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

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

Barbora Ančincová　　Tomáš Čapek

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

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

Barbora Ančincová
Red Hat Customer Content Services
bancinco@redhat.com

Tomáš Čapek
Red Hat Customer Content Services
tcapek@redhat.com

法律上の通知

概要

本書は 2 部構成になっています。前半の SELinux では、SELinux 機能の基本と原則について説明しています。後半の 制限のあるサービスの管理では、様々なサービスの設定に関する実際のタスクにフォーカスしています。

目次

パート I. SELinux

第1章 はじめに

SELinux (Security-Enhanced Linux) は Linux カーネルに *MAC (Mandatory Access Control)* を実装するもので、標準の *Discretionary Access Controls (DAC: 任意アクセス制御)* を確認した後で許可される操作をチェックします。これは米国国家安全保障局 (National Security Agency) が開発したもので、定義されたポリシーを基に Linux システム内のファイルやプロセスおよびその他のアクションにルールを強制できます。

SELinux を使用すると、ファイル (ディレクトリーやデバイスを含む) はオブジェクトとして参照されます。ユーザーによるコマンドや Mozilla Firefox アプリケーションなどの実行といったプロセスは、サブジェクトとして参照されます。ほとんどのオペレーティングシステムでは DAC (任意アクセス制御) が使われており、これはサブジェクトとオブジェクト、およびサブジェクト同士の情報交換方法を制御するものです。DAC を使用するオペレーティングシステムでは、ユーザーは自身が所有するファイルのパーミッション (オブジェクト) を制御します。例えば、Linux オペレーティングシステム上では、ユーザーは自身のホームディレクトリーを全ユーザー読み取り可能にすることができ、新たな保護を加えることなく、ユーザーおよびプロセス (サブジェクト) に機密性の高い可能性のある情報へのアクセスを与えることになります。

DAC メカニズムにのみ依存することは、強力なシステムセキュリティとしては基本的に不十分です。DAC のアクセスに関する決定は、ユーザー ID と所有権にのみ基づいており、ユーザーのロールやプログラムの機能および信頼性、データの機密性および整合性といったその他のセキュリティ関連情報を考慮していません。各ユーザーは通常、自身のファイルに対して完全な裁量権を有しており、システム全体にセキュリティポリシーを強制することが困難になっています。さらに、ユーザーが実行するプログラムはすべて、そのユーザーに許可された全パーミッションを継承していて、ユーザーのファイルへのアクセスを変更することは自由にできます。このため、悪意のあるソフトウェアに対する保護は最低限のものしか与えられていません。多くのシステムサービスおよび権限が与えられているプログラムは、要件をはるかに超える雑な権限で実行されているため、プログラムのうちのどれかに欠点があると悪用されて、システムへのさらなるアクセスが取得される可能性があります。 [1]

以下は、SELinux を実行していない Linux オペレーティングシステムで使われているパーミッションの例です。システムによっては、パーミッションおよび出力はこの例とは多少異なる場合があります。ファイルパーミッションを表示するには、以下のコマンドを実行します。

```
~]$ ls -l file1
-rwxrw-r-- 1 user1 group1 0 2009-08-30 11:03 file1
```

この例では、最初の 3 つのパーミッション **rwx** が、Linux **user1** ユーザー (この例では所有者) の **file1** へのアクセスを制御します。次の 3 つのパーミッション **rw-** は、Linux **group1** グループの **file1** へのアクセスを制御します。最後の 3 つのパーミッション **r--** は、その他のユーザーの **file1** へのアクセスを制御します。その他のユーザーには、すべてのユーザーとプロセスが含まれます。

SELinux を使用すると Linux カーネルに MAC (強制アクセス制御) が追加され、Red Hat Enterprise Linux ではデフォルトで有効になります。汎用の MAC アーキテクチャーは、各種のセキュリティ関連情報を含むラベルを決定基準として、管理者が設定したセキュリティポリシーをシステム内の全プロセスおよびファイルに対して強制する能力を必要とします。これが適切に実装されると、システム自体が的確に自己防御され、アプリケーションを改ざんから保護し、迂回することでアプリケーションの安全性に必須のサポートを提供します。MAC ではアプリケーション同士が確実に分離されるため、信頼性の低いアプリケーションでも安全に実行することができます。プロセス実行に関する特権を制限する機能により、アプリケーションやシステムサービス内の脆弱性を悪用することで発生する可能性のある被害の範囲を限定することができます。限られた権限しか持たない正規ユーザーだけでなく、権限を与えられたユーザーが不正なアプリケーションを知らずに実行してしまった場合でも、MAC で情報を保護することができます。 [2]

以下は、SELinux を実行する Linux オペレーティングシステム上でプロセス、Linux ユーザー、ファイルに使用されるセキュリティ関連の情報を含むラベルの例です。この情報は SELinux **コンテキスト**と呼ばれ、以下のコマンドを実行すると表示できます。

```
~]$ ls -Z file1
-rwxrw-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**) を示しています。この情報は、アクセス制限の決定に使用されます。DAC では、アクセスは Linux ユーザー ID とグループ ID のみ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** でチェックされることを覚えておくことが重要です。DAC ルールが最初に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。

注記

SELinux を実行する Linux オペレーティングシステム上には、Linux ユーザーと SELinux ユーザーがいます。SELinux ユーザーは、SELinux ポリシーの一部です。Linux ユーザーは SELinux ユーザーにマッピングされています。混乱を避けるために本ガイドでは、**Linux ユーザー**と **SELinux ユーザー**という用語で区別します。

1.1. SELinux を実行する利点

- プロセスおよびファイルがすべて、タイプでラベル付けられます。タイプはプロセスのドメインを定義し、ファイルのタイプもあります。プロセスはそれぞれのドメインで実行することで互いに分離しており、SELinux ポリシールールはプロセスがファイルと対話する方法と、プロセス同士が対話する方法を定義します。アクセスは、明確にアクセスを許可するSELinux ポリシールールが存在する場合にのみ、許可されます。
- 粒度の細かいアクセス制御。ユーザーの判断に任され、Linux ユーザーおよびグループ ID に基づいて制御されている従来の UNIX パーミッションにとどまらず、SELinux のアクセス決定は、SELinux ユーザーやロール、タイプ、さらにはオプションとしてレベルなどの利用可能なすべての情報に基づいて判断されます。
- SELinux ポリシーは管理者が定義し、システム全体にわたって強制されるもので、ユーザーの判断で設定されるものではありません。
- 権限のあるエスカレーション攻撃に対する脆弱性が低減されます。プロセスはドメイン内で実行されるので、それぞれが分離されます。SELinux ポリシールールは、プロセスがファイルおよび他のプロセスにアクセスする方法を定義します。あるプロセスが危険にさらされても、攻撃者がアクセスできるの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権を持つ設定になっているファイルのみになります。例えば、Apache HTTP サーバーが危険にさらされても、特定の SELinux ポリシールールで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取る許可が追加されているかそのような設定になっていなければ、攻撃者はそのプロセスを使ってホームディレクトリーにあるファイルを読み取ることはできません。
- SELinux を使用すると、データの秘密性と整合性が強化され、プロセスを信頼できない入力から保護します。

ただし、SELinux は以下のものではありません。

- アンチウィルスソフトウェア
- パスワードやファイアウォール、その他のセキュリティシステムなどの代わりとなるもの
- オールインワンのセキュリティソリューション

SELinux は既存のセキュリティソリューションを強化するように設計されており、これらに代わるものではありません。SELinux の実行中でも、ソフトウェアを最新のもの更新したり、分かりにくいパスワードやファイアウォールを使うなどのすぐれたセキュリティ対策を継続することが重要です。

1.2. SELinux の使用例

以下では、SELinux によるセキュリティ強化の具体例を示しています。

- デフォルトのアクションは拒否になります。ファイルを開くプロセスなどでアクセスを許可する SELinux ポリシールールがない場合は、アクセスが拒否されます。
- SELinux は Linux ユーザーを制限できます。SELinux ポリシーには、制限のある SELinux ユーザーが多く存在します。Linux ユーザーを制限のある SELinux ユーザーにマッピングして、これらのユーザーに適用されているセキュリティルールとメカニズムを活用することができます。例えば、ある Linux ユーザーを SELinux **user_u** ユーザーにマッピングすると、この Linux ユーザーは**sudo** や **su** といったセットユーザー ID (setuid) アプリケーションを (実行可能と設定されている場合以外は) 実行できず、ホームディレクトリーにあるファイルやアプリケーションも実行できません。この設定では、ユーザーが悪意のあるファイルを自身のホームディレクトリーから実行することを防ぎます。
- プロセス分離が使用されます。プロセスはそれぞれのドメインで実行されるので、他のプロセスが使用するファイルやそれらのプロセスに別のプロセスがアクセスすることを防ぎます。例えば SELinux 実行中の場合、攻撃者が Samba サーバーに侵入しても、この Samba サーバーを攻撃者のベクターとして利用して、MariaDB が使用するデータベースなどの他のプロセスが使用するファイルの読み取りや書込みはできません。
- SELinux は、設定ミスによる破損の制限に役立ちます。ドメインネームシステム (DNS) サーバーは、ゾーン転送と呼ばれる DNS サーバー間での情報複製を頻繁に行います。攻撃者は、ゾーン転送を使って、DNS サーバーを偽の情報で更新できます。Red Hat Enterprise Linux で BIND (Berkeley Internet Name Domain) を DNS サーバーとして稼働している場合、ゾーン転送を実行できるサーバーの制限を管理者が忘れても、デフォルトの SELinux ポリシーは、ゾーンファイル [3] が BIND **named** デーモン自体や他のプロセスによってゾーン転送経由で更新されることを防ぎます。
- SELinux についてのバックグラウンド情報と SELinux が防いだ多種のエクスプロイトについての情報は、NetworkWorld.com の記事、A seatbelt for server software: SELinux blocks real-world exploits [4] を参照してください。

1.3. SELinux アーキテクチャー

SELinux は、Linux カーネルに組み込まれた Linux セキュリティモジュールです。SELinux は、読み込み可能なポリシールールで稼働します。プロセスがファイルを開こうとするといったセキュリティ関連のアクセスが発生すると、その操作は SELinux がカーネルで傍受します。SELinux ポリシールールがこの操作を許可するとそのまま続けられますが、許可しないとこの操作は遮断され、プロセスはエラーを受け取ります。

アクセスを許可する/許可しないといった SELinux の決定は、キャッシュされます。このキャッシュは、AVC (アクセスベクターキャッシュ) と呼ばれます。このキャッシュされた決定を使用すると、SELinux ポリシールールをチェックする頻度が減り、その結果、パフォーマンスが向上します。DAC ルールが最初にアクセスを拒否すると SELinux ポリシールールは効果がないことに留意してください。

1.4. SELinux モード

SELinux には以下の 3 つのモードがあります。

- Enforcing: SELinux ポリシーが強制されます。SELinux は SELinux ポリシールールに基づいてアクセスを拒否します。
- Permissive: 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、enforcing モードでは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。
- Disabled: SELinux が無効化されます。DAC ルールのみが使用されます。

enforcing モードと permissive モードの切り替えには、**setenforce** ユーティリティーを使います。**setenforce** を使った変更は、再起動されると維持されません。enforcing モードへの変更は、Linux root ユーザーで **setenforce 1** コマンドを実行します。permissive モードへの変更は、**setenforce 0** コマンドを実行します。現在の SELinux モードを表示するには、 **getenforce** ユーティリティーを実行します。

永続的なモード変更は、「SELinux の有効化および無効化」で説明されています。

1.5. Red Hat Enterprise Linux 7 での新機能

本セクションでは、Red Hat Enterprise Linux 7 で導入された SELinux の新機能の概要を説明します。以下で説明する各機能については、本ガイド内で詳細に説明されている文書へのリンクがあります。

File Name Transition

これまでは、あるディレクトリー内でそのディレクトリーのラベル以外のラベル付けをした特定のオブジェクトクラスを作成すると、間違ったラベル付けに関連した問題が発生していました。*file name transition* 機能を使うと、ポリシー作成者はルール作成時にファイル名を指定できます。これで以下の状態を記述するルールを作成することが可能になります。**A_t** のラベルの付いたプロセスが **B_t** のラベルが付いたディレクトリー内で特定のオブジェクトクラスを作成し、この特定のオブジェクトクラスを **objectname** と命名すると、これは **C_t** のラベルが付けられます。このメカニズムは間違ったラベルによる問題を減らし、システムプロセスに関してより細かい制御をもたらします。

File Name Transition に関する詳細は、「File Name Transition (ファイル名の移行)」を参照してください。

OpenShift

OpenShift Platform as a Service (PaaS) が SELinux で制限されるサービスに追加されました。OpenShift は、開発者やチームがアプリケーションを構築、テスト、導入、実行できるクラウドにおけるプラットホームを提供します。OpenShift における SELinux 設定についての詳細は、25章*OpenShift* を参照してください。

ID 管理

ID 管理 (IdM) が SELinux で制限されるサービスに追加されました。IdM は、コンピューターネットワークのセキュリティ面の管理に必要となるユーザーやグループ、ホスト、その他のオブジェクトに関するデータを保存することで、集中化された認証、承認、およびアカウント情報を提供します。IdM における SELinux 設定についての詳細は、26章*ID 管理* を参照してください。

ptrace() の無効化

新たなブール値 **deny_ptrace** が SELinux のブール値に追加されました。このブール値を使うと、ユーザーは **ptrace()** システムコールを無効にできます。**deny_ptrace** についての詳細は、「ptrace() の無効化」 を参照してください。

新たな制限のあるドメイン

Red Hat Enterprise Linux 7 では、多くの製品やサービスが制限のある別個のドメインを使用します。たとえば、SELinux は以下をサポートするようになっています。

OpenStack

OpenStack はクラウドコンピューティングプラットホームで、Swift、Nova、または Glance などの様々なコンポーネントで構成されます。**swift_t**、**nova_***、および **glance_*** ドメインが SELinux ポリシーに追加されました。

OpenShift

OpenShift はクラウドコンピューティングプラットホームです。**openshift_*** ドメインが SELinux ポリシーに追加されました。

realmd

realmd ユーティリティーは DBus サービスで、Active Directory や ID 管理などの領域における発見および登録を管理します。**realmd_t** ドメインが SELinux ポリシーに追加されました。

glusterd

glusterd ユーティリティーはボリューム管理のデーモンです。**glusterd_t** ドメインが SELinux ポリシーに追加されました。

stapserver

stapserver ユーティリティーはインストルメンテーションシステムサーバーを提供します。**stapserver_t** ドメインが SELinux ポリシーに追加されました。

OpenLMI

OpenLMI (Open Linux Management Infrastructure) は、Linux システムの管理用に共通インフラストラクチャーを提供します。**pegasus_openlmi_*** ドメインが SELinux ポリシーに追加されました。

制限のあるドメインについての詳細は、3章ターゲットポリシーを参照してください。サービスのマニュアルページについての詳細は、「サービスの man ページ」を参照してください。

縮小ポリシー

これまでは、ポリシールールの指定には M4 マクロが使用されており、SELinux ポリシーはカーネルメモリーを大量に消費して、起動時のポリシーの読み込みに多くの時間がかかっていました。Red Hat Enterprise Linux 7 では、マクロの代わりにタイプのグループ化を可能にする属性が使われており、SELinux ポリシーの簡素化とそのサイズの縮小を図っています。たとえば、主にクラスタとアンチウイルスサービスでは、SELinux ドメインがいくつか結合されました。

- **amavis.pp** と **clamav.pp** の各モジュールは **antivirus.pp** モジュールに統合され、**antivirus_t** タイプにエイリアス化されました。

```
typealias antivirus_t alias { amavis_t clamd_t clamscan_t
freshclam_t }
```

- **pacemaker.pp**、**corosync.pp**、および **aisexec.pp** モジュールは **rhcs.pp** モジュールに統合され、**cluster_t** タイプにエイリアス化されました。

```
typealias cluster_t alias { aisexec_t corosync_t pacemaker_t
rgmanager_t }
```

SELinux ポリシーのサイズは 80 パーセントに縮小し、読み込み時間は大幅に短縮され、ブート時のカーネルメモリー消費量も抑えられています。

事前作成のポリシー

selinux-policy パッケージに事前作成のポリシーが含まれるようになりました。このポリシーにユーザーがローカルのカスタマイズを加えた時のみ、SELinux ポリシーはインストール後に再コンパイルされます。この変更により、パッケージのインストールプロセスが迅速になり、メモリー使用量が抑えられます。

Secure Linux コンテナー

Linux コンテナーは低レベルの仮想化機能で、これを使うことでシステム上で同時に同一サービスの複数コピーを実行することが可能になります。セキュアな Linux コンテナーを作成する際は、**virt-sandbox-service** ユーティリティーが systemd ユニットファイルを使用してコンテナー内にサービスを適切に設定します。その後に、SELinux ポリシーで指定された SELinux コンテキストで **libvirt** ライブラリーがこのコンテナーを開始します。

セキュアな Linux コンテナーの詳細は、8章*Secure Linux コンテナー*を参照してください。

sepolicy スイート

sepolicy ユーティリティーは、インストール済みの SELinux ポリシーをクエリする機能のスイートを提供します。このスイートを使うと、移行レポートや man ページ、さらには新ポリシーのモジュールを作成できるようになり、ユーザーは SELinux ポリシーへのアクセスが容易になり、理解が深まります。

sepolicy 検査スイートの詳細は、5章***sepolicy* スイート** を参照してください。

サムネイル保護

これまでは、画面がロックされている際にサムネイルドライバーはロックされていませんでした。この結果、パスワードを入力することなく、サムネイルドライバーコードを使って画面のロックを迂回することが可能になります。新たな SELinux ポリシーはこの種の攻撃を防ぎ、システムのセキュリティを高めます。サムネイル保護についての詳細は、「サムネイル保護」を参照してください。

Permissive ドメインの無効化

すべての permissive ドメイン宣言は新たな **permissivedomains.pp** モジュールにリファクタリングされています。その結果、**semodule** ユーティリティーを使ってすべての permissive ドメインを無効化することが可能になっています。このモジュールについての詳細は、「Permissive ドメインを無効にする」を参照してください。

ラベル付き NFS

NFS クライアントとサーバー間で SELinux ラベルの受け渡しがサポートされるようになりました。この新たなラベル付き NFS 機能を使うと、ユーザーは単一 NFS ボリューム上で様々な SELinux ラベルを区別できるようになり、NFS ボリューム にアクセスする制限のあるドメインのセキュリティ制御が向上します。

ラベル付き NFS についての詳細は、「NFS と SELinux」を参照してください。

SELinux Systemd アクセス制御

Red Hat Enterprise Linux 7 では、システムサービスを開始、停止するすべてのコールを **systemd** デーモンが管理します。**systemd** 方式のコールは SELinux アクセスチェックにマッピングされ、サービスが自動または手動で開始された際にこれが一貫性のある SELinux ラベルを提供します。

SELinux と **systemd** 統合についての詳細は、9章*SELinux **systemd** によるアクセス制御*を参照してください。

[1] "Integrating Flexible Support for Security Policies into the Linux Operating System", by Peter Loscocco and Stephen Smalley. この論文は当初、国家安全保障局向けに書かれましたが、現在は公開されています。詳細および初回リリースの文書については オリジナル論文 を参照してください。編集および変更は、Murray McAllister 氏が行なっています。

[2] 「Meeting Critical Security Objectives with Security-Enhanced Linux」、Peter Loscocco および Stephen Smalley 著。この論文は当初、 国家安全保障局向けに作成されましたが、その後公開されています。詳細と当初公開されたドキュメントについては、 オリジナル論文 を参照してください。編集および変更は Murray McAllister 氏が行なっています。

[3] IP アドレスマッピングへのホスト名などの情報を含むテキストファイルで、DNS サーバーが使用するもの

[4] Marti, Don. "A seatbelt for server software: SELinux blocks real-world exploits". 2008 年 2 月 24 日発行。Accessed 27 August 2009: http://www.networkworld.com/news/2008/022408-selinux.html.

第2章 SELinux コンテキスト

プロセスとファイルは、SELinux ユーザーやロール、タイプ、レベル (オプション) などの追加情報を含む SELinux コンテキストでラベル付けされています。SELinux 実行中は、これらすべての情報を使ってアクセス制御が決定されます。Red Hat Enterprise Linux では SELinux は、RBAC (ロールベースアクセス制御) と TE (Type Enforcement)、さらにオプションで MLS (複数レベルのセキュリティ) の組み合わせを提供します。

以下は、SELinux コンテキストの例です。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上のプロセスや Linux ユーザー、ファイルに使用されます。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するには、以下のコマンドを実行します。

```
~]$ 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** という構文になります。各フィールドは以下のようになります。

SELinux user

SELinux user ID は、特定のロールセットおよび特定の MLS/MCS 範囲への権限があるポリシーに既知の ID です。各 Linux ユーザーは、SELinux ポリシー経由で SELinux ユーザーにマッピングされます。これにより、SELinux ユーザーに課された制限が Linux ユーザーに継承されます。マッピングされた SELinux ユーザー ID は、ユーザーが入ることができるロールやレベルを定義するためにそのセッションのプロセスにおいて SELinux コンテキストで使用されます。SELinu ユーザーアカウントと Linux ユーザーアカウント間のマッピング一覧を表示するには、root で以下のコマンドを実行します。(*policycoreutils-python* パッケージのインストールが必要になります)。

```
~]# 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       *
```

システムによって出力は多少異なります。

- **Login Name** コラムは Linux ユーザーを一覧表示します。
- **SELinux User** コラムでは、どの SELinux ユーザーに Linux ユーザーがマッピングされているかを一覧表示します。プロセスについてアクセス可能なロールとレベルを SELinux ユーザーが制限します。
- **MLS/MCS Range** コラムは、MLS (複数レベルセキュリティ) と MCS (複数カテゴリセキュリティ) が使用するレベルです。
- **Service** コラムは、Linux ユーザーがシステムにログインするはずの適切な SELinux コンテキストを決定します。デフォルトではアスタリスク (*) 記号が使用され、すべてサービスを表します。

role

SELinux の一部は RBAC (ロールベースアクセス制御) であり、ロールは RBAC の属性です。SELinux ユーザーはロールに対する権限を有しており、ロールはドメインに対する権限を持っています。ロールは、ドメインと SELinux ユーザーの媒介として機能します。入ることができるロールはどのドメインに入ることができるかを決定し、最終的には、これがどのオブジェクトタイプがアクセス可能かを制御します。これが、権限のあるエスカレーション攻撃における脆弱性の低減に役立ちます。

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を定義し、ファイルのタイプを定義します。SELinux ポリシールールは、ドメインがタイプにアクセスする場合でも、ドメインが別のドメインにアクセスする場合でも、タイプ同士がアクセスする方法を定義します。アクセスを許可する特定の SELinux ポリシールールが存在する場合にのみ、アクセスは許可されます。

level

レベルは、MLS および MCS の属性です。MLS 範囲は、レベルが異なる場合は **lowlevel-highlevel**、レベルが同一の場合は **lowlevel** と書かれる、一対のレベルです (**s0-s0** は **s0** と同じものです)。各レベルは、秘密度-カテゴリのペアで、カテゴリはオプションです。カテゴリがある場合、レベルは **sensitivity:category-set** と書かれます。カテゴリがない場合は、**sensitivity** と書かれます。

カテゴリセットが連続したものである場合は、短縮が可能です。例えば、**c0.c3** は **c0,c1,c2,c3** と同じことになります。**/etc/selinux/targeted/setrans.conf** ファイルは、レベル (**s0:c0**) をヒューマンリーダブルな形式にマッピングしています (すなわち、**CompanyConfidential**)。Red Hat Enterprise Linux では、ターゲットポリシーは MCS を強制し、MCS には **s0** という秘密度しかありません。Red Hat Enterprise Linux の MCS は、**c0** から **c1023** までの 1024 の異なるカテゴリをサポートします。**s0-s0:c0.c1023** の秘密度は **s0** で、すべてのカテゴリーに権限があります。

MLS は、Bell-La Padula 必須アクセスモデルを強制し、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 をデフォルトの SELinux ポリシーとするように設定します。
Red Hat Enterprise Linux で出荷される MLS ポリシーは、評価済み設定の一部ではないプログラムドメインの多くを省略するので、デスクトップワークステーション上の MLS は使用できません (X Window System ではサポートなし)。しかし、upstream SELinux Reference Policy からの MLS ポリシーは構築が可能で、これにはすべてのプログラムドメインが含まれます。MLS 設定の詳細については、「マルチレベルのセキュリティ (MLS)」を参照してください。

2.1. ドメイン移行

あるドメインのプロセスは、移行先のドメインの **entrypoint** タイプがあるアプリケーションを実行することで、別のドメインに移行できます。**entrypoint** パーミッションは SELinux ポリシーで使用され、ドメインに入るためにどのアプリケーションを使用するかを制御します。以下にドメイン移行の例を示します。

手順2.1 ドメイン移行の例

1. ユーザーはパスワードの変更を希望しています。これを行うには、**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** タイプがラベル付けされています。

```
~]$ 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ている**/etc/shadow** ファイルにアクセスします。

```
~]$ 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** ドメインで実行中のプロセスが**shadow_t** タイプのラベルが付いているファイルの読み取りおよび書き込みを許可されています。この **shadow_t** タイプはパスワード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** と **/etc/shadow** ファイル、およびこれらのバックアップファイルが含まれます。

3. SELinux ポリシールールでは、**passwd_t** ドメインには**passwd_exec_t** タイプへの **entrypoint** パーミッションがあるとしています。

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が**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が拒否となっていますが、**passwd_t** ドメインで実行中のアプリケーションが **shadow_t** タイプのラベルが付いたファイルにアクセスすることを許可するルールが存在することから、**passwd** アプリケーションは **/etc/shadow** ファイルへのアクセスが許可され、ユーザーのパスワードを更新することができます。

この例は包括的なものではなく、あくまでドメイン移行を説明する基本的な例として使われています。**passwd_t** ドメインで実行中のサブジェクトが**shadow_t** ファイルタイプのラベルが付けられたオブジェクトへアクセスすることを許可するルールは実際にありますが、サブジェクトが新たなドメインに移行する前に、他の SELinux ポリシールールが満たされる必要があります。この例では、Type Enforcement が以下のことを確認します。

- **passwd_t** ドメインには、**passwd_exec_t** タイプのラベルが付いたアプリケーションを実行することでしか、入ることができない。このドメインは、**lib_t** タイプのような権限のある共有ライブラリからしか実行できない。また、他のいかなるアプリケーションも実行できない。

- **passwd_t** のような、権限のあるドメインしか**shadow_t** タイプのラベルが付けられたファイルに書き込めない。他のプロセスがスーパーユーザー権限で実行されていても、**passwd_t** ドメインで実行されているわけではないので、これらのプロセスは **shadow_t** タイプのラベルが付けられたファイルには書き込めない。

- **passwd_t** ドメインに移行できるのは、権限のあるドメインのみ。例えば、**sendmail_t** ドメインで実行中の **sendmail** プロセスには**passwd** を実行する正当な理由がないので、**passwd_t** ドメインに移行することは決してありません。

- **passwd_t** ドメインで実行中のプロセスが読み取りおよび書き込みができる権限タイプは、**etc_t** または **shadow_t** タイプといったラベルが付けられたファイルのみです。これにより、**passwd** アプリケーションがだまされて任意のファイルを読み取りまたは書き込みすることを防ぎます。

2.2. プロセスでの SELinux コンテキスト

プロセスの SELinux コンテキストを表示するには、**ps -eZ** コマンドを実行します。例を示します。

手順2.2 passwd ユーティリティーの SELinux コンテキストを表示する

1. **アプリケーション** → **システムツール** → **端末** の順に選択して、端末を開きます。

2. **passwd** ユーティリティーを実行します。新たな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 user_name.
(current) UNIX password:
```

3. 新しいタブか別の端末を開いて、以下のコマンドを実行します。出力は以下のようになります。

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

4. 最初のタブまたは端末で **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** ユーティリティーの実行時 (**passwd_exec_t** タイプのラベルが付けられている) にユーザーのシェルプロセスが **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留意してください。

実行中のすべてのプロセスについての SELinux コンテキストを表示するには、再度 **ps** ユーティリティーを実行します。以下の出力例は省略されており、システムによっては異なる場合があることに注意してください。

```
]$ 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 ロールがデーモンなどのシステムプロセスに使われています。その後に、Type Enforcement が各ドメインを分離しています。

2.3. ユーザーの SELinux コンテキスト

以下のコマンドを使って、Linux ユーザーに関連する SELinux コンテキストを一覧表示します。

```
~]$ 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無制限の実行が可能です。この SELinux コンテキストでは、Linux ユーザーが SELinux **unconfined_u** ユーザーにマッピングされ、**unconfined_r** ロールとして実行し、**unconfined_t** ドメインで実行していることを示しています。 **s0-s0** は MLS 範囲で、このケースでは **s0** と同じです。ユーザーにアクセス権があるカテゴリは **c0.c1023** で定義され、これは全カテゴリになります (**c0** から **c1023** まで)。

第3章 ターゲットポリシー

ターゲットポリシーは、Red Hat Enterprise Linux で使われるデフォルトの SELinux ポリシーです。ターゲットポリシー使用時には、ターゲットとなるプロセスは制限されたドメインで実行され、ターゲット外のプロセスは制限のないドメインで実行されます。例えば、デフォルトではログインしたユーザーは **unconfined_t** ドメインで実行し、init で開始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。このドメインは両方とも、制限のないものです。

制限のないドメイン (制限のあるドメインも) は、実行可能および書き込み可能なメモリーチェックに制限されます。デフォルトでは、制限のないドメインで実行するサブジェクトは、書き込み可能なメモリーを割り当てることができず、その実行もできません。これにより、バッファオーバーフロー攻撃への脆弱性が低減されます。これらのメモリーチェックは、ブール値の設定で無効化されます。これにより、SELinux ポリシーのランタイムでの修正が可能になります。ブール値の設定は、後ほど説明されます。

3.1. 制限のあるプロセス

Red Hat Enterprise Linux では、**sshd** や **httpd** といったネットワーク上でリッスンするサービスは、ほとんどすべて制限があります。また、**passwd** ユーティリティーなど、root ユーザーとして実行し、ユーザーのためのタスクを実行するプロセスはほとんど制限があります。プロセスに制限があると、プロセス自体のドメイン内で実行されます。例えば、**httpd_t** ドメイン内で **httpd** プロセスが実行される、といったようにです。制限のあるプロセスが攻撃者によって危険にさらされても、SELinux ポリシーの設定によって、攻撃者のリソースへのアクセスや攻撃による損害は限定されます。

以下の手順を完了して、SELinux が有効となり、システムが以下の例を実行できる用意ができていることを確認してください。

手順3.1 SELinux ステータスの確認方法

1. SELinux が有効で enforcing モードで稼働しており、ターゲットポリシーが使用されていることを確認します。正常な出力は、以下のようになります。

```
~]$ sestatus
SELinux status:                 enabled
SELinuxfs mount:                /selinux
Current mode:                   enforcing
Mode from config file:          enforcing
Policy version:                 24
Policy from config file:        targeted
```

   SELinux の有効化および無効化についての詳細は、「SELinux の有効化および無効化」を参照してください。

2. root で **/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. 作成されたファイルの SELinux コンテンツを表示するには、以下のコマンドを実行します。

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

   Red Hat Enterprise Linux ではデフォルトで、Linux ユーザーには制限がありません。そのため、**testfile** ファイルに SELinux **unconfined_u** ユーザーのラベルが付けられてます。

RBAC はファイルでなくプロセスに使用されます。ロールはファイルにとって意味がありません。**object_r** ロールは、ファイルに使われる一般的なロールです (永続的なストレージおよびネットワークファイルシステム)。**/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** タイプは、**httpd** プロセスがこのファイルにアクセスすることを許可します。

以下では、Samba が使用するファイルなど、正確にラベル付けされていないファイルを Apache HTTP Server (**httpd**) が読み取らないように SELinux が防ぐ例を示します。これはあくまで例であり、本番環境では用いないでください。ここでは、*httpd* および *wget* パッケージがインストールされ、SELinux ターゲットポリシーが使われ、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3.2 制限のあるプロセスの例

1. root で **httpd** デーモンを起動します。

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

サービスが稼働していることを確認します。出力は以下のようになり、タイムスタンプのみが異なります。

```
~]$ 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 ユーザーでの書き込みアクセスがあるディレクトリーに切り替え、以下のコマンドを実行します。デフォルト設定に変更がなければ、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** コマンドでファイルのラベルを付け換えます。ただし、ファイルシステムのラベルが付け換えられると、この変更は失われます。ファイルシステムのラベルが付け換えられた場合でも、こうした変更を永続的に維持するには、**semanage** ユーティリティーを使用します。このコマンドについては後で説明します。root で以下のコマンドを実行し、タイプを Samba で使用されるタイプに変更します。

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

変更を表示するには、以下のコマンドを実行します。

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

4. 現行の DAC パーミッションは、**httpd** プロセスによる **testfile** へのアクセスを許可することに留意してください。ユーザーとしての書き込みアクセスがあるディレクトリーに切り替え、以下のコマンドを実行します。デフォルト設定に変更がなければ、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** の実行が必要がない場合は、root で以下のコマンドを実行して停止します。

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

この例では SELinux によって追加された新たなセキュリティを説明しました。ステップ 2 では、DAC ルールは **httpd** プロセスによる **testfile** へのアクセスを許可しますが、このファイルは**httpd** プロセスにアクセス権のないタイプでラベル付けされているので、SELinux はアクセスを拒否しました。

auditd デーモンが稼働していれば、以下のようなエラーが、**/var/log/audit/audit.log** にログ記録されます。

```
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** にログ記録されます。

```
[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。例えば、**init** で実行される制限のないサービスは **unconfined_service_t** ドメインで、カーネルで実行される制限のないサービスは**kernel_t** ドメインで、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ることになります。制限のないプロセスでは SELinux ポリシールールが適用されますが、既存のポリシールールは制限のないドメイン内で実行中のプロセスにほとんどすべてのアクセスを許可します。制限のないドメイン内で実行中のプロセスは、ほとんど DAC ルールにフォールバックします。制限のないプロセスが危険にさらされても、SELinux は攻撃者によるシステムリソースやデータへのアクセス獲得を阻止しません。しかし、もちろん DAC ルールは常に使われます。SELinux は DAC ルールの上に加わるもので、DAC ルールに取って代わるものではありません。

SELinux が有効であることを確認し、システムが以下の例を実行できるようにするには、「制限のあるプロセス」にある 手順3.1「SELinux ステータスの確認方法」を完了してください。

以下の例では、制限なしで実行中の場合、Apache HTTP Server (**httpd**) が Samba 向けのデータにアクセスできる様子を示します。Red Hat Enterprise Linux ではデフォルトで、**httpd** プロセスは制限のある **httpd_t** ドメイン内で実行されることに留意してください。これはあくまで例であり、本番環境では用いないでください。ここでは *httpd*、*wget*、*dbus*、*audit* パッケージがインストールされ、SELinux ターゲットポリシーが使われ、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** コマンドでファイルのラベルを付け換えます。ただし、ファイルシステムのラベルが付け換えられると、この変更は失われます。ファイルシステムのラベルが付け換えられた場合でも、こうした変更を永続的に維持するには、**semanage** ユーティリティーを使用します。このコマンドについては後で説明します。root ユーザーで以下のコマンドを実行し、タイプを Samba で使用され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、**httpd** プロセスが稼働し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 ユーザーで以下のコマンドを実行し、**httpd** プロセスを停止します。

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

3. **httpd** プロセスを制限なしで実行する場合は、 root ユーザーで以下のコマンドを実行し、**/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ものに変更します。

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

4. **/usr/sbin/httpd** に **bin_t** タイプがラベル付けさ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** プロセスを起動し、これが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. 以下のコマンドを実行し、**httpd** が **unconfined_service_t** ドメインで実行中であることを確認します。

```
~]$ 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 ユーザーでの書き込みアクセスがあるディレクトリーに切り替え、以下のコマンドを実行します。デフォルト設定に変更がなければ、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.1'

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

2009-05-07 01:41:10 (0.00 B/s) - `testfile.1' saved [0/0]
```

httpd プロセスには **samba_share_t** タイプのラベルが付いたファイルへのアクセス権はありませんが、**httpd** は制限のない **unconfined_service_t** ドメインで実行しており、DAC ルールにフォールバックします。このため、**wget** コマンドは成功します。もし **httpd** が制限のある **httpd_t** ドメインで実行していたら、**wget** コマンドは失敗していたでしょう。

8. **restorecon** ユーティリティーは、 ファイルのデフォルト SELinux コンテキストを復元します。root で以下のコマンドを実行すると、**/usr/sbin/httpd** のデフォルトの SELinux コンテキストが復元されます。

```
~]# 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** タイプがラベル付けさ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 で以下のコマンドを実行して **httpd** を再起動します。再起動したら、**httpd** が制限のある **httpd_t** ドメインで実行し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** の実行が必要がない場合は、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
```

