Red Hat Enterprise Linux 6

Security-Enhanced Linux

ユーザーガイド

エディッション 4

Last Updated: 2017-10-12

Red Hat Enterprise Linux 6 Security-Enhanced Linux

ユーザーガイド
エディッション 4

Red Hat Engineering Content Services

法律上の通知

概要

本ガイドは、ユーザーおよび管理者が Security-Enhanced Linux (SELinux) を管理、使用する際に手助けとなるものです。

目次

前書き

Red Hat Enterprise Linux 6 SELinux ユーザーガイドは、SELinux の経験がほとんどまたは全くない方のためのものです。システム管理の経験は必要ではありませんが、本ガイドのコンテンツはシステム管理のタスク向けに書かれています。本ガイドは、SELinux の基本的な概念と実際の適用方法を紹介します。本ガイドを通読すると、SELinux について中級の理解が得られるようになっています。

励ましやヘルプ、テストを提供していただいたすべての方に感謝いたします。以下の方々には特にお世話になりました。

- Dominick Grift、Stephen Smalley、Russell Coker の 3 氏の貢献、ヘルプ、忍耐には特に感謝いたします。

第1章 商標に関する情報

Linux は、米国およびその他の国における Linus Torvalds の登録商標です。

UNIX は、The Open Group の登録商標です。

Type Enforcement は、McAfee, Inc. の完全所有子会社である Secure Computing, LLC の商標で、米国およびその他の国で登録されています。McAfee と Secure Computing, LLC のいずれも、本ガイド外での本書著者による当該商標の使用もしくは言及には同意していません。

Apache は、The Apache Software Foundation の商標です。

MySQL は、米国およびその他の国における MySQL AB の商標もしくは登録商標です。

言及されている他の製品は、それらの会社の商標である場合があります。

第2章 はじめに

SELinux (Security-Enhanced Linux) は Linux カーネルに *MAC (Mandatory Access Control)*を追加するもので、標準の *Discretionary Access Controls (DAC: 任意アクセス制御)*がチェックされた後で許可された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 Linx 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、ディレクトリーやデバイスを含むファイル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、この望ましくないアクションに追加の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) に秘密度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 メカニズムのみへの依存は、強力なシステムセキュリティー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、データの秘密度および整合性といったその他のセキュリティー関連情報を無視しています。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、システム全体にわたるセキュリティーポリシーの強制を困難にしています。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。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粒度の荒い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、システムへのさらなるアクセスを取得するために利用されかねません。[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。**ls -l** コマンドを使って、ファイルパーミッションを表示させます。

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

この例では、パーミッションの最初の 3 ビットである **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 ビット **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 ビット **r--** は、その他全員の **file1** へのアクセスを制御し、これには全ユーザーとプロセスが含まれます。

