd/system/postgresql.service
PGDATA=/opt/postgresql/data
PGLOG=/opt/postgresql/data/pgstartup.log
```

このファイルを保存し、テキストエディターを終了します。

/etc/systemd/system/postgresql.service ファイルが存在しない場合は作成し、次の内容を挿入します。

```
.include /lib/systemd/system/postgresql.service
[Service]
```

```
# Location of database directory
Environment=PGDATA=/opt/postgresql/data
Environment=PGLOG=/opt/postgresql/data/pgstartup.log
```

6. **新しい場所でデータベースを初期化します。**

```
~]$ su - postgres -c "initdb -D /opt/postgresql/data"
```

7. **データベースの場所を変更すると、この時点でサービスの起動に失敗します。**

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

SELinux によりサービスが起動しなくなりました。これは、新しい場所に適切なラベルが付けられていないためです。以下の手順では、新しい場所 (/opt/postgresql/) にラベルを付け、postgresql サービスを適切に起動する方法を説明します。

8. **semanage ユーティリティーを使用して、/opt/postgresql/ およびその中のその他のディレクトリーやファイルに対するコンテキストマッピングを追加します。**

```
~]# semanage fcontext -a -t postgresql_db_t "/opt/postgresql(/.*)?"
```

9. **このマッピングは、/etc/selinux/targeted/contexts/files/file_contexts.local ファイルに書き込まれます。**

```
~]# grep -i postgresql /etc/selinux/targeted/contexts/files/file_contexts.local

/opt/postgresql(/.*)?    system_u:object_r:postgresql_db_t:s0
```

10. **これで restorecon ユーティリティーを使用して、このコンテキストマッピングを実行中のシステムに適用します。**

```
~]# restorecon -R -v /opt/postgresql
```

11. **/opt/postgresql/ の場所に、PostgreSQL の正しいコンテキストのラベルが付けられるようになり、postgresql サービスが正常に起動します。**

```
~]# systemctl start postgresql.service
```

12. **/opt/postgresql/ のコンテキストが正しいことを確認します。**

```
~]$ ls -lZ /opt
drwxr-xr-x. root root system_u:object_r:postgresql_db_t:s0 postgresql
```

13. **postgresql 処理で新しい場所が表示されることを、ps コマンドで確認します。**

```
~]# ps aux | grep -i postmaster

postgres 21564  0.3  0.3  42308  4032 ?        S    10:13   0:00 /usr/bin/postmaster -p 5432 -D
/opt/postgresql/data/
```

14. 場所が変更され、ラベルが付けられ、postgresql が正常に起動しました。この時点で、実行中のすべてのサービスをテストして、通常の操作を確認してください。

[19] 詳細は、PostgreSQL プロジェクトページを参照してください。

第22章 RSYNC

rsync ユーティリティーは、高速なファイル転送を実行し、システム間でデータの同期に使用されます。[20]

Red Hat Enterprise Linux を使用する場合は、rsync パッケージにより rsync が提供されます。以下のコマンドを実行して、rsync パッケージがインストールされているかどうかを確認します。

```
~]$ rpm -q rsync
package rsync is not installed
```

インストールされていない場合は、root で **yum** ユーティリティーを使用してインストールします。

```
~]# yum install rsync
```

22.1. RSYNC および SELINUX

SELinux では、ファイルタイプを定義するために、ファイルに拡張属性が必要です。ポリシーは、これらのファイルに必要なアクセスデーモンを管理します。**rsync** デーモンを使用してファイルを共有する場合は、ファイルおよびディレクトリーに **public_content_t** タイプのラベルを付ける必要があります。ほとんどのサービスと同様に、SELinux が **rsync** 経由でその防御メカニズムを実行するには、正しいラベリングが必要になります。[21]

22.2. タイプ

高度なプロセス分離を提供するために、SELinux Targeted ポリシーで使用される主要なパーミッション制御方法は Type Enforcement です。すべてのファイルとプロセスにはタイプのラベルが付いています。タイプはプロセスの SELinux ドメインを定義し、ファイルの SELinux タイプを定義します。SELinux ポリシールールは、タイプにアクセスするドメインであるか、別のドメインにアクセスするドメインであるかにかかわらず、タイプが相互にアクセスする方法を定義します。アクセスは、それを許可する特定の SELinux ポリシールールが存在する場合にのみ許可されます。

rsync では、以下のタイプが使用されます。フレキシブルアクセスを設定するには、以下のいずれかを行います。

public_content_t

これは、**rsync** を使用して共有するファイル (および実際のファイル) の場所に使用される一般的なタイプです。**rsync** と共有するファイルを格納する特殊なディレクトリーを作成する場合は、そのディレクトリーとその内容に、このラベルを適用する必要があります。

rsync_exec_t

このタイプは、**/usr/bin/rsync** システムバイナリーに使用されます。

rsync_log_t

このタイプは、**rsync** ログファイルに使用されます。デフォルトでは **/var/log/rsync.log** にあります。rsync ログファイルの場所を変更するには、実行時に **rsync** コマンドの **--log-file=FILE** を使用します。

rsync_var_run_t

このタイプは、**/var/run/rsyncd.lock** にある **rsyncd** ロックファイルに使用されます。このロックファイルは、**rsync** サーバーが接続制限を管理するために使用されます。

rsync_data_t

このタイプは、rsync ドメインとして使用し、その他のサービスのアクセス範囲から分離するファイルおよびディレクトリーに使用されます。また、**public_content_t** は、ファイルまたはディレクトリーが複数のサービス (たとえば、rsync ドメインとしての FTP および NFS ディレクトリー) と相互作用する場合に使用できる、一般的な SELinux コンテキストタイプです。

rsync_etc_t

このタイプは、**/etc** ディレクトリー内の rsync 関連のファイルに使用されます。

22.3. ブール値

SELinux は、サービスの実行に必要な最小アクセスレベルに基づいています。サービスはさまざまな方法で実行できます。そのため、サービスの実行方法を指定する必要があります。以下のブール値を使用して SELinux を設定します。

rsync_anon_write

ブール値を有効にすると、**rsync_t** ドメイン内の**rsync** が、**public_content_rw_t** タイプがあるファイル、リンク、およびディレクトリーを管理できるようになります。多くの場合、これは公共のファイル転送サービスに使用される公共ファイルです。ファイルおよびディレクトリーには、このタイプのラベルを付ける必要があります。

rsync_client

ブール値を有効にすると、**rsync** は、**rsync_port_t** として定義されたポートへの接続を開始するだけでなく、**rsync_data_t** タイプがあるファイル、リンク、およびディレクトリーを管理できるようになります。SELinux がその制御を行えるようにするには、**rsync** が**rsync_t** ドメインに存在する必要があることに注意してください。本章の設定例は、**rsync_t** ドメインで実行している**rsync**を示しています。

rsync_export_all_ro

このブール値を有効にすると、**rsync_t** ドメインの **rsync** が、クライアントへの読み取り専用アクセスを持つ NFS および CIFS ボリュームをエクスポートできるようになります。

注記

SELinux ポリシーは継続的に開発されているため、上記のリストには、サービスに関連するすべてのブール値が常に含まれているとは限りません。これらを一覧表示するには、以下のコマンドを入力します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の説明を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

このコマンドが機能するには、**sepolicy** ユーティリティーを提供する追加の policycoreutils-devel パッケージが必要であることに注意してください。

22.4. 設定の例

22.4.1. デーモンとしての Rsync

Red Hat Enterprise Linux を使用する場合は、rsync をデーモンとして使用できるため、複数のクライアントが中央サーバーとして直接通信し、集中ファイルを格納して同期を維持できます。以下の例は、正しいドメインのネットワークソケットで rsync をデーモンとして実行し、SELinux がこのデーモンを事前定義済み (SELinux ポリシー) の TCP ポートで実行する方法を示しています。次に、**rsync** デーモンが標準以外のポートで通常どおり実行できるように SELinux ポリシーを変更する方法を示します。

この例は、1つのシステムで実行し、SELinux ポリシーと、そのローカルデーモンおよびプロセスに対する制御を示します。これは単なる例で、SELinux が rsync に与える影響を示していることに注意してください。rsync の包括的なドキュメントは、このドキュメントの範囲外です。詳細は、公式の rsync のドキュメント を参照してください。この例では、rsync、setroubleshoot-server、および audit パッケージがインストールされ、SELinux Targeted ポリシーが使用され、SELinux が Enforcing モードで実行されていることを前提としています。

手順22.1 rsync を rsync_t として起動させる

1. **getenforce** コマンドを実行して、SELinux が Enforcing モードで実行されていることを確認します。