このセクションの例は、危険にさらされた制限のあるプロセスからデータがどのように保護されるか (SELinux で保護)、また危険にさらされた制限のないプロセスから攻撃者がよりデータにアクセスしやすいか (SELinux で保護されていない) を示しています。

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 ポリシーを使って SELinux ユーザーにマッピングされます。これにより、SELinux ユーザーに課された制限が Linux ユーザーに継承されます。root で **semanage login -l** を実行すると、この Linux ユーザーマッピングが表示されます。

```
~]# semanage login -l

Login Name           SELinux User         MLS/MCS Range        Service

__default__          unconfined_u         s0-s0:c0.c1023       *
root                 unconfined_u         s0-s0:c0.c1023       *
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 **__default__** ログインにマッピングされ、これはさらに SELinux **unconfined_u** ユーザーにマッピングされます。以下の行で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、新規 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示しています。ここでは Red Hat Enterprise Linux のデフォルトにあるように、root ユーザーが制限なしで実行中であることを前提としています。

手順3.4 新規 Linux ユーザーを SELinux unconfined_u ユーザーにマッピングする

1. root で以下のコマンドを実行し、ユーザー名 **newuser** という新規 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、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. 現行セッションから一旦ログアウトし、Linux **newuser** ユーザーでログインし直します。ログインすると、**pam_selinux** PAM モジュールが自動的にこの Linux ユーザーを SELinux ユーザーにマッピングし (このケースでは **unconfined_u**)、SELinux コンテキストを設定します。その後は、このコンテキストで Linux ユーザーのシェルが起動されます。以下のコマンドを実行して、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

システム上で **newuser** ユーザーが不要になれば、Linux **newuser** のセッションからログアウトし、自分の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。これで **newuser** がこのユーザーのホームディレクトリーとともに削除されます。

制限のあるユーザーおよび制限のない Linux ユーザーは、実行可能および書き込み可能なメモリーチェックに影響を受け、また MCS と MLS に制限されます。

unconfined_t ドメインから自身の制限のあるドメインへの移行が可能と SELinux ポリシーが定義しているアプリケーションを、制限のない Linux ユーザーが実行しても、この制限のない Linux ユーザーはまだその制限のあるドメインの制約に影響を受けます。ここでのセキュリティの利点は、Linux ユーザーが制限なしで実行していてもアプリケーションには制限が残っているという点です。このため、アプリケーションの欠点が悪用されても、ポリシーで制限できます。

同様に、これらのチェックを制限のあるユーザーに適用することもできます。しかし、制限のあるユーザーはそれぞれ、**unconfined_t** ドメインに対して制限のあるユーザードメインで制限されます。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の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。その場合は、制限のある Linux ユーザーはターゲットの制限のあるドメインの制約の影響を受けることになります。つまり、特別の権限は、そのロールにしたがって制限のあるユーザーに関連付けられるということです。下記の表では、Red Hat Enterprise Linux における Linux ユーザーの基本的な制限のあるドメインの例を示しています。

表3.1 SELinux ユーザーの権限

ユーザー	ロール	ドメイン	X Window System	su または sudo	ホームディレクトリーおよび /tmp/ (デフォルト) で実行	ネットワーキング
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	いいえ	いいえ
xguest_u	xguest_r	xguest_t	はい	いいえ	いいえ	Firefox のみ

- **user_t**、**guest_t**、**xguest_t** ドメインの Linux ユーザーは、SELinux ポリシーが許可する場合に、決まったユーザー ID (setuid) アプリケーションしか実行できません (例、**passwd**)。これらのユーザーは **su** や **sudo** setuid アプリケーションを実行できないので、これらのアプリケーションを使って roo になることができません。
- **sysadm_t**、**staff_t**、**user_t**、**xguest_t** ドメインの Linux ユーザーは、X Window System と端末経由でログインできます。
- デフォルトでは、**guest_t** と **xguest_t** ドメインの Linux ユーザーは自身のホームディレクトリーや **/tmp/** 内のアプリケーションを実行できず、書き込みアクセスのあるディレクトリーにあるアプリケーションで、ユーザーのパーミッションを継承しているアプリケーションが実行できません。これにより、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のファイルを修正できないようにしています。
- デフォルトでは、**staff_t** と **user_t** ドメインの Linux ユーザーは自身のホームディレクトリーや **/tmp/** 内のアプリケーションの実行が可能です。ユーザーによるホームディレクトリーと**/tmp/** のアプリケーション実行の許可と阻止に関する情報は、「アプリケーションを実行するユーザーのためのブール値」 を参照してください。
- **xguest_t** ドメインの Linux ユーザーにある唯一のネットワークアクセスは、ウェブページに接続する **Firefox** です。

既に説明した SELinux ユーザーの他に、これらのユーザーにマッピング可能な特別ロールがあります。これらのロールは、SELinux がユーザーに許可するものを決定します。

- **webadm_r** は、Apache HTTP サーバーに関連する 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処理のみが可能です。

利用可能なロールを一覧表示するには、以下のコマンドを実行します。

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

seinfo コマンドはデフォルトではインストールされない*setools-console* パッケージが提供することに注意してください。

3.3.1. sudo 移行および SELinux ロール

ケースによっては、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ることもあります。これを実行するには、制限のあるユーザーが **sudo** コマンドを使って *制限のある管理者*の SELinux ロールを獲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者アクセスを付与するために使用されます。ユーザーが **sudo** を管理者コマンドの前に置いた場合、このユーザーは**ユーザー自身の** パスワードを要求されます。ユーザーが認証され、コマンドが許可されると、管理者コマンドは root ユーザーであるかのように実行されます。

表3.1「SELinux ユーザーの権限」 にあるように、**staff_u** および **sysadm_u** の制限のある SELinux ユーザーのみがデフォルトで **sudo** の使用を許可されています。それらのユーザーが**sudo** を使ってコマンドを実行すると、ユーザーのロールは **/etc/sudoers** 設定ファイルか、ある場合は **/etc/sudoers.d/** ディレクトリー内の各ファイルで指定されているルール基づいて変更されます。

sudo についての詳細情報は、Red Hat Enterprise Linux 7 システム管理者のガイド の 『権限の取得』 の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、**sudo** を設定して、新規作成の *SELinux_user_u* の制限のあるユーザーを *administrator_r* の制限のある管理者に移行する方法を説明します。既存の SELinux ユーザーに対して制限のある管理者ロールを設定するには、最初の 2 ステップを省略してください。また、以下のコマンドは root ユーザーで実行する必要があることに注意してください。

1. 新規 SELinux ユーザーを作成し、そのユーザーに対してデフォルトの SELinux ロールと補助的な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
```

   下記の例では、新規作成の **confined_u** SELinux ユーザーのデフォルトのロールは**staff_r** で、制限のある管理者ロールは **webadm_r** になります。

```
~]# semanage user -a -r s0-s0:c0.c1023 -R "staff_r webadm_r" confined_u
```

2. デフォルトの SElinux ポリシーコンテキストファイルをセットアップします。たとえば、**staff_u** SELinux ユーザーと同じ SELinux ルールを用意するには、**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
```

3. 新規作成の Linux 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/** ディレクトリー内に Linux ユーザーと同じ名前で新規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=administaror_t ROLE=administrator_r /bin/sh " > /etc/sudoers.d/linux_user
```

   例：

```
~]# echo "linux_user ALL=(ALL) TYPE=webadm_t ROLE=webadm_r /bin/sh " > /etc/sudoers.d/linux_user
```

5. **restorecon** ユーティリティーを使って *linux_user* ホームディレクトリーのラベルを付け替えます。

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

6. システムを再起動します。

```
~]# systemctl reboot
```

7. 新規作成の Linux ユーザーとしてログインすると、このユーザーはデフォルトの SELinux ロールでラベル付けされます。

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

sudo を実行すると、そのユーザーの SELinux コンテキストは **/etc/sudoers.d/*linux_user*** で指定されている補助的な SELinux ロールに変更されます。**sudo** で **-i** オプションを使用すると、インタラクティブシェルが実行されます。

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

最初のステップで指定された例の **confined_u** SELinux ユーザーの場合、出力は以下のようになります。

```
~]$ 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 を使った作業

ここからのセクションでは、Red Hat Enterprise Linux における主要 SELinux パッケージの概要を説明します。内容は以下の通りです。パッケージのインストールおよび更新、使用されるログファイル、主要 SELinux 設定ファイル、SELinux の有効および無効化、SELinux モード、ブール値の設定、ファイルおよびディレクトリーラベルの一時的および永続的変更、**mount** コマンドによるファイルシステムラベルの上書き、NFS ボリュームのマウント、ファイルおよびディレクトリーのコピーおよびアーカイブ時における SELinux コンテキストの保存方法。

4.1. SELinux パッケージ

Red Hat Enterprise Linux の完全インストールでは、インストール中に手動で除外しない限り、デフォルトで SELinux パッケージがインストールされます。テキストモードでの最小構成インストールだと、デフォルトでは *policycoreutils-python* と *policycoreutils-gui* はインストールされません。またデフォルトでは、SELinux ターゲットポリシーが使用され、SELinux は enforcing モードで実行されます。以下の SELinux パッケージ は、デフォルトでインストールされます。

- *policycoreutils* は、**restorecon**、**secon**、**setfiles**、**semodule**、**load_policy**、および **setsebool** を提供して SELinux を操作、管理します。
- *selinux-policy* は、SELinux Reference ポリシーの設定を提供します。SELinux Reference ポリシーは完全な SELinux ポリシーで、SELinux ターゲットポリシーなどの他のポリシーのベースとして使われます。詳細は、Tresys Technology SELinux Reference Policy のページを参照してください。このパッケージには、**selinux-policy.conf** ファイルや RPM マクロが含まれています。
- *selinux-policy-targeted* は、SELinux ターゲットポリシーを提供します。
- *libselinux* は、SELinux アプリケーション用の API を提供します。
- *libselinux-utils* は、**avcstat**、**getenforce**、**getsebool**、**matchpathcon**、**selinuxconlist**、**selinuxdefcon**、**selinuxenabled**、および **setenforce** のユーティリティーを提供します。
- *libselinux-python* は、SELinux アプリケーション開発用の 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ませんが、**yum install *<package-name>*** コマンドを実行するとオプションでインストールできます。

- *selinux-policy-devel* は、カスタム SELinux ポリシーおよびポリシーモジュール作成用のユーティリティーを提供します。また、SELinux と他のサービスを合わせて設定する方法を記述した man ページも含まれています。
- *selinux-policy-mls* は、MLS (複数レベルのセキュリティ) SELinux ポリシーを提供します。
- *setroubleshoot-server* は、 SELinux がアクセスを拒否した際に作成される拒否メッセージを、**sealert** ユーティリティーで表示可能な詳細な記述に変換します。このユーティリティーも本パッケージで提供されます。
- *setools-console* は、ポリシー分析およびクエリ、監査ログモニタリングおよびレポーティング、ファイルコンテキスト管理用の数多くのユーティリティーとライブラリーである Tresys Technology SETools distribution を提供します。*setools* パッケージは、SETools 用のメタパッケージです。*setools-gui* パッケージは、**apol** と **seaudit** の各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** の各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に関する詳細情報は、Tresys Technology SETools ページを参照してください。*setools* と *setools-gui* の各パッケージは、Red Hat Network Optional チャンネルが有効になっている時のみ利用可能であ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** のようなレベルを **SystemLow-SystemHigh** といった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の **semanage**、**audit2allow**、**audit2why**、**chcat** といった各種ユーティリティーを提供します。
- *policycoreutils-gui* は、SELinux 管理用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. 使用するログファイル

Red Hat Enterprise Linux では、*dbus* および *audit* のパッケージは、デフォルトのパッケージ選択から削除されなければ、デフォルトでインストールされます。*setroubleshoot-server* は Yum (**yum install setroubleshoot** コマンドを使用) でインストールする必要があります。

auditd が実行中であれば、以下のような SELinux 拒否メッセージはデフォルトで **/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** はすでに定期的なサービスとしては稼働していませんが、AVC メッセージの分析にはまだ使われています。必要に応じて以下の 2 つのプログラムが **setroubleshoot** を開始する方法として作動します。

- **sedispatch** ユーティリティーは、**audit** サブシステムの一部として実行されます。AVC 拒否メッセージが返されると、**sedispatch** は **dbus** を使ってメッセージを送信します。これらのメッセージは、**setroubleshootd** が実行中であればそこに直接送られます。実行中でなければ、**sedispatch** がこれを自動的に開始します。
- **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** 内の dbus メッセージを待機します。通知バブルを開始して、ユーザーが AVC メッセージを検討できるようにします。

手順4.1 デーモンの自動開始

1. **auditd** および **rsyslog** デーモンが起動時に自動的に開始するように設定するには、root ユーザーで以下のコマンドを実行します。

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

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

2. これらのデーモンが有効であることを確認するには、シェルプロンプトで次のコマンドを入力します。

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

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

別の方法では、**systemctl status *service-name*.service** コマンドを使って**enabled** というキーワードをコマンド出力で検索します。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 デーモンでシステムサービスを管理する方法についての詳細情報は、システム管理者のガイドのシステムサービスの管理の章を参照してください。

4.3. 主要設定ファイル

/etc/selinux/config は、主要 SELinux 設定ファイルです。使用するSELinux モードと SELinux ポリシーを管理します。

```
# 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=enforcing

SELINUX オプションは、SELinux が稼働するモードを設定します。SELinux には、enforcing、permissive、disabled の 3 つのモードがあります。enforcing モードでは SELinux ポリシーが強制され、SELinux ポリシールールに基づいて SELinux はアクセスを拒否します。拒否メッセージは、ログに記録されます。permissive モードでは、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 が enforcing モードであったら拒否されたであろうアクションについては拒否がログに記録されます。disabled モードでは、SELinux は無効化され (SELinux モジュールが Linux カーネルに登録されない)、DAC ルールのみが使用されます。

SELINUXTYPE=targeted

SELINUXTYPE オプションは、使用する SELinux ポリシーを設定します。ターゲットポリシーがデフォルトのポリシーです。MLS ポリシーを使用する場合にのみ、このオプションを変更してください。MLS ポリシーの有効化については、「SELinux における MLS の有効化」を参照してください。

重要

SELinux の permissive または disabled モードでシステムが稼働している場合、ユーザーにはファイルに誤ったラベル付けをするパーミッションがあります。また、SELinux が無効の間に作成されたファイルにはラベルが付けられません。enforcing モードに変更すると、これが問題になります。間違ったラベルが付いたファイルやラベルなしのファイルが問題を起こさないようにするために、disabled モードから permissive モードや enforcing モードに変更するとファイルシステムでは自動的に再ラベル付けが行われます。

4.4. SELinux の有効化および無効化

SELinux のステータスをチェックするには、**getenforce** または **sestatus** コマンドを使います。**getenforce** コマンドは、**Enforcing**、**Permissive**、**Disabled** のいずれかを返します。

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:                /selinux
Current mode:                   enforcing
Mode from config file:          enforcing
Policy version:                 24
Policy from config file:        targeted
```

4.4.1. SELinux の有効化

重要

システムが最初に SELinux なしで、特に *selinux-policy* パッケージなしでインストールされ、これが後でシステムに追加された場合、SELinux を有効にするには追加のステップが必要になります。システムのスタートアップ時に SELinux が初期化されたことを確認するには、**dracut** ユーティリティーを実行して SELinux 認識を initramfs ファイルシステムに記載する必要があります。これを行わないと、SELinux がシステムのスタートアップ時に起動しません。

SELinux が無効になっているシステムでは、**SELINUX=disabled** オプションが **/etc/selinux/config** で設定されています。

```
# 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
```

また、**getenforce** コマンドは、**Disabled** を返します。

```
~]$ getenforce
Disabled
```

以下の手順で、SELinux を有効にします。

手順4.2 SELinuxの有効化

1. ここでは、以下のパッケージがインストール済みであることを前提としています。
   - *selinux-policy-targeted*
   - *selinux-policy*
   - *libselinux*
   - *libselinux-python*
   - *libselinux-utils*
   - *policycoreutils*
   - *policycoreutils-python*
   - *setroubleshoot*
   - *setroubleshoot-server*
   - *setroubleshoot-plugins*

   上記の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、**rpm** ユーティリティーを使用します。

   ```
   ~]$ rpm -qa | grep selinux
   selinux-policy-3.12.1-136.el7.noarch
   libselinux-2.2.2-4.el7.x86_64
   selinux-policy-targeted-3.12.1-136.el7.noarch
   libselinux-utils-2.2.2-4.el7.x86_64
   libselinux-python-2.2.2-4.el7.x86_64
   ```

   ```
   ~]$ rpm -qa | grep policycoreutils
   policycoreutils-2.2.5-6.el7.x86_64
   policycoreutils-python-2.2.5-6.el7.x86_64
   ```

   ```
   ~]$ rpm -qa | grep setroubleshoot
   setroubleshoot-server-3.2.17-2.el7.x86_64
   setroubleshoot-3.2.17-2.el7.x86_64
   setroubleshoot-plugins-3.0.58-2.el7.noarch
   ```

   これら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 package_name
   ```

   以下のパッケージはオプションになります。

   - *policycoreutils-gui*

- *setroubleshoot*
- *mcstrans*

2. SELinux を有効にする前に、ファイルシステム上の全ファイルを SELinux コンテキストでラベル付けする必要があります。これが行われないと、制限のあるドメインはアクセスが拒否される場合があり、システムの正常な起動を妨げます。これを避けるには、**/etc/selinux/config** ファイルで **SELINUX=permissive** と設定します。

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

3. root でシステムを再起動します。次回の起動時に、ファイルシステムがラベル付けされます。このラベルプロセスでは、全ファイルに SELinux コンテキストがラベル付けされます。

```
~]# reboot
```

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

一番下の行の * (アスタリスク) 記号はそれぞれ、ラベル付けされた 1000 ファイルを表します。上記の例では、4 つの * 記号はラベル付けされた 4000 ファイルを表しています。全ファイルにラベル付けする時間はシステム上のファイル数とハードディスクドライブの速度によって異なります。最近のシステムでは、このプロセスは 10 分程度で終わります。

4. permissive モードでは SELinux ポリシーは強制されませんが、enforcing モードであれば拒否されたはずのアクションについては拒否がログに記録されます。enforcing モードに変更する前に、root で以下のコマンドを実行して、SELinux が最後の起動時にアクセスを拒否しなかったことを確認します。最後の起動時にアクセス拒否がなかった場合は、このコマンドはなにも返しません。起動時に SELinux がアクセスを拒否した場合は、トラブルシューティング情報を 10章 *トラブルシューティング* で参照してください。

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

5. **/var/log/messages** ファイルに拒否メッセージがなければ、**/etc/selinux/config** で **SELINUX=enforcing** と設定します。

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=enforcing
```

```
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

6. システムを再起動して、**getenforce** で **Enforcing** が返されることを確認します。

```
~]$ getenforce
Enforcing
```

7. root で以下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__          unconfined_u         s0-s0:c0.c1023       *
root                 unconfined_u         s0-s0:c0.c1023       *
system_u             system_u             s0-s0:c0.c1023       *
```

このような出力にならない場合は、root で以下のコマンドを実行してユーザーマッピングを修正します。**SELinux-user *username* is already defined** 警告を無視しても問題ありません。ここでの *username* は、**unconfined_u**、**guest_u**、**xguest_u** のいずれかになります。

手順4.3 ユーザーマッピングの修正

1. **~]# semanage user -a -S targeted -P user -R "unconfined_r system_r" -r s0-s0:c0.c1023 unconfined_u**
2. **~]# semanage login -m -S targeted -s "unconfined_u" -r s0-s0:c0.c1023 __default__**
3. **~]# semanage login -m -S targeted -s "unconfined_u" -r s0-s0:c0.c1023 root**
4. **~]# semanage user -a -S targeted -P user -R guest_r guest_u**
5. **~]# semanage user -a -S targeted -P user -R xguest_r xguest_u**

重要

SELinux の permissive または disabled モードでシステムが稼働している場合、ユーザーにはファイルを誤ってラベル付けするパーミッションがあります。また、SELinux が無効の間に作成されたファイルにはラベル付けがされません。enforcing モードに変更すると、これが問題になります。間違ったラベルが付いたファイルやラベルなしのファイルが問題を起こさないように disabled モードから permissive モードや enforcing モードに変更すると、ファイルシステムは自動的に再ラベル付けが行われます。

4.4.2. SELinux の無効化

SELinux を無効にするには、**/etc/selinux/config** ファイルで **SELINUX=disabled** と設定します。

```
# 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
```

システムを再起動して、**getenforce** コマンドが **Disabled** を返すことを確認します。

```
~]$ getenforce
Disabled
```

4.5. ブール値

ブール値を使うと、SELinux ポリシー記述の知識がなくても、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せずに、NFS ボリュームへのサービスのるアクセスを許可するといった変更が可能になります。

4.5.1. ブール値の一覧表示

ブール値の各項目が何であるかやそれらがオンかオフかについてなどの説明がある一覧を表示するには、Linux root ユーザーで **semanage boolean -l** コマンドを実行します。以下の例では、すべてのブール値が表示されているわけではなく、出力は省略されています。

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

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

SELinux boolean コラムは、ブール値の名前を表示します。**Description** コラムは、ブール値がオンかオフか、またそれらが何をするかを表示します。

以下の例では、**ftp_home_dir** ブール値はオフで、FTP デーモン (**vsftpd**) がユーザーのホームディレクトリーにあるファイルに読み取り/書き込みをしないようにしています。

```
ftp_home_dir                    (off  ,  off)  Determine whether ftpd can
read...
```

getsebool -a コマンドはブール値を一覧表示し、オンかオフかを表示しますが、個別の説明はありません。以下の例は、すべてのブール値を表示しているわけではありません。

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

getsebool *boolean-name* コマンドを実行すると、*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
```

複数のブール値を表示するには、空白で区切られたリストを使います。

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

4.5.2. ブール値の設定

ブール値を有効、無効にするには、**setsebool** ユーティリティーを **setsebool *boolean_name* on/off** の形式で実行します。

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

手順4.4 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て、Apache HTTP Server スクリプトとモジュールがデータベースサーバーに接続できないようにしています。

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

2. Apache HTTP Server スクリプトとモジュールが一時的にデータベースサーバーに接続できるようにするには、root で以下のコマンドを実行します。

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

3. ブール値が有効になったことを確認するには、**getsebool** ユーティリティーを使用します。

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

   これで Apache HTTP Server スクリプトとモジュールが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には維持されません。再起動後も変更を維持するには、root で **setsebool -P *boolean-name* on** コマンドを実行します。 [5]

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

4.5.3. Shell のオートコンプリート機能

getsebool、**setsebool**、**semanage** の各ユーティリティーでは Shell のオートコンプリート機能を使用することができます。**getsebool** と **setsebool** では、コマンドラインパラメーターとブール値にオートコンプリート機能が使用可能です。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記号 ("-") を付けて、**Tab** キーを押します。

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

ブール値でオートコンプリート機能を使用するには、ブール値名の入力を開始したところで **Tab** 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と使用され、これらはひとつずつ記入されます。**semanage** コマンドの最初の引数はオプションで、SELinux ポリシーのどの部分を管理するかを指定します。

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

その後にコマンドラインパラメーターが続きます。

```
~]# 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、ブール値や SELinux ユーザー、ドメインなどの特定の SELinux エントリー名を記入します。エントリー名の最初の部分を入力したら、**Tab** 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
```

コマンドラインパッケージは、コマンド内でチェーンすることができます。

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

4.6. SELinux コンテキスト – ファイルのラベル付け

SELinux 実行中のシステム上では、すべてのプロセスとファイルにセキュリティ関連の情報を表示する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。ファイルに関しては、**ls -Z** コマンドでこれを表示できます。

```
~]$ 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**) を示しています。この情報は、アクセス制限の決定に使用されます。DAC システムでは、アクセスは Linux ユーザー ID とグループ ID に基づいて制御されます。SELinux ポリシールールは、DAC ルールの後でチェックされます。DAC ルールが最初に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。

注記

デフォルトでは、新規作成の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 タイプを引き継ぎます。たとえば、**etc_t** タイプのラベルが付けられた**/etc/** ディレクトリー内に新規ファイルを作成すると、この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
```

ファイルの SELinux コンテキストを管理するには、 **chcon**、**semanage fcontext**、**restorecon** といった複数のコマンドがあります。

4.6.1. 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** コマンドによる変更は、ファイルシステムの再ラベル付けや **restorecon** コマンドが実行されると維持されません。SELinux ポリシーは、特定のファイルの SELinux コンテキストをユーザーが修正できるかどうかを制御します。**chcon** を使うと、ユーザーは変更する SELinux コンテキストの一部または全部を提供します。SELinux がアクセスを拒否する一般的な原因は、ファイルタイプが間違っているためです。

クイックリファレンス

- ファイルタイプを変更するには、**chcon -t *type file-name*** コマンドを実行します。ここでの *type* は **httpd_sys_content_t** などの SELinux タイプで、*file-name* はファイル名またはディレクトリー名になります。

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

- ディレクトリーのタイプとそのコンテンツを変更するには、**chcon -R -t *type* *directory-name*** コマンドを実行します。ここでの*type* は **httpd_sys_content_t** などの SELinux タイプで、*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4.5 ファイルまたはディレクトリーのタイプ変更

以下では SELinux コンテキストのタイプを変更し、他の属性はそのままにしておく手順を説明します。このセクションの例は、ディレクトリーにも適用できます。例えば、**file1** をディレクトリーに置き換えます。

1. ホームディレクトリーへ移動します。
2. 新規ファイルを作成し、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**unconfined_u** ユーザー、**object_r** ロール、**user_home_t** タイプ、**s0** レベルが含まれます。SELinux コンテキストの各パーツの説明は、2章*SELinux コンテキスト* を参照してください。

3. 以下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のみを変更します。そして、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. **file1** ファイルの SELinux コンテキストを復元するには、以下のコマンドを実行します。変更内容を表示するには、**-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、以前のタイプである **samba_share_t** が、正しい **user_home_t** に復元されました。ターゲットポリシー (Red Hat Enterprise Linux ではデフォルトの SELinux ポリシー) を使用している場合は、**restorecon** コマンドが **/etc/selinux/targeted/contexts/files/** ディレクトリー内のファイルを読み取り、どの SELinux コンテキストファイルにするかをチェックします。

手順4.6 ディレクトリーおよびコンテンツタイプの変更

以下の例では、新規ディレクトリーの作成と、そのディレクトリーのファイルタイプを (そのコンテンツとともに) Apache HTTP Server が使用するタイプに変更する方法を示します。この例で使用される設定は、Apache HTTP Server で (**/var/www/html/** ではなく) 異なるドキュメントルートを使用する場合に適用します。

1. root ユーザーとして新規ディレクトリー **/web/** を作成し、この中に 3 つの空のファイル (**file1**、**file2**、**file3**) を作成します。**/web/** ディレクトリーとその中のファイルは、**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 で以下のコマンドを実行し、**/web/** ディレクトリー (およびそのコンテンツ) 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 についての詳細は、chcon(1) の man ページを参照してください。

注記

Type Enforcement は、SELinux ターゲットポリシーで使われる主要なパーミッション制御です。ほとんどの場合、SELinux ユーザーとロールは無視することができます。

4.6.2. 永続的な変更: semanage fcontext

semanage fcontext コマンドは、ファイルの SELinux コンテキスト変更に使用します。ターゲットポリシーを使用の際は、変更は **/etc/selinux/targeted/contexts/files/** ディレクトリーにあるファイルに書き込まれます。

- **file_contexts** は多くのファイルのデフォルトのコンテキストに加え、**semanage fcontext** でアップデートされたコンテキストも指定します。
- **file_contexts.local** ファイルは、新規作成ファイルと **file_contexts** で見つからないディレクトリーのコンテキストを保存します。

これらのファイルは、2 つのユーティリティーが読み込みます。ファイルシステムのラベル変更には **setfiles** ユーティリティーを使用し、デフォルトの SELinux コンテキストを復元するには **restorecon** ユーティリティーを使用します。つまり、ファイルシステムのラベル変更が行われても、**semanage fcontext** による変更は維持されます。SELinux ポリシーは、ユーザーが特定ファイルの SELinux コンテキストを修正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ラベル変更が行われても SELinux コンテキストの変更が維持されるようにするには、以下の手順を実行します。

1. 以下のコマンドを実行します。ファイルまたはディレクトリーの完全パスを使用します。

```
~]# 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変更を適用します。

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

手順4.7 ファイルまたはディレクトリーのタイプ変更

以下ではファイルのタイプを変更し、SELinux コンテキストの他の属性はそのままにしておく例を示しています。このセクションの例は、ディレクトリーにも適用できます。例えば、**file1** をディレクトリーに置き換えます。

1. root ユーザーとして、**/etc/** ディレクトリー内に新規ファイルを作成します。デフォルトでは、**/etc/** ディレクトリー内の新規作成ファイルには **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確認するには、以下のコマンドを実行します。

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

2. root で以下のコマンドを実行し、**file1** のタイプを **samba_share_t** に変更します。**-a** オプションは新規レコードを追加し、**-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しても、直ちにタイプが変更されるわけではないことに留意してください。**file1** には **etc_t** タイプのラベルが付けられたままです。

```
~]# 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 -a -t samba_share_t /etc/file1 コマンドが以下のエントリーを **/etc/selinux/targeted/contexts/files/file_contexts.local** に追加します。

```
/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** が **/etc/file1** のエントリーを **file_contexts.local** に追加したので、**restorecon** によりタイプが **samba_share_t** に変更されます。

```
~]# 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4.8 ディレクトリーおよびコンテンツタイプの変更

以下の例では、新規ディレクトリーの作成と、そのディレクトリーのファイルタイプを (そのコンテンツとともに) Apache HTTP Server が使用するタイプに変更する方法を示します。この例で使用される設定は、Apache HTTP Server で (**/var/www/html/** ではなく) 異なるドキュメントルートを使用する場合に適用します。

1. root ユーザーとして新規ディレクトリー **/web/** を作成し、この中に 3 つの空のファイル (**file1**、**file2**、**file3**) を作成します。**/web/** ディレクトリーとその中のファイルは、**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 で以下のコマンドを実行し、**/web/** ディレクトリーとその中にあるファイルのタイプを **httpd_sys_content_t** に変更します。**-a** オプションは新規レコードを追加し、**-t** オプション

はタイプ (httpd_sys_content_t) を定義します。**"/web(/.*)?"** の正規表現を使うことで、**semanage** が変更を **/web/** とその中のファイルに適用します。このコマンドを実行しても、直接にはタイプを変更しないことに留意してください。**/web/** およびその中のファイルは **default_t** タイプのラベルが付けられたままです。

```
~]# 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(/.*)?" コマンドが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/** とその中のすべてのファイルのタイプを変更します。**-R** オプションは再帰的なので、**/web/** ディレクトリー下のすべてのファイルとディレクトリーが **httpd_sys_content_t** タイプでラベル付けされます。**semanage** で **/web(/.*)?** のエントリーを **file.contexts.local** に追加したので、**restorecon** により **httpd_sys_content_t** にタイプが変更されます。

```
~]# 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、新規作成の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 タイプを引き継ぎます。

手順4.9 追加されたコンテキストの削除

以下では、SELinux コンテキストの追加と削除の例を示しています。**/web(/.*)?** のようにコンテキストが正規表現の一部である場合、正規表現の前後に引用符を使います。

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

1. コンテキストを削除するには、root ユーザーで以下のコマンドを実行します。ここでの *file-name|directory-name* は、**file_contexts.local** の最初の部分です。

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

以下は、**file_contexts.local** 内のコンテキスト例です。

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

最初の部分は **/test** になっています。**restorecon** 実行後もしくはファイルシステムのラベル交換後に **/test/** ディレクトリーへの **httpd_sys_content_t** のラベル付けを防ぐには、root で以下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 についての詳細は、semanage(8) の man ページを参照してください。

重要

semanage fcontext -a で SELinux のコンテキストを変更する場合、ファイルシステムの再ラベル付け後もしくは **restorecon** コマンド実行後におけるファイルの誤ったラベル付けを避けるために、ファイルもしくはディレクトリーへの完全パスを使用してください。

4.7. file_t 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際は、EA 値を割り当てられていないファイルのデフォルトタイプは、**file_t** タイプになります。このタイプはこの目的のみに使用され、適切にラベル付けされたファイルシステム上には存在しません。これは、SELinux を実行しているシステム上の全ファイルには適切な SELinux コンテキストがあるはずで、**file_t** タイプはファイル-コンテキストの設定には決して使用されないためです。 [6]。

default_t タイプは、ファイル-コンテキスト設定内の他のパターンのいずれにも合致しないファイルに使用され、これによってこれらのファイルをディスク上のコンテキストのないファイルから区別できるようになり、通常は制限のあるドメインはアクセスできません。たとえば、**/mydirectory/** のようなトップレベルのディレクトリーを新たに作成すると、default_t タイプのラベルが付けられます。このディレクトリーにサービスがアクセスする必要がある場合、このロケーション用にファイル-コンテキスト設定を更新する必要があります。ファイル-コンテキスト設定にコンテキストを追加することに関しては、「永続的な変更: semanage fcontext」 を参照してください。

4.8. ファイルシステムのマウント

デフォルトでは、拡張属性をサポートするファイルシステムがマウントされる際は、各ファイルのセキュリティ-コンテキストがファイルの **security.selinux** 拡張属性から取得されます。拡張属性をサポートしないファイルシステムのファイルは、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設定コンテキストが割り当てられます。

既存の拡張属性を上書きしたり、拡張属性をサポートしないファイルシステムの異なるデフォルトコンテキストを特定するには、**mount -o context** コマンドを使います。例えば、複数システムで使用するリムーバブルメディアなどの正しい属性を提供するファイルシステムを信頼できない場合に、これは便利です。**mount -o context** コマンドは、File Allocation Table (FAT) や NFS ボリュームなど、拡張属性を

サポートしないファイルシステムのラベル付けのサポートにも使用できます。**context** オプションで指定されたコンテキストは、ディスクに書き込まれません。オリジナルのコンテキストは保持され、**context** なしでマウントされるとこれを見ることができます (最初にファイルシステムが拡張属性を持っている場合)。

ファイルシステムのラベル付けに関する情報については、James Morris の記事 "Filesystem Labeling in SELinux" : http://www.linuxjournal.com/article/7426 を参照してください。