SELinux は Linux カーネルに MAC を追加するもので、Red Hat Enterprise Linux ではデフォルトで有効になっています。一般目的の MAC アーキテクチャーは、システム内のすべてのプロセスとファイルに対して多種多様のセキュリティー関連情報を含むラベルを決定土台とする、管理者が設定したセキュリティポリシーを強制する能力を必要とします。適切に実装すると、システムは十分にシステム自身を保護でき、安全なアプリケーションの悪用や回避に対する保護により、アプリケーションのセキュリティーに必須のサポートを提供します。MAC は、信頼性の低いアプリケーションの安全な実行を許可する、強力なアプリケーションの分離を提供します。プロセス実行に関する権限を制限する能力により、アプリケーションおよびシステムサービスの脆弱性の悪用から発生する可能性のある損害の領域を制限します。MAC の使用により、限定的な権限を持つ正規ユーザーや不正なアプリケーションを知らずに実行してしまった、権限を持つユーザーから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、ファイルに対して使用されるセキュリティー関連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、**ls -Z** コマンドで表示させます。

```
~]$ 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**)、レベル (**s0**) を提供します。この情報は、アクセス制御の決定に使われます。DAC では、アクセスは Linux ユーザーおよびグループの 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いてください。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」 という用語で区別することにします。

2.1. 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されています。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行われます。

- 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での脆弱性が低減されます。一例を挙げると、プロセスがドメイン内で実行されるのでそれぞれ分離され、かつ SELinux ポリシールールがプロセスによるファイルおよび他のプロセスへのアクセス方法を定義することから、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 ポリシールールがユーザーのホームディレクトリーにあるファイルへのアクセスを許可するように追加・設定されていなければ、攻撃者はそのファイルを読み出すプロセスを使うことはできません。

- SELinux は、データの秘密性と整合性を強化し、プロセスを信頼できない入力から守るために使用できます。

しかし、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ーシステムなどに取って代わるもの

- オールインワンのセキュリティーソリューション

SELinux は既存のセキュリティーソリューションを強化する設計になっており、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ー対策を継続することが重要です。

2.2. SELINUX の使用例

以下では、SELinux がセキュリティーを強化する具体例を示しています。

- デフォルトのアクションは、拒否。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、制限された SELinux ユーザーが多く存在します。Linux ユーザーを制限された 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ールールとメカニズムの利点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）実行できません。また、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の実行を防止して、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しないようにします。

- プロセス分離の使用。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、MySQL が使用するデータベースなどの他のプロセスによって使われるファイルの読み取りや書込みはできません。

- SELinux は、設定ミスによる損害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。

- Red Hat Enterprise Linux 4 におけるデフォルトの SELinux 対象ポリシーによるエクスプロイトの制限については、Red Hat Magazine の記事、Risk report: Three years of Red Hat Enterprise Linux 4[4] を参照してください。

- 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[5] を参照してください。

- Red Hat Enterprise Linux 4 および 5 と出荷された SELinux が軽減した OpenPegasus のエクスプロイトについての情報は、James Morris 氏のブログ記事 SELinux mitigates remote root vulnerability in OpenPegasus を参照してください。

2.3. SELINUX アーキテクチャー

SELinux は、Linux カーネルに組み込まれた Linux セキュリティーモジュールです。SELinux は、ロード可能なポリシールールで駆動しています。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ー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。決定をキャッシュすることで、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 ポリシールールは影響しないことに留意してください。

2.4. 他のオペレーティングシステム上での SELINUX

他の Linux ディストリビューション上での SELinux 実行については、以下のリンクを参照してください。

- Fedora: http://fedoraproject.org/wiki/SELinux および Fedora SELinux FAQ.
- Hardened Gentoo: http://www.gentoo.org/proj/en/hardened/selinux/selinux-handbook.xml.
- Debian: http://wiki.debian.org/SELinux.
- Ubuntu: https://wiki.ubuntu.com/SELinux および https://help.ubuntu.com/community/SELinux.

---

[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", 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、その後公開されています。詳細および初回リリース時の文書に関しては、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] Cox, Mark. "Risk report: Three years of Red Hat Enterprise Linux 4". 2008 年 2 月 26 日発行。Accessed 27 August 2009: http://magazine.redhat.com/2008/02/26/risk-report-three-years-of-red-hat-enterprise-linux-4/.

[5] 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3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ー) 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、**ls -Z** コマンドを使用します。

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

SELinux コンテキストは、**SELinux user:role:type:level** 構文にしたがいます。フィー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。SELinux 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、Linux root ユーザーで **semanage login -l** 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

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

システムによっては、出力が多少違う場合があります。**Login Name** コラムは Linux ユーザーを一覧表示し、**SELinux User** コラムは Linux ユーザーがマッピングされている SELinux ユーザーを一覧表示します。プロセスでは、SELinux ユーザーはアクセス可能なロールとレベルを制限します。最後のコラムである **MLS/MCS Range** は、MLS (複数レベルセキュリティー) と MCS (複数カテゴリセキュリティー) が使用するレベルです。

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。入力可能なロールは、入力可能なドメインを決定し、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

スは、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、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**秘密度:カテゴリのセット** と書かれます。カテゴリがない場合は、**秘密度** と書かれます。

カテゴリセットは連続したもので、短縮が可能です。例えば、**c0.c3** は **c0,c1,c2,c3** と同じことです。**/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**)。**setrans.conf** はテキストエディターで編集せず、**semanage** コマンドを使って変更してください。詳細については、 semanage(8) man ページを参照してください。Red Hat Enterprise Linux では、対象ポリシーは MCSを強制し、MCS には **s0** の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** は秘密度 **s0** で、全カテゴリに対する権限が与えられています。

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 がデフォルトの SELinux ポリシーになるように設定します。Red Hat Enterprise Linux と出荷される MLS ポリシーは、評価済み設定の一部となってい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ー (MLS)」 を参照してください。

3.1. 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。

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** アプリケーションを実行します。**/usr/bin/passwd** 実行可能ファイルは、**passwd_exec_t** タイプで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 アプリケーションは、**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による**shadow_t** タイプのラベルが付けられたファイルの読み取り/書き込みを許可するとしています。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** アプリケーションを実行すると、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で拒否されるため、**passwd_t** ドメインで実行中のアプリケーション (ほかのものと共に) による**shadow_t** タイプのラベルが付けられたファイルへのアクセスを許可するルールが存在し、**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します。

この例は詳しいものではなく、ドメイン移行を説明する基本的な例として使われています。**passwd_t** ドメインで実行中のサブジェクトによる **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のアクセス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は、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けられ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ー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** といった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/書き込みできるのは、**etc_t** や **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/書き込みすることを防ぎます。

3.2. 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。例えば、

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** コマンドを実行します。新たなパスワードを入力しないでください。

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