```
~]$ getenforce
Enforcing
```

   SELinux が Enforcing モードで実行されていると、このコマンドは **Enforcing** を返します。

2. **which** コマンドを実行して、rsync バイナリーがシステムパスにあることを確認します。

```
~]$ which rsync
/usr/bin/rsync
```

3. rsync をデーモンとして実行する場合は、設定ファイルを使用して、**/etc/rsyncd.conf** として保存する必要があります。この例で使用されている以下の設定ファイルは非常に単純であり、使用可能なすべてのオプションを示しているわけではなく、**rsync** デーモンを示すだけで十分であることに注意してください。

```
log file = /var/log/rsync.log
pid file = /var/run/rsyncd.pid
lock file = /var/run/rsync.lock
[files]
 path = /srv/rsync
        comment = file area
        read only = false
        timeout = 300
```

4. rsync がデーモンモードで動作するための簡単な設定ファイルが存在するようになったため、以下のコマンドを実行して起動できます。

```
~]# systemctl start rsyncd.service
```

   rsyncd が正常に起動したことを確認します (出力は以下のようになります。タイムスタンプのみが異なります)。

```
~]# systemctl status rsyncd.service
rsyncd.service - fast remote file copy program daemon
```

```
   Loaded: loaded (/usr/lib/systemd/system/rsyncd.service; disabled)
   Active: active (running) since Thu 2014-02-27 09:46:24 CET; 2s ago
 Main PID: 3220 (rsync)
   CGroup: /system.slice/rsyncd.service
           └─3220 /usr/bin/rsync --daemon --no-detach
```

SELinux は、**rsync_t** ドメインで実行しているときに、**rsync** デーモンにプロテクションメカニズムを適用できるようになりました。

```
~]$ ps -eZ | grep rsync
system_u:system_r:rsync_t:s0     3220 ?        00:00:00 rsync
```

この例は、**rsync_t** ドメインで実行している**rsyncd**を取得する方法を示しています。Rsync は、ソケットが有効なサービスとして実行することもできます。これにより、クライアントがサービスへの接続を試行するまで、**rsyncd**が実行されません。**rsyncd** がソケットにより起動したサービスとして実行できるようにするには、上記の手順を行います。**rsyncd** をソケット起動サービスとして起動するには、root で以下のコマンドを実行します。

```
~]# systemctl start rsyncd.socket
```

以下の例は、デフォルト以外のポートでこのデーモンを正常に実行する方法を示しています。次の例では、TCP ポート 10000 が使用されています。

手順22.2 デフォルト以外のポートでの rsync デーモンの実行

1. **/etc/rsyncd.conf** ファイルを変更し、グローバル設定領域 (つまりファイル領域が定義する前) のファイルの一番上にある **port = 10000** 行を追加します。新しい設定ファイルは以下のようになります。

```
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
lock file = /var/run/rsync.lock
port = 10000
[files]
        path = /srv/rsync
        comment = file area
        read only = false
 timeout = 300
```

2. この新しい設定で **rsync** デーモンを起動すると、次のような拒否メッセージが SELinux によりログに記録されます。