4.8.1. コンテキストのマウント

ファイルシステムを指定されたコンテキストでマウントする、または既存のコンテキストがある場合はこれを上書きする、拡張属性をサポートしないファイルシステムの異なるデフォルトのコンテキストを指定するには、希望するファイルシステムのマウント時に root ユーザーで **mount -o context=*SELinux_user:role:type:level*** コマンドを実行します。コンテキストの変更は、ディスクに書き込まれません。デフォルトでは、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。共通ポリシーでは、このデフォルトのコンテキストは **nfs_t** タイプを使います。追加のマウントオプションがないと、これによって Apache HTTP Server などの他のサービスを使用する NFS ボリュームを共有することが妨げられる可能性があります。以下の例では NFS ボリュームをマウントすることで、Apache HTTP Server 経由での共有が可能になっています。

```
~]# 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上にある新規作成ファイルおよびディレクトリーには、**-o context** で指定された SELinux コンテキストがあるように見えます。しかし、これらの変更はディスクに書き込まれていないため、このオプションで指定されたコンテキストは新たなマウントがあると維持されません。このため、このオプションのコンテキストを保持するには、マウント時に指定されたものと同一のコンテキストと使用する必要があります。コンテキストを新たなマウントの後にも維持する方法については、「コンテキストのマウントを永続的にする」 を参照してください。

Type Enforcement は、SELinux ターゲットポリシーで使われる主要なパーミッション制御です。ほとんどの場合、SELinux ユーザーとロールは無視することができます。このため、**-o context** で SELinux コンテキストを上書きする際は、SELinux **system_u** ユーザーと **object_r** ロールを使って、このタイプに集中します。MLS ポリシーや複数カテゴリのセキュリティを使用していない場合は、**s0** レベルを使います。

注記

ファイルシステムを **context** オプションでマウントする場合は、(ユーザーやプロセスによる) コンテキスト変更は禁止されます。例えば、**context** オプションでマウントされたファイルシステム上で **chcon** コマンドを実行すると、**Operation not supported** エラーが出ます。

4.8.2. デフォルトコンテキストの変更

「file_t および default_t タイプ」 の説明にあるように、拡張属性をサポートするファイルシステムでは、ディスク上に SELinux コンテキストがないファイルにアクセスがあった場合、SELinux ポリシーが定義するデフォルトのコンテキストを持っているものとして扱われます。共通ポリシーでは、このデフォルトのコンテキストは **file_t** タイプを使います。別のデフォルトコンテキストが望ましい場合は、**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** 上の) 新規作成ファイルを新規作成の **/test/** ディレクトリーにマウントします。ここでは、**/test/** ディレクトリーを定義するルールが **/etc/selinux/targeted/contexts/files/** にないことを前提としています。

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

この例では、

- **system_u:object_r:samba_share_t:s0** が「ラベルのないファイルのデフォルトの説明コンテキスト」 [7] であることを、**defcontext** オプションが定義します。
- マウントされると、ファイルシステムの root ディレクトリー (**/test/**) は、 **defcontext** が指定するコンテキストでラベル付けされたかのように扱われます (このラベルはディスク上で保存されない)。これは、**/test/** 下で作成されたファイルのラベリングに影響します。新規作成ファイルは **samba_share_t** タイプを継承し、これらのラベルはディスク上で保存されます。
- **defcontext** オプションでファイルシステムがマウントされている間に**/test/** 下で作成されたファイルは、そのラベルを保持します。

4.8.3. NFS ボリュームのマウント

デフォルトでは、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。共通ポリシーでは、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れない場合もあります。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がマウントされて、他のサービスがこれを読み取ったりエクスポートしたりすることを防ぐことができます。

NFS ボリュームをマウントし、別のサービスでこれを読み取ったりエクスポートしたい場合は、マウントの際に **context** オプションを使って **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って NFS ボリュームをマウントすることで、Apache HTTP Server 経由の共有が可能になります。

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

これらの変更はディスクに書き込まれないため、このオプションで指定されたコンテキストは新たなマウントがあると維持されません。このため、このオプションのコンテキストを保持するには、マウント時に指定されたものと同一のコンテキストと使用する必要があります。コンテキストを新たなマウントの後にも維持する方法については、「コンテキストのマウントを永続的にする」を参照してください。

context オプションを使ったファイルシステムのマウントの代替方法として、ブール値を有効にして **nfs_t** タイプのラベルが付いたファイルシステムへのサービスのアクセスを許可することもできます。**nfs_t** タイプへのサービスのアクセスを許可するブール値の設定については、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。

4.8.4. 複数の NFS マウント

同一の NFS エクスポートから複数のマウントを行う場合、各マウントの SELinux コンテキストを異なるコンテキストで上書きしようとすると、マウントコマンドの失敗につながります。以下の例では、NFS サーバーには単一エクスポートである **/export/** があり、これには**web/** と **database/** の 2 つのサブディレクトリーがあります。以下のコマンドで単一 NFS エクスポートから 2 つのマウントを試みて、それぞれの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 つ目のマウントコマンドが失敗し、以下が **/var/log/messages** にログ記録されます。

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

コンテキストが異なる複数のマウントを単一 NFS エクスポートから行うには、**-o nosharecache,context** オプションを使用します。以下の例では、コンテキストが異なる複数のマウントを単一 NFS エクスポートから行います (各マウントへの単一サービスアクセスを許可)。

```
~]# 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** がローカルで **/local/web/** にマウントされ、すべてのファイルが **httpd_sys_content_t**タイプでラベル付けされており、Apache HTTP Server へのアクセスを許可しています。**server:/export/database** はローカルで **/local/database** にマウントされ、すべてのファイルが **mysqld_db_t** タイプでラベル付けされており、MariaDB へのアクセスを許可しています。これらのタイプ変更はディスクに書き込まれません。

重要

nosharecache オプションを使うと、あるエクスポートの同一のサブディレクトリーを異なるコンテキストで複数回マウントすることができます (例えば、**/export/web/** を複数回マウントする)。ファイルが 2 つの異なるコンテキストでアクセス可能な場合は、エクスポートの同一のサブディレクトリーを異なるコンテキストで複数回マウントしないでください。重複するマウントを作成することになってしまいます。

4.8.5. コンテキストのマウントを永続的にする

コンテキストのマウントを再マウントや再起動後も維持するには、**/etc/fstab** ファイル内のファイルシステムのエントリーまたは自動マウント機能のマップを追加し、希望するコンテキストをマウントオプションとして使用します。以下の例では、NFS コンテキストマウントでエントリーを **/etc/fstab** に追加します。

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

4.9. SELinux ラベルの維持

このセクションでは、ファイルおよびディレクトリーのコピー、移動、アーカイビングによる SELinux コンテキストへの影響を説明します。また、コピーおよびアーカイブ時にコンテキストを維持する方法も説明します。

4.9.1. 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がない場合にこれらをコピーすると、新たなファイルまたはディレクトリーが作成されます。この新規作成のファイルまたはディレクトリーのコンテキストは、(オリジナルコンテキストを維持するオプションが使用されていなければ) オリジナルのファイルまたはディレクトリーのコンテキストではなく、デフォルトのラベリングルールに基づくことになります。例えば、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
```

このファイルが **/etc/** という別のディレクトリーにコピーされたとすると、この新しいファイルは**/etc/** のデフォルトのラベル付けルールにしたがって作成されます。(追加オプションなしで) ファイルをコピーすると、オリジナルのコンテキストは保持されない可能性があります。

```
~]$ 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
```

/etc/file1 が存在しない状況で、**file1** が **/etc/** にコピーされると、**/etc/file1** は新規ファイルとして作成されます。上の例にあるように、**/etc/file1** はデフォルトのラベル付けルールにしたがって、**etc_t** タイプでラベル付けされます。

ファイルが既存ファイル上にコピーされると、ユーザーが **--preserve=context** などの **cp** オプションを指定してオリジナルファイルのコンテキストを維持しない限り、既存ファイルのコンテキストが維持されます。SELinux ポリシーは、コピー時にコンテキストの維持を妨げる場合があります。

手順4.10 SELinux コンテキストを維持せずにコピーする

この手順では、**cp** コマンドでオプションなしでファイルをコピーすると、ターゲットの親ディレクトリーからタイプを継承することを示しています。

1. ユーザーのホームディレクトリーでファイルを作成します。ファイルは **user_home_t** タイプでラベル付けされます。

   ```
   ~]$ touch file1
   ```

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

2. 以下のコマンドで示すように、**/var/www/html/** ディレクトリーは **httpd_sys_content_t** タイプでラベル付けされています。

```
~]$ 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4.11 SELinux コンテキストを維持してコピーする

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維持する方法を示しています。

1. ユーザーのホームディレクトリーでファイルを作成します。ファイルは **user_home_t** タイプでラベル付けされます。

```
~]$ touch file1
```

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

2. 以下のコマンドで示すように、**/var/www/html/** ディレクトリーは **httpd_sys_content_t** タイプでラベル付けされています。

```
~]$ 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** オプションを使うと、コピー時に SELinux コンテキストが維持されます。以下で示すように、**file1** の **user_home_t** タイプは、このファイルを **/var/www/html/** にコピーしても維持されます。

```
~]# 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4.12 コンテキストのコピーおよび変更

この手順では、**--context** オプションを使ってコピー先のコンテキストを変更する方法を示しています。以下の例は、ユーザーのホームディレクトリーで行われています。

1. ユーザーのホームディレクトリーでファイルを作成します。ファイルは **user_home_t** タイプでラベル付けされます。

```
~]$ touch file1
```

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

2. **--context** オプションを使って 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** を使用しないと、**file2** は **unconfined_u:object_r:user_home_t** コンテキストでラベル付けされます。

```
~]$ 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4.13 既存ファイル上へのファイルのコピー

この手順では、既存ファイル上にファイルをコピーする際に、(オプションを使ってコンテキストを維持する場合を除いて) 既存ファイルのコンテキストが維持されることを示しています。

1. root で新規ファイル **file1** を **/etc/** ディレクトリーに作成します。以下のように、このファイルは **etc_t** タイプでラベル付けされます。

```
~]# 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。以下のように、このファイルは **user_tmp_t** タイプでラベル付けされます。

```
~]$ 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. コピー後に以下のコマンドを実行すると、**file1** は **etc_t** タイプでラベル付けされており、**/etc/file1** を上書きした **/tmp/file2** の **user_tmp_t** タイプではないことが分かります。

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

重要

ファイルやディレクトリーは移動するのではなく、コピーしてください。こうすることで、正しい SELinux コンテキストでのラベル付けが確保されます。SELinux コンテキストが間違っていると、プロセスがそれらのファイルやディレクトリーにアクセスできなくなります。

4.9.2. ファイルおよびディレクトリーの移動

ファイルとディレクトリーは、移動すると現行の SELinux コンテキストを維持します。多くの場合、これは移動先の場所で間違ったものとなります。以下の例では、ファイルをユーザーのホームディレクトリーから Apache HTTP Server が使用する **/var/www/html/** ディレクトリーに移動します。ファイルは移動されたため、正しい SELinux コンテキストを継承しません。

手順4.14 ファイルおよびディレクトリーの移動

1. ユーザーのホームディレクトリーに移動して、ファイルを作成します。ファイルは **user_home_t** タイプでラベル付けされます。

```
~]$ touch file1
```

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

2. 以下のコマンドを実行して、**/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** タイプがラベル付けされています。**/var/www/html/** 下で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するため、このタイプでラベル付けされます。

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、現行の **user_home_t** タイプを維持します。

```
~]# 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** タイプでラベル付けされたファイルを読み取れません。Web ページを構成するすべてのファイルが **user_home_t** タイプ、もしくは Apache HTTP Server が読み取り不可能な別のタイプでラベル付けされている場合、それらに **Mozilla Firefox** のような Web ブラウザ経由でアクセスしようとすると、パーミッションは拒否されます。

重要

ファイルやディレクトリーを **mv** コマンドで移動すると、誤った SELinux コンテキストとなり、Apache HTTP Server や Samba などのプロセスがそれらの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。

4.9.3. デフォルト SELinux コンテキストのチェック

ファイルやディレクトリーの SELinux コンテキストが正しいかどうかは、 **matchpathcon** ユーティリティーを使ってチェックします。このユーティリティーは、システムポリシーにクエリを行い、ファイルパスに関連するデフォルトのセキュリティコンテキストを提供します。 [8] 以下の例では、**matchpathcon** を使って **/var/www/html/** ディレクトリーのファイルが正しくラベル付けされているかを検証しています。

手順4.15 matchpathcon を使ってデフォルトの SELinux コンテキストをチェックする

1. root ユーザーとして **/var/www/html/** ディレクトリーに 3 つのファイルを作成します (**file1**、**file2**、**file3**)。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** のタイプを **samba_share_t** に変更します。Apache HTTP Server は、**samba_share_t** タイプでラベル付けされたファイルやディレクトリーを読み取れ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、現行の SELinux コンテキストを SELinux ポリシーの正しいデフォルトのコンテキストと比較します。以下のコマンドを実行すると、**/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.
```

以下の **matchpathcon** コマンドの出力は、**file1** は **samba_share_t** タイプでラベル付けされていますが、**httpd_sys_content_t** タイプでラベル付けされるべきであることを示しています。

```
/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
```

このラベル問題を解決して Apache HTTP Server が **file1** にアクセスできるようにするには、root で **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
```

4.9.4. tar を使ったファイルのアーカイブ作成

tar ユーティリティーはデフォルトでは拡張属性を維持しません。SELinux コンテキストは拡張属性に保存されるので、ファイルをアーカイブするとコンテキストは失われます。コンテキストを維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するには、**tar --selinux** を使います。**tar** アーカイブに拡張属性のないファイルが含まれる、もしくはシステムデフォルトに拡張属性を適合させたい場合は、**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、root ユーザーで **restorecon** を実行する必要があることもあります。

以下の例では、SELinux コンテキストを保持する **tar** アーカイブの作成方法を説明します。

手順4.16 tar アーカイブを作成する

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

```
~]$ 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 ユーザーとして **/var/www/html/** ディレクトリーに 3 つのファイルを作成します (**file1**、**file2**、**file3**)。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、**test.tar** という名前の **tar** アーカイブを作成します。SELinux コンテキストを保持するには、**--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、全ユーザーに完全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、以下のコマンドを実行して **tar** アーカイブを抽出します。**--selinux** オプションを指定してください。これを行わないと、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 コンテキストを確認します。**httpd_sys_content_t** タイプが維持されたことが分かります。**--selinux** を使用していなければ、**default_t** に変更されていました。

```
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/** ディレクトリーが不要になったら、root で以下のコマンドを実行し、ディレクトリーとその中の全ファイルを削除します。

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

拡張属性すべてを保持する **--xattrs** オプションなどの **tar** に関する詳細情報は、tar(1) man ページを参照してください。

4.9.5. star を使ったファイルのアーカイブ作成

star ユーティリティーは、デフォルトでは拡張属性を維持しません。SELinux コンテキストは拡張属性に保存されるので、ファイルをアーカイビングするとコンテキストは失われます。コンテキストを維持するアーカイブを作成するには、**star -xattr -H=exustar** コマンドを使用します。*star* パッケージはデフォルトではインストールされません。**star** をインストールするには、**yum install star** コマンドを root ユーザーで実行します。

以下の例では、SELinux コンテキストを保持する **star** アーカイブの作成方法を説明します。

手順4.17 star アーカイブを作成する

1. root で **/var/www/html/** ディレクトリーに 3 つのファイルを作成します (**file1**、**file2**、**file3**)。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/** ディレクトリーに移動し、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、全ユーザーに完全アクセスを許可します。

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

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

4. 以下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/** ディレクトリーに移動し、以下のコマンドを実行して**star** アーカイブを抽出します。

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

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

6. SELinux コンテキストを確認します。**httpd_sys_content_t** タイプが維持されたことが分かります。**-xattr -H=exustar** オプションを使用していなければ、**default_t** に変更されていました。

```
~]$ 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/** ディレクトリーが不要になったら、root で以下のコマンドを実行し、ディレクトリーとその中の全ファイルを削除します。

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

8. **star** が不要になったら、root でパッケージを削除します。

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

star についての詳細は、star(1) の man ページを参照してください。

4.10. 情報収集ツール

以下に挙げるユーティリティーは、アクセスベクターキャッシュの統計情報やクラス、タイプ、ブール値の数などの情報を便利な形式で提供するコマンドラインツールです。

avcstat

このコマンドは、ブート以降のアクセスベクターキャッシュの統計値を短い出力で提供します。時間間隔を秒に指定すると、統計をリアルタイムで見ることができます。これで、初期出力以降の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**/selinux/avc/cache_stats** で、**-f /path/to/file** オプションを使うと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、クラスやタイプ、ブール値、allow ルールの数などのポリシーの内訳を説明する際に便利です。**seinfo** は、policy.conf ファイル (バージョン 12 から 21 までのポリシーソースを含む単一テキストファイル) やバイナリーポリシーファイル、ポリシーパッケージのモジュラー一覧、入力としてのポリシー一覧ファイルを使う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うには、*setools-console* がインストールされている必要があります。

seinfo の出力は、バイナリーとソースファイル間では異なります。例えば、ポリシーソースファイルは**{ }** の括弧で複数のルール要素を単一行にまとめます。属性に関しても同様の働きをし、単一属性が一つまたは複数のタイプに拡大します。これらは拡張されたものでバイナリーポリシーファイルとは関連がなくなるため、検索結果ではゼロの値が返されます。しかし、最初は括弧を使っていた単一行のルールが複数の個別行となると、ルールの数は大幅に増大します。

バイナリーポリシーにはないアイテムもあります。例えば、neverallow ルールはランタイム中ではなく、ポリシーのコンパイル中にのみチェックされます。また、最初のセキュリティ ID (SID) はブート中にカーネルがポリシーを読み込む前に必要となることから、バイナリーポリシーの一部ではありません。

```
~]# seinfo

Statistics for policy file: /etc/selinux/targeted/policy/policy.24
```

```
Policy Version  & Type: v.24 (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
```

また **seinfo** ユーティリティーは、ドメイン属性を持つタイプの数を一覧表示することも可能で、制限のある異なるプロセスの数を予測します。

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

すべてのドメインタイプに制限があるわけではありません。制限のないドメイン数を確認するには、**unconfined_domain** 属性を使います。

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

Permissive ドメインは、**--permissive** オプションで数えられます。

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

完全なリストを表示するには、上記のコマンドから **| wc -l** を除きます。

sesearch

sesearch ユーティリティーを使うと、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検索ができます。例を示します。

```
~]$ sesearch --role_allow -t httpd_sys_content_t
/etc/selinux/targeted/policy/policy.24
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** ルールの数を提示します。

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

dontaudit ルールの数も提供可能です。

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

4.11. マルチレベルのセキュリティ (MLS)

マルチレベルのセキュリティ技術とは、Bell-La Padula Mandatory Access Model を強制するセキュリティ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 (subjects)* と呼ばれ、ファイル、デバイス、システムのその他のパッシブコンポーネントは *オブジェクト (objects)* 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。各セキュリティレベルは *sensitivity (秘密度)* と *category (カテゴリ)* で構成されています。例えば、社内のリリーススケジュールは、社内ドキュメントカテゴリの部外秘の秘密度で保管されています。

図4.1「クリアランスレベル」は、米国の国防コミュニティーが最初に設計したクリアランスレベルを示しています。上記の例の社内スケジュールに当てはめると、部外秘カテゴリのドキュメントを閲覧できるのは、部外秘クリアランスを取得しているユーザーのみとなります。しかし、部外秘クリアランスしかないユーザーは、より高いレベルのクリアランスを必要とするドキュメントの閲覧はできません。このようなユーザーは、より低いレベルのクリアランスのドキュメントには読み取り専用アクセスが許可され、より高いレベルのクリアランスのドキュメントには書き込みアクセスが許可されます。

図4.1 クリアランスレベル

図4.2「MLS を使用したデータフローの許可」は、「秘密」セキュリティレベルで実行しているサブジェクトと異なるセキュリティレベルのオブジェクト間で許可されるすべてのデータフローを示しています。簡潔に説明すると、Bell-LaPadula モデルは *no read up (上方読み取りは不可)* と *no write down (下方書き込みは不可)* という 2 つの特性を強制します。

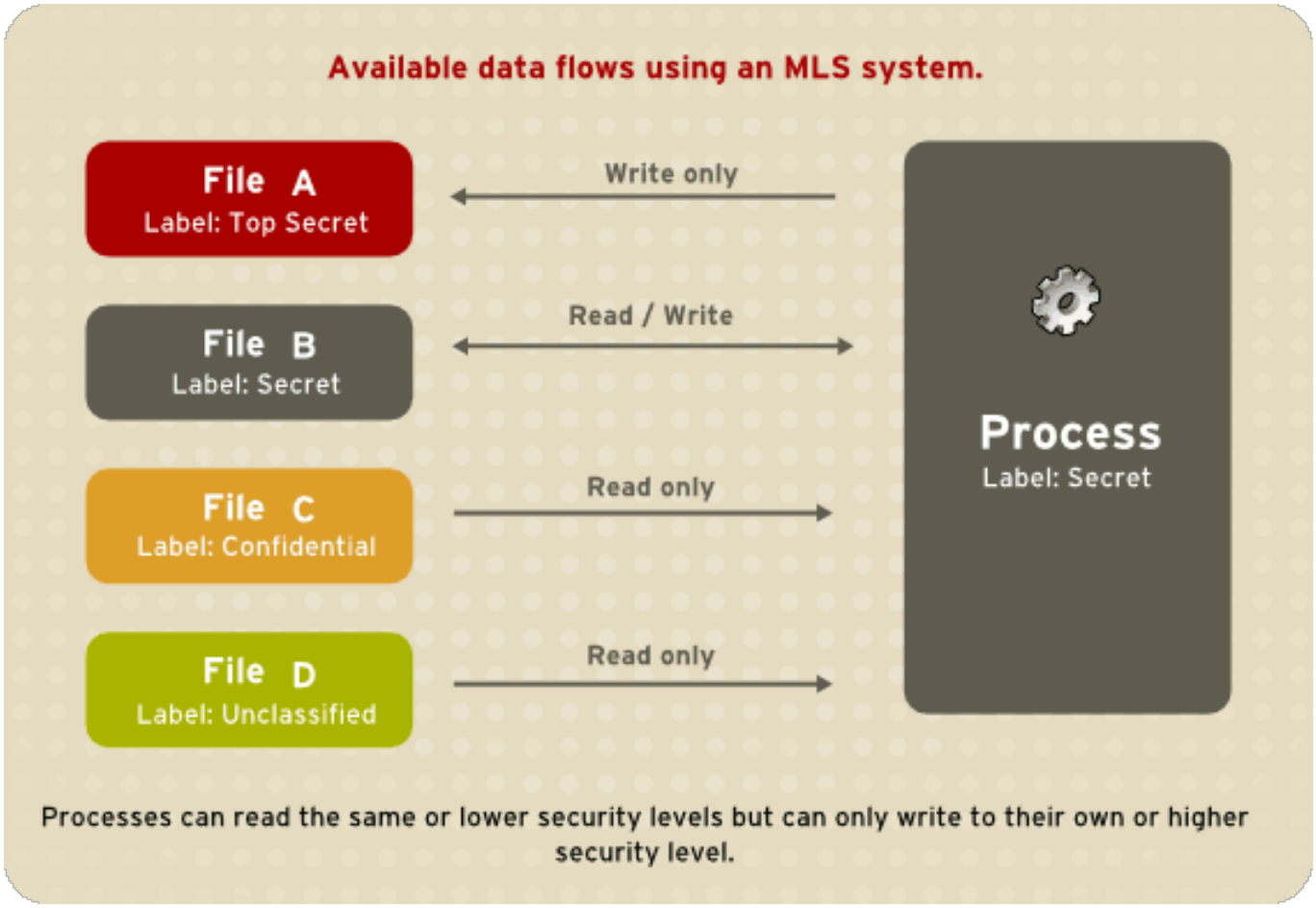


図4.2 MLS を使用したデータフローの許可

4.11.1. MLS とシステム権限

MLS アクセスルールは、常に従来のアクセスパーミッション (ファイルパーミッション) と組み合わせて使われます。例えば、「秘密」のセキュリティレベルを持つユーザーが任意アクセス制御 (DAC) を使って他のユーザーによるファイルへのアクセスを遮断すると、「最高秘密」のセキュリティレベルを持つユーザーのアクセスも遮断されます。SELinux の MLS 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です。より高いセキュリティクリアランスがあるからといって、任意にファイルシステムを閲覧する許可が自動的に与えられるわけではありません。

トップレベルのクリアランスを持つユーザーは、マルチレベルシステム上で自動的に管理者権限を獲得するわけではありません。このようなユーザーは、コンピューター上の全情報へのアクセスがありますが、これは管理者権限とは別のものです。

4.11.2. SELinux における MLS の有効化

注記

X Window System 実行中のシステム上では、MLS ポリシーの使用は推奨されません。

システム上で SELinux の MLS ポリシーを有効にするには、以下のステップにしたがいます。

手順4.18 SELinux MLS ポリシーを有効にする

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、ファイルシステム上のすべてのファイルが MLS ラベルで再ラベル付けされる必要があります。ファイルシステムが再ラベル付けされると、制限のあるドメインはアクセスが拒否され、システムが正常に起動できない可能性があります。これを回避するには、**/etc/selinux/config** ファイルで **SELINUX=permissive** と設定します。また、**SELINUXTYPE=mls** と設定して 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 モードで稼働していることを確認します。

```
~]# setenforce 0
```

```
~]$ getenforce
```

```
Permissive
```

4. **.autorelabel** ファイルを root のホームディレクトリーに作成し、次回のリブート時にファイルが再ラベル付けされていることを確認します。

```
~]# touch /.autorelabel
```

このファイルには **-F** オプションを追加する必要があることに留意してください。以下のコマンドを実行すると、これができます。

```
~]# echo "-F" >> /.autorelabel
```

5. システムをリブートします。次回の起動時にすべてのファイルシステムが MLS ポリシーにしたがって再ラベル付けされます。ラベルプロセスでは、全ファイルが適切な SELinux コンテキストでラベル付けされます。

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

一番下の行の * (アスタリスク) 記号はそれぞれ、ラベル付けされた 1000 ファイルを表します。上記の例では、11 個の * 記号はラベル付けされた 11000 ファイルを表しています。全ファイルにラベル付けする時間はシステム上のファイル数とハードディスクドライブの速度によって異なります。最近のシステムでは、このプロセスは 10 分程度で終わります。ラベリングプロセスが完了すると、システムは自動で再起動します。

6. permissive モードでは SELinux ポリシーは強制されませんが、enforcing モードであれば拒否されたはずのアクションについては拒否がログに記録されます。enforcing モードに変更する前に、root で以下のコマンドを実行して、SELinux が最後の起動時にアクセスを拒否しなかったことを確認します。最後の起動時にアクセス拒否がなかった場合は、このコマンドはなにも返しません。起動時に SELinux がアクセスを拒否した場合は、トラブルシューティング情報を 10章トラブルシューティング で参照してください。

```
~]# 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場合、または既存の拒否をすべて解決した場合は、**/etc/selinux/config** ファイルで **SELINUX=enforcing** と設定します。

```
# 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 モードで稼働していることを確認します。

```
~]$ getenforce
Enforcing
```

MLS ポリシーが有効であることも確認します。

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

4.11.3. 特別の MLS 範囲を持つユーザーの作成

特別の MLS 範囲を持つ新規 Linux ユーザーを作成するには、以下のステップにしたがいます。

手順4.19 特別の MLS 範囲を持つユーザーの作成

1. **useradd** コマンドで新規Linux ユーザーを追加し、このユーザーを既存の SELinux ユーザーにマッピングします (このケースでは **user_u**)。

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

2. 新規作成の Linux ユーザーにパスワードを割り当てます。

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

3. root で以下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__          unconfined_u         s0-s0:c0.c1023       *
john                 user_u               s0                   *
root                 unconfined_u         s0-s0:c0.c1023       *
system_u             system_u             s0-s0:c0.c1023       *
```

4. ユーザー **john** の特定範囲を定義します。

```
~]# semanage login --modify --seuser user_u --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** に特定の MLS 範囲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__          unconfined_u         s0-s0:c0.c1023       *
john                 user_u               s2:c100              *
root                 unconfined_u         s0-s0:c0.c1023       *
system_u             system_u             s0-s0:c0.c1023       *
```

6. john のホームディレクトリーのラベルを修正するには、以下のコマンドを実行します (必要な場合)。

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

4.11.4. Polyinstantiated ディレクトリーの設定

/tmp 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、ユーザーが一時的なストレージとして使用します。しかしこの設定では、これらのディレクトリーは競合状態の攻撃やファイル名に基づく情報漏えいに対して脆弱となってしまいます。SELinux は、*polyinstantiated* ディレクトリーという形で解決法を提供します。これはつまり、**/tmp** と **/var/tmp/** の両方がインスタンス化され、各ユーザーにはプライベートのように見えるということです。ディレクトリーのインスタンス化が有効になると、各ユーザーの **/tmp** と **/var/tmp/** ディレクトリーは自動的に **/tmp-inst** および **/var/tmp/tmp-inst** 下にマウントされます。

ディレクトリーの polyinstantiation を有効にするには、以下のステップにしたがいます。

手順4.20 Polyinstantiation ディレクトリーを有効にする

1. **/etc/security/namespace.conf** ファイルの最後の 3 行をコメント解除し、**/tmp**、**/var/tmp/**、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** ファイルで **pam_namespace.so** がセッション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。

4.12. File Name Transition (ファイル名の移行)

file name transition 機能を使うと、ポリシー作成者はポリシー移行ルール作成時にファイル名を指定できます。これで以下の状態を記述するルールを書き込むことが可能になります。**A_t** のラベルの付いたプロセスが **B_t** のラベルが付いたディレクトリー内で特定のオブジェクトクラスを作成し、この特定のオブジェクトクラスを **objectname** と命名すると、これに **C_t** のラベルが付けられます。このメカニズムは、システム上のプロセスに関してより細かい制御をもたらします。

file name transition がない場合、オブジェクトにラベル付けするには以下の 3 つの方法があります。

- 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** のラベルが付いているディレクトリー内でユーザーがファイルを作成すると、そのファイルにも **etc_t** のラベルが付けられます。しかし、この方法はディレクトリー内で異なるラベルが付いた複数のファイルを格納したい場合は役に立たないことになります。

- ポリシー作成者は以下の状態を記述するルールをポリシーで作成することができます。タイプ **A_t** のプロセスが **B_t** のラベルが付いたディレクトリー内で特定のオブジェクトクラスを作成すると、このオブジェクトは新たな **C_t** のラベルが付けられます。単一プログラムが同一のディレクトリー内に複数のオブジェクトを作成し、このオブジェクトがそれぞれ別個のラベルを必要とする場合、この方法は問題になります。さらに、作成されたオブジェクトの名前が指定されないので、これらのルールは部分的な制御しかできません。

- アプリケーションのなかには、特定のパスのラベルが何であるかをアプリケーションがシステムに尋ねることができる SELinux 認識を備えているものもあります。このようなアプリケーションは、必要なラベルが付いたオブジェクトを作成するようにカーネルに要求します。SELinux 認識を備えたアプリケーションには、**rpm** パッケージマネジャー、**restorecon** ユーティリティー、**udev** デバイスマネジャーなどがあります。ただし、すべてのアプリケーションに SELinux 認識のあるファイルやディレクトリーを作成するように指示することは可能です。オブジェクトの作成後に正しいラベルに交換する必要が頻繁にあります。これを行わないと、制限のあ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。

file name transition の機能は、間違ったラベルに関する問題を減らし、システムの安全性を高めます。ポリシー作成者は、あるアプリケーションが特定の名前で特定のディレクトリーにのみ作成できることを適切に記述できます。ルールが勘案するのはファイルパスではなく、ファイル名です。これがファイルパスの basename になります。file name transition は **strcmp()** 関数が実行する完全一致を使用す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勘案されません。

注記

ファイルパスはカーネルで異なる場合があり、file name transition はラベルの判断にこのパスを使用しません。その結果、この機能が影響を与えるのは当初のファイル作成のみで、既存の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4.1 File Name Transition を使ったポリシールール作成の例

以下の例では、file name transition を使った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** ラベルの付いたディレクトリー内に **~/.ssh/** ディレクトリーを作成すると、この**~/.ssh/** ディレクトリーは **ssh_home_t** ラベルが付けられることを記述しています。

以下も、file name transition を使って作成されたポリシールールの例です。

```
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

file name transition の機能は主にポリシー作成者に影響します。ただし、ファイルオブジェクトがほとんど常にそれを格納しているディレクトリーのデフォルトラベルで作成される代わりに、ファイルオブジェクトのなかにはポリシーで指定されたラベルとは異なるものでラベル付けされているものがあることにユーザーは気付くでしょう。

4.13. ptrace() の無効化

ptrace() システムコールを使うと、あるプロセスが別のプロセスの実行を監視および制御できるようにな り、メモリーとレジスタの変更を可能にします。このコールは主に開発者がデバッグする際に使用します。 たとえば、**strace** の使用時などです。**ptrace()** が必要ない時は、これを無効にしてシステムセキュリ ティを高めることができます。これを行うには **deny_ptrace** ブール値を有効にして全プロセスが他のプ ロセスで **ptrace()** を使用することを拒否します。これは**unconfined_t** ドメインで実行中のものにも 適用されます。

deny_ptrace ブール値はデフォルトでは無効になっています。これを有効にするには、root ユーザーで **setsebool -P deny_ptrace on** コマンドを実行します。

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

ブール値が有効になったかどうかを確認するには、以下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** コマンドを実行します。

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

注記

setsebool -P コマンドは、変更を永続的なものにします。再起動後に変更を維持したくない場 合は、**-P** オプションを使用しないでください。

ブール値が影響するのは、Red Hat Enterprise Linux の一部となっているパッケージのみです。このため、 サードパーティーのパッケージはその後も **ptrace()** システムコールを使用できます。**ptrace()** の使用 が可能なドメインを一覧表示するには、以下のコマンドを実行します。*setools-console* パッケージが **sesearch** ユーティリティーを提供しますが、この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
```

4.14. サムネイル保護

サムネイルアイコンは、潜在的に攻撃者が 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 ロックされたマシンに侵入することを許してしまう可能性があります。システムがリムーバブルメディアを 検出すると、マシンがロックされていても、Nautilus ファイルマネジャーがサムネイルドライバーコード を実行して適切なファイルブラウザ内にサムネイルアイコンを表示します。サムネイルの実行可能ファイル に脆弱性がある場合、攻撃者はサムネイルドライバーコードを使ってパスワード入力をせずにロックされた 画面を迂回できるので、この動作は安全ではありません。

このため、このような攻撃を防ぐには新規の SELinux ポリシーを使用します。このポリシーは、画面が ロックされている際には、確実にすべてのサムネイルドライバーがロックされるようにします。このサムネ 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に有効です。このポリシーは、以下のアプリ 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

---

[5] 一時的にデフォルトの動作に戻すには、Linux root ユーザーで **setsebool httpd_can_network_connect_db off** コマンドを実行します。リブート後も変更を維持するには、**setsebool -P httpd_can_network_connect_db off** コマンドを実行します。

[6] **/etc/selinux/targeted/contexts/files/** ディレクトリー内のファイルが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内のファイルは **restorecon** および **setfiles** ユーティリティーが読み取り、ファイルおよびディレクトリーをデフォルトのコンテキストに復元します。

[7] Morris, James. "Filesystem Labeling in SELinux". Published 1 October 2004. Accessed 14 October 2008: http://www.linuxjournal.com/article/7426.

[8] **matchpathcon** についての詳細情報は、the matchpathcon(8) の 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、インストール済みの SELinux ポリシーをクエリする機能のスイートを提供します。これらの機能は新規のものか、これまでは **sepolgen** や **setrans** などの別個のユーティリティーが提供していたものです。このスイートを使うと、移行レポートや man ページ、さらには新ポリシーのモジュールを作成できるようになり、ユーザーは SELinux ポリシーへのアクセスが容易になり、理解が深まります。

policycoreutils-devel パッケージが **sepolicy** を提供します。root ユーザーで以下のコマンドを実行して、**sepolicy** をインストールします。

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

sepolicy スイートは以下の機能を提供し、これらはコマンドラインパラメーターとして起動されます。

表5.1 sepolicy の機能

機能	説明
booleans	SELinux ポリシーに問い合わせてブール値の詳細を表示する
communicate	ドメインが相互通信を行えるかどうかを SELinux ポリシーに問い合わせる
generate	SELinux ポリシーモジュールのテンプレートを生成する
gui	SELinux ポリシーのグラフィカルユーザーインターフェース
interface	SELinux ポリシーインターフェースを一覧表示する
manpage	SELinux man ページを生成する
network	SELinux ポリシーネットワーク情報を問い合わせる
transition	SELinux ポリシーに問い合わせ、プロセス移行レポートを生成する

5.1. sepolicy Python バインディング

以前のバージョンの Red Hat Enterprise Linux では、*setools* パッケージに **sesearch** および **seinfo** ユーティリティーが含まれていました。**sesearch** ユーティリティーは SELinux ポリシー内でのルール検索に使用し、**seinfo** ユーティリティーはポリシー内の他のコンポーネントへのクエリを可能にしていました。

Red Hat Enterprise Linux 7 では、**sesearch** および **seinfo** に Python バインディングが追加され、これらユーティリティーの機能を **sepolicy** スイート経由で使用することができます。例を示します。

```
> 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 ポリシーの生成に **sepolgen** または **selinux-polgengui** ユーティリティーが使われていました。これらのツールは、**sepolicy** スイートに統合されました。Red Hat Enterprise Linux 7 では、**sepolicy generate** コマンドを使って最初の SELinux ポリシーモジュールテンプレートを生成します。

sepolgen とは違い、**sepolicy generate** は root ユーザーで実行する必要はありません。このユーティリティーは RPM 仕様ファイルも作成します。これは、ポリシーパッケージファイル (***NAME*.pp**) および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し、SELinux ポリシーのカーネルへのインストールを提供し、ラベルの修正を行う RPM パッケージの構築に使用することができます。設定スクリプトが SELinux ポリシーのインストールを継続し、ラベリングを設定します。さらに、**sepolicy manpage** コマンドを使うと、インストールされたポリシーに基づいた man ページが生成されます。[9] 最後に、**sepolicy generate** は SELinux ポリシーと man ページを RPM パッケージに構築、コンパイルして、他のシステムにインストールする用意をします。

sepolicy generate が実行されると、以下のファイルが作成されます。

***NAME*.te – タイプ強制ファイル**

このファイルは、特定のドメインにおけるタイプおよびルールすべてを定義します。

***NAME*.if – インターフェイスファイル**

このファイルは、システム用にデフォルトのファイルコンテキストを定義します。**NAME.te** ファイル内で作成されたファイルタイプを取り、ファイルパスをタイプに関連付けます。**restorecon** や **rpm** といったユーティリティーは、これらのパスを使ってラベルを書き込みます。

***NAME*_selinux.spec – RPM 仕様ファイル**

このファイルは、SELinux ポリシーをインストールし、ラベル付けを設定する RPM 仕様ファイルです。また、インターフェイスファイルとポリシーを記述する man ページもインストールします。**sepolicy manpage -d *NAME*** コマンドを使うと man ページを生成することができます。