```
unconfined_u:unconfined_r:passwd_t:s0-s0:c0.c1023 13212 pts/1
00:00:00 passwd
```

4. 最初のタブ/端末で **Ctrl+C** を押して、**passwd** アプリケーションをキャンセルします。

この例では、**passwd** アプリケーションの実行時 (**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。

ps -eZ コマンドを使って、実行中のプロセスの SELinux コンテキストを一覧表示します。以下は、省略された出力例で、システムによって異なります。

```
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ます。その後に、Type Enforcement が各ドメインを分離します。

3.3. ユーザーの SELINUX コンテキスト

id -Z 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。

```
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4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**initrc_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。ブール値設定は、後で説明されます。

4.1. 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** アプリケーションなど、Linux 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 ポリシー設定によっては、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。

以下の例では、Samba が使用するファイルなどの正確にラベル付けら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 が防いだ方法を示します。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。

1. **sestatus** コマンドを実行し、SELinux が有効になっていること、 enforcing モードで実行中であること、ターゲットポリシーが使われていること、を確認します。

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

SELinux が有効だと、**SELinux status: enabled** が返されます。SELinux が enforcing モードで実行中だと、**Current mode: enforcing** が返されます。SELinux ターゲットポリシーが使用されていると、**Policy from config file: targeted** が返されます。

2. Linux root ユーザーで、**touch /var/www/html/testfile** コマンドを実行してファイルを作成します。

3. **ls -Z /var/www/html/testfile** コマンドを実行し、SELinux コンテキストを表示します。

```
-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。[6] **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。

4. Linux root ユーザーで **service httpd start** コマンドを実行し、**httpd** プロセスを開始します。**httpd** が正常にスタートすると、出力は以下のようになります。

```
~]# service httpd start
Starting httpd:                                            [  OK
]
```

5. 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、**wget http://localhost/testfile** 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]
```

6. **chcon** コマンドはラベルを張り替えますが、このラベル変更は、ファイルシステムのラベルが張り替えられると残りません。ファイルシステムのラベル張り替え後にも変更が残るようにするには、**semanage** コマンドを使います。これは後で説明します。Linux root ユーザーで以下のコマンドを実行し、タイプを Samba が使用するタイプに変更します。

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

ls -Z /var/www/html/testfile コマンドを実行し、変更を表示します。

```
-rw-r--r--  root root unconfined_u:object_r:samba_share_t:s0
/var/www/html/testfile
```

7. 注記: 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します。Linux 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、**wget http://localhost/testfile** 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.
```