```
Jul 22 10:46:59 localhost setroubleshoot: SELinux is preventing the rsync (rsync_t) from
binding to port 10000. For complete SELinux messages, run sealert -l c371ab34-639e-45ae-
9e42-18855b5c2de8
```

3. **semanage** ユーティリティーを使用して、**rsync_port_t** の SELinux ポリシーに TCP ポート 10000 を追加します。

```
~]# semanage port -a -t rsync_port_t -p tcp 10000
```

4. **rsync_port_t** の SELinux ポリシーに TCP ポート 10000 が追加され、**rsyncd** はこのポートで通常どおり起動して動作するようになりました。

```
~]# systemctl start rsyncd.service
```

```
~]# netstat -lnp | grep 10000
tcp        0      0 0.0.0.0:10000   0.0.0.0:*       LISTEN      9910/rsync
```

SELinux ではポリシーが変更され、rsyncd の TCP ポート 10000 での動作が許可されるようになりました。

---

[20] 詳細は、Rsync プロジェクトページを参照してください。

[21] rsync および SELinux の詳細は、rsync_selinux(8) の man ページを参照してください。

第23章 POSTFIX

Postfix はオープンソースのメールトランスポートエージェント (MTA) で、LDAP、SMTP AUTH (SASL)、TLS などのプロトコルに対応します。[22]

Red Hat Enterprise Linux では、postfix パッケージにより Postfix が提供されます。以下のコマンドを実行して、postfix パッケージがインストールされているかどうかを確認します。

```
~]$ rpm -q postfix
package postfix is not installed
```

インストールされていない場合は、**yum** ユーティリティー root を使用して、インストールします。

```
~]# yum install postfix
```

23.1. POSTFIX および SELINUX

Postfix が有効になると、デフォルトで制限ありで実行されます。制限のあるプロセスは、独自のドメインで実行され、他の制限のあるプロセスから分離されます。制限のあるプロセスが攻撃者によって侵害された場合、SELinux ポリシーの設定に応じて、攻撃者のリソースへのアクセスと、攻撃者が行う可能性のある損害は制限されます。以下の例は、Postfix と、そのドメインで実行している関連プロセスを示しています。この例では、postfix パッケージがインストールされ、Postfix サービスが起動していることを前提としています。

1. **getenforce** コマンドを実行して、SELinux が Enforcing モードで実行されていることを確認します。

   ```
   ~]$ getenforce
   Enforcing
   ```

   SELinux が Enforcing モードで実行されていると、このコマンドは **Enforcing** を返します。

2. root で次のコマンドを入力して、**postfix** を起動します。

   ```
   ~]# systemctl start postfix.service
   ```

   サービスが実行中であることを確認します。出力には以下の情報が含まれている必要があります (タイムスタンプのみは異なります)。

   ```
   ~]# systemctl status postfix.service
   postfix.service - Postfix Mail Transport Agent
     Loaded: loaded (/usr/lib/systemd/system/postfix.service; disabled)
     Active: active (running) since Mon 2013-08-05 11:38:48 CEST; 3h 25min ago
   ```

3. 次のコマンドを実行して、**postfix** プロセスを表示します。

   ```
   ~]$ ps -eZ | grep postfix
   system_u:system_r:postfix_master_t:s0 1651 ?   00:00:00 master
   system_u:system_r:postfix_pickup_t:s0 1662 ?   00:00:00 pickup
   system_u:system_r:postfix_qmgr_t:s0 1663 ?     00:00:00 qmgr
   ```

   ここでは、Postfix **master** プロセスに関連する SELinux コンテキストが

system_u:system_r:postfix_master_t:s0 になっています。文脈の最後の部分である **postfix_master_t** は、この処理のタイプです。タイプは、プロセスのドメインとファイルのタイプを定義します。この例では、**master** プロセスが **postfix_master_t** ドメインで実行しています。

23.2. タイプ

高度なプロセス分離を提供するために、SELinux Targeted ポリシーで使用される主要なパーミッション制御方法は Type Enforcement です。すべてのファイルとプロセスにはタイプのラベルが付いています。タイプはプロセスの SELinux ドメインを定義し、ファイルの SELinux タイプを定義します。SELinux ポリシールールは、タイプにアクセスするドメインであるか、別のドメインにアクセスするドメインであるかにかかわらず、タイプが相互にアクセスする方法を定義します。アクセスは、それを許可する特定の SELinux ポリシールールが存在する場合にのみ許可されます。

Postfix では、以下のタイプが使用されます。フレキシブルアクセスを設定するには、以下のいずれかを行います。

postfix_etc_t

このタイプは、**/etc/postfix/** ディレクトリーの Postfix の設定ファイルに使用されます。

postfix_data_t

このタイプは、**/var/lib/postfix/** ディレクトリーの Postfix データファイルに使用されます。

postfix_var_run_t

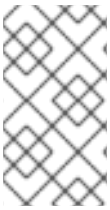
このタイプは、**/run/** ディレクトリーに保存されている Postfix ファイルに使用されます。

postfix_initrc_exec_t

Postfix 実行ファイルには、**postfix_initrc_exec_t** タイプにラベルが付いています。実行すると、**postfix_initrc_t** ドメインに移行します。

postfix_spool_t

このタイプは、**/var/spool/** ディレクトリーに保存されている Postfix ファイルに使用されます。



注記

Postfix 用のファイルとそのタイプの一覧を表示するには、次のコマンドを実行します。