***NAME*.sh – ヘルパーシェルスクリプト**

このスクリプトは、システム上のラベル付けをコンパイル、インストール、修正する手助けとなります。また、インストールされたポリシーに基づいた man ページを生成し、他のシステムにインストールできる RPM パッケージをコンパイル、構築します。

SELinux ポリシーモジュールを生成できる場合は、**sepolicy generate** はソースドメインからターゲットドメインへの生成されたすべてのパスをプリントアウトします。**sepolicy generate** についての詳細は、sepolicy-generate(8) の man ページを参照してください。

5.3. ドメイン移行について:sepolicy transition

これまでは、2 つのドメインタイプまたはプロセスタイプの間で移行が買おうかどうかを調べるためには **setrans** ユーティリティーを使ってこれらのドメインもしくはプロセス間の移行に使用する中間タイプをすべてプリントアウトしていました。Red Hat Enterprise Linux 7 では、**setrans** が **sepolicy** スイートの一部として提供され、**sepolicy transition** コマンドが使用されます。

sepolicy transition コマンドは SELinux ポリシーにクエリを行い、プロセス移行レポートを作成します。**sepolicy transition** コマンドは、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) という 2 つのコマンドライン引数を必要とします。ソースドメインのみが入力された場合は、**sepolicy transition** はソースドメインが移行可能なドメインすべてを一覧表示します。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。「@」 記号は 「実行」 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ると、**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関して SELinux ポリシーを調査し、これらのパスを一覧表示します。以下の出力は、完全なものではありません。

```
~]$ 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 についての詳細は、sepolicy-transition(8) の man ページを参照してください。

5.4. Man ページの生成: sepolicy manpage

sepolicy manpage コマンドは、SELinux ポリシーに基づいてプロセスドメインを文書化した man ページを生成します。このため、このドキュメンテーションは常に最新のものになります。自動生成された man ページの名前は **httpd_selinux** のように、プロセスドメイン名と **_selinux** 接尾辞からなります。

Man ページには、制限のあるドメイン用の SELinux ポリシーの様々な部分についての情報を提供するいくつかのセクションが含まれます。

- **Entrypoints** セクションには、ドメイン移行時に実行する必要のある実行可能ファイルすべてが含まれています。
- **Process Types** セクションには、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すべてが含まれています。
- **Booleans** セクションには、ドメインに関連するブール値が一覧表示されています。
- **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が記述されています。
- **Managed Files** セクションでは、ドメインが書き込み可能な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が説明されています。
- **File Contexts** セクションにはドメインに関連付けられたファイルタイプすべてが含まれ、システム上でデフォルトのパスラベリングと一緒に使用するファイルタイプ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** のようなドメイン共有タイプを使用する方法が説明されています。

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

[9] **sepolicy manpage** についての詳細は、「Man 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、数多くの制限のある SELinux ユーザーを利用することができます。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、SELinux ユーザーに課された制限が Linux ユーザーに継承されます。制限の例は、(ユーザーによりますが) X Window System が実行できない、ネットワーキングが使用できない、(SELinux ポリシーが許可していなければ) setuid アプリケーションを実行できない、**su** や **sudo** などのコマンドを実行できない、などがあります。これによって、システムをユーザーから保護することができます。制限のあるユーザーについての詳細は、「制限のあるユーザーおよび制限のないユーザー」を参照してください。

6.1. Linux および SELinux ユーザーのマッピング

root ユーザーで以下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。

```
~]# 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 **__default__** ログインにマッピングされ、これはさらに SELinux **unconfined_u** ユーザーにマッピングされます。**useradd** コマンドで Linux ユーザーが作成され、オプションが特定されないと、このユーザーは SELinux **unconfined_u** にマッピングされます。以下でデフォルトのマッピングを定義します。

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

6.2. 新規 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** ドメインで稼働します。**unconfined_u** にマッピングされた Linux ユーザーでログインし、**id -Z** コマンドを実行すると、以下の出力が表示されます。

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

Linux ユーザーが **unconfined_t** ドメインで稼働すると SELinux ポリシールールが適用されますが、**unconfined_t** ドメインで稼働する Linux ユーザーにほとんど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。SELinux ポリシーで **unconfined_t** ドメインから自身の制限のあるドメインへの移行が可能だと定義されているアプリケーションを、制限のない Linux ユーザーが実行しても、この制限のない Linux ユーザーは制限のあるドメインの規定に拘束されます。ここでのセキュリティの利点は、Linux ユーザーは制限なしで実行していてもアプリケーションには制限があることから、アプリケーションの欠点を悪用しようとしてもポリシーで制限できる、という点です。

注記

上記の点は、システムがユーザーから保護されるということではありません。ユーザーとシステムがアプリケーションの欠点による損害の可能性から守られるということです。

useradd コマンドで Linux ユーザーを作成する場合は、**-Z** オプションを使ってどの SELinux ユーザーにマッピングするかを指定します。以下の例では、新規の Linux ユーザー **useruuser** を作成し、そのユーザーを SELinux **user_u** ユーザーにマッピングしています。SELinux**user_u** ユーザーにマッピングされた Linux ユーザーは、**user_t** ドメインで稼働します。このドメインでは、(**passwd** など) SELinux ポリシーが許可しない限り、Linux ユーザーは setuid アプリケーションを実行できず、**su** や **sudo** コマンドも実行できないので、これらのコマンドで root ユーザーになることを防いでいます。

手順6.1 新規 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された新規 Linux ユーザー (**useruuser**) を作成します。

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

2. **useruuser** と **user_u** 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 でパスワードを Linux **useruuser** ユーザーに割り当てます。

```
~]# 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一旦ログアウトし、Linux **useruuser** ユーザーでログインし直します。ログインすると、**pam_selinux** モジュールがこの Linux ユーザーを SELinux ユーザーにマッピングし (このケースでは **user_u**)、SELinux コンテキストを設定します。その後は、このコンテキストで Linux ユーザーのシェルが起動されます。以下のコマンドを実行して、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** のセッションからログアウトし、自分のアカウントでログインし直します。Linux **useruuser** ユーザーが不要な場合は、root で以下のコマンドを実行し、そのホームディレクトリーとともに削除します。

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

6.3. 既存 Linux ユーザーの制限: semanage login

Linux ユーザーが SELinux **unconfined_u** ユーザーにマッピングされ (デフォルトの動作)、マッピング先の SELinux ユーザーを変更したい場合は、**semanage login** コマンドを使います。以下の例では、**newuser** という名前の新規 Linux ユーザーが作成され、SELinux**user_u** ユーザーにマッピングされます。

手順6.2 Linux ユーザーを SELinux ユーザーにマッピングする

1. root ユーザーで、ユーザー名 **newuser** という新規 Linux ユーザーを作成します。このユーザーはデフォルトマッピングを使用しているため、**semanage login -l** 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 オプションは新規レコードを追加し、**-s** オプションは Linux ユーザーがマッピングされる SELinux ユーザーを指定します。最後の引数である **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** と **user_u** の間のマッピングを表示するには、再度**semanage** ユーティリティーを使用します。

```
~]# 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. 現行セッションから一旦ログアウトし、Linux **newuser** ユーザーでログインし直します。以下のコマンドを実行し、**newuser** の SELinux コンテキストを表示します。

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

6. Linux **newuser** のセッションからログアウトし、自分のアカウントでログインし直します。Linux **newuser** ユーザーが不要な場合は、root で以下のコマンドを実行し、その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** ユーザーと **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 **__default__** ログインにマッピングされます (このログインは、SELinux **unconfined_u** ユーザーにマッピングされます)。新規 Linux ユーザーの場合で特に SELinux ユーザーにマッピングされておらず、デフォルトで制限をかけたい場合、デフォルトマッピングを **semanage login** コマンドで変更します。

例えば以下のコマンドを root で実行して、デフォルトマッピングを **unconfined_u** から **user_u** に変更します。

```
~]# 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       *
```

新規 Linux ユーザーが作成され、SELinux ユーザーが特定されていない場合、もしくは既存の Linux ユーザーがログインし **semanage login -l** 出力からの特定のエントリーに適合しない場合、**__default__** ログインの場合のように **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 で以下のコマンドを実行して **__default__** ログインを SELinux **unconfined_u** ユーザーにマッピングします。

```
~]# 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 パッケージはキオスクユーザーアカウントを提供します。このアカウントは、図書館や銀行、空港、情報キオスク、コーヒーショップなどの場所で、誰もが立ち寄って使えるマシンを確保するために使われます。キオスクユーザーアカウントは非常に限定的なもので、基本的にユーザーができるのはログインして **Firefox** でインターネットの Web サイトを閲覧することだけです。ファイルの作成や設定変更など、ログイン中にアカウントで行われた変更は、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、以下の手順にしたがいます。

1. root ユーザーで *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. 誰もがキオスクアカウントを使えるようにするため、アカウントはパスワード保護されていません。このため、SELinux が enforcing モードで実行されている場合のみ、アカウントが保護されます。このアカウントにログインする前に、**getenforce** ユーティリティーを使って 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ない場合は、「SELinux の有効化および無効化」を参照して enforcing モードに変更します。SELinux が permissive モードだったり無効だったりすると、このアカウントにログインすることができません。

3. このアカウントには、GNOME Display Manager (GDM) 経由でしか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、**ゲスト** 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のためのブール値

Linux ユーザーが自分のホームディレクトリーや書き込みアクセスのある **/tmp/** で (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ことで、ユーザー所有のファイルに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修正できないようになっています。Red Hat Enterprise Linux ではデフォルトで、**guest_t** と **xguest_t** ドメインの Linux ユーザー はホームディレクトリーや**/tmp/** でアプリケーションを実行できません。しかしデフォルトでは、**user_t** と **staff_t** ドメインでは実行可能となっています。

この動作の変更はブール値で可能となっており、 **setsebool** ユーティリティーで設定します。これは、root ユーザーで実行する必要があります。**setsebool -P** コマンドは、変更を永続的なものにします。再起動後に変更を維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。

guest_t

guest_t ドメインの Linux ユーザーがホームディレクトリーと**/tmp/** でアプリケーションを実行**できる**ようにするには、以下を実行します。

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

xguest_t

xguest_t ドメインの Linux ユーザーがホームディレクトリーと**/tmp/** でアプリケーションを実行**できる**ようにするには、以下を実行します。

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

user_t

user_t ドメインの Linux ユーザーがホームディレクトリーと**/tmp/** でアプリケーションを実行**できない**ようにするには、以下を実行します。

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

staff_t

staff_t ドメインの Linux ユーザーがホームディレクトリーと**/tmp/** でアプリケーションを実行**できない**ようにするには、以下を実行します。

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

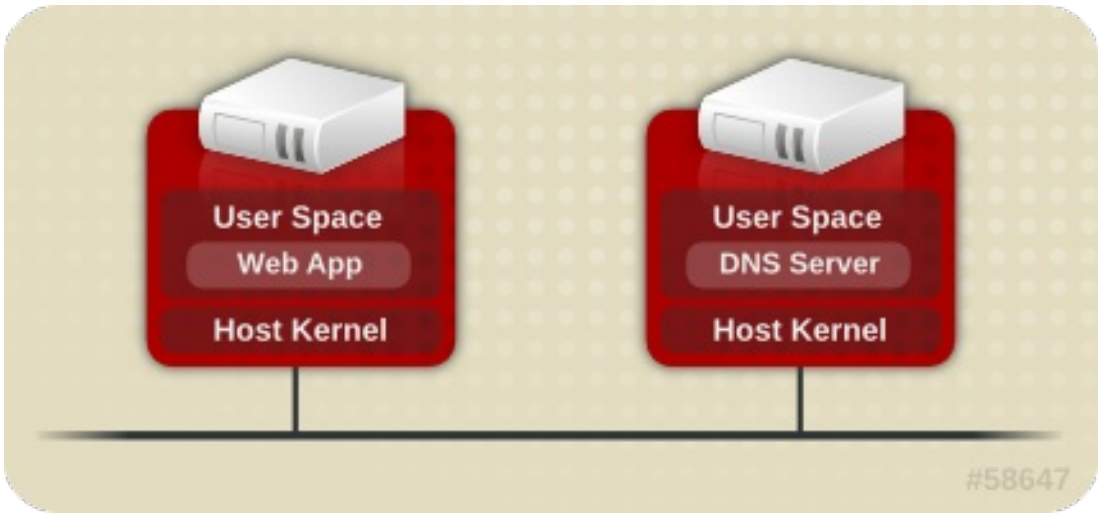
第7章 sVirt

sVirt は Red Hat Enterprise Linux に導入されている技術で、SELinux と仮想化を統合します。仮想マシンの使用時には Mandatory Access Control (MAC) を適用してセキュリティを高めます。これらの技術を統合する主な理由は、ホストや他の仮想マシンを目標とした攻撃経路として使用される可能性のあるハイパーバイザー内のバグに対してシステムを堅牢にし、セキュリティを高めるためです。

本章では、Red Hat Enterprise Linux での sVirt による仮想化技術の統合について説明します。

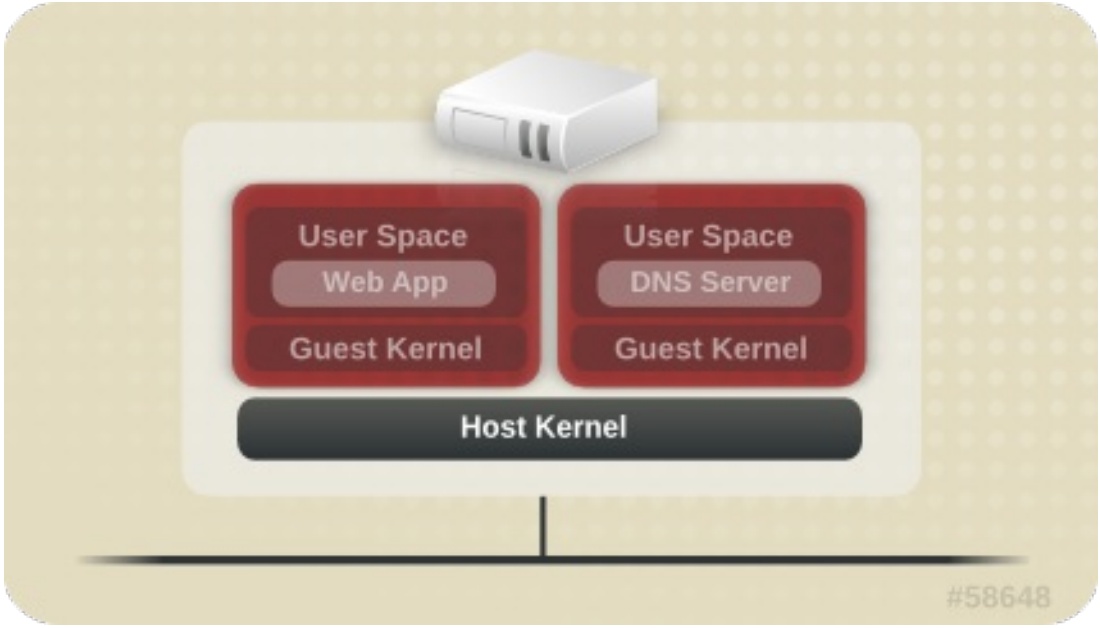
非仮想化環境

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構成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。下の図は、非仮想化環境を示したものです。



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 (「ゲスト」として) 単一のホストカーネルおよび物理ホストに格納することができます。下の図は仮想化環境を示したものです。

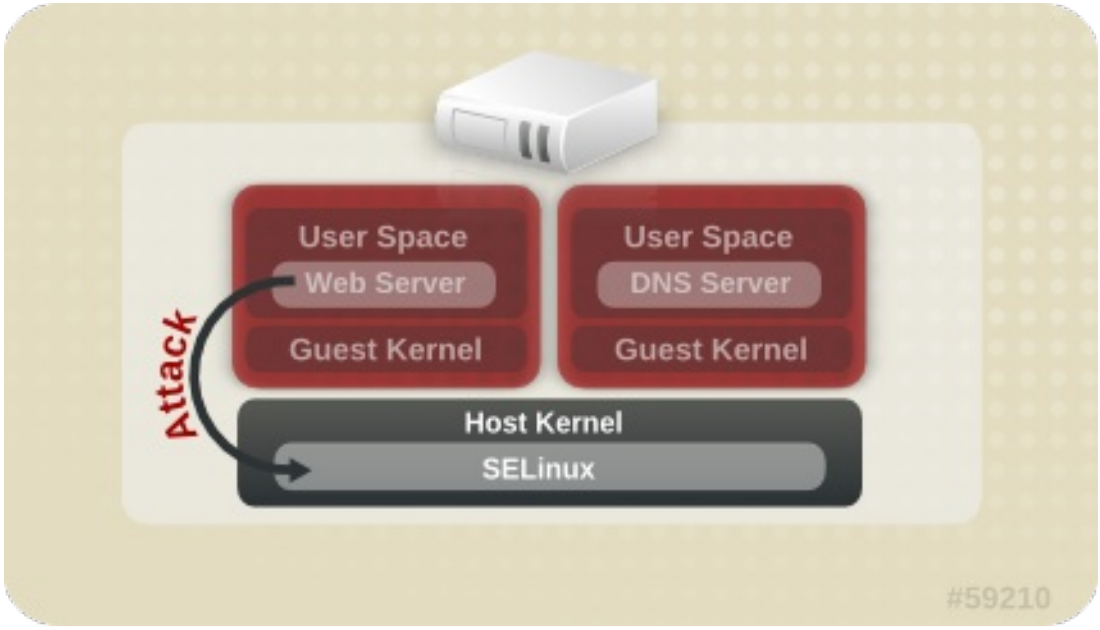


7.1. セキュリティと仮想化

7.1. セキュリティと仮想化

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。エクスプロイトは通常、影響を受けたマシンで封じ込められます。ただし、ネットワーク攻撃は明らかに例外となります。仮想化環境内でサービスがグループ化されると、システムに新たな脆弱性が出現します。ハイパーバイザーのセキュリティに不備があって、ゲストインスタンスによるエクスプロイトを受ける可能性がある場合、そのゲストにホストのみならず、そのホスト上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。これは理論上の話ではありません。攻撃はすでにハイパーバイザー上に存在しているのです。これらの攻撃はゲストインスタンスを超えて拡大し、他のゲストを攻撃にさらす可能性があります。

sVirt は、ゲストを隔離して、悪用された場合にさらなる攻撃を開始する能力を抑制するためのものです。以下のイメージで示すように、攻撃は仮想マシンから出ることができず、他のゲストインスタンスにも届きません。



SELinux は、MAC (Mandatory Access Control) の実装内で仮想化インスタンス向けのプラグ可能なセキュリティフレームワークを導入します。 sVirt のフレームワークにより、ゲストとそのリソースに固有のラベル付けが可能になります。ラベルが付けられると、ルールの適用が可能になり、異なるゲスト間のアクセスを拒否できるようになります。

7.2. sVirt のラベル付け

SELinux の保護下にある他のサービスと同様に、sVirt はプロセスベースのメカニズムと制約を使用して、ゲストインスタンス全体に追加のセキュリティ層を提供します。通常の使用では、sVirt がバックグラウンドで作動していることすら分かりません。このセクションはで、sVirt のラベル付け機能について説明します。

以下の出力にあるように、sVirt を使用すると各仮想マシンのプロセスにラベルが付けられ、動的生成のレベルで稼働するようになります。各プロセスは異なるレベルで他の仮想マシンから隔離されています。

```
~]# 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
```

以下の出力で示すように、実際のディスクイメージはプロセスに一致するよう自動的にラベル付けされます。

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

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

以下の表では、sVirt の使用時に割り当て可能な各種のラベルの概要を示しています。

表7.1 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 は無作為に選択されたフィールドです。現時点では、約 50 万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	これらのイメージファイルやデバイスの読み取り/書き込みができるのは、同じ MCS フィールドの **svirt_t** ラベルが付いたプロセスだけです。
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** のラベルが付いたプロセスはすべて、svirt_image_t:s0 のファイルおよびデバイスに書き込み可能です。
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 の使用時に、静的なラベル付けを行なうこともできます。 静的なラベルを使用すると、管理者は仮想化ゲストに特定のラベルを選択することができます。これには MCS/MLS フィールドも含まれます。静的にラベル付けした仮想化ゲストを実行する場合は、管理者はイメージファイルにも正しいラベルを設定する必要があります。仮想マシンは常にそのラベルで起動し、静的にラベル付けした仮想化マシンのコンテンツの修正を sVirt システムが行なうことはありません。 これにより、sVirt コンポーネントが MLS 環境で実行できるようになります。また、要件に応じてひとつのシステム上で異なる機密性レベルを持つ複数の仮想マシンを実行することもできます。

第8章 Secure Linux コンテナー

Linux コンテナー (LXC) は低レベルの仮想化機能で、これを使うことでシステム上で同時に同一サービスの複数コピーを実行することが可能になります。完全な仮想化と比べるとコンテナーは新システム全体が起動する必要がなく、メモリー消費量が少なくてすみ、読み取り専用でベースのオペレーティングシステムが使用できます。たとえば、LXC だと複数の web サーバーを同時に稼働することが可能で、これらはシステムデータを共有する一方で独自のデータも備えています。また、root ユーザーとして実行することも可能です。ただし、コンテナー内で権限のあるプロセスを実行すると、コンテナー外で実行中の他のプロセスや他のコンテナー内で実行中のプロセスに影響する場合があります。Secure Linux コンテナーは SELinux コンテキストを使用するため、コンテナー内で実行するプロセスが相互に対話したり、ホストと対話することを防ぎます。

Red Hat Enterprise Linux におけるLinux コンテナー管理のメインユーティリティーは **Docker** アプリケーションです。代替方法としては、*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も使用できます。

Linux コンテナーに関する詳細は、Resource Management and Linux Containers Guide を参照してください。

第9章 SELinux systemd によるアクセス制御

Red Hat Enterprise Linux 7 では、システムサービスは **systemd** デーモンで制御します。
Red Hat Enterprise Linux の以前のリリースでは、デーモンは以下の 2 通りの方法で起動されていました。

- ブート時に System V **init** デーモンが **init.rc** スクリプトを開始し、このスクリプトが希望するデーモンを開始しました。たとえば、ブート時に起動される Apache サーバーには、以下の SELinux ラベルがありました。

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

- 管理者が手動で **init.rc** スクリプトを開始し、デーモンが実行されていました。たとえば、**service httpd restart** コマンドが Apache サーバー上で開始されると、その結果、SELinux ラベルは以下のようになりました。

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

プロセスは手動で開始されると、それを開始した SELinux ラベルのユーザーの部分を採用し、上記の 2 つのシナリオにおけるラベリングに食い違いをもたらします。**systemd** デーモンを使うと、移行は非常に異なります。**systemd** がシステム上で開始および停止するコールを**init_t** を使ってすべて処理するため、デーモンが手動で再起動された際にラベルのユーザーの部分を上書きできます。その結果、上記の両方のシナリオでラベルが期待どおりに **system_u:system_r:httpd_t:s0** となり、どのドメインがどのユニットを制御するかについての SELinux ポリシーが改善されます。

9.1. サービスに関する SELinux アクセスパーミッション

Red Hat Enterprise Linux の以前のバージョンでは、管理者は System V Init スクリプトに基づいてどのユーザーやアプリケーションがサービスを開始、停止できるかを制御することが可能でした。現在は、**systemd** がすべてのサービスを開始、停止し、ユーザーとプロセスは**systemctl** ユーティリティーを使って **systemd** と通信します。**systemd** デーモンには SELinux ポリシーを参考にし、呼び出しているプロセスのラベルと発信元が操作しようとしているユニットファイルのラベルをチェックした後で、SELinux に対して発信元のアクセスを許可するかどうかを尋ねる機能があります。このアプローチは、システムサービスを開始、停止するといったものを含む重大なシステム機能へのアクセス制御を強化します。

たとえば、これまでは管理者は NetworkManager が **systemctl** を実行して D-Bus メッセージを **systemd** に送信できるようして、NetworkManager が要求したサービスをこのデーモンが開始したり停止していました。実際、NetworkManager は **systemctl** が実行可能なすべてのことをできるように許可されていました。また、特定のサービスを開始したり停止したりすることが可能な制限ある管理者を設定することは不可能でした。

これらの問題を解決するために、**systemd** は SELinux Access Manager としても機能するようになりました。これは、D-Bus メッセージを **systemd** に送信するプロセスや**systemctl** を実行しているプロセスのラベルを取得することができます。このデーモンはその次にプロセスが設定を希望するユニットファイルのラベルを探します。最後に、SELinux ポリシーがプロセスラベルとユニットファイルのラベルの間で特定のアクセスを許可する場合、**systemd** はカーネルから情報を取得することができます。つまり、特定のサービスについて **systemd** と対話する必要のあるアプリケーションで危険にされされているものは、SELinux 経由で制限ができるようになっています。ポリシー作成者は、これらの細かい制御を使って管理者を制限することができます。ポリシー変更には **service** と呼ばれる新たなクラスが関わり、以下のパーミッションを伴います。

```
class service
{
        start
        stop
```

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

たとえば、ポリシー作成者はドメインがサービスの状態を獲得したり、サービスを開始、停止することを許可できるようになりましたが、サービスを有効、無効にすることはできません。SELinux および **systemd** でのアクセス制御の操作は、すべてのケースで一致するわけではありません。マッピングは、**systemd** メソッド呼び出しと SELinux アクセスチェックが並ぶように定義されています。表9.1「systemd ユニットファイルメソッド呼び出しと SELinux アクセスチェックのマッピング」では、ユニットファイルにおけるアクセスチェックのマッピングを表示しています。表9.2「systemd の全般的なシステム呼び出しと SELinux アクセスチェックのマッピング」では、システム全般におけるアクセスチェックを表示しています。これらの表で一致するものがない場合は、**undefined** システムチェックが呼び出されます。

表9.1 systemd ユニットファイルメソッド呼び出しと SELinux アクセスチェックのマッピング

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
ReloadOrRestart	start
ReloadOrRestartUnit	start
ReloadOrTryRestart	start
ReloadOrTryRestartUnit	start
ReloadUnit	reload
ResetFailed	stop
ResetFailedUnit	stop
Restart	start
RestartUnit	start
Start	start
StartUnit	start
StartUnitReplace	start
Stop	stop
StopUnit	stop
TryRestart	start
TryRestartUnit	start

systemd ユニットファイルメソッド	SELinux アクセスチェック
UnmaskUnitFiles	enable

表9.2 systemd の全般的なシステム呼び出しと SELinux アクセスチェックのマッピング

systemd の全般的なシステム呼び出し	SELinux アクセスチェック
ClearJobs	reboot
FlushDevices	halt
Get	status
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
Reboot	reboot
SetUserLinger	halt
TerminateSeat	halt
TerminateSession	halt
TerminateUser	halt

例9.1 システムサービス用の SELinux ポリシー

sesearch ユーティリティーを使うと、システムサービス用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** コマンドを実行すると、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 } ;
```

9.2. SELinux と journald

9.2. SELinux と journald

systemd では、**journald** デーモン (**systemd-journal** とも呼ぶ) が **syslog** ユーティリティーの代わりとなり、これはロギングデータを収集、保存するシステムサービスになります。カーネルや **libc** **syslog()** 機能を使ってユーザープロセスから受け取ったロギング情報、システムサービスの標準およびエラー出力から受け取ったロギング情報、またはネイティブの API から受け取ったロギング情報を元に構造化およびインデックス化されたジャーナルを作成、維持します。また、暗黙的に安全な方法で各ロングメッセージの多くのメタデータフィールドを収集します。

systemd-journal サービスは SELinux と使うことでセキュリティを高めることができます。SELinux は、プロセスが設計されたことのみを実行するようにすることでこれらを制御します。ポリシー作成者の制御目標によっては、実行できるものがこれよりも少なくなることもあります。たとえば SELinux は、危険にさらされた **ntpd** プロセスが Network Time 以外の処理をできないようにします。しかし、**ntpd** プロセスは **syslog** メッセージを送信するので、SELinux は危険にさらされたこのプロセスがこれらのメッセージを送信し続けることを許可します。危険にさらされた **ntpd** は **syslog** メッセージをフォーマットして他のデーモンに一致させ、管理者の判断を誤らせる可能性があります。さらには、**syslog** ファイルを読み込むユーティリティーの判断を誤らせ、システム全体を危険にさらす可能性もあります。

systemd-journal デーモンは、すべてのログメッセージを検証するとともに、それらに SELinux ラベルを追加します。こうすることでログメッセージにおける矛盾の検出が容易になり、このタイプの攻撃が発生する前に防ぐことができます。**journalctl** ユーティリティーを使うと、**systemd** ジャーナルのログにクエリを実行することができます。コマンドライン引数を指定せずにこのコマンドを実行すると、ジャーナルのすべてのコンテンツが古いエントリーから順に一覧表示されます。システムコンポーネントのログを含むシステム上で生成されたすべてのログを見るには、root で **journalctl** を実行します。root 以外のユーザーでこのコマンドを実行すると、出力は現在ログイン中のユーザーに関連するログのみに限定されます。

例9.2 journalctl を使ったログの一覧表示

journalctl を使って特定の SELinux ラベルに関連するすべてのログを一覧表示することができます。たとえば、以下のコマンドは、**system_u:system_r:policykit_t:s0** ラベルで記録されたすべてのログを一覧表示します。

```
~]# 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 についての詳細は、journalctl(1) の man ページを参照してください。

第10章 トラブルシューティング

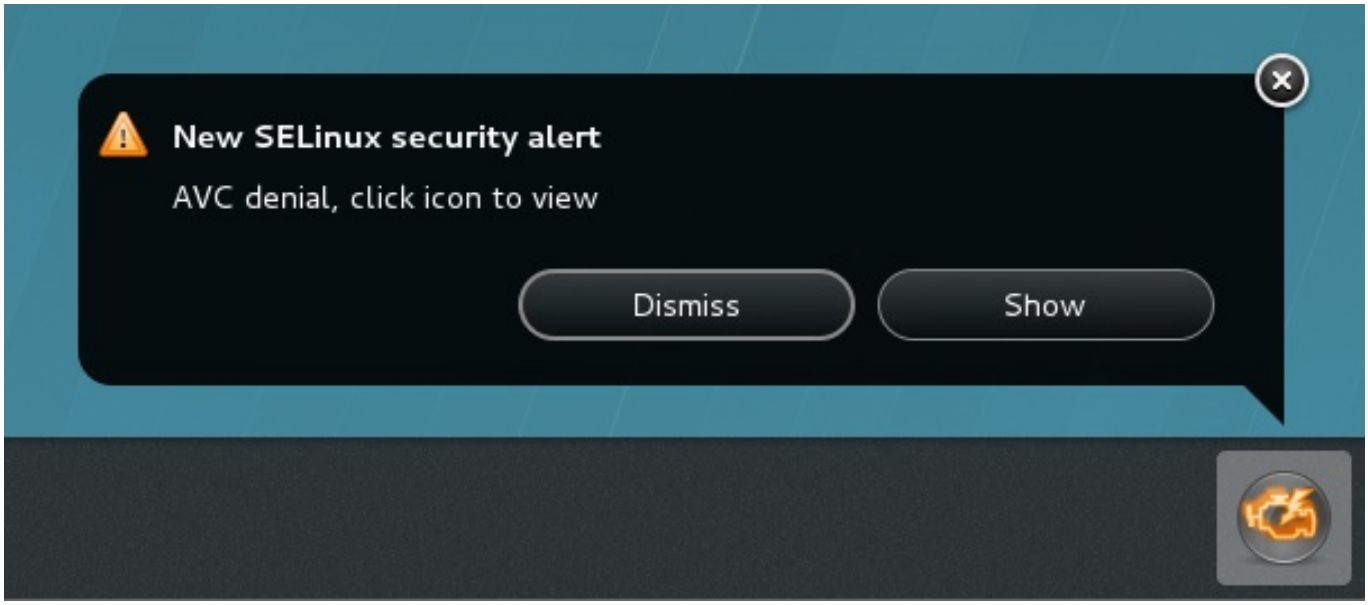
本章では、SELinux がアクセスを拒否した場合に何が起こるか、問題の 3 つの主要原因、正しいラベリングについての情報の場所、SELinux 拒否の分析、**audit2allow** を使ったカスタムポリシーモジュールの作成について説明します。

10.1. アクセス拒否の場合

アクセスを許可する、しないといった SELinux の決定は、キャッシュされます。このキャッシュは、AVC (アクセスベクターキャッシュ) と呼ばれます。SELinux がアクセスを拒否すると、拒否メッセージはログに記録されます。これらの拒否は「AVC拒否」とも呼ばれ、実行中のデーモンに応じて別の場所にログ記録されます。

デーモン	ログ記録の場所
auditd オン	**/var/log/audit/audit.log**
auditd オフ; rsyslogd オン	**/var/log/messages**
setroubleshootd、rsyslogd、auditd すべてオン	**/var/log/audit/audit.log** 読みやすい拒否メッセージが **/var/log/messages** にも送信されます。

X Window System を実行中で *setroubleshoot* と *setroubleshoot-server* パッケージがインストールされ、**setroubleshootd** と **auditd** デーモンが稼働している場合、SELinux がアクセスを拒否すると警告が表示されます。



表示 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解決法が示されます。X Window System を実行していないと、SELinux のアクセス拒否は分かりにくくなります。例えば、Web サイトをブラウジングしているユーザーが以下のようなエラーを受け取る場合があります。

```
Forbidden

You don't have permission to access file name on this server
```

このような状況では、DAC ルール (標準の Linux パーミッション) がアクセスを許可していれば、**"SELinux is preventing"** エラーの場合は **/var/log/messages** を、**"denied"** エラーの場合は **/var/log/audit/audit.log** をそれぞれチェックします。これは root ユーザーで以下のコマンドで実行できます。

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

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

10.2. 問題の原因トップ 3

以下のセクションでは、問題の原因のトップ 3 を説明します。これらは、ラベル付けの問題、ブール値およびサービスのポートの設定、SELinux ルールの展開になります。

10.2.1. ラベル付けの問題

SELinux 実行中のシステム上では、すべてのプロセスとファイルにセキュリティ関連の情報を含む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。このラベルが間違っていると、アクセスは拒否されます。アプリケーションのラベルが間違っていると、移行先のプロセスにも正しいラベルがないことになり、結果として SELinux がアクセスを拒否することになりかねません。さらにはこのプロセスが、間違ったラベルの付いたファイルを作成することにもなります。

一般的なラベル付けの問題は、非標準のディレクトリーをサービスに使う場合に発生します。例えば、Web サイトに **/var/www/html/** を使うのではなく、管理者は **/srv/myweb/** を使いたかったとします。Red Hat Enterprise Linux では、**/srv/** ディレクトリーは **var_t** タイプでラベル付けされます。作成されたファイルとディレクトリーおよび **/srv/** はこのタイプを継承します。また、(**/myserver/** のような) 新規作成のトップレベルのディレクトリーは **default_t** タイプでラベル付けされます。SELinux は、Apache HTTP Server (**httpd**) がこれら両方のタイプにアクセスすることを禁止します。アクセスを許可するには、**httpd** が **/srv/myweb/** にあるファイルにアクセス可能であることを SELinux が認識している必要があります。

```
~]# 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 (およびその下にある全ファイルとディレクトリー) のコンテキストを SELinux ファイル設定に追加します [10]。**semanage** ユーティリティーはコンテキストを変更しません。変更を適用するには、root で **restorecon** ユーティリティーを実行します。

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

ファイルコンテキスト設定へのコンテキスト追加に関する詳細情報は、「永続的な変更: semanage fcontext」 を参照してください。

10.2.1.1. 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チェックし、そのパスのデフォルトラベルと比較します。以下の例では、間違ったラベル付けがされているファイルを含んだディレクトリーでの **matchpathcon** の使用を説明しています。

```
~]$ 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** および **page1.html** ファイルは **user_home_t** タイプでラベル付けされています。このタイプは、ユーザーのホームディレクトリーで使われるものです。**mv** コマンドを使っ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** タイプのラベル付けがされます。このタイプはホームディレクトリーの外にあってはならないので、**restorecon** ユーティリティーを使って、ファイルを正しいタイプに戻します。

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

ディレクトリー下の全ファイルのコンテキストを復元するには、**-R** を使います。

```
~]# 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 の詳細例に関しては、「デフォルト SELinux コンテキストのチェック」を参照してください。

10.2.2. 制限のあるサービスの実行方法

サービスは様々な方法で実行可能なので、サービスの実行方法を指定する必要があります。ランタイム時に SELinux ポリシーの一部変更を許可するブール値でこれを実行でき、SELinux ポリシー記述の知識がなくても可能です。これにより、SELinux ポリシーの再ロードや再コンパイルをせずに、NFS ボリュームへのサービスによるアクセスを許可するといった変更が可能になります。また、デフォルトでないポート番号でのサービス実行は、**semanage** コマンドでポリシー設定を更新する必要があります。