8. Linux root ユーザーで **rm -i /var/www/html/testfile** コマンドを実行し、**testfile** を削除します。

9. **httpd** の実行が不要ならば、Linux root ユーザーで **service httpd stop** コマンドを実行し、**httpd** を停止します。

```
~]# service httpd stop
Stopping httpd:                                            [  OK
]
```

この例では SELinux によって追加された新たなセキュリティーを説明しました。ステップ 7 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、ファイルは**httpd** プロセスが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す。

以下のようなエラーは、**/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は、**/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
```

4.2. 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 プログラムは制限のない **initrc_t** ドメインで、制限のないカーネルプロセスは**kernel_t** ドメインで、制限のない Linux ユーザーは **unconfined_t** ドメインで実行されます。制限のないプロセスでは、SELinux ポリシールールは適用されますが、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、もっぱら 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。

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。注記: 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、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。

1. **sestatus** コマンドを実行し、SELinux が有効になっていること、 enforcing モードで実行中であること、ターゲットポリシーが使われていること、を確認します。

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

SELinux が有効だと、**SELinux status: enabled** が返されます。SELinux が enforcing モードで実行中だと、**Current mode: enforcing** が返されます。SELinux ターゲットポリシーが使用されていると、**Policy from config file: targeted** が返されます。

2. Linux root ユーザーで、**touch /var/www/html/testfile** コマンドを実行してファイルを作成します。

3. **ls -Z /var/www/html/testfile** コマンドを実行し、SELinux コンテキストを表示します。

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

Red Hat Enterprise Linux ではデフォルトで、Linux ユーザーは制限なしで実行します。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。[7] **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。

4. **chcon** コマンドはラベルを張り替えますが、このラベル変更は、ファイルシステムのラベルが張り替えられると残りません。ファイルシステムのラベル張り替え後にも変更が残るようにするには、**semanage** コマンドを使います。これは後で説明します。Linux root ユーザーで以下のコマンドを実行し、タイプを Samba が使用するタイプに変更します。

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

ls -Z /var/www/html/testfile コマンドを実行し、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
```

5. **service httpd status** 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。

```
~]$ service httpd status
httpd is stopped
```

出力が異なる場合は、Linux root ユーザーで **service httpd stop** コマンドを実行し、**httpd** プロセスを停止します。

```
~]# service httpd stop
```

```
Stopping httpd:                                            [  OK
]
```

6. **httpd** プロセスを制限なしで実行するには、Linux root ユーザーで以下のコマンドを実行し、**/usr/sbin/httpd** のタイプを制限のある ドメインに移行しないタイプに変更します。

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

7. **ls -Z /usr/sbin/httpd** コマンドを実行し、**/usr/sbin/httpd** が **unconfined_exec_t** タイプでラベル付けられていることを確認します。

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

8. Linux root ユーザーで **service httpd start** コマンドを実行し、**httpd** プロセスを開始します。**httpd** が正常にスタートすると、出力は以下のようになります。

```
~]# service httpd start
Starting httpd:                                            [  OK
]
```

9. **ps -eZ | grep httpd** コマンドを実行し、**httpd** が **unconfined_t** ドメインで実行していることを表示します。

```
~]$ ps -eZ | grep httpd
unconfined_u:unconfined_r:unconfined_t:s0 7721 ?      00:00:00 httpd
unconfined_u:unconfined_r:unconfined_t:s0 7723 ?      00:00:00 httpd
unconfined_u:unconfined_r:unconfined_t:s0 7724 ?      00:00:00 httpd
unconfined_u:unconfined_r:unconfined_t:s0 7725 ?      00:00:00 httpd
unconfined_u:unconfined_r:unconfined_t:s0 7726 ?      00:00:00 httpd
unconfined_u:unconfined_r:unconfined_t:s0 7727 ?      00:00:00 httpd
unconfined_u:unconfined_r:unconfined_t:s0 7728 ?      00:00:00 httpd
unconfined_u:unconfined_r:unconfined_t:s0 7729 ?      00:00:00 httpd
unconfined_u:unconfined_r:unconfined_t:s0 7730 ?      00:00:00 httpd
```