```
~]$ grep postfix /etc/selinux/targeted/contexts/files/file_contexts
```

23.3. ブール値

SELinux は、サービスの実行に必要な最小アクセスレベルに基づいています。サービスはさまざまな方法で実行できます。そのため、サービスの実行方法を指定する必要があります。以下のブール値を使用して SELinux を設定します。

postfix_local_write_mail_spool

ブール値を有効にすると、Postfix はシステムのローカルメールスプールに書き込むことができるようになります。Postfix では、ローカルのスプールを使用する場合に、通常の操作でこのブール値を有効にする必要があります。

注記

SELinux ポリシーは継続的に開発されているため、上記のリストには、サービスに関連するすべてのブール値が常に含まれているとは限りません。これらを一覧表示するには、以下のコマンドを入力します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の説明を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

このコマンドが機能するには、**sepolicy** ユーティリティーを提供する追加の policycoreutils-devel パッケージが必要であることに注意してください。

23.4. 設定の例

23.4.1. SpamAssassin および Postfix

SpamAssasin は、オープンソースのメールフィルターで、受信したメールから迷惑メール (spam メッセージ) をフィルターにかける方法を提供します。[23]

Red Hat Enterprise Linux を使用する場合は、spamassassin パッケージにより SpamAssassin が提供されます。以下のコマンドを実行して、spamassassin パッケージがインストールされているかどうかを確認します。

```
~]$ rpm -q spamassassin
package spamassassin is not installed
```

インストールされていない場合は、root で **yum** ユーティリティーを使用してインストールします。

```
~]# yum install spamassassin
```

SpamAssassin は、Postfix などのメーラーと一緒に動作し、スパムフィルターリング機能を提供します。SpamAssassin がメールを効果的に傍受、分析、フィルターリングするには、ネットワークインターフェイスでリッスンする必要があります。SpamAssassin のデフォルトポートは TCP/783 ですが、これは変更できます。以下の例は、SELinux が、デフォルトで特定のポートへのアクセスのみを許可することで、SpamAssassin を補完する実際のデモンストレーションを示しています。次に、この例では、ポートを変更し、SpamAssassin がデフォルト以外のポートで動作するようにする方法を示します。

これは単なる例で、SELinux が SpamAssassin の単純な設定にどのように影響するかを示していることに注意してください。SpamAssassin に関する包括的なドキュメントは、このドキュメントの範囲外です。詳細は、公式の SpamAssassin のドキュメント を参照してください。この例では、spamassassin がインストールされていることを前提としています。ファイアウォールは、使用中のポートへのアクセスを許可するように設定されており、SELinux Targeted ポリシーが使用され、SELinux が Enforcing モードで実行されていることを前提としています。

手順23.1 デフォルト以外のポートで SpamAssassin を実行する

1. **semanage** ユーティリティーを root で実行し、SELinux により**spamd** デーモンがデフォルトでリッスンできるようになっているポートを表示します。

```
~]# semanage port -l | grep spamd
spamd_port_t  tcp 783
```

これは、TCP/783 が **spamd_port_t** で SpamAssassin の動作ポートとして定義されていることを示しています。

2. **/etc/sysconfig/spamassassin** 設定ファイルを編集して、サンプルポート TCP/10000 で SpamAssassin を起動するように変更します。

```
# Options to spamd
SPAMDOPTIONS="-d -p 10000 -c m5 -H"
```

この行では、SpamAssassin がポート 10000 で動作することを指定するようになりました。以下の例では、このソケットを開くことができるように SELinux ポリシーを変更する方法を説明します。

3. SpamAssassin を起動すると、以下のようなエラーメッセージが表示されます。

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

この出力は、SELinux がこのポートへのアクセスをブロックしたことを示しています。

4. 以下のような拒否メッセージが SELinux により記録されます。

```
SELinux is preventing the spamd (spamd_t) from binding to port 10000.
```

5. root で **semanage** を実行して SELinux ポリシーを変更し、SpamAssassin がサンプルポート (TCP/10000) で動作できるようにします。

```
~]# semanage port -a -t spamd_port_t -p tcp 10000
```

6. SpamAssassin が起動し、TCP ポート 10000 で動作していることを確認します。

```
~]# systemctl start spamassassin.service

~]# netstat -lnp | grep 10000
tcp 0 0 127.0.0.1:10000 0.0.0.0:* LISTEN 2224/spamd.pid
```

7. この時点で、**spamd** は、SELinux ポリシーでそのポートへのアクセスが許可されているため、TCP ポート 10000 で適切に動作しています。

---

[22] 詳細は、System Administrator's Guide の Postfix セクションを参照してください。

[23] 詳細は、System Administrator's Guide の Spam Filters セクションを参照してください。

第24章 DHCP

dhcpd デーモンは、Red Hat Enterprise Linux で使用され、クライアントのレイヤー 3 の TCP/IP 詳細を動的に配信して設定します。

dhcp は、DHCP サーバーおよび dhcpd デーモンを提供します。以下のコマンドを実行して、dhcp パッケージがインストールされているかどうかを確認します。

```
~]# rpm -q dhcp
package dhcp is not installed
```

インストールされていない場合は、root で yum ユーティリティーを使用してインストールします。