例えば、Apache HTTP Server の MariaDB との通信を許可するには、**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** および **grep** ユーティリティーを使って、アクセスを許可するブール値が利用可能かどうかを調べます。例えば、**getsebool -a | grep ftp** コマンドと使って 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
ftp_home_dir --> off
ftpd_connect_db --> off
httpd_enable_ftp_server --> off
tftp_anon_write --> off
```

ブール値の一覧表示とそれらがオンかオフかを表示するには、**getsebool -a** コマンドを実行します。ブール値の一覧表示、各ブール値の説明、それらがオンかオフかについては、root で **semanage boolean -l** を実行します。ブール値の一覧表示と設定については、「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が許可されます。サービスが実行されているポートをポリシーを変更せずに変えようとすると、サービスのスタート失敗につながる場合があります。例えば、root で **semanage port -l | grep http** コマンドを実行し、**http** 関連ポートを一覧表示します。

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

http_port_t ポートタイプは、Apache HTTP Server がリッスン可能なポートを定義します。このケースでは、TCP ポート 80、443、488、8008、8009、8443 になります。管理者が **httpd.conf** を設定し **httpd** がポート 9876 (**Listen 9876**) をリッスンするようにしても、ポリシーがこれを反映するように更新されていないと、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 拒否メッセージは、**/var/log/audit/audit.log** にログ記録されます。

```
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
```

http_port_t ポートタイプに一覧表示されていないポートを**httpd** が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、ポートをポリシー設定に追加します [11]。

```
~]# 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ます。**-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。

10.2.3. ルールの発展と壊れたアプリケーション

アプリケーションが壊れると、SELinux はアクセスを拒否します。また、SELinux ルールは発展しており、SELinux が見たことのない方法でアプリケーションが稼働する場合もあります。この場合、アプリケーションが期待通りの動作をしていても、SELinux にアクセスを拒否される可能性があります。例えば、PostgreSQL の新バージョンがリリースされ、現行ポリシーが見たことのないアクションを実行すると、アクセスは本来許可されるべきなのに拒否されます。

こういった場合、アクセスが拒否された後で、 **audit2allow** ユーティリティーを使っ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** の使用については、「アクセス許可: audit2allow」 を参照してください。

10.3. 問題の修正

以下のセクションでは、問題の解決方法を説明します。取り上げるトピックは以下のとおりです。Linux パーミッションのチェック - これは SELinux ルールの前にチェックされます。拒否がログ記録されない場合に SELinux がアクセスを拒否する理由。サービスの man ページ - これにはラベル付けとブール値の情報が含まれます。あるプロセスがシステム全体ではなく permissive で実行することを許可するための permissive ドメイン。拒否メッセージの検索方法および表示方法。拒否の分析。**audit2allow** によるカスタムポリシーモジュールの作成。

10.3.1. Linux パーミッション

アクセスが拒否されたら、標準 Linux パーミッションをチェックしてください。1章*はじめに*の説明にあるように、ほとんどのオペレーティングシステムでは任意アクセス制御 (DAC) を使ってアクセスを制御しており、ユーザーが所有しているファイルのパーミッションを自分で管理できるようになっています。SELinux ポリシールールは DAC ルールの後にチェックされます。最初に DAC ルールがアクセスを拒否すれば、SELinux ポリシールールは使われません。

アクセスが拒否され、SELinux 拒否がログ記録されていない場合、以下のコマンドを使って標準 Linux パーミッションを表示します。

```
~]$ 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 ユーザーとグループが所有しています。root ユーザーには読み取りおよび書き込みパーミッション (**-rw**) があり、root グループのメンバーには読み取りパーミッション (**-r-**) があります。それ以外の人にはアクセスがありません (**---**)。デフォルトでは、これらのパーミッションは **httpd** によるこのファイルの読み取りを許可しません。この問題を解決するには、**chown** コマンドで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
```

ここでは、**httpd** を Linux Apache ユーザーとして実行するというデフォルト設定を前提としています。**httpd** を別のユーザーで実行する場合は、**apache:apache** をそのユーザーで置き換えます。

Linux パーミッションの詳細に関しては、Fedora Documentation Project "Permissions" のドラフトを参照してください。

10.3.2. サイレント拒否の原因

状況によっては、SELinux がアクセスを拒否した際に AVC 拒否メッセージがログ記録されない場合もあります。アプリケーションやシステムライブラリー機能は、タスクの実行に必要なアクセス以上のものをプローブすることがよくあります。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。このルールは、標準ポリシーに共通のものです。**dontaudit** のマイナス面は、SELinux はアクセスを拒否するものの拒否メッセージがログ記録されないため、トラブルシューティングがより難しくなるという点です。

一時的に **dontaudit** ルールを無効にしてすべての拒否をログ記録できるようにするには、以下のコマンドを root で実行します。

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

-D オプションは **dontaudit** ルールを無効にし、**-B** オプションはポリシーを再構築します。**semodule -DB** を実行した後、パーミッション問題があったアプリケーションを試します。そのアプリケーションに関連した SELinux 拒否がログ記録されているかどうかをチェックします。どの拒否を許可するかという決定は、注意して行なってください。なかには、無視して **dontaudit** ルールで扱われるべきものもあります。わからない場合やアドバイスが必要な場合は、fedora-selinux-list のような SELinux リストに掲載されている他の SELinux ユーザーや開発者に連絡してください。

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

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

これでポリシーが元の状態に復元されます。 **dontaudit** ルールの完全なリストを表示させるには、**sesearch --dontaudit** コマンドを実行します。検索結果を絞り込むには、**-s *domain*** オプションと **grep** コマンドを使います。以下に例を挙げます。

```
~]$ 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 ;
```

拒否の分析に関する情報は、「Raw Audit Messages」 と 「sealert メッセージ」 を参照してください。

10.3.3. サービスの man ページ

サービスの man ページには、特定の状況で使うべきファイルタイプやサービスにあるアクセスを変更するブール値 (NFS ボリュームにアクセスする **httpd** など) といった価値のある情報が含まれています。この情報は、通常の man ページや、**sepolicy manpage** ユーティリティーを使って各サービスドメインに SELinux ポリシーから自動で生成可能な man ページにあります。このような man ページは、***service-name*_selinux** という形式の名前が付けられます。またこれらの man ページは*selinux-policy-devel* パッケージと出荷されます。この *selinux-policy-doc* パッケージには、HTML バージョンの ***service-name*_selinux** man ページが含まれています。

例えば、httpd_selinux(8) man ページには、特定の状況で使うべきファイルタイプやスクリプトを許可するブール値、共有ファイル、ユーザーのホームディレクトリー内にあるディレクトリーへのアクセスなどに関する情報があります。サービスに関する SELinux 情報の man ページには、以下のものがあります。

- Samba: samba_selinux(8) man ページは、たとえば、**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) man ページは、SELinux nfsd ポリシーを使うとユーザーが自身の nfsd プロセスを可能な限り安全な方法で設定できることを説明しています。

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、SELinux によるアクセス拒否を防ぎます。

sepolicy manpage についての詳細は、「Man ページの生成: **sepolicy manpage**」 を参照してください。

10.3.4. Permissive ドメイン

SELinux が permissive モードで実行されていると、SELinux はアクセスを拒否しませんが、enforcing モードでは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。以前は、単一ドメインを permissive にすることはできませんでした (プロセスはドメイン内で実行されます)。特定の状況ではこの結果、システム全体を permissive にして問題の解決を図っていました。

Permissive ドメインは、管理者がシステム全体を permissive にするのではなく、単一プロセス (ドメイン) を permissive で実行する設定を可能にするものです。permissive ドメインでは SELinux チェックは引き続き行われますが、カーネルがアクセスを許可し、SELinux がアクセスを拒否したであろう状況の AVC 拒否をレポートします。

Permissive ドメインには以下の利点があります。

- システム全体を permissive にして危険にさらすことなく、単一のプロセス (ドメイン) を permissive にして問題解決ができます。
- 管理者が新たなアプリケーション用のポリシーを作成できます。以前は最低限のポリシーを作成し、マシン全体を permissive モードにすることでアプリケーションが実行できるようにすることが推奨されていましたが、SELinux 拒否はログ記録されていました。そして **audit2allow** を使ってポリシーを記述することができました。これは、システム全体を危険にさらしていました。permissive ドメインでは、新規ポリシー内のドメインのみが permissive でマークされるので、システム全体を危険にさらすことはありません。

10.3.4.1. 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。ここでの *domain* は、permissive にするドメインのことです。例えば、root で以下のコマンドを実行し、**httpd_t** ドメイン (Apache HTTP Server が稼働するドメイン) を permissive にします。

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

permissive にしたドメインを一覧表示するには、root で **semodule -l | grep permissive** コマンドを実行します。以下のようになります。

```
~]# semodule -l | grep permissive
permissive_httpd_t 1.0
permissivedomains 1.0.0
```

ドメインが permissive である必要がなければ、**semanage permissive -d *domain*** コマンドを root で実行します。以下のようになります。

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

10.3.4.2. Permissive ドメインを無効にする

permissivedomains.pp モジュールには、システム上で提示されるすべての permissive ドメイン宣言が含まれています。これらの permissive ドメインすべてを無効にするには、root で以下のコマンドを実行します。

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

10.3.4.3. Permissive ドメイン での拒否

SYSCALL メッセージは、permissive ドメインでは違ったものになります。以下は、Apache HTTP Server からの AVC 拒否 (および関連するシステムコール) の例です。

```
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)
```

デフォルトでは **httpd_t** ドメインは permissive ではないので、アクションは拒否され**SYSCALL** メッセージに **success=no** が含まれます。以下の例は、同じ状況での AVC 拒否ですが、**semanage permissive -a httpd_t** コマンドを実行して **httpd_t** ドメインを permissive にしてある点が異なります。

```
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)
```

このケースでは、AVC 拒否はログ記録されましたが、**SYSCALL** メッセージの **success=yes** にあるように、アクセスは拒否されませんでした。

permissive ドメインに関する詳細情報は、Dan Walsh のブログ記事 "Permissive Domains" を参照してください。

10.3.5. 拒否の検索および表示

このセクションでは、*setroubleshoot*、*setroubleshoot-server*、*dbus*、*audit* のパッケージがインストールされ、**auditd**、**rsyslogd**、**setroubleshootd** のデーモンが実行中であることを前提としています。これらのデーモンのスタート方法に関しては、「使用するログファイル」を参照してください。SELinux AVC メッセージの検索および表示には、**ausearch**、**aureport**、**sealert** などの数多くのユーティリティーが利用できます。

ausearch

audit パッケージが **ausearch** ユーティリティーを提供します。これは、異なる検索条件に基づいて **audit** デーモンログイベントにクエリを行うことができます。[12] **ausearch** ユーティリティーは **/var/log/audit/audit.log** にアクセスするので、root ユーザーで実行する必要があります。

検索対象	コマンド
すべての拒否	**ausearch -m avc**
当日の拒否	**ausearch -m avc -ts today**
過去 10 分間の拒否	**ausearch -m avc -ts recent**

特定のサービスの SELinux AVC メッセージを検索するには、**-c *comm-name*** オプションを使います。ここでの *comm-name* は実行可能ファイルの名前です。例えば、Apache HTTP Server の場合は**httpd**、Samba の場合は **smbd** になります。

```
~]# ausearch -m avc -c httpd
```

```
~]# ausearch -m avc -c smbd
```

ausearch コマンドでは、読みやすくするためには**--interpret** (**-i**) オプションを、スクリプト処理には **--raw** (**-r**) オプションを使用することが推奨されます。**ausearch** オプションの詳細については、ausearch(8) 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、これは監査システムログのサマリーレポートを作成します。 [13] **aureport** ユーティリティーは **/var/log/audit/audit.log** にアクセスするので、root ユーザーで実行する必要があります。SELinux 拒否メッセージの一覧を表示し、その発生頻度を確認するには、**aureport -a** コマンドを実行します。以下の例では出力に 2 つの拒否があります。

```
~]# 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
```

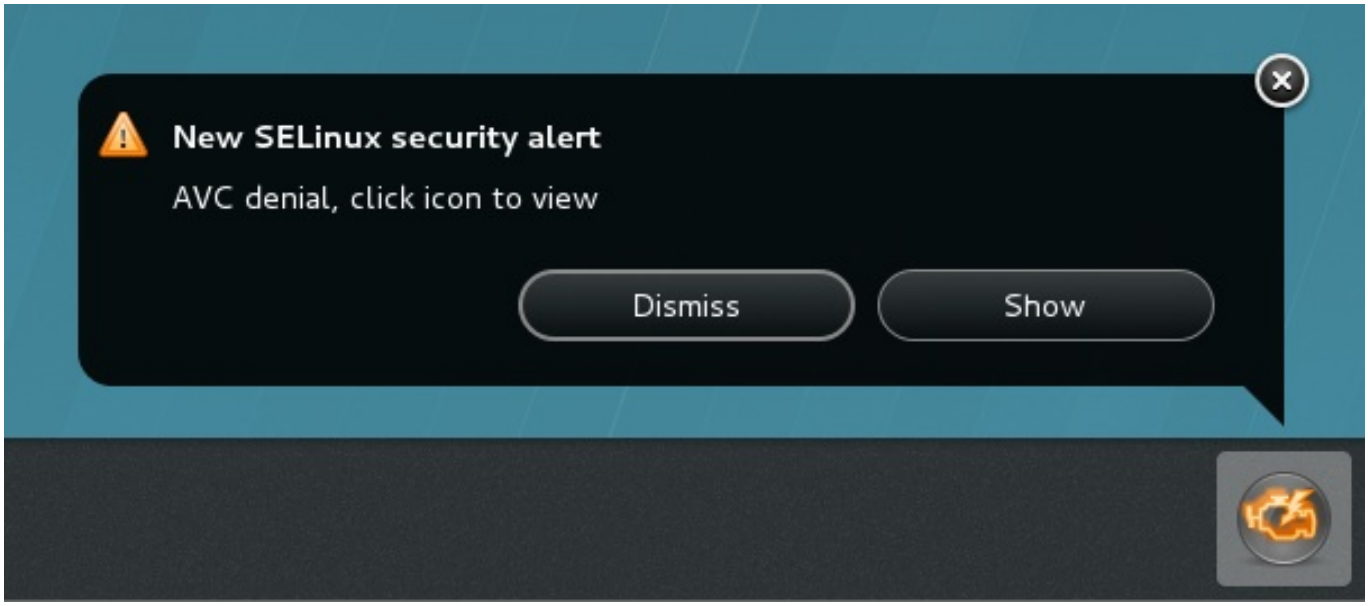
sealert

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* が変換した拒否メッセージを読み取ります。 [14] **/var/log/messages** にあるように、拒否には ID が割り当てられます。以下の例は、**messages** からの拒否です。

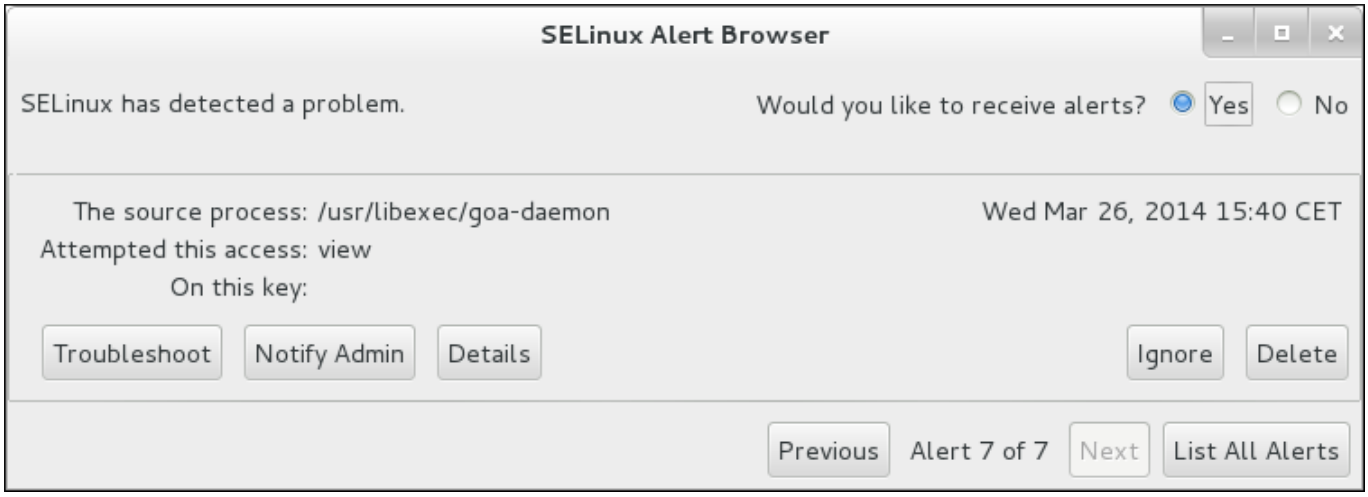
```
setroubleshoot: SELinux is preventing /usr/sbin/httpd from name_bind
access on the tcp_socket. For complete SELinux messages. run sealert -l
8c123656-5dda-4e5d-8791-9e3bd03786b7
```

この例の拒否 ID は、**8c123656-5dda-4e5d-8791-9e3bd03786b7** です。**-l** オプションは、ID を引数 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詳細な分析とアクセスを許可するソリューションが提示されます。

X Window System を実行中で *setroubleshoot* と *setroubleshoot-server* パッケージがインストールされ、**setroubleshootd**、**dbus**、および **auditd** デーモンが稼働している場合、SELinux がアクセスを拒否すると警告が表示されます。



表示 をクリックすると **sealert** GUI が起動し、問題の解決を図ることができます。



別の方法では **sealert -b** コマンドを実行すると、**sealert** GUI を開始することができます。拒否メッセージすべての詳細な分析を表示するには、**sealert -l *** コマンドを実行します。

10.3.6. Raw Audit Messages

Raw Audit Messages は **/var/log/audit/audit.log** に記録されます。以下の例は、Apache HTTP Server (**httpd_t** ドメインで稼働中) が **/var/www/html/file1** ファイル (**samba_share_t** タイプでラベル付け) にアクセスしようとした際に発生したAVC 拒否メッセージ (および関連のシステムコール) 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 }

中括弧内のこのアイテムは、拒否されたパーミッションを示します。**getattr** エントリーは、ソースプロセスがターゲットファイルのステータス情報の読み取りを試みたことを示します。これは、ファイルの読み取り前に発生します。このアクションが拒否されたのは、アクセスされたファイルに間違ったラベルが付けられていたためです。一般的に見られるパーミッションは、**getattr**、**read**、**write** などです。

comm="*httpd*"

プロセスを開始した実行可能ファイルです。このファイルの完全パスは、システムコール (**SYSCALL**) メッセージの **exe=** セクションにあります。このケースでは、**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) へのパスです。

scontext="*unconfined_u:system_r:httpd_t:s0*"

拒否されたアクションを試みたプロセスの SELinux コンテキストです。このケースでは、Apache HTTP Server のSELinux コンテキストで、これは **httpd_t** ドメインで実行中です。

tcontext="*unconfined_u:object_r:samba_share_t:s0*"

プロセスがアクセスを試みたオブジェクト (ターゲット) のSELinux コンテキストです。このケースでは、**file1** のコンテキストです。**httpd_t** ドメインで実行中のプロセスは **samba_share_t** タイプにはアクセスできないことに注意してください。

状況によっては、**tcontext** が **scontext** と一致する場合もあります。例えば、プロセスがユーザー ID など、その実行中のプロセスの特徴を変更することになるシステムサービスの実行を試みる場合などです。また、プロセスが通常の制限で許されている以上のリソース (メモリーなど) を使おうとして、そのプロセスが制限超過を許されているかどうかのセキュリティチェックにつながる場合、**tcontext** が **scontext** と一致する可能性があります。

システムコール (**SYSCALL**) メッセージでは、2 つの点に注目します。

- **success=*no*** は、拒否 (AVC) が強制されたかどうかを示します。**success=no** は、システムコールが成功しなかったことを示します (SELinux がアクセスを拒否)。**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** などの permissive ドメインや制限のないドメインで見られます。
- **exe="*/usr/sbin/httpd*"** は、プロセスを開始した実行可能ファイルへの完全パスです。このケースでは、**exe="/usr/sbin/httpd"** です。

SELinux がアクセスを拒否することになる原因の多くは、ファイルタイプが間違っていることです。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**) とターゲットコンテキスト (**tcontext**) を比べます。プロセス (**scontext**) がそのようなオブジェクト (**tcontext**) にアクセスしてもよいかどうかを確認します。例えば、Apache HTTP Server (**httpd_t**) は特定の設定がない限り、**httpd_sys_content_t** や **public_content_t** など、httpd_selinux(8) man ページで指定されたタイプ以外にはアクセスすべきではありません。

10.3.7. sealert メッセージ

拒否には ID が割り当てられ、**/var/log/messages** で見ることができます。以下の例は、Apache HTTP Server (**httpd_t** ドメインで稼働中) が **/var/www/html/file1** ファイル (**samba_share_t** タイプでラベル付け) にアクセスしようとした際に発生したAVC 拒否 (**messages** にログ記録) 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 84e0b04d-d0ad-4347-8317-22e74f6cd020
```

以下のように、**sealert -l 84e0b04d-d0ad-4347-8317-22e74f6cd020** コマンドを実行して完全なメッセージを表示します。このコマンドはローカルマシン上でのみ機能し、**sealert** GUI と同じ情報を提示します。

```
~]$ sealert -l 84e0b04d-d0ad-4347-8317-22e74f6cd020

Summary:

SELinux is preventing httpd (httpd_t) "getattr" to /var/www/html/file1
(samba_share_t).

Detailed Description:

SELinux denied access to /var/www/html/file1 requested by httpd.
/var/www/html/file1 has a context used for sharing by different program.
If you
would like to share /var/www/html/file1 from httpd also, you need to
change its
file context to public_content_t. If you did not intend to this access,
this
could signal a intrusion attempt.

Allowing Access:

You can alter the file context by executing chcon -t public_content_t
'/var/www/html/file1'

Fix Command:

chcon -t public_content_t '/var/www/html/file1'

Additional Information:

Source Context                unconfined_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                   /usr/sbin/httpd
Port                          <Unknown>
Host                          hostname
Source RPM Packages           httpd-2.2.10-2
Target RPM Packages
Policy RPM                    selinux-policy-3.5.13-11.fc12
Selinux Enabled               True
Policy Type                   targeted
MLS Enabled                   True
Enforcing Mode                Enforcing
Plugin Name                   public_content
Host Name                     hostname
Platform                      Linux hostname 2.6.27.4-68.fc12.i686 #1
SMP Thu Oct
```

```
30 00:49:42 EDT 2008 i686 i686
Alert Count                    4
First Seen                     Wed Nov  5 18:53:05 2008
Last Seen                      Wed Nov  5 01:22:58 2008
Local ID                       84e0b04d-d0ad-4347-8317-22e74f6cd020
Line Numbers

Raw Audit Messages

node=hostname type=AVC msg=audit(1225812178.788:101): avc:  denied  {
getattr } for  pid=2441 comm="httpd" path="/var/www/html/file1" dev=dm-0
ino=284916 scontext=unconfined_u:system_r:httpd_t:s0
tcontext=unconfined_u:object_r:samba_share_t:s0 tclass=file

node=hostname type=SYSCALL msg=audit(1225812178.788:101): arch=40000003
syscall=196 success=no exit=-13 a0=b8e97188 a1=bf87aaac a2=54dff4
a3=2008171 items=0 ppid=2439 pid=2441 auid=502 uid=48 gid=48 euid=48
suid=48 fsuid=48 egid=48 sgid=48 fsgid=48 tty=(none) ses=3 comm="httpd"
exe="/usr/sbin/httpd" subj=unconfined_u:system_r:httpd_t:s0 key=(null)
```

Summary

拒否されたアクションの簡潔なサマリーです。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けられたファイル (**file1**) へのアクセスを拒否されました。

Detailed Description

より詳細な説明です。この例では、**file1** に **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 でエクスポートするファイルおよびディレクトリーに使われます。説明では、Apache HTTP Server および Samba によるアクセスが望まれる場合、タイプを Apache HTTP Server および Samba がアクセス可能なものに変更することを提案しています。

Allowing Access

アクセスを可能にする方法を提案しています。ファイルの再ラベル付けやブール値を有効にする、ローカルポリシーモジュールの作成、などの方法があります。このケースでは、Apache HTTP Server および Samba の両方がアクセス可能なタイプでファイルにラベル付けすることを提案しています。

Fix Command

アクセスを可能にし、拒否を解決するコマンドを提案しています。この例では、**file1** タイプを Apache HTTP Server と Samba の両方がアクセス可能な **public_content_t** に変更するコマンドを提示しています。

Additional Information

ポリシーパッケージ名やバージョン (**selinux-policy-3.5.13-11.fc12**) などのバグレポートに便利な情報です。ただ、拒否が発生した原因の解決には役立たない可能性があります。

Raw Audit Messages

/var/log/audit/audit.log からの拒否に関連した raw 監査メッセージです。AVC 拒否の各アイテムに関しては、「Raw Audit Messages」 を参照してください。

10.3.8. アクセス許可: 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説明する目的でのみ、使われています。

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 policy allow ルールを生成します。 [15] 「sealert メッセージ」 にあるように拒否メッセージを分析し、ラベル変更がないもしくはブール値で許可されたアクセスがない場合は、**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 にアクセスを拒否された場合は、**audit2allow** を実行すると以前は拒否されたアクセスを許可する Type Enforcement ルールが生成されます。

以下の例では、**audit2allow** を使ってポリシーモジュールを作成します。

1. 拒否メッセージおよび関連するシステムコールは、**/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が拒否されました。「sealert メッセージ」 にあるように拒否メッセージを分析します。ラベル変更がないもしくはブール値で許可されたアクセスがない場合は、**audit2allow** を使ってローカルポリシーモジュールを作成します。

2. 以下のコマンドを実行して、アクセスが拒否された理由についてヒューマンリーダブルな記述を作成します。**audit2allow** ユーティリティーは **/var/log/audit/audit.log** を読み取るので、root ユーザーで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により、すべてを監査ログが読み取られます。**-w** オプションでは、ヒューマンリーダブルな記述が作成されます。上記では Type Enforcement ルールがないのでアクセスが拒否されました。

3. 以下のコマンドを実行して、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

Type Enforcement ルールの欠如は通常、SELinux ポリシーのバグによって引き起こされ、Red Hat Bugzilla で報告されるべきです。Red Hat Enterprise Linux の場合、**Red Hat Enterprise Linux** 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。バグ報告では、**audit2allow -w -a** および **audit2allow -a** コマンドの出力も報告してください。

4. **audit2allow -a** が表示したルールを使うには、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、現在作業中のディレクトリーに **-M** で指定された名前のついた 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、以下のコマンドを root で実行します。

```
~]# semodule -i mycertwatch.pp
```

重要

audit2allow で作成したモジュールは、必要以上にアクセスを許可する場合があります。**audit2allow** で作成されたモジュールは、アップストリームの SELinux リストに公表してレビューされ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 でバグを作成してください。

複数のプロセスから複数の拒否メッセージがあって、そのうちの 1 つのプロセスに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って **audit2allow** の入力を絞り込みます。以下の例では、**grep** を使って **certwatch** に関連した拒否メッセージのみを **audit2allow** に送信する方法を示しています。

```
~]# grep certwatch /var/log/audit/audit.log | audit2allow -R -M
mycertwatch2
****************** IMPORTANT ***********************
```

```
To make this policy package active, execute:

semodule -i mycertwatch2.pp
```

[10] **/etc/selinux/targeted/contexts/files/** 内のファイルは、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内のファイルは **restorecon** および **setfiles** ユーティリティーが読み取り、ファイルとディレクトリーをデフォルトのコンテキストに復元します。

[11] **semanage port -a** コマンドは、エントリーを **/etc/selinux/targeted/modules/active/ports.local** ファイルに追加します。デフォルトでは、このファイルは root のみが読み取れることに留意してください。

[12] **ausearch** についての詳細情報は、ausearch(8) の man ページを参照してください。

[13] **aureport** についての詳細は、aureport(8) の man ページを参照してください。

[14] **sealert** についての詳細は、sealert(8) の man ページを参照してください。

[15] **audit2allow** についての詳細は、audit2allow(1) man ページを参照してください。

第11章 追加情報

11.1. 貢献者

- Dominick Grift – テクニカルエディター
- Murray McAllister – Red Hat プロダクトセキュリティ
- James Morris – テクニカルエディター
- Eric Paris – テクニカルエディター
- Scott Radvan – Red Hat カスタマーコンテンツサービス
- Daniel Walsh – Red Hat セキュリティエンジニアリング

11.2. その他のリソース

米国国家安全保障局 (NSA)

NSA は SELinux の開発元です。NSA のNational Information Assurance Research Laboratory (NIARL) の研究者らは、Linux カーネルの主要サブシステムにおける柔軟性のある強制アクセス制御を設計、実装しました。また、Flask アーキテクチャーが提供する新たなオペレーティングシステムのコンポーネントを実装しました。セキュリティサーバーとアクセスベクターキャッシュのことです。NSA 研究者は Linux 2.6 で LSM ベースの SELinux 含めるように改訂しました。NSA は、X Window System (XACE/XSELinux) と Xen (XSM/Flask) でも同様の制御の開発を進めました。 [16]

- SELinux メイン Web サイト: http://www.nsa.gov/research/selinux/index.shtml
- SELinux ドキュメンテーション: http://www.nsa.gov/research/selinux/docs.shtml
- SELinux バックグラウンド: http://www.nsa.gov/research/selinux/background.shtml

Tresys Technology

Tresys Technology は以下のアップストリームです。

- SELinux userland libraries and tools.
- SELinux Reference Policy.

SELinux ニュース

- ニュース: http://selinuxnews.org/wp/.
- Planet SELinux (ブログ): http://selinuxnews.org/planet/.

SELinux プロジェクト Wiki

- メインページ: http://selinuxproject.org/page/Main_Page.
- ドキュメンテーション、メールリスト、Web サイト、ツールへのリンクを含むユーザーリソース: http://selinuxproject.org/page/User_Resources.

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 の SELinux FAQ: http://fedoraproject.org/wiki/SELinux_FAQ/.

非公式の SELinux FAQ

http://www.crypt.gen.nz/selinux/faq.html

The SELinux Notebook - The Foundations - 第 3 版

http://www.freetechbooks.com/the-selinux-notebook-the-foundations-t785.html

IRC

Freenode について:

- #selinux
- #fedora-selinux
- #security

---

[16] 詳細は、NSA Contributors to SELinux のページを参照してください。

パート II. 制限のあるサービスの管理

第12章 はじめに

本ガイドのパート II ではより実用的なタスクにフォーカスしており、様々なサービスの設定方法についての情報を提供しています。各サービスは、最も一般的なタイプとブール値とともに仕様を表示しています。また、こうしたサービスを設定する場合の実例を挙げながら、SELinux でどのようにサービスの動作を補完しているのかについて見ていきます。

SELinux が enforcing モードの場合は、Red Hat Enterprise Linux で使用されるデフォルトのポリシーはターゲットポリシーになります。ターゲットとなるプロセスが制御のあるドメインで実行され、ターゲット以外のプロセスは制限のないドメインで実行されます。ターゲットポリシーと制限のあるプロセスおよび制御のないプロセスについての詳細は、3章*ターゲットポリシー*を参照してください。

第13章 Apache HTTP Server

Apache HTTP Server は、現行の HTTP 標準を備えたオープンソースの HTTP サーバーを提供します。[17]

Red Hat Enterprise Linux では、 *httpd* パッケージが Apache HTTP Server を提供します。*httpd* 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、以下のコマンドを実行します。

```
~]$ rpm -q httpd
package httpd is not installed
```

パッケージがインストールされておらず Apache HTTP Server を使用したい場合は、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
```

13.1. Apache HTTP Server と SELinux

SELinux を有効にすると、Apache HTTP Server (**httpd**) はデフォルトで制限のあるサービスとして実行されます。制限のあるプロセスはそのプロセス自体のドメインで実行され、 他の制限のあるプロセスとは分離されます。 制限のあるプロセスが攻撃を受けると、 SELinux ポリシー設定に応じて、 攻撃側がリソースにアクセスして加えることができる被害は限定されます。以下の例では、 **httpd** プロセス自体のドメイン内で実行しているプロセスを示します。ここでは、*httpd*、*setroubleshoot*、*setroubleshoot-server*、*policycoreutils-python* の各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 モードで実行していることを確認します。

   ```
   ~]$ getenforce
   Enforcing
   ```

   SELinux が enforcing モードで実行していれば、**Enforcing** が返されます。

2. root で以下のコマンドを実行し、**httpd** を起動します。

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

   サービスが稼働していることを確認します。出力は以下のようになり、タイムスタンプのみが異なります。

   ```
   ~]# 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 プロセスに関連する SELinux コンテキストは**system_u:system_r:httpd_t:s0** です。コンテキストの末尾から 2 番目の部分である **httpd_t** がタイプになります。タイプはプロセスのドメインやファイルのタイプを定義します。この例の場合、**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、**httpd_t** などの制限のあるドメイン内で実行しているプロセスがファイルや他のプロセス、システムなどとどのように交信するのかを定義します。**httpd** がファイルにアクセスができるよう、ファイルには適切なラベルを付ける必要があります。たとえば、**httpd_sys_content_t** タイプのラベルが付いたファイルの場合、**httpd** はこのファイルの読み取りはできますが書き込みはできません。この場合、Linux (DAC) のパーミッションで書き込みのアクセスが許可されていても書き込みはできません。特定の動作を許可する場合、たとえば、スクリプトによるネットワークへのアクセスを許可する、**httpd** による NFS や CIFS ファイルシステムへのアクセスを許可する、**httpd** による CGI (Common Gateway Interface) スクリプトの実行を許可するなどの場合には、ブール値を有効にする必要があります。

httpd が TCP ポート 80、443、488、8008、8009、8443 以外のポートでリッスンするように **/etc/httpd/conf/httpd.conf** ファイルを設定する場合は、**semanage port** コマンドを使って SELinux ポリシー設定に新しいポート番号を追加する必要があります。以下では、まだ SELinux ポリシー設定で **httpd** 用には定義されていないポートでリッスンするよう**httpd** を設定した結果、**httpd** の起動に失敗する例を示します。また、**httpd** がポリシーにまだ定義されていない非標準のポートで正しくリッスンするよう SELinux システムを設定する方法についても示します。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。各コマンドは root ユーザーで実行してください。

1. 以下のコマンドを実行し、**httpd** が稼働し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)
```

出力が上記と異なる場合は、この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って、SELinux で **httpd** にリッスンを許可してい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. root で **/etc/httpd/conf/httpd.conf** を編集します。**Listen** オプションを設定し、SELinux ポリシー設定で **httpd** 用に設定されていないポートを記入します。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 127.0.0.1:12345
```

4. 以下のコマンドを実行し、**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.
```

次のような SELinux 拒否メッセージがログ記録されます。

```
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. この例で **httpd** がポート 12345 をリッスンできるように SELinux で許可するには、以下のコマンドが必要になります。

```
~]# semanage port -a -t http_port_t -p tcp 12345
```

6. 再度 **httpd** を起動して、新しいポートをリッスンするようにします。

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

7. これで **httpd** が非標準ポート (この例では TCP 12345) をリッスンできるようにする SELinux 設定が完了したので、**httpd** がこのポートで正常に起動するようになります。

8. **httpd** が TCP ポート 12345 でリッスンし通信しているかを確認するには、以下のようにそのポートに telnet 接続を開き HTTP GET コマンドを発行します。

```
~]# 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

Type Enforcement が SELinux のターゲットポリシーで使用されるメインのパーミッション制御になります。すべてのファイルおよびプロセスにタイプのラベルが付けられます。ファイルの場合はタイプ、プロセスの場合はドメインを定義します。タイプにアクセスするドメインであっても、別のドメインにアクセスするドメインであっても、SELinux ポリシールールがタイプ同士のアクセス方法を定義します。アクセスを許可する特定の SELinux ポリシールールが存在する場合にのみ、そのアクセスが許可されます。

以下では **/var/www/html/** ディレクトリーに新規ファイルを作成し、このファイルが親ディレクトリー (**/var/www/html/**) から **httpd_sys_content_t** タイプを継承していることを例示します。