10. 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、**wget http://localhost/testfile** 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けられ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_t** ドメインで実行しており、DAC ルールの使用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なら、**wget** コマンドは失敗していたでしょう。

11. **restorecon** コマンドは、デフォルトのファイル向け SELinux コンテキストを復元します。Linux root ユーザーで **restorecon -v /usr/sbin/httpd** コマンドを実行し、**/usr/sbin/httpd** のデフォルトの SELinux コンテキストを復元します。

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

ls -Z /usr/sbin/httpd コマンドを実行し、**/usr/sbin/httpd** が **httpd_exec_t** タイプでラベル付けられていることを確認します。

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

12. Linux root ユーザーで**service httpd restart** コマンドを実行し、**httpd** を再スタートさせます。再スタート後に **ps -eZ | grep httpd** コマンドを実行し、**httpd** が制限のある **httpd_t** ドメインで実行中であることを確認します。

```
~]# service httpd restart
Stopping httpd:                                            [  OK
]
Starting httpd:                                            [  OK
]
~]# ps -eZ | grep httpd
unconfined_u:system_r:httpd_t:s0    8880 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t:s0    8882 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t:s0    8883 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t:s0    8884 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t:s0    8885 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t:s0    8886 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t:s0    8887 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t:s0    8888 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t:s0    8889 ?        00:00:00 httpd
```

13. Linux root ユーザーで **rm -i /var/www/html/testfile** コマンドを実行し、**testfile** を削除します。

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

14. **httpd** の実行が不要ならば、Linux root ユーザーで **service httpd stop** コマンドを実行し、**httpd** を停止します。

```
~]# service httpd stop
Stopping httpd:                                            [  OK
]
```

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。

4.3. 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。Linux root ユーザーで **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

__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 6 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 6 のデフォルトにあるように、Linux 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。

1. Linux root ユーザーで **useradd newuser** 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。

2. Linux root ユーザーで **passwd newuser** コマンドを実行し、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 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。**id -Z** コマンドを実行し、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 Linux root ユーザーで **userdel -r newuser** コマンドを実行します。これで**newuser** ユーザーは、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。

SELinux ポリシーが**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ーの利点は、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 6 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4.1 SELinux ユーザーの能力

ユーザー	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_t	はい	**su** および **sudo**	はい	はい
staff_u	staff_t	はい	**sudo** のみ	はい	はい
user_u	user_t	はい	いいえ	はい	はい
guest_u	guest_t	いいえ	いいえ	いいえ	はい
xguest_u	xguest_t	はい	いいえ	いいえ	Firefox のみ

- **user_t**、**guest_t**、**xguest_t**、**git_shell_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) アプリケーションのみを実行できます (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 Linux root ユーザー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、書き込みアクセス権のあるディレクト

リーにあり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実行が妨げられます。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することを防いでいます。

- 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 ユーザーが持ち得る唯一のネットワークアクセスは、**Firefox** による Web ページへの接続です。

---

[6] MLS のような他のポリシーを使用している場合、**secadm_r** のような他のロールが使われることがあります。

[7] MLS のような他のポリシーを使用している場合、**secadm_r** のような他のロールが使われることがあります。

第5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。内容は以下のとおりです；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/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 ファイルシステムのマウント、ファイルおよびディレクトリーのコピー/アーカイブ時における SELinux コンテキストの保存方法。

5.1. 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。以下は、デフォルトでインストールされる SELinux パッケージの概要です。

- policycoreutils – SELinux の操作および管理用の**restorecon**、**secon**、**setfiles**、**semodule**、**load_policy**、**setsebool** といったユーティリティーを提供します。
- selinux-policy – SELinux Reference Policy を提供します。SELinux Reference Policy は完全な SELinux ポリシーで、SELinux ターゲットポリシーといった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 ページを参照してください。このパッケージは、**/usr/share/selinux/devel/policygentool** 開発ユーティリティーやポリシーファイルの例も提供します
- selinux-policy-targeted – SELinux ターゲットポリシーを提供します。
- libselinux – SELinux アプリケーション用の API を提供します。
- libselinux-utils – **avcstat**、**getenforce**、**getsebool**、**matchpathcon**、**selinuxconlist**、**selinuxdefcon**、**selinuxenabled**、**setenforce**、 **togglesebool** の各種ツールを提供します。
- libselinux-python – 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。