```
~]# yum install dhcp
```

24.1. DHCP および SELINUX

dhcpd が有効になると、既定で制限付きで実行されます。制限のあるプロセスは、独自のドメインで実行され、他の制限のあるプロセスから分離されます。制限のあるプロセスが攻撃者によって侵害された場合、SELinux ポリシーの設定に応じて、攻撃者のリソースへのアクセスと、攻撃者が行う可能性のある損害は制限されます。以下の例は、dhcpd と、そのドメインで実行している関連プロセスを示しています。この例では、dhcp がインストールされ、dhcpd サービスが開始していることを前提としています。

1. **getenforce コマンドを実行して、SELinux が Enforcing モードで実行されていることを確認します。**

   ```
   ~]$ getenforce
   Enforcing
   ```

   SELinux が Enforcing モードで実行されていると、このコマンドは Enforcing を返します。

2. **root で次のコマンドを入力して、dhcpd を起動します。**

   ```
   ~]# systemctl start dhcpd.service
   ```

   サービスが実行中であることを確認します。出力には以下の情報が含まれている必要があります (タイムスタンプのみは異なります)。

   ```
   ~]# systemctl status dhcpd.service
   dhcpd.service - DHCPv4 Server Daemon
     Loaded: loaded (/usr/lib/systemd/system/dhcpd.service; disabled)
     Active: active (running) since Mon 2013-08-05 11:49:07 CEST; 3h 20min ago
   ```

3. **次のコマンドを実行して、dhcpd プロセスを表示します。**

   ```
   ~]$ ps -eZ | grep dhcpd
   system_u:system_r:dhcpd_t:s0 5483 ?        00:00:00 dhcpd
   ```

   dhcpd プロセスに関連付けられた SELinux コンテキストは system_u:system_r:dhcpd_t:s0 です。

24.2. タイプ

高度なプロセス分離を提供するために、SELinux Targeted ポリシーで使用される主要なパーミッション制御方法は Type Enforcement です。すべてのファイルとプロセスにはタイプのラベルが付いています。タイプはプロセスの SELinux ドメインを定義し、ファイルの SELinux タイプを定義します。SELinux ポリシールールは、タイプにアクセスするドメインであるか、別のドメインにアクセスするドメインであるかにかかわらず、タイプが相互にアクセスする方法を定義します。アクセスは、それを許可する特定の SELinux ポリシールールが存在する場合にのみ許可されます。

DHCP では、以下のタイプが使用されます。

dhcp_etc_t

このタイプは、主に、設定ファイルを含む /etc ディレクトリーのファイルに使用されます。

dhcpd_var_run_t

このタイプは、/var/run/ ディレクトリーの dhcpd の PID ファイルに使用されます。

dhcpd_exec_t

このタイプは、DHCP 実行ファイルを dhcpd_t ドメインに移行するために使用されます。

dhcpd_initrc_exec_t

このタイプは、DHCP 実行ファイルを dhcpd_initrc_t ドメインに移行するために使用されます。

注記

dhcpd ファイルの一覧とそのタイプを表示するには、次のコマンドを実行します。

```
~]$ grep dhcp /etc/selinux/targeted/contexts/files/file_contexts
```

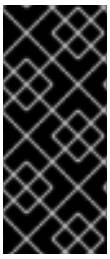
第25章 OPENSHIFT BY RED HAT

OpenShift by Red Hat は、開発者が Web アプリケーションを構築してデプロイできるようにする PaaS (Platform as a Service) です。OpenShift では、Java、Ruby、PHP など、さまざまなプログラミング言語およびフレームワークを使用できます。また、Eclipse の統合、JBoss Developer Studio、Jenkins など、アプリケーションのライフサイクルをサポートする統合開発者ツールも提供します。OpenShift は、オープンソースの ecosystem を使用して、モバイルアプリケーションやデータベースサービスなどに使用されるプラットフォームを提供します。[24]

Red Hat Enterprise Linux では、openshift-clients パッケージに OpenShift クライアントツールが同梱されています。次のコマンドを入力して、インストールされているかどうかを確認します。

```
~]$ rpm -q openshift-clients
package openshift-clients is not installed
```

openshift-clients パッケージがインストールされていない場合は、OpenShift Enterprise Client Tools Installation Guide および OpenShift Online Client Tools Installation Guide で OpenShift クライアントツールのインストールプロセスの詳細を確認してください。



重要

rhc パッケージは、以前は OpenShift クライアントツールを提供していました。最新のバージョンの OpenShift では、このパッケージが非推奨になり、Red Hat ではサポートされなくなりました。したがって、OpenShift バージョン 2 以降、rhc パッケージは、対応している OpenShift バージョンに使用される OpenShift クライアントツールを提供するopenshift-clients パッケージに置き換えられます。

25.1. OPENSHIFT および SELINUX

SELinux では、すべてのプロセスに SELinux ポリシーに従ってラベルが付けられているため、OpenShift を使用するアプリケーションよりもセキュリティー制御が向上します。したがって、SELinux は、同じノードで実行しているさまざまな歯車における悪意のある攻撃から OpenShift を保護します。

SELinux および OpenShift の詳細は、Dan Walsh's presentation を参照してください。

25.2. タイプ

高度なプロセス分離を提供するために、SELinux Targeted ポリシーで使用される主要なパーミッション制御方法は Type Enforcement です。すべてのファイルとプロセスにはタイプのラベルが付いています。タイプはプロセスの SELinux ドメインを定義し、ファイルの SELinux タイプを定義します。SELinux ポリシールールは、タイプにアクセスするドメインであるか、別のドメインにアクセスするドメインであるかにかかわらず、タイプが相互にアクセスする方法を定義します。アクセスは、それを許可する特定の SELinux ポリシールールが存在する場合にのみ許可されます。

以下のタイプは、OpenShift で使用されます。異なるタイプを使用すると、柔軟なアクセスを設定できます。

プロセスのタイプ

openshift_t

OpenShift プロセスは、**openshift_t** SELinux のタイプに関連付けられています。

実行ファイルのタイプ

openshift_cgroup_read_exec_t

SELinux では、このタイプのファイルを使用して、実行ファイルを **openshift_cgroup_read_t** ドメインに移行できます。

openshift_cron_exec_t

SELinux では、このタイプのファイルを使用して、実行ファイルを **openshift_cron_t** ドメインに移行できます。

openshift_initrc_exec_t

SELinux では、このタイプのファイルを使用して、実行ファイルを **openshift_initrc_t** ドメインに移行できます。

書き込み可能なタイプ

openshift_cgroup_read_tmp_t

このタイプでは、**/tmp** ディレクトリー内の OpenShift 制御グループ (cgroup) による一時ファイルの読み取りとアクセスが可能になります。

openshift_cron_tmp_t

このタイプでは、OpenShift cron ジョブの一時ファイルを **/tmp** に保存できます。

openshift_initrc_tmp_t

このタイプでは、OpenShift **initrc** 一時ファイルを **/tmp** に保存できます。

openshift_log_t

このタイプのファイルは、OpenShift ログデータとして扱われ、通常は **/var/log/** ディレクトリーに保存されます。

openshift_rw_file_t

OpenShift には、このタイプのラベルが付いたファイルの読み取りと書き込みのパーミッションがあります。

openshift_tmp_t

このタイプは、OpenShift 一時ファイルを **/tmp** に保存するために使用されます。

openshift_tmpfs_t

このタイプを使用すると、OpenShift データを tmpfs ファイルシステムに保存できます。

openshift_var_lib_t

このタイプを使用すると、**/var/lib/** ディレクトリーに OpenShift ファイルを保存できます。

openshift_var_run_t

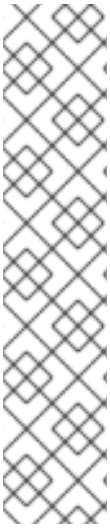
このタイプを使用すると、**/run/** ディレクトリーまたは **/var/run/** ディレクトリーに OpenShift ファイルを保存できます。

25.3. ブール値

SELinux は、サービスの実行に必要な最小アクセスレベルに基づいています。サービスはさまざまな方法で実行できます。そのため、サービスの実行方法を指定する必要があります。以下のブール値を使用して SELinux を設定します。

openshift_use_nfs

ブール値を有効にすると、NFS 共有に OpenShift をインストールできるようになります。



注記

SELinux ポリシーは継続的に開発されているため、上記のリストには、サービスに関連するすべてのブール値が常に含まれているとは限りません。これらを一覧表示するには、以下のコマンドを入力します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の説明を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

このコマンドが機能するには、**sepolicy** ユーティリティーを提供する追加の policycoreutils-devel パッケージが必要であることに注意してください。

25.4. 設定の例

25.4.1. デフォルトの OpenShift ディレクトリーの変更

デフォルトでは、OpenShift はそのデーターを **/var/lib/openshift/** ディレクトリーに保存します。このディレクトリーには **openshift_var_lib_t** SELinux のタイプのラベルが付いています。OpenShift が別のディレクトリーにデータを保存できるようにするには、適切な SELinux コンテキストで、新しいディレクトリーにラベルを付けます。

以下の手順では、データを**/srv/openshift/** に保存するために、デフォルトの OpenShift ディレクトリーを変更する方法を示しています。

手順25.1データを保存するためのデフォルトの OpenShift ディレクトリーの変更

1. root で、**/srv** ディレクトリーに新しい**openshift/** ディレクトリーを作成します。新しいディレクトリーには、**var_t** のタイプにラベルが付けられます。