1. 以下のコマンドを実行して、**/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
```

/var/www/html/ が **httpd_sys_content_t** タイプでラベル付けされていることが分かります。

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. 以下のコマンドを実行して 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 コマンドを使用すると **file1** には **httpd_sys_content_t** タイプのラベルが付けられていることが分かります。SELinux では、 **httpd** がこのタイプのラベルが付いたファイルを読み込めるよう許可していますが、書き込みは許可していません。Linux のパーミッションが書き込みアクセスを許可していても、書き込みは許可されません。SELinux ポリシーでは、**httpd_t** ドメイン (**httpd** が実行されるドメイン) で実行しているプロセスが読み取りと書き込みができるタイプを定義しています。これにより、プロセスが別のプロセス用のファイルにアクセスすることを防いでいます。

たとえば、**httpd** は **httpd_sys_content_t** タイプ (Apache HTTP Server 用) のラベルが付いたファイルを読み込むことはできますが、デフォルトでは **samba_share_t** タイプ (Samba 用) 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、ユーザーのホームディレクトリーにあるファイルには **user_home_t** タイプのラベルが付けられます。これにより、デフォルトで**httpd** がユーザーのホームディレクトリーにあるファイルの読み取りや書き込みをすることを防いでいます。

以下で **httpd** で使用されるタイプを例示します。タイプを使い分けることで柔軟なアクセス設定ができるようになります。

httpd_sys_content_t

このタイプは、静的な Web サイトで使用される **.html** ファイルなどの Web コンテンツに使用します。このタイプのラベルが付けられたファイルは、**httpd** および **httpd** で実行されるスクリプトによるアクセスが可能となります (読み取り専用)。デフォルトでは、このタイプのラベルが付けられたファイルおよびディレクトリーには、**httpd** や他のプロセスは書き込みや編集ができません。デフォルトでは、**/var/www/html/** ディレクトリー内に作成またはコピーされたファイルには **httpd_sys_content_t** タイプのラベルが付けられることに注意してください。

httpd_sys_script_exec_t

このタイプは、**httpd** で実行するスクリプトに使用します。一般的には**/var/www/cgi-bin/**内の CGI (Common Gateway Interface) スクリプトに使用されます。デフォルトでは、SELinux ポリシーにより、**httpd** は CGI スクリプトの実行が禁止されています。これを許可するには、スクリプトに **httpd_sys_script_exec_t** タイプのラベルを付け、**httpd_enable_cgi** のブール値を有効にします。**httpd_sys_script_exec_t** のラベルが付けられたスクリプトは、**httpd** で実行されると **httpd_sys_script_t** ドメインで実行されます。**httpd_sys_script_t** ドメインには、**postgresql_t** や **mysqld_t** などの他のシステムドメインへのアクセスがあります。

httpd_sys_rw_content_t

このタイプのラベルが付けられたファイルには、**httpd_sys_script_exec_t** タイプのラベルが付いたスクリプトは書き込み可能となりますが、これ以外のラベルタイプのスクリプトによる

編集はできません。**httpd_sys_script_exec_t** タイプのラベルが付いたスクリプトで読み込みや書き込みをするファイルには、**httpd_sys_rw_content_t** タイプのラベルを使用する必要があります。

httpd_sys_ra_content_t

このタイプのラベルが付けられたファイルは、**httpd_sys_script_exec_t** タイプのラベルが付いたスクリプトによる追加が可能になりますが、これ以外のラベルタイプのスクリプトによる編集はできません。**httpd_sys_script_exec_t** タイプのラベルが付いたスクリプトで読み込みや追加をするファイルには、**httpd_sys_ra_content_t** タイプのラベルを使用する必要があります。

httpd_unconfined_script_exec_t

このタイプのラベルが付いたスクリプトは SELinux の保護なしで実行されます。他のオプションをすべて試してもうまくいかない複雑なスクリプトにのみ、このタイプを使用してください。**httpd** の SELinux 保護を無効にする、またはシステム全体の SELinux 保護を無効にするよりは、このタイプの使用が望まれます。

注記

httpd で使用可能な他のタイプを確認するには、以下のコマンドを実行します。

```
~]$ grep httpd /etc/selinux/targeted/contexts/files/file_contexts
```

手順13.1 SELinux のコンテキストを変更する

ファイルやディレクトリーのタイプは **chcon** コマンドを使用して変更できます。**chcon** による変更は、ファイルシステムの再ラベルや **restorecon** コマンドを実行すると失われます。特定ファイルの SELinux コンテキストをユーザーが変更できるかどうかは、SELinux ポリシーで制御します。以下の例では、**httpd** 用に **index.html** ファイルと新規ディレクトリーを作成し、**httpd** がこれらにアクセスできるようにするラベルを付けます。

1. root で **mkdir** ユーティリティーを使用し、**httpd** が使用するファイルを保存する最上位のディレクトリーを作成します。

```
~]# mkdir -p /my/website
```

2. ファイルコンテキスト設定のパターンに合致しないファイルやディレクトリーには、**default_t** タイプのラベルが付いている場合があります。制限のあるサービスは、このタイプのファイルやディレクトリーに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実行し、**/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** がアクセス可能なタイプに変更します。これで**/my/website/** の下に作成されるファイルは、**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り、httpd がアクセスできるようになります。

```
~]# 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 についての詳細は、「一時的な変更: chcon」 を参照してください。

再ラベル付けや **restorecon** コマンドの実行後もこのラベル変更を維持するには、**semanage fcontext** コマンド (**semanage** は *policycoreutils-python* パッケージで提供) を使用します。このコマンドにより、変更がファイルコンテキスト設定に追加されます。この後に **restorecon** を実行すると、ファイルコンテキスト設定が読み込まれ、ラベル変更が適用されます。以下の例では、**httpd** が使用する新規ディレクトリーと **index.html** ファイルを作成し、**httpd** がアクセスできるようにラベルを永続的に変更します。

1. root で **mkdir** ユーティリティーを使用し、**httpd** が使用するファイルを保存する最上位のディレクトリーを作成します。

```
~]# mkdir -p /my/website
```

2. root で以下のコマンドを実行して、ラベル変更をファイルコンテキスト設定に追加します。

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

"/my(/.*)?" は、ラベル変更が **/my/** ディレクトリーとその下のファイルおよびディレクトリーすべてに適用されることを意味します。

3. root で **touch** 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実行し、ラベルの変更を適用します (ステップ 2 の **semanage** コマンドで変更されたファイルコンテキスト設定が **restorecon** により読み込まれます)。

```
~]# 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 に関する詳細情報は、「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、実行するサービスに最低限必要なレベルのアクセスに基づいています。サービスの実行手段に複数あるため、サービスの実行方法を指定する必要があります。これには、ブール値を使用します。ブール値を使用すると、SELinux ポリシーの記述方法の知識がなくてもランタイム時に SELinux ポリシーの一部変更ができます。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なうことなく、サービスの NFS ボリュームへのアクセス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** ブール値をオンにするには、以下のコマンドを root ユーザーで実行します。

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

同じ例でブール値を無効にするには、下記の様にコマンドの **on** を **off** にします。

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

注記

再起動後に **setsebool** による変更を維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。

以下では、**httpd** の動作を指定する一般的なブール値について説明します。

httpd_anon_write

このブール値を無効にすると、**httpd** は **public_content_rw_t** タイプのラベルが付いたファイルへのアクセスが読み取り専用に限定されます。有効にすると、パブリックファイル転送サービス用のファイルを含むパブリックディレクトリーなど、**public_content_rw_t** タイプのラベルが付いたファイルへの書き込みが可能になります。

httpd_mod_auth_ntlm_winbind

このブール値を有効にすると、 **httpd** で **mod_auth_ntlm_winbind** モジュールを介した NTLM および Winbind 認証メカニズムへのアクセスが許可されます。

httpd_mod_auth_pam

このブール値を有効にすると、**httpd** で **mod_auth_pam** モジュールを介した PAM 認証メカニズムへのアクセスが許可されます。

httpd_sys_script_anon_write

このブール値は、パブリックファイル転送サービスで使用されるような、**public_content_rw_t** タイプのラベルが付いたファイルへの書き込みアクセスを HTTP スクリプトに許可するかどうかを定義します。

httpd_builtin_scripting

httpd スクリプト機能へのアクセスを定義するブール値です。PHP コンテンツの場合、このブール値を有効にすることが必要とされることが多くあります。

httpd_can_network_connect

このブール値を無効にすると、HTTP スクリプトやモジュールがネットワークやリモートポートに接続開始することができなくなります。接続の開始を許可する場合は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 スクリプトやモジュールによるデータベースサーバーへの接続開始が阻止されます。接続の開始を許可する場合はブール値を有効にします。

httpd_can_network_relay

httpd をフォワードプロキシまたはリバースプロキシとして使用する場合、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送信をできなくなります。これにより、**httpd** に脆弱性が見つかった場合にスパム攻撃を阻止することができます。HTTP モジュールにメール送信を許可する場合は、このブール値を有効にします。

httpd_dbus_avahi

このブール値を無効にすると、**httpd** による **D-Bus** 経由の **avahi** サービスへのアクセスが拒否されます。このアクセスを許可する場合は、このブール値を有効にします。

httpd_enable_cgi

このブール値を無効にすると、**httpd** が CGI スクリプトの実行をできなくなります。**httpd** に CGI スクリプトの実行を許可する場合は、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** タイプのラベルを付けておく必要があります)。

httpd_enable_ftp_server

このブール値を有効にすると、**httpd** が FTP ポートでリッスンできるようになり、FTPサーバーとしての動作が可能になります。

httpd_enable_homedirs

このブール値を無効にすると、**httpd** がユーザーのホームディレクトリーにアクセスできなくなります。ユーザーのホームディレクトリー (**/home/*/** 内のコンテンツなど) へのアクセスを許可する場合は、このブール値を有効にします。

httpd_execmem

このブール値を有効にすると、 **httpd** が実行可能かつ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るようになります。バッファのオーバーフローに対する保護が低下するため、安全面からはこのブール値の有効化は推奨されません。ただし、特定のモジュールやアプリケーションではこの権限を必要とするものもあります (Java や Mono アプリケーションなど)。

httpd_ssi_exec

このブール値は、Web ページ内の SSI (server side include) 要素を実行可能にするかどうかを定義します。

httpd_tty_comm

このブール値は、**httpd** が制御ターミナルへアクセスできるかどうかを定義します。通常、このアクセスは必要とされませんが、SSL 証明書ファイルを設定する場合などに、パスワードのプロンプトを表示させ処理するため、ターミナルへの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** による **httpd** の全タイプへの完全アクセスが許可されます (つまり、sys_content_t の実行、読み込み、書き込み)。これを無効にすると、読み取り専用 web コンテンツ、書き込み可能 web コンテンツ、実行可能 web コンテンツが分離されます。このブール値を無効にすると安全性は高くなりますが、各ファイルに付与するアクセス権に応じてスクリプトや他の web コンテンツを個別にラベル付けするという管理オーバーヘッドが生じます。

httpd_use_cifs

このブール値を有効にすると、Samba でマウントされるファイルシステムなど、**cifs_t** タイプのラベルが付いている CIFS ボリューム上にあるファイルに **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 を使ってマウントされるファイルシステムなど、**nfs_t** タイプのラベルが付いている NFS ボリューム上にあるファイルに **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、上記のリストでは常にこのサービスに関連するブール値がすべて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、以下のコマンドを実行します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の記述を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

sepolicy ユーティリティーを提供する *policycoreutils-devel* パッケージが追加で必要になることに留意してください。

13.4. 設定例

以下では、SELinux がどのように Apache HTTP Server を補完するのか、Apache HTTP Server の全機能をどのように維持するのかを実践的な例を用いて示します。

13.4.1. 静的なサイトを稼働させる

静的な web サイトを作成する場合は、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 Server は **httpd_sys_content_t** タイプのラベルが付いたファイルに書き込みはできません。以下の 例では、読み取り専用 web サイト向けのファイルを保存する新規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 で **/mywebsite/index.html** ファイルを作成します。以下のコンテンツを **/mywebsite/index.html** にコピーして貼り付けます。

```
<html>
<h2>index.html from /mywebsite/</h2>
</html>
```

3. **/mywebsite/** およびその配下のファイルやサブディレクトリーへの読み取り専用アクセスを Apache HTTP Server に許可するために、このディレクトリー 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実行してラベルの変更をファイルコンテキスト設定に追加します。

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

4. root で **restorecon** を使用してラベル変更を適用します。

```
~]# 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. この例の場合、root で **/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、**DocumentRoot "/mywebsite"** オプションを追加します。編集後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実行して Apache HTTP Server の状態を確認します。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
```

サーバーが稼働している場合は、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** への変更にもこれを適用)。

```
~]# 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 ブラウザで **http://localhost/index.html** に移動します。以下のように表示されます。

```
index.html from /mywebsite/
```

13.4.2. NFS および CIFS ボリュームの共有

クライアント側の NFS マウントはデフォルトで、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。共通ポリシーでは、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、クライアント側にマウントされた Samba 共有は、ポリシーが定義したデフォルトのコンテキストでラベル付けされます。共通ポリシーでは、このデフォルトのコンテキストに **cifs_t** タイプを使用します。

ポリシー設定によっては、サービスが **nfs_t** または **cifs_t** タイプのラベルが付けられたファイルを読み取れない場合もあります。これにより、これら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、他のサービスが読み取ったり、エクスポートすることを防ぐことができます。ブール値をオンやオフに切り替えて、**nfs_t** や **cifs_t** タイプにアクセス可能なサービスを制御することができます。

(**nfs_t** タイプのラベルが付けられている) NFS ボリュームへのアクセスと共有を**httpd** に許可する場合は、**httpd_use_nfs** ブール値を有効にします。

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

(**cifs_t** タイプのラベルが付けられている) CIFS ボリュームへのアクセスと共有を**httpd** に許可する場合は、**httpd_use_cifs** ブール値を有効にします。

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

注記

再起動後に **setsebool** による変更を維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共有

Type Enforcement を使用すると、プロセスが別のプロセス用のファイルにアクセスしてしまうのを防ぐのに役立ちます。たとえば、デフォルトでは Samba は **httpd_sys_content_t** タイプのラベルが付いたファイルを読み込みことはできません。このタイプは Apache HTTP Server での使用を目的としています。目的のファイルに **public_content_t** または **public_content_rw_t** タイプのラベルを付けると、Apache HTTP Server、FTP、rsync、Samba 間でファイルを共有することができるようになります。

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、 Samba でその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** を使用して、複数サービス間でファイルを共有するための最上位の新規ディレクトリーを作成します。

   ```
   ~]# mkdir /shares
   ```

2. ファイルコンテキスト設定のパターンに合致しないファイルやディレクトリーには、**default_t** タイプのラベルが付いている場合があります。制限のあるサービスは、このタイプのファイルやディレクトリーに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。以下のコンテンツをコピーして **/shares/index.html** に貼り付けます。

```
<html>
<body>
<p>Hello</p>
</body>
</html>
```

4. **/shares/** に **public_content_t** タイプのラベルを付けることで、Apache HTTP Server、FTP、rsync、Samba による読み取り専用アクセスを許可します。root で以下のコマンドを実行し、ラベルの変更をファイルコンテキスト設定に追加します。

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

5. root で **restorecon** ユーティリティーを使用してラベル変更を適用します。

```
~]# 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 で **/shares/** を共有する場合は、以下の手順にしたがいます。

1. *samba*、*samba-common*、*samba-client* の各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 (バージョン番号は使用しているバージョンによって異なります)。

```
~]$ 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
```

上記のパッケージ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、これらをインストールします。

```
~]# yum install package-name
```

2. root で **/etc/samba/smb.conf** ファイルを編集します。Samba で **/shares/** ディレクトリーを共有するために、以下のエントリーをこのファイルの末尾に追加します。

```
[shares]
comment = Documents for Apache HTTP Server, FTP, rsync, and Samba
path = /shares
public = yes
writeable = no
```

3. Samba ファイルシステムのマウントには Samba アカウントが必要になります。root で以下のコマンドを実行し、Samba アカウントを作成します。*username* は既存の Linux ユーザーにします。たとえば、**smbpasswd -a testuser** を実行すると、Linux の **testuser** ユーザー用の Samba アカウントが作成されます。

```
~]# 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する際に、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 サービスを開始します。

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

5. 以下のコマンドを実行し、利用可能な共有を表示します。*username* はステップ 3 で追加した Samba アカウントにします。パスワードの入力を求められたら、ステップ 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 (バージョン番号は使用しているバージョンによって異なります)。

```
~]$ 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って新規ディレクトリーを作成します。このディレクトリーは Samba 共有の **shares** をマウントする際に使用します。

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

7. root で以下のコマンドを実行して、Samba 共有の **shares** を **/test/** にマウントします。*username* はステップ 3 のユーザー名にしてください。

```
~]# mount //localhost/shares /test/ -o user=username
```

ステップ 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コンテンツを表示します。

```
~]$ cat /test/index.html
<html>
<body>
<p>Hello</p>
</body>
</html>
```

Apache HTTP Server で **/shares/** を共有する場合は、以下の手順にしたがいます。

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 (バージョン番号は使用しているバージョンによって異なります)。

```
~]$ 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/** ディレクトリーに移動します。root で以下のコマンドを実行して**/shares/** ディレクトリーへのリンク (**shares** という名前にします) を作成します。

```
html]# ln -s /shares/ shares
```

3. Apache HTTP Server を起動します。

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

4. web ブラウザを使って **http://localhost/shares** に移動します。 **/shares/index.html** が表示されます。

デフォルトでは、**index.html** ファイルが存在していれば、Apache HTTP Server はこれを読み込みます。**/shares/** に **file1**、**file2**、**file3** しかなく **index.html** がない場合、**http://localhost/shares** にアクセスするとディレクトリー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. root で **touch** ユーティリティーを使用して **/shares/** に新規ファイルを 3 つ作成します。

```
~]# 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
```

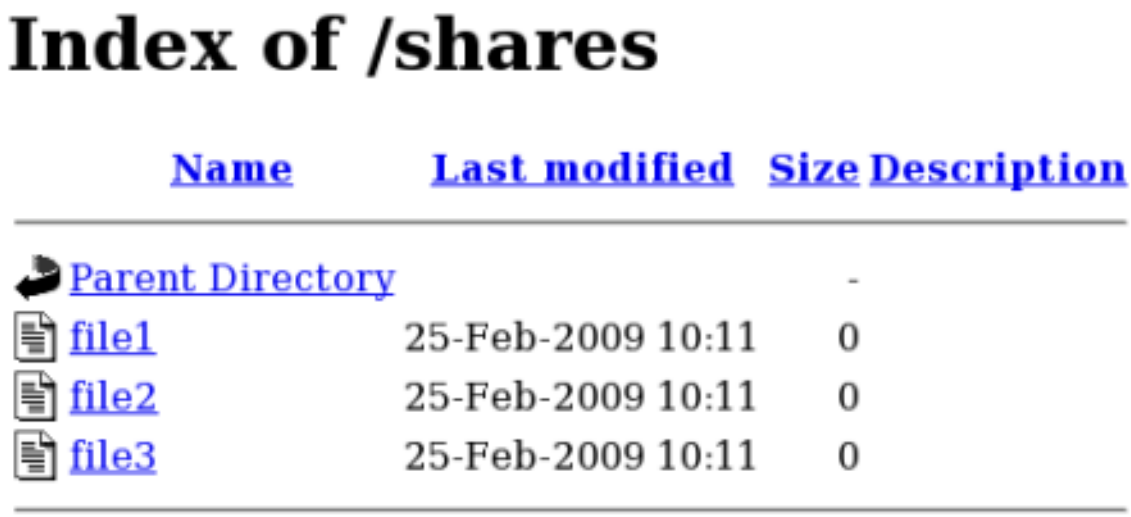
3. root で以下のコマンドを実行して Apache HTTP Server の状態を確認します。

```
~]# 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、これを起動します。

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

4. web ブラウザで **http://localhost/shares** に移動します。ディレクトリー一覧が表示されます。



13.4.4. ポート番号を変更する

ポリシー設定によっては、サービスが特定のポート番号でのみ実行できるようにすることが可能です。ポリシーを変更せずサービスが実行されるポートを変えようとすると、サービスの起動に失敗する場合があります。root ユーザーで **semanage** ユーティリティーを使用して、SELinux が **httpd** にリッスンを許可してい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
```

デフォルトでは、 SELinux で **httpd** にリッスンを許可している TCP ポートは 80、443、488、8008、8009、8443 になります。**httpd** で **http_port_t** 用に記載されていないポートをリッスンするよう **/etc/httpd/conf/httpd.conf** に設定を行なうと、**httpd** の起動に失敗します。

httpd が TCP ポート 80、443、488、8008、8009、8443 以外のポートで実行するようにするには、以下の手順で設定します。

1. root で **/etc/httpd/conf/httpd.conf** ファイルを編集し、SELinux ポリシーでは **httpd** 用に設定されていないポートを **Listen** オプションに記載します。以下の例では、**httpd** が IP アドレス 10.0.0.1、TCP ポート 12345 でリッスンするよう設定します。

```
# 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 で以下のコマンドを実行し、SELinux ポリシーの設定にこの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** を実行しないようになったら、root ユーザーで**semanage** ユーティリティーを実行してポリシー設定からそのポートを削除します。

```
~]# semanage port -d -t http_port_t -p tcp 12345
```

[17] 詳細は、システム管理者のガイド の **Apache HTTP サーバー** のセクションを参照してください。

第14章 Samba

Samba は Server Message Block (SMB) および Common Internet File System (CIFS) プロトコルのオープンソース実装で、多様なオペレーティングシステムにまたがるクライアント間でのファイルおよびプリントサービスを提供します。 [18]

Red Hat Enterprise Linux では、Samba サーバーは *samba* パッケージが提供します。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ことを確認します。

```
~]$ rpm -q samba
package samba is not installed
```

パッケージがインストールされておらず Samba を使用したい場合は、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 samba
```

14.1. Samba と SELinux

SELinux を有効にすると、Samba サーバー (**smbd**) はデフォルトで制限のあるサービスとして実行されます。制限のあるサービスはそのサービス自体のドメイン内で実行され、他の制限のあるサービスとは分離されます。以下の例では、サービス自体のドメイン内で実行している **smbd** プロセスを示しています。この例では、*samba* 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 モードで実行していることを確認します。

   ```
   ~]$ getenforce
   Enforcing
   ```

   SELinux が enforcing モードで実行している場合は、**Enforcing** が返されます。

2. root で 以下のコマンドを実行して **smbd** を起動します。

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

   サービスが稼働していることを確認します。出力は以下のようになり、タイムスタンプのみが異なります。

   ```
   ~]# 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 プロセスに関連する SELinux コンテキストは**system_u:system_r:smbd_t:s0** です。このコンテキストの最後から 2 番目の部分、 **smbd_t** がタイプになります。タイプは、プロセスのドメインやファイルのタイプを定義します。この例の場合、**smbd** プロセスは **smbd_t** ドメイン内で実行しています。

smbd がファイルにアクセスおよび共有をできるようにするには、ファイルに適切な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込みと書き込みができますが、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。このタイプは Apache HTTP Server での使用を目的としているためです。Samba でホームディレクトリーや NFS ボリュームのエクスポートを可能にしたり、Samba がドメインコントローラとしての動作できるようにするなど、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。

14.2. タイプ

ファイルに **samba_share_t** タイプのラベルを付けて Samba によるファイル共有ができるようにします。このタイプのラベル付けはユーザー作成のファイルに限定してください。システムファイルには **samba_share_t** タイプのラベルは付けないよう注意してください。ブール値を有効にすると、これらのラベル付けしたファイルやディレクトリーを共有できるようになります。SELinux では、**/etc/samba/smb.conf** ファイルと Linux パーミッションが適切に設定されていれば、Samba は **samba_share_t** タイプのラベルが付いたファイルに書き込みができるようになります。

samba_etc_t タイプは、**/etc/samba/** 内にある **smb.conf** などの特定ファイルに使用されます。**samba_etc_t** タイプのラベル付けは手作業では行なわないでください。このディレクトリー内のファイルに適切なラベルが付けられていない場合、root で **restorecon -R -v /etc/samba** コマンドを実行して、そのファイルをデフォルトのコンテキストに復元します。**/etc/samba/smb.conf** に **samba_etc_t** タイプのラベルが付いていない場合、Samba サービスの起動が失敗し、SELinux 拒否メッセージがログ記録される可能性があります。以下で、**/etc/samba/smb.conf** に **httpd_sys_content_t** タイプのラベルが付いている場合の拒否メッセージの例を示します。

```
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限レベルのアクセスに基づいています。サービスの実行手段は複数あるため、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、特別なアクセス制限がなく共通ファイル用に予約されている領域などのパブリックディレクトリーに **smbd** が書き込めるようになります。

samba_create_home_dirs

このブール値を有効にすると、Samba が単独で新規のホームディレクトリーを作成できるようになります。これは、PAM などのメカニズムで実行されることが多くあります。

samba_domain_controller

このブール値を有効にすると、Samba がドメインコントローラーとして機能するとともに、**useradd**、**groupadd**、**passwd** などの関連コマンドの実行パーミッションを付与することになります。

samba_enable_home_dirs

この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ります。

samba_export_all_ro

あらゆるファイルやディレクトリーをエクスポートし、読み取り専用のパーミッションを付与します。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やディレクトリーを Samba で共有できるようになります。**samba_export_all_ro** ブール値が有効になっていて **samba_export_all_rw** ブール値が無効の場合、**/etc/samba/smb.conf** で書き込みアクセスが設定され Linux パーミッションでも書き込みアクセスが許可されていても、Samba 共有への書き込みアクセスは拒否されます。

samba_export_all_rw

あらゆるファイルやディレクトリーをエクスポートし、読み取りと書き込みのパーミッションを付与します。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やディレクトリーを Samba でエクスポートできるようになります。**/etc/samba/smb.conf** のパーミッションおよび Linux パーミッションで書き込みアクセスを許可する設定にする必要があります。

samba_run_unconfined

このブール値を有効にすると、Samba が **/var/lib/samba/scripts/** ディレクトリー内で制限の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場合は、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** が Samba 経由で NFS 共有に完全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 が NFS ボリュームを共有できるようになります。

use_samba_home_dirs

Samba のホームディレクトリー用にリモートサーバーを使用する場合、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、上記のリストでは常にこのサービスに関連するブール値がすべて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、以下のコマンドを実行します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の記述を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

sepolicy ユーティリティーを提供する *policycoreutils-devel* パッケージが追加で必要になることに留意してください。

14.4. 設定例

SELinux でどのように Samba サーバーを補完するのか、 Samba サーバーの全機能をどのように管理するのかなど、 実践的な例を以下に示します。

14.4.1. 作成したディレクトリーを共有する

新規のディレクトリーを作成し、そのディレクトリーを Samba で共有します。

1. *samba*、*samba-common*、*samba-client* の各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
   ```

   上記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、これらをインストールします。

   ```
   ~]# yum install package-name
   ```

2. root で **mkdir** を使用して、Samba 経由でファイルを共有するための最上位の新規ディレクトリーを作成します。

   ```
   ~]# mkdir /myshare
   ```

3. root で **touch** ユーティリティー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かを確認する際に使用します。

   ```
   ~]# touch /myshare/file1
   ```

4. SELinux では、**/etc/samba/smb.conf** ファイルおよび Linux パーミッションが適切に設定されていれば、Samba は **samba_share_t** タイプのラベルが付いたファイルの読み取りおよび書き込みが可能になります。root で以下のコマンドを実行し、ファイルコンテキスト設定にラベルの変更を追加します。

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

5. root で **restorecon** ユーティリティーを使用してラベル変更を適用します。

```
~]# 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** ファイルを編集します。Samba で **/myshare/** ディレクトリーを共有するために、以下をこのファイルの末尾に追加します。

```
[myshare]
comment = My share
path = /myshare
public = yes
writeable = no
```

7. Samba ファイルシステムのマウントには Samba アカウントが必要になります。root で以下のコマンドを実行し、Samba アカウントを作成します。 *username* は既存の Linux ユーザーにします。たとえば、**smbpasswd -a testuser** を実行すると、Linux の **testuser** ユーザー用の Samba アカウントが作成されます。

```
~]# 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する際に、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 サービスを開始します。

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

9. 以下のコマンドを実行し、利用可能な共有を表示します。*username* はステップ 7 で追加した Samba アカウントにします。パスワード入力を求められたら、ステップ 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 (バージョン番号は使用しているバージョンによって異なります)。

```
~]$ 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. root で **mkdir** ユーティリティーを使って新規ディレクトリーを作成します。このディレクトリーは Samba 共有の **myshare** をマウントする際に使用します。

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

11. root で以下のコマンドを実行して、Samba 共有の **myshare** を **/test/** にマウントします。*username* はステップ 7 のユーザー名にしてください。

```
~]# mount //localhost/myshare /test/ -o user=username
```

ステップ 7 で設定した *username* のパスワードを入力します。

12. 以下のコマンドを実行して ステップ 3 で作成した **file1** を表示します。

```
~]$ ls /test/
file1
```

14.4.2. web サイトを共有する

/var/www/html/ ディレクトリーで web サイトを共有したい場合などは、ファイルに**samba_share_t** タイプのラベルが付けられないことがあります。このような場合には、**samba_export_all_ro** ブール値を使用して読み取り専用パーミッションを付与して (現在のラベルに関係なく) すべてのファイルやディレクトリーを共有するか、**samba_export_all_rw** を使用して読み取りおよび書き込みパーミッションを付与して (現在のラベルに関係なく) すべてのファイルやディレクトリーを共有します。

以下の例では、**/var/www/html/** 内に web サイトのファイルを作成してから、そのファイルに読み取りおよび書き込みパーミッションを与えて Samba で共有します。ここでは、*httpd*、*samba*、*samba-common*、*samba-client*、*wget* の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 ユーザーで **/var/www/html/file1.html** ファイルを作成します。以下のコンテンツをコピーしてこのファイルに貼り付けます。

```
<html>
<h2>File being shared through the Apache HTTP Server and Samba.
</h2>
</html>
```

2. 以下のコマンドを実行して、**file1.html** の SELinux コンテキストを表示します。

```
~]$ 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
```

このファイルには **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、Apache HTTP Server 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 Server を起動します。

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

4. ユーザーでの書き込みアクセスがあるディレクトリーに移動し、以下のコマンドを実行します。デフォルト設定に変更がなければ、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** ファイルを編集します。Samba で **/var/www/html/** ディレクトリーを共有するために、以下をこのファイルの末尾に追加します。

```
[website]
comment = Sharing a website
path = /var/www/html/
public = no
writeable = 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。 Samba はデフォルトでは、このタイプのラベルが付いたファイルやディレクトリーには、Linux パーミッションで許可されていても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
```

再起動後に変更を維持したくない場合は、**-P** 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 からいずれのタイプにもアクセスもできるようになることに注意してください。

7. Samba サービスを開始します。

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

[18] 詳細は、システム管理者のガイド 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。その目的は、ユーザーがリモートホストに直接ログインしたり、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、高速で安定性があり、また重要な点として安全性を確保するため、土台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能力があることから、**vsftpd** は Red Hat Enterprise Linux に同梱されている唯一のスタンドアロン FTP となります。

Red Hat Enterprise Linux では、Very Secure FTP デーモンは *vsftpd* パッケージで提供されます。以下のコマンドを実行して *vsftpd* がインストールされているか確認します。

```
~]$ rpm -q vsftpd
package vsftpd is not installed
```

FTP サーバーを利用する必要があり、*vsftpd* 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 vsftpd
```

15.1. FTP と SELinux

FTP デーモンの **vsftpd** はデフォルトで制限のあるサービスとして実行されます。**vsftpd** とファイルやプロセス、またシステムとの交信方法は SELinux ポリシーで定義されます。たとえば、認証ユーザーが FTP 経由でログインすると、そのユーザーは自分のホームディレクトリーの読み取りや書き込みはできません。SELinux は、**vsftpd** がユーザーのホームディレクトリーにアクセスすることをデフォルトで禁止するためです。またデフォルトでは、**vsftpd** は NFS や CIFS ボリュームへのアクセスもありません。このため、**/etc/vsftpd/vsftpd.conf** ファイル内で書き込みアクセスが設定されていても、匿名ユーザーには書き込みアクセスが与えられません。ブール値を有効にすると、こうしたアクセスを許可することができます。

以下では、認証ユーザーがログインして、自分のホームディレクトリーを表示しようとすると SELinux が拒否する例を示します。この例では、*vsftpd* パッケージがインストール済みで SELinux ターゲットポリシーが使用されており、SELinux が enforcing モードで実行中であることを前提としています。

1. Red Hat Enterprise Linux では、**vsftpd** はデフォルトでは anonymous ユーザーによるログインしか許可していません。認証ユーザーによるログインを許可するには、root で **/etc/vsftpd/vsftpd.conf** を編集します。**local_enable=YES** オプションをコメント解除します。

   ```
   # Uncomment this to allow local users to log in.
   local_enable=YES
   ```

2. **vsftpd** サービスを起動します。

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

   サービスが稼働していることを確認します。出力は以下のようになり、タイムスタンプのみが異なります。

```
~]# systemctl status vsftpd.service
vsftpd.service - Vsftpd ftp daemon
   Loaded: loaded (/usr/lib/systemd/system/vsftpd.service; disabled)
   Active: active (running) since Tue 2013-08-06 14:42:07 CEST; 6s
ago
```

vsftpd.conf の編集前にサービスが稼働していた場合は、サービスを再起動して設定変更を適用します。

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

3. 現在ログインしているユーザーのまま、以下のコマンドを実行します。ユーザー名のプロンプトが表示されたら、 ログインしているユーザー名になっているか確認します。ユーザー名が正しければ **Enter** を押します。ユーザー名が違う場合は、正しいユーザー名を入力します。

```
~]$ ftp localhost
Connected to localhost (127.0.0.1).
220 (vsFTPd 2.1.0)
Name (localhost:username):
331 Please specify the password.
Password: Enter your password
500 OOPS: cannot change directory:/home/username
Login failed.
ftp>
```

4. 以下のような SELinux 拒否メッセージがログ記録されます。

```
setroubleshoot: SELinux is preventing the ftp daemon from reading
users home directories (username). For complete SELinux messages.
run sealert -l c366e889-2553-4c16-b73f-92f36a1730ce
```

5. ホームディレクトリーへのアクセスが SELinux で拒否されました。これを解決するには、**ftp_home_dir** ブール値を有効にします。root ユーザーで以下のコマンドを実行して、ブール値を有効にします。

```
~]# setsebool -P ftp_home_dir=1
```

注記

再起動後に変更を維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。

再度ログインを試みます。SELinux が **ftp_home_dir** ブール値を使用してホームディレクトリーへのアクセスを許可しているので、ログインが成功します。

15.2. タイプ

デフォルトでは、匿名ユーザーは FTP でログインすると **/var/ftp/** ディレクトリー内のファイルへの読み取りアクセスが与えられます。このディレクトリーには **public_content_t** タイプのラベルが付いているため、**/etc/vsftpd/vsftpd.conf** で書き込みアクセスが設定されていても、許可されるのは読み取り専用アクセスのみになります。**public_content_t** タイプには、Apache HTTP Server、Samba、NFS など他のサービスがアクセス可能です。

FTP 経由でファイルを共有する場合は、以下のいずれかのタイプを使用します。

public_content_t

ユーザーが作成したファイルやディレクトリーを vsftpd 経由の読み取り専用で共有する場合に、**public_content_t** タイプのラベルを付けます。このタイプのラベルが付いているファイルには、Apache HTTP Server、Samba、NFS など、他のサービスからもアクセスすることができます。**public_content_t** タイプのラベルが付いたファイルへの書き込みは、Linux パーミッションで書き込みが許可されていてもできません。書き込みアクセスが必要な場合は、**public_content_rw_t** タイプを使用してください。

public_content_rw_t

ユーザーが作成したファイルやディレクトリーを **vsftpd** 経由の読み取りおよび書き込みのパーミッションで共有する場合に、**public_content_rw_t** タイプのラベルを付けます。このタイプのラベルが付いているファイルには、Apache HTTP Server、Samba、NFS など、他のサービスからもアクセスすることができます。このタイプのラベルが付いたファイルに書き込みを行う場合は、まず最初に各サービスのブール値を有効にしておく必要がある点に注意してください。

15.3. ブール値

SELinux は、サービスの実行に必要な最小限レベルのアクセスに基づいています。サービスの実行手段は複数あるため、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** タイプのラベルが付いたファイルおよびディレクトリーへの書き込みが禁止されます。有効にすると、ユーザーが FTP 経由でファイルのアップロードをできるようになります。ファイルのアップロード先となるディレクトリーには **public_content_rw_t** タイプのラベルを付け、Linux パーミッションも適切に設定しておく必要があります。

ftpd_full_access

このブール値を有効にすると、アクセス制御に Linux (DAC) のパーミッションのみが使用されるので、認証ユーザーはファイルに **public_content_t** や **public_content_rw_t** のタイプのラベルが付いていなくてもファイルの読み取りおよび書き込みが可能になります。

ftpd_use_cifs

このブール値を有効にすると、**vsftpd** が **cifs_t** タイプのラベルが付いたファイルやディレクトリーにアクセスできるようになります。したがって、このブール値を有効にすると、Samba でマウントしたファイルシステムを **vsftpd** で共有することができるようになります。

ftpd_use_nfs

このブール値を有効にすると、**vsftpd** が **nfs_t** タイプのラベルが付いたファイルやディレクトリーにアクセスできるようになります。したがって、このブール値を有効にすると、NFS でマウントしたファイルシステムを **vsftpd** で共有することができるようになります。

ftp_home_dir

このブール値を有効にすると、認証ユーザーによるユーザーのホームディレクトリー内のファイルの読み取りと書き込みが許可されます。このブール値が無効だと、ホームディレクトリーからファイルのダウンロードを試みた場合、**550 Failed to open file** などのエラーが発生します。SELinux 拒否メッセージがログ記録されます。

ftpd_connect_db

FTP デーモンによるデータベースへの接続開始を許可します。

httpd_enable_ftp_server

httpd デーモンによる FTP ポートでのリッスンおよび FTP サーバーとしての動作を許可します。

tftp_anon_write

このブール値を有効にすると、特別なアクセス制限がなく共通ファイル用に予約されている領域などのパブリックディレクトリーへの TFTP によるアクセスが許可されます。

注記

SELinux ポリシーは継続的に開発されているため、上記のリストでは常にこのサービスに関連するブール値がすべて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、以下のコマンドを実行します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の記述を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

sepolicy ユーティリティーを提供する *policycoreutils-devel* パッケージが追加で必要になることに留意してください。

15.4. 設定例

15.4.1. FTP サイトにアップロードする

以下では、特定のユーザーによるファイルのアップロードが可能な FTP サイトを作成している例を示します。ディレクトリー構造を作成し、必要となる SELinux 設定の変更を行なっています。

1. root ユーザーで以下のコマンドを実行して FTP ホームディレクトリーへのアクセスができるようにします。