以下は、**yum install *<package-name>*** コマンドでインストールする必要のある主要オプションパッケージの概要です。

- selinux-policy-mls – MLS SELinux ポリシーを提供します。
- setroubleshoot-server – SELinux にアクセスを拒否された際に生成される拒否メッセージを **sealert** (本パッケージで提供) で表示される詳細な説明に変換します。
- setools-console – このパッケージは、ポリシー分析および問い合わせや監査ログの監視およびレポーティング、ファイルコンテキストの管理に使用する数多くのツールおよびライブラリーである Tresys Technology SETools distribution を提供します。[8] 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**、**seaudit**、**sediffx** の各種ツールを提供します。setools-console パッケージは、**seaudit-report**、**sechecker**、**sediff**、**seinfo**、**sesearch**、**findcon**、**replcon**、**indexcon** の各種コマンドラインツールを提供します。これらのツールに関する情報は、Tresys Technology SETools ページを参照してください。

- mcstrans – **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。このパッケージは、デフォルトではインストールされません。
- policycoreutils-python – 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 – SELinux 管理用のグラフィカルツールである **system-config-selinux** を提供します。

5.2. 使用するログファイル

Red Hat Enterprise Linux 6 では、dbus および audit 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**) でインストールする必要があります。

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
```

```
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 6 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 **sedispatch** と **seapplet** の 2 つの新プログラムが setroubleshoot を開始する方法として作動します。**sedispatch** は監査サブシステムの一部として稼働し、AVC 拒否が起こると **dbus** 経由でメッセージを送信します。このメッセージは、**setroubleshootd** がすでに稼働中の場合は即座にそこに送信され、まだ稼働していない場合は **setroubleshootd** を開始します。**seapplet** はシステムのツールバーで稼働するツールで、**setroubleshootd** で dbus メッセージを待機し、吹き出し通知でユーザーが拒否を確認できるようにしています。

デーモンの自動開始

auditd および **rsyslogd** 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、Linux root ユーザーで以下のコマンドを実行します。

```
~]# chkconfig --levels 2345 auditd on
~]# chkconfig --levels 2345 rsyslog on
```

service *service-name* status コマンドを使って、サービスが稼働しているかをチェックします。以下に例を挙げます。

```
~]# service auditd status
auditd (pid  1318) is running...
```

上記のサービスが稼働していない場合 (***service-name* is stopped**)、Linux root ユーザーで **service *service-name* start** コマンドを使い、サービスを開始します。以下に例を挙げます。

–

```
~]# service auditd start
Starting auditd:                                           [  OK  ]
```

5.3. 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。

5.4. 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。**getenforce** コマンドは、SELinux が有効な際に (SELinux ポリシールールが強制されている際に) **Enforcing** を返します。

```
~]$ getenforce
```

```
Enforcing
```

getenforce コマンドは、SELinux が有効ではあるものの SELinux ポリシールールが強制されておらず DAC ルールのみが使用されている場合に **Permissive** を返します。SELinux が無効だと、**getenforce** コマンドは、**Disabled** を返します。

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

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

SELinux が有効だと、**SELinux status: enabled** が返されます。SELinux が enforcing モードで実行中だと、**Current mode: enforcing** が返されます。SELinux ターゲットポリシーが使用されていると、**Policy from config file: targeted** が返されます。

5.4.1. 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** オプションは **/etc/selinux/config** で設定します。

```
# 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
```

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

```
~]$ getenforce
Disabled
```