```
~]# mkdir /srv/openshift
```

```
~]$ ls -Zd /srv/openshift
drwxr-xr-x. root root unconfined_u:object_r:var_t:s0   openshift/
```

2. root で、**semanage** ユーティリティーを使用して、**/srv/openshift/** を適切な SELinux コンテキストにマッピングします。

```
~]# semanage fcontext -a -e /var/lib/openshift /srv/openshift
```

3. **次に、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/openshift
```

4. **これで、/srv/openshift/ ディレクトリーに、正しいopenshift_var_lib_t タイプのラベルが付けられます。**

```
~]$ ls -Zd /srv/openshift
drwxr-xr-x. root root unconfined_u:object_r:openshift_var_lib_t:s0   openshift/
```

[24] OpenShift の詳細は、Product Documentation for OpenShift Container Platform および Product Documentation for OpenShift Online を参照してください。

第26章 ID 管理

Identity Management (IdM) は、標準定義の一般的なネットワークサービス (PAM、LDAP、Kerberos、DNS、NTP、証明書サービスなど) で統合環境を提供します。IdM を使用すると、Red Hat Enterprise Linux システムがドメインコントローラーとして機能できます。[25]

Red Hat Enterprise Linux では、ipa-server パッケージにより IdM サーバーが提供されます。以下のコマンドを実行して、ipa-server パッケージがインストールされているかどうかを確認します。

```
~]$ rpm -q ipa-server
package ipa-server is not installed
```

インストールされていない場合は、root ユーザーになり、以下のコマンドを入力してインストールします。

```
~]# yum install ipa-server
```

26.1. ID 管理と SELINUX

Identity Management では、IdM アクセス権に SELinux コンテキストを指定できるように、ホストごとに設定した SELinux ロールに、IdM ユーザーをマッピングできます。ユーザーログインプロセス中に、**SSSD** (System Security Services Daemon) は特定の IdM ユーザーに定義したアクセス権限をクエリーします。そして、**pam_selinux** モジュールは、IdM のアクセス権 (**guest_u:guest_r:guest_t:s0** など) に従って、適切な SELinux コンテキストでユーザープロセスを起動するように、カーネルにリクエストを送信します。

Identity Management および SELinux の詳細は、Red Hat Enterprise Linux 7 の Linux Domain, Identity, Authentication, and Policy Guide を参照してください。

26.1.1. Active Directory ドメインへの信頼

以前のバージョンの Red Hat Enterprise Linux では、Identity Management で **WinSync** ユーティリティーを使用して、Active Directory (AD) ドメインのユーザーが IdM ドメインに保存されているデータにアクセスできるようにしました。これを行うには、**WinSync** が、AD サーバーからローカルサーバーにユーザーとグループのデータを複製し、データの同期を維持する必要がありました。

Red Hat Enterprise Linux 7 では、**SSSD** デーモンが AD と連携するように拡張され、ユーザーは IdM ドメインと AD ドメイン間に信頼関係を作成できるようになりました。ユーザーおよびグループのデータは、AD サーバーから直接読み取られます。また、AD ドメインおよび IdM ドメイン間でのシングルサインオン (SSO) 認証を可能にする Kerberos レルム間の信頼も提供されます。SSO を設定すると、AD ドメインのユーザーは、パスワードを必要とせずに、IdM ドメインに保存されている Kerberos で保護されているデータにアクセスできます。

この機能は、デフォルトではインストールされません。これを使用するには、追加の ipa-server-trust-ad パッケージをインストールします。

26.2. 設定の例

26.2.1. IdM ユーザーへの SELinux ユーザーのマッピング

以下の手順では、新しい SELinux マッピングを作成する方法と、このマッピングに新しい IdM ユーザーを追加する方法を示します。

手順26.1 SELinux マッピングにユーザーを追加する方法

1. **新しい SELinux マッピングを作成するには、以下のコマンドを実行します。SELinux_mapping は新しい SELinux マッピングの名前で、--selinuxuser オプションは特定の SELinux ユーザーを指定します。**

```
~]$ ipa selinuxusermap-add SELinux_mapping --selinuxuser=staff_u:s0-s0:c0.c1023
```