```
~]# setsebool ftp_home_dir=1
```

2. root で **mkdir** ユーティリティーを使用して新規の最上位ディレクトリーを作成します。

```
~]# mkdir -p /myftp/pub
```

3. **/myftp/pub/** ディレクトリーで Linux パーミッションを設定し、Linux ユーザーの書き込みアクセスを許可します。以下の例では、所有者とグループを root から所有者 **user1** とグループ root に変更します。**user1** を書き込みアクセスを付与するユーザーで置き換えます。

```
~]# chown user1:root /myftp/pub
~]# chmod 775 /myftp/pub
```

chown コマンドは、所有者とグループのパーミッションを変更します。このコマンドはモードを変更し、**user1** ユーザーには読み取り、書き込み、実行のパーミッションを許可し、root グループのメンバーには読み取り、書き込み、実行のパーミッションを許可しています。これ以外のユーザーには読み取りと実行のパーミッションを許可しています。このパーミッションにより、Apache HTTP Server はこのディレクトリー配下にあるファイルを読み取ることができます。

4. SELinux を実行する場合は、ファイルやディレクトリーに適切なアクセス許可のラベルを付ける必要があります。Linux パーミッションの設定だけでは不十分です。**public_content_t** タイプのラベルが付いているファイルは、FTP、Apache HTTP Server、Samba、および rsync による読み込みが可能です。**public_content_rw_t** タイプのラベルが付いているファイルは、FTP による書き込みが可能です。Samba など FTP 以外のサービスが **public_content_rw_t** タイプのラベルが付いているファイルに書き込みできるようにするには、ブール値を設定しておく必要があります。最上位のディレクトリー (**/myftp/**) に **public_content_t** タイプのラベルを付けると、このディレクトリー配下にあるコピーまたは新規作成されたファイルには、サービスによる書き込みや変更ができなくなります。root で以下のコマンドを実行し、ラベルの変更をファイルコンテキスト設定に追加します。

```
~]# semanage fcontext -a -t public_content_t /myftp
```

5. **restorecon** ユーティリティーを使用してラベル変更を適用します。

```
~]# restorecon -R -v /myftp/
restorecon reset /myftp context unconfined_u:object_r:default_t:s0-
>system_u:object_r:public_content_t:s0
```

6. **/myftp/** には **public_content_t** タイプのラベルが、**/myftp/pub/** には **default_t** タイプのラベルが付いていることを確認します。

```
~]$ ls -dZ /myftp/
drwxr-xr-x. root root system_u:object_r:public_content_t:s0 /myftp/
~]$ ls -dZ /myftp/pub/
drwxrwxr-x. user1 root unconfined_u:object_r:default_t:s0
/myftp/pub/
```

7. ユーザーが FTP でファイルをアップロードできるようにするには、まず先に FTP がディレクトリーに書き込み可能となっていることが必要です。SELinux では、FTP は **public_content_rw_t** タイプのラベルが付いたディレクトリーへの書き込みが可能になっています。この例では、FTP が書き込み可能とするディレクトリーに **/myftp/pub/** を使用しています。root ユーザーで以下のコマンドを実行し、ラベルの変更をファイルコンテキスト設定に追加します。

```
~]# semanage fcontext -a -t public_content_rw_t "/myftp/pub(/.*)?"
```

8. root で **restorecon** ユーティリティーを使用してラベル変更を適用します。

```
~]# restorecon -R -v /myftp/pub
restorecon reset /myftp/pub context system_u:object_r:default_t:s0-
>system_u:object_r:public_content_rw_t:s0
```

9. **vsftpd** が **public_content_rw_t** タイプのラベルの付いたファイルに書き込みできるようにするには、**ftpd_anon_write** ブール値を有効にする必要があります。root で以下のコマンドを実行して、ブール値を有効にします。

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

注記

再起動後に変更を維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。

以下では、FTP でログインしてからファイルをアップロードする例を示します。ユーザーは前述の例と同じ **user1** ユーザーを使用しています。**/myftp/pub/** ディレクトリーは **user1** が専用所有者となるディレクトリーです。

1. ホームディレクトリーに移動します。次に FTP 経由でアップロードするファイルを格納するディレクトリーを作成します。

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

```
~]$ mkdir myftp
```

2. **~/myftp/** ディレクトリーに移動します。

```
~]$ cd ~/myftp
```

このディレクトリー内に **ftpupload** 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ます。

```
File upload via FTP from a home directory.
```

3. **ftpd_anon_write** ブール値が有効になっていることを確認します。

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

ブール値が無効になっている場合は、root で以下のコマンドを実行して有効にします。

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

注記

再起動後に変更を維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。

4. **vsftpd** サービスを起動します。

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

5. 以下のコマンドを実行します。ユーザー名の入力が求められたら、書き込みアクセスのあるユーザーのユーザー名を入力し、そのユーザーの正しいパスワードを入力します。

```
~]$ ftp localhost
Connected to localhost (127.0.0.1).
220 (vsFTPd 2.1.0)
Name (localhost:username):
331 Please specify the password.
```

```
Password: Enter the correct password
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd myftp
250 Directory successfully changed.
ftp> put ftpupload
local: ftpupload remote: ftpupload
227 Entering Passive Mode (127,0,0,1,241,41).
150 Ok to send data.
226 File receive OK.
ftp> 221 Goodbye.
```

ftpd_anon_write ブール値が有効になっているので、アップロードが成功します。

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) を使うと、リモート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かのように操作することができます。これにより、システム管理者はネットワーク上のサーバーにリソースを統合することができるようになります。 [19]

Red Hat Enterprise Linux では、NFS の完全サポートに *nfs-utils* パッケージが必要になります。以下のコマンドを実行して、*nfs-utils* がインストールされているか確認します。

```
~]$ 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 を使用したい場合は、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 nfs-utils
```

16.1. NFS と SELinux

SELinux の実行時には、NFS デーモンはデフォルトで制限されています。例外は **nfsd** プロセスで、これには制限のない **kernel_t** ドメインタイプのラベルが付いています。SELinux ポリシーはデフォルトで、NFS によるファイル共有を許可します。また、クライアントとサーバー間での SELinux ラベルの受け渡しもサポートしており、これにより NFS ボリュームにアクセスする制限のあるドメインのセキュリティ制御が向上します。たとえば、NFS ボリューム上にホームディレクトリーを設定する際に、そのボリューム上の他のディレクトリーにはアクセスできず、このホームディレクトリーにのみアクセス可能な制限のあるドメインを指定することができます。同様に、Secure Virtualization といったアプリケーションが NFS ボリューム上で画像ファイルのラベルを設定できることで、仮想マシンの分離レベルが高まります。

ラベルが付いた NFS のサポートは、デフォルトでは無効になっています。これを有効にする方法については、「SELinux ラベルが付いた NFS サポートを有効にする」を参照してください。

16.2. タイプ

デフォルトでは、クライアント側にマウントした NFS ボリュームには、ポリシーで定義された NFS 用のデフォルト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あれば、このデフォルトのコンテキストには **nfs_t** タイプが使用されます。root ユーザーは、**mount -context** オプションを使用してこのデフォルトタイプを上書きすることができます。NFS で使用されるタイプは以下のとおりです。異なるタイプを使用することで、柔軟なアクセス設定ができます。

var_lib_nfs_t

このタイプは、**/var/lib/nfs/** ディレクトリー内の既存ファイルおよびこのディレクトリーにコピーまたは新規作成されるファイルに使用されます。通常の操作では、このタイプを変更する必要はありません。加えられた変更をデフォルトの設定に復元する場合は、root ユーザーで **restorecon -R -v /var/lib/nfs** コマンドを実行します。

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ます。また、NFS 関連の実行可能なシステムファイルやライブラリにも、このタイプのラベルが付けられます。ユーザーはこのタイプをファイルにラベル付けしないでください。**nfsd_exec_t** は **nfsd_t** に切り替わります。

16.3. ブール値

SELinux は、サービスの実行に必要な最小限レベルのアクセスに基づいています。サービスの実行手段は複数あるため、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** デーモンが 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** デーモンが 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 システムデーモンが NFS ボリューム上のシステム共有リポジトリーを読み取ることが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、**httpd** が NFS ボリューム上に格納された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。無効にすると、**smbd** は Samba を介した NFS 共有へのフルアクセスが禁止されます。

sanlock_use_nfs

このブール値を有効にすると、**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** スケジューラーが 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、制限のある仮想ゲストが NFS ボリューム上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** が NFS ボリューム上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、Git Common Gateway Interface (CGI) が 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、上記のリストでは常にこのサービスに関連するブール値がすべて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、以下のコマンドを実行します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の記述を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

sepolicy ユーティリティーを提供する *policycoreutils-devel* パッケージが追加で必要になることに留意してください。

16.4. 設定例

16.4.1. SELinux ラベルが付いた NFS サポートを有効にする

以下の例では、SELinux ラベルが付いた NFS サポートを有効にする方法を示しています。ここでは、*nfs-utils* パッケージがインストール済みで SELinux ターゲットポリシーが使用されており、SELinux が enforcing モードで実行中であることを前提としています。

注記

次のステップ 1 からステップ 3 までは、NFS サーバー **nfs-srv** で行います。

1. NFS サーバーが稼働している場合は、これを停止します。

   ```
   [nfs-srv]# systemctl stop nfs
   ```

   サーバーが停止したことを確認します。

   ```
   [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、稼働していることを確認します。出力は以下のようになり、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. クライアント側で 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渡されました。

```
[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

ラベルが付いた NFS サポートをホームディレクトリーやその他のコンテンツに有効にすると、そのコンテンツは EXT ファイルシステム上にある場合と同様のラベルが付けられます。また、異なるバージョンの NFS があるシステムをマウントしたり、ラベルが付いた NFS をサポートしないサーバーのマウントを試みると、エラーが返されることに留意してください。

[19] 詳細は、ストレージ管理ガイド の **Network File System (NFS)** の章を参照してください。

第17章 BIND (Berkeley Internet Name Domain)

BIND では **named** デーモンを使って名前解決サービスを行ないます。BIND を使うと、ユーザーは数値アドレスではなく名前でコンピューターリソースやサービスを検索することができます。

Red Hat Enterprise Linux では、*bind* パッケージが DNS サーバーを提供しています。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ことを確認します。

```
~]$ rpm -q bind
package bind 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 bind
```

17.1. BIND と SELinux

/var/named/slaves/、**/var/named/dynamic/**、**/var/named/data/** ディレクトリーのデフォルトパーミッションでは、ゾーン転送およびダイナミック DNS 更新によるゾーンファイルの更新が許可されます。**/var/named/** 内のファイルには **named_zone_t** タイプのラベルが付けられ、マスターゾーンファイルに使用されます。

スレーブサーバーの場合、**/etc/named.conf** ファイルでスレーブゾーンを **/var/named/slaves/** に配置するよう設定します。以下に、スレーブ DNS サーバーの **/etc/named.conf** 内にあるドメインエントリーの例を示します。このスレーブ DNS サーバーは、**/var/named/slaves/** 内に **testdomain.com** 用のゾーンファイルを格納します。

```
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** のラベルが付けられている場合は、**named_write_master_zones** ブール値を有効にして、ゾーンファイル更新のためのゾーン転送とダイナミック DNS を許可する必要があります。また、親ディレクトリーのモードを変更して、**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必要があります。

/var/named/ 内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や **restorecon -R /var/** を実行するとそのタイプが**named_zone_t** に変更されます。

17.2. タイプ

BIND で使用されるタイプを以下に示します。 タイプに応じて柔軟なアクセス設定ができます。

named_zone_t

マスターゾーンファイルに使用されます。他のサービスは、このタイプのファイルを変更することはできません。**named_write_master_zones** のブール値が有効な場合に **named** デーモンのみがこのタイプのファイルを変更できます。

named_cache_t

このタイプのラベルが付いたファイルには、特にブール値の設定がなくてもデフォルトで named による書き込みが可能です。**/var/named/slaves/**、**/var/named/dynamic/**、**/var/named/data/** のディレクトリー内にコピーまたは作成されるファイルには、**named_cache_t** タイプのラベルが自動的に付けられます。

named_var_run_t

/var/run/bind/、**/var/run/named/**、**/var/run/unbound/** のディレクトリー内にコピーまたは作成されるファイルには、**named_var_run_t** タイプのラベルが自動的に付けられます。

named_conf_t

BIND 関連の設定ファイル (通常 **/etc/** ディレクトリーに格納) には、**named_conf_t** タイプのラベルが自動的に付けられます。

named_exec_t

BIND 関連の実行可能ファイル (通常 **/usr/sbin/** ディレクトリーに格納) には、**named_exec_t** タイプのラベルが自動的に付けられます。

named_log_t

BIND 関連のログファイル (通常 **/var/log/** ディレクトリーに格納) には、**named_log_t** タイプのラベルが自動的に付けられます。

named_unit_file_t

/usr/lib/systemd/system/ ディレクトリー内にある実行可能な BIND 関連のファイルには、**named_unit_file_t** タイプのラベルが自動的に付けられます。

17.3. ブール値

SELinux は、サービスの実行に必要な最小限レベルのアクセスに基づいています。サービスの実行手段は複数あるため、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** タイプのラベルが付いたゾーンファイルやディレクトリーに書き込みができなくなります。このデーモンは通常、ゾーンファイルへの書き込みを必要としません。ただし、セカンダリーサーバーがゾーンファイルへの書き込みを必要とする場合などには、このブール値を有効にして書き込みを許可します。

named_tcp_bind_http_port

このブール値を有効にすると、BIND が Apache ポートをバインド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、上記のリストでは常にこのサービスに関連するブール値がすべて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、以下のコマンドを実行します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の記述を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

sepolicy ユーティリティーを提供する *policycoreutils-devel* パッケージが追加で必要になることに留意してください。

17.4. 設定例

17.4.1. ダイナミック DNS

BIND を使用すると、ホストがゾーンファイルや DNS 内の記録を動的に更新することができるようになります。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 レコードでリアルタイムの修正が必要となる場合に BIND を使用します。

ダイナミック DNS で更新するゾーンファイル用には、**/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に作成またはコピーされるファイルは、**named** による書き込みを許可する Linux パーミッションを継承します。また、こうしたファイルには **named_cache_t** タイプのラベルが付けられるため、SELinux は **named** がこれらのファイルに書き込むことを許可します。

/var/named/dynamic/ 内のゾーンファイルに **named_zone_t** タイプのラベルが付けられている場合、動的 DNS 更新がマージされる前にまずジャーナルに書き込まれる必要があるため、一定期間この動的 DNS の更新に失敗することがあります。ジャーナルのマージ試行時にゾーンファイル に**named_zone_t** タイプのラベルが付けられていると、以下のようなエラーがログ記録されます。

```
named[PID]: dumping master file: rename: /var/named/dynamic/zone-name:
permission denied
```

また、以下のような SELinux 拒否メッセージもログ記録されます。

```
setroubleshoot: SELinux is preventing named (named_t) "unlink" to zone-
name (named_zone_t)
```

このラベル付け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 CVS (Concurrent Versioning System)

CVS (Concurrent Versioning System) は、無料のバージョン管理システムです。中央に置かれた複数ファイルのセットに対する変更の監視および追跡に使用します。一般的に複数のユーザーがアクセスします。ソースコードリポジトリの管理などによく使用され、オープンソースの開発者の間では幅広く使用されています。

Red Hat Enterprise Linux では、*cvs* パッケージが CVS を提供します。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ことを確認します。

```
~]$ 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 を使用したい場合は、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 cvs
```

18.1. CVS と SELinux

cvs デーモンは **cvs_t** タイプのラベルが付けられて実行されます。Red Hat Enterprise Linux ではデフォルトで、CVS が読み取りと書き込み可能なのは特定のディレクトリーに限られます。**cvs_data_t** のラベルが、**cvs** の読み取りと書き込みのアクセス領域を定義します。SELinux で CVS を使用する場合、クライアントが CVS データ用に予約されている領域に完全にアクセスできるようにするには、適切なラベルの割り当てが必須になります。

18.2. タイプ

CVS で使用されるタイプを以下に示します。 タイプに応じて柔軟なアクセス設定ができます。

cvs_data_t

このタイプは CVS リポジトリ内のデータに対して使用されます。CVS が完全にアクセスできるのはこのタイプのデータのみです。

cvs_exec_t

このタイプは **/usr/bin/cvs** バイナリに対して使用されます。

18.3. ブール値

SELinux は、サービスの実行に必要な最小限レベルのアクセスに基づいています。サービスの実行手段は複数あるため、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

このブール値は、**cvs** デーモンがユーザー認証用 **/etc/shadow** ファイルにアクセスできるようにします。

注記

SELinux ポリシーは継続的に開発されているため、上記のリストでは常にこのサービスに関連するブール値がすべて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、以下のコマンドを実行します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の記述を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

sepolicy ユーティリティーを提供する *policycoreutils-devel* パッケージが追加で必要になることに留意してください。

18.4. 設定例

18.4.1. CVS のセットアップ

以下の例では、リモートアクセスを許可する SELinux 設定と簡単な CVS セットアップを示しています。使用する 2 台のホストは、ホスト名が **cvs-srv** で IP アドレスが **192.168.1.1** 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** のクライアントです。いずれのホストも同一サブネット上にあります (192.168.1.0/24)。 これは一例に過ぎず、*cvs* と *xinetd* パッケージがインストールされていること、SELinux ターゲットポリシーを使用していること、SELinux は enforced モードで実行していることを前提としています。

ここでは、DAC の全パーミッションが許可されている場合でも、SELinux ではファイルのラベルに基づくポリシールールが強制でき、明確に CVS アクセス用のラベルが付けられている特定領域へのアクセスのみを許可することができることを例示しています。

注記

ステップ 1 から 9 は CVS サーバー**cvs-srv** で行ないます。

1. この例では、*cvs* と *xinetd* のパッケージが必要になります。これらの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
   ```

   これら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。

   ```
   [cvs-srv]# yum install cvs xinetd
   ```

2. root で以下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** ユーティリティーで行うこともできます。

3. **cvsuser** というユーザー名のユーザーを作成し、このユーザーを CVS グループのメンバーにします。**system-config-users** を使用します。

4. **/etc/services** ファイルを編集して、以下のように CVS サーバーのエントリをコメント解除します。

```
cvspserver 2401/tcp   # CVS client/server operations
cvspserver 2401/udp   # CVS client/server operations
```

5. CVS リポジトリーをファイルシステムの root 領域に作成します。SELinux を使用する場合、リポジトリーは root ファイルシステムに配置するのが最適です。こうすることで、他のサブディレクトリーに影響を与えることなく、再帰的なラベルを与えることができます。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. 誰でもアクセスできるように **/cvs/** ディレクトリーにすべてのパーミッションを与えます。

```
[root@cvs-srv]# chmod -R 777 /cvs
```

警告

これは説明を目的とした例に過ぎません。実稼働システムでは、ここで示すパーミッションを使用しないでください。

7. **/etc/xinetd.d/cvs** ファイルを編集し、CVS セクションをコメント解除して**/cvs/** ディレクトリーを使用するよう設定します。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って、ポート 2401 上で TCP を使用した着信接続を許可するルールを追加します。

10. クライアント側では、**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. これで CVS は設定されましたが、SELinux ではログインおよびファイルのアクセスが拒否されます。これを確認するため、 **cvs-client** で **$CVSROOT** 変数を設定し、リモートによるログインを試行します。以下のステップは **cvs-client** で行なってください。

```
[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 でこのアクセスを許可するためには、以下のステップを **cvs-srv** で行なってください。

12. root で **/cvs/** ディレクトリーのコンテキストを変更し、**cvs_data_t** タイプを付与して、**/cvs/** 内の既存のデータおよび新規のデータすべてに再帰的にラベル付けが行なわれるようにします。

```
[root@cvs-srv]# semanage fcontext -a -t cvs_data_t '/cvs(/.*)?'
[root@cvs-srv]# restorecon -R -v /cvs
```

13. これで、クライアント **cvs-client** はログインして、このリポジトリ内のすべての CVS リソースにアクセスできるようになりました。

```
[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

Squid とは、HTTP、Gopher、FTP データオブジェクトに対応する、Web クライアント用の高パフォーマンスなプロキシキャッシングサーバーです。頻繁に要求される Web ページをキャッシングして再利用することで、帯域幅を抑え、応答時間を改善します。 [20]

Red Hat Enterprise Linux では、*squid* パッケージが Squid キャッシングプロキシを提供します。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ことを確認します。

```
~]$ 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 を使用したい場合は、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
```

19.1. Squid キャッシングプロキシと SELinux

SELinux を有効にすると、squid はデフォルトで制限のあるサービスとして実行されます。制限のあるプロセスはそれ自体のドメイン内で実行され、他の制限のあるプロセスとは分離されます。制限のあるプロセスが攻撃を受けると、SELinux ポリシー設定に応じて、攻撃側のリソースへのアクセスと攻撃者による被害は限定されます。以下で squid が自身のドメイン内で実行している squid プロセスの例を示します。ここでは *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 モードで実行していることを確認します。

   ```
   ~]$ getenforce
   Enforcing
   ```

   SELinux が enforcing モードで実行している場合は、このコマンドは **Enforcing** を返します。

2. root ユーザーで以下のコマンドを実行し、**squid** デーモンを起動します。

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

   サービスが稼働していることを確認します。出力は以下のようになり、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 Mon 2013-08-05 14:45:53 CEST; 2s
   ago
   ```

3. 以下のコマンドを実行して、**squid** プロセスを表示します。

   ```
   ~]$ 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 プロセスに関連する SELinux コンテキストは**system_u:system_r:squid_t:s0** です。コンテキストの最後から 2 番目の部分、 **squid_t** がタイプになります。タイプは、プロセスのドメインやファイルのタイプを定義します。この例の場合、squid プロセスは **squid_t** ドメイン内で実行しています。

SELinux ポリシーは、**squid_t** などのように、制限のあるドメイン内で実行しているプロセスがファイルや他のプロセス、システム全般などとどのように対話するのかを定義します。squid がファイルにアクセス可能とするには、ファイルに適切なラベルを付ける必要があります。

/etc/squid/squid.conf ファイルを設定して、**squid** がデフォルトの TCP ポート 3128、3401、4827 以外のポートでリッスンするようにするには、**semanage port** コマンドを使って SELinux ポリシー設定にそのポート番号を追加する必要があります。以下では、SELinux ポリシー設定では最初に **squid** 用に定義されていなかったポートでリッスンするように設定したため、このサーバーの起動に失敗する例を示します。また、SELinux システムを設定し、ポリシーではまだ定義されていなかった非標準のポートでこのデーモンがリッスンできるようにする方法についても示します。ここでは、*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。各コマンドは root ユーザーで実行してください。

1. **squid** が実行中では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)
```

出力が上記と異なる場合は、以下のコマンドを実行してこのプロセスを停止します。

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

2. 以下のコマンドを実行して、SELinux で **squid** にリッスンを許可しているポートを表示します。

```
~]# 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。SELinux ポリシー設定では**squid** 用に設定していないポートをリッスンするよう **http_port** オプションを設定します。この例では、このデーモンがポート 10000 でリッスンするよう設定します。

```
# Squid normally listens to port 3128
http_port 10000
```

4. **setsebool** コマンドを実行し、**squid_connect_any** ブール値をオフに設定します。これで、**squid** の動作は特定ポート上に限られることになります。

```
~]# 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 拒否メッセージがログ記録されます。

```
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 **squid** がこの例で使用しているポート 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になります。

```
~]# semanage port -a -t squid_port_t -p tcp 10000
```

7. **squid** を再起動して、新規ポートをリッスンするようにします。

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

8. これで、Squid が非標準ポート (この例では TCP 10000) でリッスンできるように SELinux を設定したので、このデーモンはこのポートで正常に起動するようになります。

19.2. タイプ

Type Enforcement が SELinux のターゲットポリシーで使用されるメインのパーミッション制御になります。全ファイルおよびプロセスにタイプのラベルが付けられます。タイプはプロセスのドメインを定義し、ファイルのタイプを定義します。SELinux ポリシールールは、ドメインがタイプにアクセスする場合でも、ドメインが別のドメインにアクセスする場合でも、タイプ同士がアクセスする方法を定義します。アクセスを許可する特定の SELinux ポリシールールが存在する場合にのみ、アクセスは許可されます。

Squid で使用されるタイプを以下に示します。タイプに応じて柔軟なアクセス設定ができます。

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。Squid とその設定に関するさまざまな統計数字を提供します。

squid_cache_t

このタイプは、**/etc/squid/squid.conf** 内の **cache_dir** ディレクティブで定義しているように、squid がキャッシュするデータに使用します。デフォルトでは、**/var/cache/squid/** および **/var/spool/squid/** ディレクトリーにコピーまたは作成されるファイルには **squid_cache_t** タイプのラベルが付けられます。また、**/var/squidGuard/** ディレクトリーにコピーまたは作成される**squid** 用の squidGuard URL リダイレクトプラグインのファイルにも**squid_cache_t** タイプのラベルが付けられます。Squid がキャッシュデータ用として使用できるのは、このラベルが付いたファイルやディレクトリーのみです。

squid_conf_t

このタイプは、Squid の設定に使用されるディレクトリーおよびファイルに対して使用されます。エラーメッセージやアイコンなどを含め、**/etc/squid/** および **/usr/share/squid/** 内に既存するファイルや、ここに作成またはコピーされるファイルにはこのタイプのラベルが付けられます。

squid_exec_t

このタイプは **squid** バイナリの **/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。**/var/log/squid/** または **/var/log/squidGuard/** 内に既存するファイル、ここに作成またはコピーされるファイルにはこのタイプのラベルを付け

関に関連するファイル、ここに作成またはコピーされるファイルにはこのタイプのラベルを付けなければなりません。

squid_initrc_exec_t

このタイプは、**squid** の起動に必要となる初期設定ファイルに使用します。初期設定ファイルは **/etc/rc.d/init.d/squid** にあります。

squid_var_run_t

このタイプは **/var/run/** ディレクトリー内のファイルに使用されます。特に、Squid の実行時に作成される **/var/run/squid.pid** という名前のプロセス ID (PID) にはこのタイプが付けられます。

19.3. ブール値

SELinux は、サービスの実行に必要な最小限レベルのアクセスに基づいています。サービスの実行手段は複数あるため、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 はどのポートでもリモートホストへの接続を開始できます。

squid_use_tproxy

このブール値を有効にすると、Squid は透過プロキシ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、上記のリストでは常にこのサービスに関連するブール値がすべて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、以下のコマンドを実行します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の記述を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

sepolicy ユーティリティーを提供する *policycoreutils-devel* パッケージが追加で必要になることに留意してください。

19.4. 設定例

19.4.1. Squid の非標準ポートへの接続

以下では、上記のブール値を実行し、特定のポートにへのアクセスのみをデフォルトで許可することで SELinux が Squid を補完している実用的な例を示します。また、ブール値を変更し、その変更により許可されるアクセスについても示します。

以下に示す例は、シンプルな Squid 設定に対してどのように SELinux が影響を与えることができるのかを

示す一例に過ぎません。Squid に関する総合的な説明は本ガイドの対象外となります。詳細については、公式の Squid ドキュメント を参照してください。ここでは、Squid ホストにはインターネットアクセスがあり、2 種類のネットワークインターフェースが備わっていることを前提としています。またファイアウォールでは、Squid がリッスンするデフォルトの TCP ポート (TCP 3128) を使って内部インターフェース上のアクセスを許可するよう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. メインの設定ファイル **/etc/squid/squid.conf** を編集し、**cache_dir** ディレクティブが以下のようにコメント解除されていることを確認します。

```
cache_dir ufs /var/spool/squid 100 16 256
```

この行では、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定義しています。Squid ストレージフォーマット (**ufs**)、キャッシュを配置するシステム上のディレクトリー (**/var/spool/squid**)、キャッシュに使用するメガバイト単位のディスク領域 (**100**)、作成される第一レベルのキャッシュディレクトリー数と第二レベルのキャッシュディレクトリー数 (それぞれ **16** と **256**) の設定情報で構成されています。

3. 同じ設定ファイル内で、**http_access allow localnet** ディレクティブもコメント解除されていることを確認してください。これにより、Red Hat Enterprise Linux では 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。こうすることで、既存の RFC1918 ネットワーク上のクライアントマシンがプロキシ経由でアクセスできるようになります (この設定例では十分なものです)。

4. 同じ設定ファイル内で **visible_hostname** ディレクティブがコメント解除され、マシンのホスト名が設定されていることを確認してください。値はホストの完全修飾ドメイン名 (FQDN) にします。

```
visible_hostname squid.example.com
```

5. root で以下のコマンドを実行し、**squid** デーモンを起動します。これが**squid** の初回の起動なので、上記の **cache_dir** ディレクティブで指定したキャッシュディレクトリーがこのコマンドで初期化され、デーモンが起動します。

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

squid 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、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** プロセス ID (PID) が制限のあるサービスとして起動されていることを確認します。この例では **squid_var_run_t** の値で確認します。

```
~]# 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、前に設定していた **localnet** ACL に接続しているクライアントマシンは、そのプロキシとしてこのホストの内部インターフェースを使用できるようになります。これはシステム全体または一般的な Web ブラウザすべてのセッティングで設定することができます。これで Squid では目的のマシンのデフォルトポートでリッスンするようになりますが (TCP 3128)、目的のマシンで許可されるのは、一般的な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になります。これが SELinux 自体で定義されているポリシーになります。SELinux では、次のステップで示すように非標準のポートへのアクセスは拒否されます。

8. TCP ポート 10000 での web サイトのリスニングなど、クライアントが Squid プロキシを介して非標準のポートを使った要求を行うと、以下のような拒否がログ記録されます。

```
SELinux is preventing the squid daemon from connecting to network
port 10000
```

9. このアクセスを許可するには、デフォルトでは無効になっている **squid_connect_any** ブール値を変更する必要があります。

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

注記

再起動後に **setsebool** による変更を維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 がクライアントの代わりにどのポートでも接続を開始できるようになったので、クライアントはインターネット上の非標準のポートにアクセスできるようになります。

---

[20] 詳細情報は、Squid Caching Proxy プロジェクトページを参照してください。

第20章 MariaDB (MySQL の後継)

MariaDB データベースはマルチユーザー、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構成されています。[21]

Red Hat Enterprise Linux では、*mariadb-server* パッケージが MariaDB を提供します。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ことを確認します。

```
~]$ rpm -q mariadb-server
package mariadb-server 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 mariadb-server
```

20.1. MariaDB と SELinux

MariaDB を有効にすると、デフォルトで制限のあるサービスとして実行されます。制限のあるプロセスはそれ自体のドメイン内で実行され、他の制限のあるプロセスとは分離されます。制限のあるプロセスが攻撃を受けると、SELinux ポリシー設定に応じて、攻撃側がリソースにアクセスして加えることができる被害が限定されます。以下では、MariaDB 自体のドメイン内で実行している MariaDB プロセスの例を示します。ここでは *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 モードで実行していることを確認します。

   ```
   ~]$ getenforce
   Enforcing
   ```

   SELinux が enforcing モードで実行している場合は、**Enforcing** が返されます。

2. root ユーザーで以下のコマンドを実行し、**mariadb** を起動します。

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

   サービスが稼働していることを確認します。出力は以下のようになり、タイムスタンプのみが異なります。

   ```
   ~]# 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. 以下のコマンドを実行して、**mysqld** プロセスを表示します。

   ```
   ~]$ 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 プロセスに関連する SELinux コンテキストは**system_u:system_r:mysqld_t:s0** です。このコンテキストの最後から 2 番目の部分、 **mysqld_t** がタイプになります。タイプは、プロセスのドメインやファイルのタイプを定義します。この例の場合、**mysqld** プロセスは **mysqld_t** ドメイン内で実行しています。

20.2. タイプ

Type Enforcement が SELinux のターゲットポリシーで使用されるメインのパーミッション制御になります。ファイルおよびプロセスのすべてにタイプのラベルが付けられます。ファイルの場合はタイプ、プロセスの場合はドメインを定義します。SELinux ポリシールールは、ドメインがタイプにアクセスする場合でも、ドメインが別のドメインにアクセスする場合でも、タイプ同士がアクセスする方法を定義します。アクセスを許可する特定の SELinux ポリシールールが存在する場合にのみ、アクセスは許可されます。

mysqld で使用されるタイプを以下に示します。タイプに応じて柔軟なアクセス設定ができます。

mysqld_db_t

このタイプは MariaDB データベースの場所に使用します。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、これは変更可能です。MariaDB データベースの場所を変更する場合は、新しい場所にこのタイプのラベルを付ける必要があります。データベースのデフォルトの場所を変更し、新しいセクションに適切なラベルを付ける方法については、「MariaDB のデータベース格納場所を変更する」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** と、**/etc/mysql/** ディレクトリー内にある他の設定ファイルすべてに使用されます。

mysqld_exec_t

このタイプは **/usr/libexec/mysqld** にある **mysqld** バイナリに使用されます。Red Hat Enterprise Linux ではこれが MariaDB バイナリのデフォルトの場所になります。他のシステムでは、このバイナリは **/usr/sbin/mysqld** に配置されることがあります。この場合でも、このタイプのラベルを付けてください。

mysqld_unit_file_t

このタイプは、Red Hat Enterprise Linux ではデフォルトで **/usr/lib/systemd/system/** ディレクトリーに配置されている MariaDB 関連の実行可能ファイルに使用されます。

mysqld_log_t

MariaDB のログが正常に動作するには、このタイプのラベルを付けてる必要があります。**/var/log/** 内にあるログファイルで、**mysql.*** のワイルドカードに一致するログファイルはすべて、このタイプ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/** 内のファイルで、特に **mysqld** デーモンの実行時に作成される **/var/run/mariadb/mariadb.pid** という名前のプロセス ID (PID) に使用されます。また、**/var/lib/mysql/mysql.sock** などの関連ソケットファイルにも使用されます。これらのファイルが制限のあるサービスとして正常に動作するには、適切なラベル付けが必要になります。

20.3. ブール値

SELinux は、サービスの実行に必要な最小限レベルのアクセスに基づいています。サービスの実行手段は複数あるため、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、 ユーザーがローカルの MariaDB 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** メーラーがデータベースサーバーへの接続開始をできるようになります。

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開始をできるようになります。

httpd_can_network_connect_db

このブール値を有効にすると、データベースサーバーとの通信に web サーバーが必要になります。

注記

SELinux ポリシーは継続的に開発されているため、上記のリストでは常にこのサービスに関連するブール値がすべて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、以下のコマンドを実行します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の記述を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

sepolicy ユーティリティーを提供する *policycoreutils-devel* パッケージが追加で必要になることに留意してください。

20.4. 設定例

20.4.1. MariaDB のデータベース格納場所を変更する

Red Hat Enterprise Linux を使用する場合、MariaDB のデフォルトのデータベース格納場所は **/var/lib/mysql/** になります。 SELinux はこの場所にこのデータベースがデフォルトで配置されることを予期しているので、この領域にはすでに **mysqld_db_t** タイプを使った適切なラベル付けが行なわれています。

データベースを格納する場所は、個別の環境要件や設定に応じて変更することもできますが、適切にラベル付けを行い、SELinux が変更後の新しい場所を認識することが重要となります。以下の例では、MariaDB データベースの格納場所を変更する方法、また この新しい格納場所にラベルをつけて SELinux がコンテンツに基づいて保護メカニズムを適用できるようにする方法を説明します。