2. **次のコマンドを実行して、tuser のユーザー名を持つ IdM ユーザーを SELinux マッピングに追加します。**

```
~]$ ipa selinuxusermap-add-user --users=tuser SELinux_mapping
```

3. **ipaclient.example.com という名前の新しいホストを SELinux マッピングに追加するには、次のコマンドを実行します。**

```
~]$ ipa selinuxusermap-add-host --hosts=ipaclient.example.com SELinux_mapping
```

4. **tuser ユーザーは、*ipaclient.example.com* ホストコンピューターにログインすると、staff_u:s0-s0:c0.c1023 ラベルを取得します。**

```
[tuser@ipa-client]$ id -Z
staff_u:staff_r:staff_t:s0-s0:c0.c1023
```

---

[25] Identity Management の詳細は、Red Hat Enterprise Linux 7 の Linux Domain, Identity, Authentication, and Policy Guide を参照してください。

第27章 RED HAT GLUSTER STORAGE

Red Hat Gluster Storage は、企業向けに柔軟で低価格な構造化されていないデータストレージを提供します。Gluster の主要な設定要素である *GlusterFS* は、スタック可能なユーザー空間設計に基づいており、さまざまなストレージサーバーをネットワーク経由で集約し、1つの大きな並列ネットワークファイルシステムに相互接続します。POSIX に互換性のある GlusterFS サーバーは、XFS ファイルシステム形式を使用してディスクにデータを保存し、これらのサーバーは NFS や CIFS を含む業界標準のアクセスプロトコルを使用してアクセスすることができます。

詳細は、ガイドの『Product Documentation for Red Hat Gluster Storage』コレクションを参照してください。

glusterfs-server は、Red Hat Gluster Storage を提供します。インストールプロセスの詳細は、Red Hat Gluster Storage の『Installation Guide』を参照してください。

27.1. RED HAT GLUSTER STORAGE および SELINUX

これを有効にすると、SELinux は、Red Hat Gluster Storage の一部として、**glusterd** (GlusterFS 管理サービス) プロセスおよび **glusterfsd** (NFS サーバー) プロセスに柔軟な必須アクセス制御を提供することで、追加のセキュリティー層として機能します。このようなプロセスでは、**glusterd_t** SELinux の種類に制限されずに、高度なプロセスの分離が行われています。

27.2. タイプ

高度なプロセス分離を提供するために、SELinux Targeted ポリシーで使用される主要なパーミッション制御方法は Type Enforcement です。すべてのファイルとプロセスにはタイプのラベルが付いています。タイプはプロセスの SELinux ドメインを定義し、ファイルの SELinux タイプを定義します。SELinux ポリシールールは、タイプにアクセスするドメインであるか、別のドメインにアクセスするドメインであるかにかかわらず、タイプが相互にアクセスする方法を定義します。アクセスは、それを許可する特定の SELinux ポリシールールが存在する場合にのみ許可されます。

以下のタイプは、Red Hat Gluster Storage で使用されます。異なるタイプを使用すると、柔軟なアクセスを設定できます。

プロセスのタイプ

glusterd_t

Gluster プロセスは、**glusterd_t** SELinux のタイプに関連付けられています。

実行ファイルのタイプ

glusterd_initrc_exec_t

Gluster init スクリプトファイルの SELinux 固有のスクリプトタイプのコンテキスト。

glusterd_exec_t

Gluster 実行ファイルに対する SELinux 固有の実行タイプのコンテキスト。

ポートタイプ

gluster_port_t

このタイプは、glusterd に対して定義されます。初期設定では、glusterd は 204007-24027 および 38465-38469 の TCP ポートを使用します。

ファイルコンテキスト

glusterd_brick_t

このタイプは、glusterd ブリックデータとして脅威を受けるファイルに使用されます。

glusterd_conf_t

このタイプは、glusterd 設定データに関連付けられます。通常、/etc ディレクトリーに保存されます。

glusterd_log_t

このタイプのファイルは、glusterdログデータとして扱われ、通常は /var/log/ ディレクトリーに保存されます。

glusterd_tmp_t

このタイプは、glusterd一時ファイルを /tmp ディレクトリーに保存するために使用されます。

glusterd_var_lib_t

このタイプを使用すると、glusterdファイルを /var/lib/ ディレクトリーに保存できます。

glusterd_var_run_t

このタイプでは、/run/ ディレクトリーまたは /var/run/ ディレクトリーにglusterdファイルを保存できます。

27.3. ブール値

SELinux は、サービスの実行に必要な最小アクセスレベルに基づいています。サービスはさまざまな方法で実行できます。そのため、サービスの実行方法を指定する必要があります。以下のブール値を使用して SELinux を設定します。

gluster_export_all_ro

ブール値を有効にすると、glusterfsd はファイルとディレクトリーを読み取り専用として共有できます。このブール値は、デフォルトでは無効になっています。

gluster_export_all_rw

ブール値を有効にすると、glusterfsd は読み取りアクセスと書き込みアクセスでファイルとディレクトリーを共有できるようになります。このブール値はデフォルトで有効になっています。

gluster_anon_write

ブール値を有効にすると、glusterfsd は、public_content_rw_t SELinux タイプでラベル付けされた公開ファイルを変更できます。

注記

SELinux ポリシーは継続的に開発されているため、上記のリストには、サービスに関連するすべてのブール値が常に含まれているとは限りません。これらを一覧表示するには、以下のコマンドを入力します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の説明を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

このコマンドが機能するには、**sepolicy** ユーティリティーを提供する追加の policycoreutils-devel パッケージが必要であることに注意してください。

27.4. 設定の例

27.4.1. Gluster ブリックのラベル付け

Gluster ブリックは、信頼できるストレージプールのサーバー上のエクスポートディレクトリーです。ブリックに正しい SELinux コンテキストのラベルが付いていないと、**glusterd_brick_t** により、SELinux は特定のファイルアクセス操作を拒否し、さまざまな AVC メッセージを生成します。

以下の手順では、Gluster ブリックに正しい SELinux コンテキストのラベルを付ける方法を説明します。この手順では、事前に論理ボリューム (**/dev/rhgs/gluster** など) を作成し、フォーマットし、Gluster ブリックとして使用することを前提としています。

Gluster ブリックの詳細は、Red Hat Gluster Storage の『Administration Guide』の『Red Hat Gluster Storage Volumes』の章を参照してください。

手順27.1 Gluster ブリックにラベルを付ける方法

1. 以前にフォーマットされた論理ボリュームをマウントするディレクトリーを作成します。以下に例を示します。

   ```
   ~]# mkdir /mnt/brick1
   ```

2. 論理ボリューム (この場合は **/dev/vg-group/gluster**) を、前の手順で作成した **/mnt/brick1/** ディレクトリーにマウントします。

   ```
   ~]# mount /dev/vg-group/gluster /mnt/brick1/
   ```

   mount コマンドはデバイスを一時的にマウントすることに注意してください。デバイスを永続的にマウントするには、以下のようなエントリーを **/etc/fstab** ファイルに追加します。

   ```
   /dev/vg-group/gluster    /mnt/brick1  xfs rw,inode64,noatime,nouuid      1 2
   ```

   詳細は、fstab(5) の man ページを参照してください。

3. **/mnt/brick1/** の SELinux コンテキストを確認します。

   ```
   ~]$ ls -lZd /mnt/brick1/
   drwxr-xr-x. root root system_u:object_r:unlabeled_t:s0 /mnt/brick1/
   ```

ディレクトリーには、**unlabeled_t** SELinux タイプのラベルが付けられます。

4. **/mnt/brick1/** の SELinux タイプを、**glusterd_brick_t** SELinux タイプに変更します。

```
~]# semanage fcontext -a -t glusterd_brick_t "/mnt/brick1(/.*)?"
```

5. **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 -Rv /mnt/brick1
```

6. 最後に、コンテキストが正常に変更されたことを確認します。

```
~]$ ls -lZd /mnt/brick1
drwxr-xr-x. root root system_u:object_r:glusterd_brick_t:s0 /mnt/brick1/
```

第28章 REFERENCES

以下の参考資料は、SELinux に関連する追加情報のポインターですが、本書では扱いません。SELinux の迅速な開発により、本ガイドの一部は Red Hat Enterprise Linux の特定のリリースにのみ適用されます。

書籍

SELinux by Example

Mayer, MacMillan, and Caplan

Prentice Hall, 2007

SELinux: NSA's Open Source Security Enhanced Linux

Bill McCarty

O'Reilly Media Inc., 2004

チュートリアルおよびヘルプ

Russell Coker のチュートリアルと講演

http://www.coker.com.au/selinux/talks/ibmtu-2004/

Dan Walsh のジャーナル

http://danwalsh.livejournal.com/

Red Hat ナレッジベース

https://access.redhat.com/site/

全般情報

NSA SELinux のメイン Web サイト

https://www.nsa.gov/What-We-Do/Research/SELinux/

NSA SELinux FAQ

https://www.nsa.gov/What-We-Do/Research/SELinux/FAQs/

メーリングリスト

NSA SELinux メーリングリスト

https://www.nsa.gov/What-We-Do/Research/SELinux/Mailing-List/

Fedora SELinux メーリングリスト

http://www.redhat.com/mailman/listinfo/fedora-selinux-list

コミュニティー

SELinux Project Wiki

http://selinuxproject.org/page/Main_Page

SELinux コミュニティーページ

http://selinux.sourceforge.net/

IRC

irc.freenode.net, #selinux

付録A 更新履歴

改訂 0.3-06 Fri Aug 9 2019 Mirek Jahoda
7.7 GA 公開用バージョン

改訂 0.3-05 Sat Oct 20 2018 Mirek Jahoda
7.6 GA リリースのバージョン

改訂 0.3-03 Tue Apr 3 2018 Mirek Jahoda
7.5 GA 公開用バージョン

改訂 0.3-01 Thu Jul 13 2017 Mirek Jahoda
7.4 GA 公開用バージョン

改訂 0.2-18 Wed Nov 2 2016 Mirek Jahoda
7.3 GA リリースのバージョン

改訂 0.2-11 Sun Jun 26 2016 Mirek Jahoda
修正を含む非同期リリース

改訂 0.2-10 Sun Feb 14 2016 Robert Krátký
修正を含む非同期リリース

改訂 0.2-9 Thu Dec 10 2015 Barbora Ančincová
Red Hat Gluster Storage の章を追加

改訂 0.2-8 Thu Nov 11 2015 Barbora Ančincová
Red Hat Enterprise Linux 7.2 GA リリース用のガイド

改訂 0.2-7 Thu Aug 13 2015 Barbora Ančincová
Red Hat Enterprise Linux 7.2 Beta リリース用のガイド

改訂 0.2-6 Wed Feb 18 2015 Barbora Ančincová
Red Hat Enterprise Linux 7.1 GA リリース用のガイド

改訂 0.2-5 Fri Dec 05 2014 Barbora Ančincová
Red Hat カスタマーポータルでの並べ替えのための更新

改訂 0.2-4 Thu Dec 04 2014 Barbora Ančincová
Red Hat Enterprise Linux 7.1 Beta リリース用のガイド

改訂 0.1-41 Tue May 20 2014 Tomáš Čapek
スタイル変更に伴う再構築

改訂 0.1-1 Tue Jan 17 2013 Tomáš Čapek
Red Hat Enterprise Linux 7 の本書の初版作成