以下に示す例は、SELinux が MariaDB に対して与える影響を示す一例に過ぎません。MariaDB に関する総合的な説明は本ガイドの対象外となります。詳細については、公式の MariaDB documentation を参照し

てください。ここでは、*mariadb-server* パッケージと *setroubleshoot-server* パッケージがインストールされていること、**auditd** サービスが実行されていること、有効なデータベースがデフォルトの場所である **/var/lib/mysql/** に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格納場所の 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
```

データベースファイルの格納場所にデフォルトで付けられるコンテキスト要素の **mysqld_db_t** が表示されています。この例で使用する新しいデータベース格納場所が期待通り正常に動作するよう、このコンテキストをその新しい場所に手作業で適用する必要があります。

2. 以下のコマンドを実行し、**mysqld** の 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. **mysqld** デーモンを停止します。

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

4. データベース格納場所となるディレクトリーを新規作成します。この例では **/mysql/** を使用しています。

```
~]# mkdir -p /mysql
```

5. 古い場所にあるデータベースファイルを新しい場所にコピーします。

```
~]# cp -R /var/lib/mysql/* /mysql/
```

6. この場所の所有権を変更して、mysql ユーザーおよび mysql グループによるアクセスを許可します。これは従来の Unix パーミッションを設定するもので、SELinux はこれを順守します。

```
~]# chown -R mysql:mysql /mysql
```

7. 以下のコマンドを実行して、新規ディレクトリーの初期のコンテキストを確認します。

```
~]# ls -lZ /mysql
drwxr-xr-x. mysql mysql unconfined_u:object_r:usr_t:s0   mysql
```

新規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格納場所として SELinux に適したものではありません。コンテキストを変更すると、MariaDB がこの場所で正しく動作できるようになります。

8. MariaDB のメインとなる設定ファイル **/etc/my.cnf** をテキストエディターで開き、新しい格納場所を参照するよう **datadir** オプションを編集します。この例の場合、**/mysql** の値を入力します。

```
[mysqld]
datadir=/mysql
```

このファイルを保存してから終了します。

9. **mysqld** を起動します。サービスは起動に失敗し、拒否メッセージが**/var/log/messages** ファイルにログ記録されるはずです。

```
~]# systemctl start mariadb.service
Job for mariadb.service failed. See 'systemctl status
postgresql.service' and 'journalctl -xn' for details.
```

ただし、**audit** デーモンが **setroubleshoot** サービスとともに実行されている場合は、拒否メッセージは **/var/log/audit/audit.log** にログ記録されます。

```
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、**/mysql/** に MariaDB のデータファイル用として適切なラベルが付けられていないためです。SELinux は、MariaDB が **usr_t** タイプのラベルが付いたコンテンツにアクセスすることを禁止しています。この問題を解決するには、以下の手順にしたがいます。

10. 以下のコマンドを実行し、**/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ていないことに注意してください。インストールされていない場合は、*policycoreutils-python* パッケージをインストールします。

```
~]# 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. **restorecon** ユーティリティーを使ってこのコンテキストマッピングを稼働中のシステムに適用します。

```
~]# restorecon -R -v /mysql
```

13. これで **/mysql/** の場所に MariaDB 用の適切なコンテキストがラベル付けされたので、**mysqld** を起動できます。

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

14. **/mysql/** の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. データ格納場所が変更され、ラベルが適切に付けられたため、**mysqld** デーモンが正常に起動するようになりました。この時点で、実行中の全サービスが正常に動作しているかテストしてください。

[21] 詳細情報は、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、オブジェクト関係データベース管理システム (DBMS) です。 [22]

Red Hat Enterprise Linux では、PostgreSQL は *postgresql-server* パッケージで提供されます。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確認してください。

```
~]# rpm -q postgresql-server
```

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 postgresql-server
```

21.1. PostgreSQL と SELinux

PostgreSQL を有効にすると、デフォルトで制限のあるサービスとして実行されます。制限のあるプロセスはそれ自体のドメイン内で実行され、他の制限のあるプロセスとは分離されます。制限のあるプロセスが攻撃を受けると、SELinux ポリシー設定に応じて、攻撃側がリソースにアクセスして加えることができる被害は制限されます。以下に、PostgreSQL 自体のドメイン内で実行している PostgreSQL プロセスの例を示します。ここでは *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. **getenforce** コマンドを実行して、SELinux が enforcing モードで実行していることを確認します。

   ```
   ~]$ getenforce
   Enforcing
   ```

   SELinux が enforcing モードで実行している場合は、**Enforcing** が返されます。

2. root ユーザーで以下のコマンドを実行し、**postgresql** を起動します。

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

   サービスが稼働していることを確認します。出力は以下のようになり、タイムスタンプのみが異なります。

   ```
   ~]# 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. 以下のコマンドを実行して、**postgresql** プロセスを表示します。

   ```
   ~]$ 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 プロセスに関連する SELinux コンテキストは **system_u:system_r:postgresql_t:s0** です。このコンテキストの最後から 2 番目の部分、**postgresql_t** がタイプになります。タイプは、プロセスのドメインやファイルのタイプを定義します。この例の場合、**postgresql** プロセスは **postgresql_t** ドメイン内で実行しています。

21.2. タイプ

Type Enforcement が SELinux のターゲットポリシーで使用されるメインのパーミッション制御になります。全ファイルおよびプロセスにタイプのラベルが付けられます。タイプはプロセスのドメインを定義し、ファイルのタイプを定義します。SELinux ポリシールールは、ドメインがタイプにアクセスする場合でも、ドメインが別のドメインにアクセスする場合でも、タイプ同士がアクセスする方法を定義します。アクセスを許可する特定の SELinux ポリシールールが存在する場合にのみ、アクセスは許可されます。

postgresql で使用されるタイプを以下に示します。タイプに応じて柔軟なアクセス設定ができます。以下のリストでは、潜在的な場所に合致させるために正規表現をいくつか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。このタイプでラベル付けされた場所は 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/** ディレクトリー内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。このタイプでラベル付けされた場所は PostgreSQL のバイナリに使用されます。

- **/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/** ディレクトリー内の実行可能な PostgreSQL 関連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。このタイプでラベル付けされ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/** ディレクトリー内のプロセス 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限レベルのアクセスに基づいています。サービスの実行手段は複数あるため、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

このブール値を有効にすると、どのユーザードメイン (PostgreSQL の定義) もデータサーバーへ接続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、上記のリストでは常にこのサービスに関連するブール値がすべて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、以下のコマンドを実行します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の記述を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

sepolicy ユーティリティーを提供する *policycoreutils-devel* パッケージが追加で必要になることに留意してください。

21.4. 設定例

21.4.1. PostgreSQL のデータベース格納場所を変更する

Red Hat Enterprise Linux を使用する場合、 PostgreSQL のデフォルトのデータベース格納場所は **/var/lib/pgsql/data/** になります。SELinux はこの場所にこのデータベースがデフォルトで配置されることを予期しているので、この領域にはすでに **postgresql_db_t** タイプを使った適切なラベル付けが行なわれています。

データベースを格納する場所は、個別の環境要件や設定に応じて変更することもできますが、適切にラベル付けを行い、SELinux が変更後の新しい場所を認識することが重要となります。以下の例では、PostgreSQL データベースの格納場所を変更する方法、また この新しい格納場所にラベルをつけて SELinux がコンテンツに基づいて保護メカニズムを適用できるようにする方法を説明します。

以下に示す例は、PostgreSQL に対してどのように SELinux が影響を与えることができるのかを示す一例に過ぎません。PostgreSQL に関する総合的な説明は本ガイドの対象外となります。詳細については、公式の PostgreSQL ドキュメント を参照してください。ここでは、*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格納場所の 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
```

   データベースファイルの格納場所にデフォルトで付けられるコンテキスト要素である **postgresql_db_t** が表示されています。この例で使用する新しいデータベース格納場所が期待通り正常に動作するよう、このコンテキストをその新しい場所に手作業で適用する必要があります。

2. データベース格納場所となるディレクトリーを新規作成します。この例では **/opt/postgresql/data/** を使用します。別の場所を使用する場合は、以下のコマンドでその場所に置き換えます。

```
~]# mkdir -p /opt/postgresql/data
```

3. 新規作成したディレクトリーを表示します。このディレクトリーの初期コンテキストは **usr_t** になっている点に注意してください。このコンテキストは、SELinux が PostgreSQL に保護メカニズムを提供するには不十分です。コンテキストを変更すると、新規作成のディレクトリーが新しい領域で適切に動作することが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 postgres グループによるアクセスを許可するため所有権を変更します。これは従来の Unix パーミッションを設定するのもので、SELinux はこれを順守します。

```
~]# chown -R postgres:postgres /opt/postgresql
```

5. テキストエディターで PostgreSQL の初期設定ファイル **/etc/rc.d/init.d/postgresql** を開き、新しい場所をポイントするよう **PGDATA** と **PGLOG** 変数を変更します。

```
~]# vi /etc/rc.d/init.d/postgresql
PGDATA=/opt/postgresql/data
PGLOG=/opt/postgresql/data/pgstartup.log
```

   ファイルを保存して、テキストエディターを終了します。

6. 新しい場所にある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したことで、この時点では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.
```

サービスが起動しない原因は SELinux にあります。新しい場所に適切なラベル付けが行なわれていないためです。以下の手順で、新しい場所 (**/opt/postgresql/**) にラベルを付け、postgresql サービスを正常に起動させます。

8. **semanage** ユーティリティーを使用して、**/opt/postgresql/** およびその配下にあるすべてのディレクトリーと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. **restorecon** ユーティリティーを使ってこのコンテキストマッピングを稼働中のシステムに適用します。

```
~]# restorecon -R -v /opt/postgresql
```

11. これで **/opt/postgresql/** の場所に PostgreSQL 用の正しいコンテキストがラベル付けされたので、**postgresql** サービスが正常に起動するようになります。

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

12. **/opt/postgresql/** のコンテキストが正しくなっていることを確認します。

```
~]$ ls -lZ /opt
drwxr-xr-x. root root system_u:object_r:postgresql_db_t:s0
postgresql
```

13. **ps** コマンドを使って、**postgresql** プロセスで新しい場所が表示されるか確認します。

```
~]# 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. データ格納場所が変更され、ラベル付けが適切に行なわれたため、**postgresql** が正常に起動するようになりました。この時点で、実行中の全サービスが正常に動作しているかテストしてください。

[22] 詳細情報は、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ファイル転送を迅速に実行し、システム間のデータ同期に使用されます。 [23]

Red Hat Enterprise Linux では、*rsync* パッケージが rsync を提供します。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ことを確認します。

```
~]$ rpm -q rsync
package rsync 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 rsync
```

22.1. rsync と 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を付与する必要があります。ポリシーは、これらのファイルに対してデーモンが持つアクセスを管理します。**rsync** デーモンを使ってファイルを共有する場合、ファイルやディレクトリーに **public_content_t** タイプのラベルを付ける必要があります。他の多くのサービスと同様に、SELinux が **rsync** に対して保護メカニズムを実行するには、適切なラベリングが必要になります。 [24]

22.2. タイプ

Type Enforcement が SELinux のターゲットポリシーで使用されるメインのパーミッション制御になります。全ファイルおよびプロセスにタイプのラベルが付けられます。ファイルの場合はタイプ、プロセスの場合はドメインを定義します。SELinux ポリシールールは、ドメインがタイプにアクセスする場合でも、ドメインが別のドメインにアクセスする場合でも、タイプ同士がアクセスする方法を定義します。アクセスを許可する特定の SELinux ポリシールールが存在する場合にのみ、アクセスは許可されます。

rsync で使用されるタイプを以下に示します。タイプに応じて柔軟なアクセス設定ができます。

public_content_t

この汎用のタイプは、**rsync** を使用して共有するファイルの場所 (および実際のファイル) に使用します。**rsync** を使って共有するファイルの格納用に特別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およびそのコンテンツには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** システムバイナリに使用されます。

rsync_log_t

このタイプは、デフォルトで **/var/log/rsync.log** にある **rsync** ログファイルに使用されます。rsync がログを記録するファイルの場所を変更する場合は、ランタイム時に **rsync** コマンドに **--log-file=FILE** オプションを使用します。

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** サーバーで接続関連の制限を管理する際に使用されます。

rsync_data_t

このタイプは、ファイルやディレクトリーを rsync ドメインとして使用し、他のサービスのアクセス範囲とは分離させたい場合に使用します。また、**public_content_t** が汎用の SELinux コンテキストになり、ファイルやディレクトリーが複数のサービスと対話する際にこれを使用できます (例：rsync ドメインとしての FTP ディレクトリーおよび NFS ディレクトリー)。

rsync_etc_t

このタイプは、**/etc/** ディレクトリー内にある 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限レベルのアクセスに基づいています。サービスの実行手段は複数あるため、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

この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** タイプのファイル、リンク、ディレクトリーなどを管理できるようになります。多くの場合、これらはパブリックファイル転送サービスに使用されるパブリックファイルになります。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

このブール値を有効にすると、**rsync_port_t** で定義されるポートに **rsync** が接続を開始できるようになり、また **rsync_data_t** タイプのファイル、リンク、ディレクトリーの管理もできるようになります。SELinux が **rsync** を管理できるようにするには、このデーモンは**rsync_t** ドメイン内にある必要がある点に注意してください。本章では、**rsync_t** ドメインで実行している **rsync** の設定例を示します。

rsync_export_all_ro

このブール値を有効にすると、**rsync_t** ドメイン内の **rsync** が 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。クライアントに付与するアクセス権は読み取り専用になります。

注記

SELinux ポリシーは継続的に開発されているため、上記のリストでは常にこのサービスに関連するブール値がすべて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、以下のコマンドを実行します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の記述を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

sepolicy ユーティリティーを提供する *policycoreutils-devel* パッケージが追加で必要になることに留意してください。

22.4. 設定例

22.4.1. デーモンとして rsync を使用する

Red Hat Enterprise Linux を使用する場合、rsync をデーモンとして使用することで、複数のクライアントがセントラルサーバーとしてこのデーモンと直接通信して、一元的にファイルを格納し、継続的に同期することができます。以下では、rsync を適切なドメイン内のネットワークソケットでデーモンとして実行した場合、SELinux が期待する、事前定義された TCP ポート (SELinux ポリシー内) 上でのこのデーモンの実行を見ていきます。次に、非標準のポートでの **rsync** デーモンによる正常な実行を許可するため SELinux を編集する方法について説明していきます。

SELinux ポリシーとローカルのデーモンおよびプロセスに対するその制御を示すために、この例は単一のシステム上で行います。以下に示す例は、rsync に対してどのように SELinux が影響を与えることができるのかを示す一例に過ぎません。rsync に関する総合的な説明は本ガイドの対象外となります。詳細については、公式の rsync ドキュメント を参照してください。ここでは、*rsync* パッケージ、*setroubleshoot-server* パッケージ、*audit* パッケージがインストールされていること、SELinux のターゲットポリシーを使用していること、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 モードで実行していることを確認します。

```
~]$ getenforce
Enforcing
```

   SELinux が enforcing モードで実行している場合は、**Enforcing** が返されます。

2. **which** コマンドを実行し、rsync バイナリがシステムパス内にあるか確認します。

```
~]$ which rsync
/usr/bin/rsync
```

3. rsync をデーモンとして実行する場合、**/etc/rsyncd.conf** という名前を付けた設定ファイルを使用する必要があります。ここで使用している設定ファイルは非常に簡潔なファイルになっているため、利用できるオプションがすべて表示されているわけではありません。**rsync** デーモンの事例として必要なものを表示しています。

```
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. これで、rsync がデーモンモードで動作する簡単な設定ファイルができたので、以下のコマンドでこれを起動することができます。

```
~]# 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、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
```

rsync が **rsync_t** ドメイン内で実行するようになったため、SELinux はその保護メカニズムを **rsync** デーモンに適用できます。

```
~]$ ps -eZ | grep rsync
system_u:system_r:rsync_t:s0     3220 ?        00:00:00 rsync
```

上記の例では、**rsyncd** を **rsync_t** ドメイン内で実行する方法について説明しました。次の例では、このデーモンをデフォルト以外のポートで適切に実行する方法について見ていきます。ここでは TCP ポート 10000 を使用します。

手順22.2 デフォルト以外のポートで rsync デーモンを実行する

1. **/etc/rsyncd.conf** ファイルを変更して、**port = 10000** の行をファイルの冒頭にあるグローバル設定エリア内に追加します (つまり、file エリアが定義される前)。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. この新規設定で **rsync** デーモンを起動すると、SELinux は以下のような拒否メッセージをログ記録します。

```
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 TCP ポート 10000 を **rsync_port_t** の SELinux ポリシーに追加します。

```
~]# semanage port -a -t rsync_port_t -p tcp 10000
```

4. TCP ポート 10000 が **rsync_port_t** の SELinux ポリシーに追加されたので、**rsyncd** がこのポートで正常に起動し、動作するようになります。

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

```
~]# netstat -lnp | grep 10000
tcp        0      0 0.0.0.0:10000   0.0.0.0:*       LISTEN
9910/rsync
```

SELinux のポリシーが修正されたため、**rsyncd** による TCP ポート 10000 での動作が許可されるようになりました。

---

[23] 詳細情報は、Rsync プロジェクトページを参照してください。

[24] rsync および SELinux の詳細情報は、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転送エージェント (MTA) で、LDAP や SMTP AUTH (SASL)、TLS といったプロトコルをサポートします。 [25]

Red Hat Enterprise Linux では、*postfix* パッケージが Postfix を提供します。*postfix* 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、以下のコマンドを実行します。

```
~]$ rpm -q postfix
package postfix 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 postfix
```

23.1. Postfix と SELinux

Postfix を有効にすると、デフォルトで制限のあるサービスとして実行されます。制限のあるプロセスはそれ自体のドメイン内で実行され、他の制限のあるプロセスとは分離されます。制限のあるプロセスが攻撃を受けると、SELinux ポリシー設定に応じて、攻撃側がリソースにアクセスして加えることができる被害は限定されます。以下に、Postfix 自体のドメイン内で実行している Postfix プロセスの例を示します。 ここでは *postfix* パッケージがインストールされていること、また Postfix サービスが起動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 モードで実行していることを確認します。

   ```
   ~]$ getenforce
   Enforcing
   ```

   SELinux が enforcing モードで実行している場合は、**Enforcing** が返されます。

2. root ユーザーで以下のコマンドを実行し、**postfix** を起動します。

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

   サービスが稼働していることを確認します。出力は以下のようになり、タイムスタンプのみが異なります。

   ```
   ~]# 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. 以下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
```

上記の出力では、Postfix **master** プロセスに関連する SELinux コンテキストは **system_u:system_r:postfix_master_t:s0** です。コンテキストの最後から 2 番目の部分、**postfix_master_t** がこのプロセスのタイプになります。タイプは、プロセスのドメインやファイルのタイプを定義します。この例の場合、**master** プロセスは **postfix_master_t** ドメイン内で実行しています。

23.2. タイプ

Type Enforcement が SELinux のターゲットポリシーで使用されるメインのパーミッション制御になります。全ファイルおよびプロセスにタイプのラベルが付けられます。タイプはプロセスのドメインを定義し、ファイルのタイプを定義します。SELinux ポリシールールは、ドメインがタイプにアクセスする場合でも、ドメインが別のドメインにアクセスする場合でも、タイプ同士がアクセスする方法を定義します。アクセスを許可する特定の SELinux ポリシールールが存在する場合にのみ、アクセスは許可されます。

Postfix で使用されるタイプを以下に示します。タイプに応じて柔軟なアクセス設定ができます。

postfix_etc_t

このタイプは、**/etc/postfix/** ディレクトリー内の設定ファイルの Postfix に使用されます。

postfix_data_t

このタイプは、**/var/lib/postfix/** ディレクトリー内にある Postfix データファイルに使用されます。

postfix_var_run_t

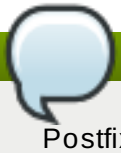
このタイプは、**/run/** ディレクトリー内の Postfix ファイルに使用されます。

postfix_initrc_exec_t

Postfix 実行可能ファイルには、**postfix_initrc_exec_t** タイプのラベルが付けられます。これらのファイルは実行されると、**postfix_initrc_t** ドメインに移行します。

postfix_spool_t

このタイプは、**/var/spool/** ディレクトリー内の Postfix ファイルに使用されます。



注記

Postfix 用のタイプとファイルの全一覧を表示するには、以下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限レベルのアクセスに基づいています。サービスの実行手段は複数あるため、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

このブール値を有効にすると、Postfix がシステム上のローカルメールスプールに書き込みできるようになります。ローカルスプールを使用する際、Postfix を正常に動作させるため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、上記のリストでは常にこのサービスに関連するブール値がすべて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、以下のコマンドを実行します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の記述を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

sepolicy ユーティリティーを提供する *policycoreutils-devel* パッケージが追加で必要になることに留意してください。

23.4. 設定例

23.4.1. SpamAssassin と Postfix

SpamAssasin はオープンソースのメールフィルターで、着信メールから未承諾 Email (スパムメッセージ) をフィルターにかける方法を提供します。 [26]

Red Hat Enterprise Linux では、*spamassassin* パッケージが SpamAssassin を提供します。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ことを確認します。

```
~]$ rpm -q spamassassin
package spamassassin 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 spamassassin
```

SpamAssassin は Postfix などのメーラーと連携してスパムフィルタリング機能を提供します。メールの効果的な遮断、分析、フィルタリングを実行するために、SpamAssassin はネットワークインターフェース上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、変更することもできます。以下では、SELinux がデフォルトで特定のポートでのみアクセスを許可することで SpamAssassin を補完している実践的な例を示します。次に、ポートを変更する方法およびデフォルト以外のポートで SpamAssassin を正常に動作させる方法について説明していきます。

以下に示す例は、シンプルな SpamAssassin 設定に対してどのように SELinux が影響を与えることがで

るのかを示す一例に過ぎません。SpamAssassin に関する総合的な説明は本ガイドの対象外となります。詳細については、公式の SpamAssassin ドキュメント を参照してください。ここでは、*spamassassin* がインストールされていること、使用しているポートでのアクセス許可がファイアウォールで設定されていること、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. root で **semanage** ユーティリティーを使用し、SELinux がデフォルトで **spamd** デーモンにリッスンすることを許可するポートを表示します。

```
~]# semanage port -l | grep spamd
spamd_port_t  tcp 783
```

上記の出力では、SpamAssassin が動作するポートとして TCP/783 が **spamd_port_t** で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、SpamAssassin が TCP/10000 で起動するよう変更します。

```
# Options to spamd
SPAMDOPTIONS="-d -p 10000 -c m5 -H"
```

上記の行では、SpamAssassin がポート 10000 で動作するよう指定しています。ここからは、このソケットを開くよう SELinux ポリシーを変更する方法を見ていきます。

3. SpamAssassin を起動すると、次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.
```

上記の出力は、このポートへのアクセスが SELinux によってブロックされたことを表しています。

4. 以下のような SELinux 拒否メッセージがログ記録されます。

```
SELinux is preventing the spamd (spamd_t) from binding to port
10000.
```

5. root で **semanage** を実行し、SpamAssassin がサンプルポート (TCP/10000) で動作できるように SELinux ポリシーを変更します。

```
~]# 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. SELinux ポリシーで **spamd** による TCP ポート 10000 へのアクセスが許可されたため、SpamAssassin がこのポートで正常に動作するようになりました。

[25] 詳細は、システム管理者のガイドの **Postfix** セクションを参照してください。

[26] 詳細は、システム管理者のガイドの **スパムフィルターPostfix** セクションを参照してください。

第24章 DHCP

dhcpd デーモンは、クライアントに第 3 層 TCP/IP を動的に提供し、詳細を設定するために Red Hat Enterprise Linux で使用されます。

dhcp パッケージが DHCP サーバーと **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確認します。

```
~]# rpm -q dhcp
package dhcp 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 dhcp
```

24.1. DHCP と SELinux

dhcpd を有効にすると、デフォルトで制限のあるサービスとして実行されます。制限のあるプロセスはそれ自体のドメイン内で実行され、他の制限のあるプロセスとは分離されます。制限のあるプロセスが攻撃を受けると、SELinux ポリシー設定に応じて、攻撃側がリソースにアクセスして加えることができる被害は限定されます。以下に、**dhcpd** 自体のドメイン内で実行している dhcpd と関連プロセスの例を示します。ここでは *dhcp* パッケージがインストールされていること、また**dhcpd** 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 モードで実行していることを確認します。

   ```
   ~]$ getenforce
   Enforcing
   ```

   SELinux が enforcing モードで実行している場合は、**Enforcing** が返されます。

2. root ユーザーで以下のコマンドを実行し、**dhcpd** を起動します。

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

   サービスが稼働していることを確認します。出力は以下のようになり、タイムスタンプのみが異なります。

   ```
   ~]# 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. 以下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 プロセスに関連する SELinux コンテキストは **system_u:system_r:dhcpd_t:s0** です。

24.2. タイプ

DHCP で使用されるタイプを以下に示します。

dhcp_etc_t

このタイプは主に、**/etc/** ディレクトリーにあるファイルに使用されます。これには設定ファイルが含まれます。

dhcpd_var_run_t

このタイプは、**/var/run/** ディレクトリー内の **dhcpd** の PID ファイルに使用されます

dhcpd_exec_t

このタイプは、DHCP 実行可能ファイルの **dhcpd_t** ドメインへの移行に使用されます。

dhcpd_initrc_exec_t

このタイプは、DHCP 実行可能ファイルの **dhcpd_initrc_t** ドメインへの移行に使用されます。

注記

dhcpd 用のファイルおよびタイプの全一覧を表示するには、以下のコマンドを実行します。

```
~]$ grep dhcp /etc/selinux/targeted/contexts/files/file_contexts
```

第25章 OpenShift

OpenShift は、開発者による Web アプリケーションの構築と導入を可能にするサービスとしてのプラットホーム (PaaS) です。OpenShift は、Java、Ruby、および PHP を含む幅広いプログラム言語およびフレームワークを提供します。また、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を提供し、これには Eclipse 統合や JBoss Developer Studio、Jenkins が含まれます。OpenShift はオープンソースのエコシステムを使用して、モバイルアプリケーションやデータベースサービス用のプラットホームを提供します。 [27]

Red Hat Enterprise Linux では、*rhc* パッケージが OpenShift クライアントツールを提供します。以下のコマンドを実行して、このパッケージがインストールされているか確認します。

```
~]$ rpm -q rhc
package rhc is not installed
```

rhc がインストールされていない場合、OpenShift のインストールプロセスの情報についてOpenShift Enterprise Client Tools Installation Guide および OpenShift Online Client Tools Installation Guide を参照してください。

25.1. OpenShift と SELinux

SELinux ポリシーにしたがってすべてのプロセスがラベル付けされているので、SELinux は OpenShift を使用するアプリケーションに対してよりすぐれたセキュリティを提供します。このため、SELinux は同一ノード上で実行中の異なるギア内で悪意のある攻撃から OpenShift を保護します。

SELinux と OpenShift についての詳細は、Dan Walsh のプレゼンテーション を参照してください。

25.2. タイプ

Type Enforcement が SELinux のターゲットポリシーで使用されるメインのパーミッション制御になります。全ファイルおよびプロセスにタイプのラベルが付けられます。タイプはプロセスのドメインを定義し、ファイルのタイプを定義します。SELinux ポリシールールは、ドメインがタイプにアクセスする場合でも、ドメインが別のドメインにアクセスする場合でも、タイプ同士がアクセスする方法を定義します。アクセスを許可する特定の SELinux ポリシールールが存在する場合にのみ、アクセスは許可されます。

OpenShift で使用されるタイプを以下に示します。タイプに応じて柔軟なアクセス設定ができます。

プロセスタイプ

`openshift_t`

OpenShift のプロセスは、**`openshift_t`** の SELinux タイプに関連付けられます。

`openshift_mail_t`

このタイプでは、OpenShift プロセスはメールプログラムの実行が可能で、ギアが送信可能なデータ量の管理に使用されます。このタイプは、OpenShift Online でのみ使用されることに注意してください。

実行可能ファイルにおけるタイプ

`openshift_cgroup_read_exec_t`

このタイプが付けられたファイルの場合、SELinux は実行可能ファイルが **openshift_cgroup_read_t** ドメインに移行することを許可します。

openshift_cron_exec_t

このタイプが付けられたファイルの場合、SELinux は実行可能ファイルが **openshift_cron_t** ドメインに移行することを許可します。

openshift_initrc_exec_t

このタイプが付けられたファイルの場合、SELinux は実行可能ファイルが **openshift_initrc_t** ドメインに移行することを許可します。

書き込み可能なタイプ

openshift_cgroup_read_tmp_t

このタイプでは、OpenShift コントロールグループ (cgroup) は **/tmp/** ディレクトリー内の一時ファイルの読み取りおよびアクセスが可能です。

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/** に保存することができます。

openshift_initrc_tmp_t

このタイプでは、OpenShift **initrc** の一時ファイルを **/tmp/** に保存することができます。

openshift_log_t

このタイプが付けられたファイルは通常、OpenShift ログデータとして扱われ、**/var/log/** ディレクトリーに保存されます。

openshift_rw_file_t

OpenShift はこのタイプのラベルが付いたファイルに読み取りおよび書き込みパーミッションがあります。

openshift_tmp_t

このタイプは、OpenShift 一時ファイルの **/tmp/** での保存に使われます。

openshift_tmpfs_t

このタイプでは、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

このタイプでは、OpenShift ファイルを **/var/lib/** ディレクトリーに保存できます。

openshift_var_run_t

このタイプでは、OpenShift ファイルを **/run/** または **/var/run/** ディレクトリーに保存できます。

25.3. ブール値

SELinux は、サービスの実行に必要な最小限レベルのアクセスに基づいています。サービスの実行手段は複数あるため、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

このブール値を有効にすると、OpenShift を NFS 共有に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、上記のリストでは常にこのサービスに関連するブール値がすべて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、以下のコマンドを実行します。

```
~]$ getsebool -a | grep service_name
```

特定のブール値の記述を表示するには、以下のコマンドを実行します。

```
~]$ sepolicy booleans -b boolean_name
```

sepolicy ユーティリティーを提供する *policycoreutils-devel* パッケージが追加で必要になることに留意してください。

25.4. 設定例

25.4.1. デフォルト OpenShift ディレクトリーの変更

デフォルトでは、OpenShift はデータを **/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** の SELinux タイプのラベルが付けられています。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、新たなディレクトリーに適切な SELinux コンテキストのラベルを付けます。

以下の手順では、OpenShift がデフォルトでデータを保存するディレクトリーを **/srv/openshift/** に変更する方法を示します。

手順25.1 データ保存用のデフォルト OpenShift ディレクトリーの変更

1. root で **/srv/** ディレクトリー内に新規の **/openshift/** ディレクトリーを作成します。この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。

```
~]# 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って、**/srv/openshift/** に適切な SELinux コンテキストをマッピングします。

```
~]# semanage fcontext -a -e /var/lib/openshift /srv/openshift
```

3. root で **restorecon** ユーティリティーを使用してラベル変更を適用します。

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

4. これで **/srv/openshift/** ディレクトリーに適切な **openshift_var_lib_t** タイプのラベルが付けられました。

```
~]$ ls -Zd /srv/openshift
drwxr-xr-x. root root unconfined_u:object_r:openshift_var_lib_t:s0
openshift/
```

[27] OpenShift についての詳細は、OpenShift Enterprise documentation および OpenShift Online documentation を参照してください。

第26章 ID 管理

ID 管理 (IdM) は、標準定義の共通ネットワークサービス (PAM、LDAP、Kerberos、DNS、NTP、および証明書サービスを含む) に統一された環境を提供します。IdM を使うことで、Red Hat Enterprise Linux システムはドメインコントローラーとして機能することができます。 [28]

Red Hat Enterprise Linux では、*ipa-server* パッケージが IdM サーバーを提供します。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確認します。

```
~]$ rpm -q ipa-server
package ipa-server 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 ユーザーで以下のコマンドを実行してインストールします。

```
~]# yum install ipa-server
```

26.1. ID 管理と SELinux

ID 管理では、ホストごとに IdM ユーザーを設定済みの SELinux ロールにマッピングすることで、IdM アクセス権の SELinux コンテキストの指定ができます。ユーザーのログインプロセス中に、System Security Services Daemon (**SSSD**) は特定の IdM ユーザー向けに定義されたアクセス権をクエリします。すると **pam_selinux** モジュールがカーネルに要求を送信し、**guest_u:guest_r:guest_t:s0** のような IdM アクセス権にしたがった適切な SELinux コンテキストでユーザープロセスを開始します。

ID 管理および SELinux についての詳細情報は、 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. アクティブディレクトリードメインへの信頼

以前のバージョンの Red Hat Enterprise Linux では、ID 管理はアクティブディレクトリー (AD) ドメインからのユーザーに、**WinSync** ユーティリティーを使って IdM ドメインで保存されているデータへのアクセスを許可していました。これを行うために、**WinSync** は AD サーバーからローカルサーバーにユーザーおよびグループのデータを複製し、このデータを同期させておく必要がありました。

Red Hat Enterprise Linux 7 では、**SSSD** デーモンと AD の連携が強化され、ユーザーが IdM と AD ドメイン間の信頼できる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データは、AD サーバーから直接読み込まれます。さらに、AD および IdM ドメイン間でシングルサインオン (SSO) 認証を可能にする Kerberos レルム間の信頼が提供されています。SSO が設定されていれば、AD ドメインからのユーザーはパスワードなしで、IdM ドメインに保存されている Kerberos 保護の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ていないので、使用する場合は *ipa-server-trust-ad* パッケージを新たにインストールします。

26.2. 設定例

26.2.1. SELinux ユーザーを IdM ユーザーにマッピングする

以下の手順では、新規 SELinux マッピングを作成し、このマッピングに新規 IdM ユーザーを追加する方法を示します。

手順26.1 ユーザーを SELinuxマッピングに追加する

1. 新規 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。ここでの **SELinux_mapping** は新規の SELinux マッピング名になり、**--selinuxuser** オプションでは特定の SELinux ユーザーを指定します。

```
~]$ ipa selinuxusermap-add SELinux_mapping --
selinuxuser=staff_u:s0-s0:c0.c1023
```

2. 以下のコマンドを実行して、ユーザー名 **tuser** の 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** という名前の新規ホストを SELinux マッピングに追加するには、以下のコマンドを実行します。

```
~]$ ipa selinuxusermap-add-host --hosts=ipaclient.example.com
SELinux_mapping
```

4. **tuser** ユーザーがホスト *ipaclient.example.com* にログインすると、**staff_u:s0-s0:c0.c1023** というラベルが付けられます。

```
[tuser@ipa-client]$ id -Z
staff_u:staff_r:staff_t:s0-s0:c0.c1023
```

[28] ID 管理の詳細情報は、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 参考文献

以下に、本ガイドの対象外となる SELinux 関連の詳細が記述されている参考文献を挙げます。SELinux は急速に開発されているため、記述の一部は Red Hat Enterprise Linux の特定リリースにのみ適用される可能性があることに注意してください。

書籍

SELinux by Example

Mayer、MacMillan、Caplan 著

2007年、Prentice Hall 出版

SELinux: NSA's Open Source Security Enhanced Linux

Bill McCarty 著

2004年、O'Reilly Media Inc. 出版

チュートリアルとヘルプ

Russell Coker 氏によるチュートリアルとトーク

http://www.coker.com.au/selinux/talks/ibmtu-2004/

Dan Walsh 氏のジャーナル

http://danwalsh.livejournal.com/

Red Hat ナレッジベース

https://access.redhat.com/site/

全般情報

NSA SELinux メイン web サイト

http://www.nsa.gov/research/selinux/index.shtml

NSA SELinux FAQ

http://www.nsa.gov/research/selinux/faqs.shtml

メーリングリスト

NSA SELinux メーリングリスト

http://www.nsa.gov/research/selinux/list.shtml

Fedora SELinux メーリングリスト

http://www.redhat.com/mailman/listinfo/fedora-selinux-list

http://www.redhat.com/mailman/listinfo/fedora-selinux-list

コミュニティ

SELinux プロジェクト Wiki

http://selinuxproject.org/page/Main_Page

SELinux コミュニティページ

http://selinux.sourceforge.net/

IRC

irc.freenode.net, #selinux

付録A 改訂履歴

改訂 0.2-6.2 **Tue Jul 21 2015** **Kenzo Moriguchi**
翻訳完成

改訂 0.2-6.1 **Tue Jul 21 2015** **Kenzo Moriguchi**
翻訳ファイルを XML ソースバージョン 0.2-6 と同期

改訂 0.2-6 **Wed Feb 18 2015** **Barbora Ančincová**
Red Hat Enterprise Linux 7.1 GA 向け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び替え順序の更新。

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**
スタイル変更のための再ビルド。

改訂 0.1-1 **Tue Jan 17 2013** **Tomáš Čapek**
Red Hat Enterprise Linux 7 用の本書ガイドの初期作成。