SELinux を有効にするには、以下の手順にしたがいます。

1. **rpm -qa | grep selinux** と **rpm -q policycoreutils**、**rpm -qa | grep**

setroubleshoot のコマンドを実行して、SELinux パッケージのインストールを確認します。本ガイドでは、以下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。selinux-policy-targeted、selinux-policy、libselinux、libselinux-python、libselinux-utils、policycoreutils、policycoreutils-python、setroubleshoot、setroubleshoot-server、setroubleshoot-plugins。これらのパッケージがインストールされていなければ、Linux root ユーザーで **yum install *package-name*** コマンドを使ってインストールします。policycoreutils-gui、setroubleshoot、mcstrans の 3 パッケージはオプションです。

2. SELinux の有効化の前に、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** 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. Linux root ユーザーで **reboot** コマンドを実行して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。

```
*** 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。enforcing モードに変更する前に、Linux root ユーザーで **grep "SELinux is preventing" /var/log/messages** 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、このコマンドに返される出力はありません。起動時に SELinux がアクセスを拒否していた場合は、トラブルシューティング情報を 8 章*トラブルシューティング*で参照してください。

5. **/var/log/messages** に拒否メッセージがない場合は、**/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** コマンドが **Enforcing** を返すことを確認します。

```
~]$ getenforce
Enforcing
```

7. Linux root ユーザーで **semanage login -l** 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。

```
Login Name                SELinux User              MLS/MCS Range

__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、Linux root で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。

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。

5.4.2. SELinux の無効化

SELinux を無効にするには、**/etc/selinux/config** 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
```

5.5. 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。

setenforce コマンドを使って enforcing モードと permissive モードを切り替えます。**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** コマンドを実行します。

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。

5.6. 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 ポリシーの再ロードや再コンパイルをせずに、NFS ファイルシステ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。

5.6.1. ブール値の一覧表示

ブール値の一覧表示で、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ありません。

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

ftp_home_dir                   -> off   Allow ftp to read and write files
in the user home directories
xen_use_nfs                    -> off   Allow xen to manage nfs files
xguest_connect_network         -> on    Allow xguest to configure Network
Manager
```

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   Allow ftp to read and write files
in the user home directories
```

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
allow_console_login --> off
allow_cvs_read_shadow --> off
allow_daemons_dump_core --> on
```

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 allow_console_login
allow_console_login --> 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 allow_console_login allow_cvs_read_shadow
allow_daemons_dump_core
allow_console_login --> off
allow_cvs_read_shadow --> off
allow_daemons_dump_core --> on
```

5.6.2. ブール値の設定

setsebool *boolean-name x* コマンドは、ブール値をオンやオフにします。ここでの*boolean-name* はブール値名で、*x* を **on** にするとブール値をオンにし、**off** にするとブール値をオフにします。

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。

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なっています。

   ```
   ~]$ 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、Linux root ユーザーで **setsebool httpd_can_network_connect_db on**コマンドを実行します。

3. **getsebool httpd_can_network_connect_db**コマンドを使って、ブール値がオンになっていることを確認します。

   ```
   ~]$ 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. この変更はリブート後には維持されません。リブート後にも変更を維持するには、Linux root ユーザーで **setsebool -P *boolean-name* on** コマンドを実行します。

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

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。

5.6.3. NFS および CIFS 用のブール値

デフォルトでは、クライアント側の NFS マウントは、NFS ファイルシステ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は **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** タイプのラベル付けされたファイルを読み取れない場合もあります。これによって、これらのタイプのラベル付けされたファイルシステムがマウントされ、他のサービスによって読み取られたり、エクスポートされることを防ぐことができます。ブール値はオンやオフにして、**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。

setsebool と **semanage** コマンドは、Linux root ユーザーが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。リブート後には変更を維持したくない場合は、**-P** オプションを使わないでください。

Apache HTTP Server

NFS ファイルシステムへのアクセスを許可するには (**nfs_t** タイプのラベルが付けられたファイル)、以下を実行します。

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

Samba ファイルシステムへのアクセスを許可するには (**cifs_t** タイプのラベルが付けられたファイル)、以下を実行します。

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

Samba

NFS ファイルシステムをエクスポートするには、以下を実行します。

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

FTP (vsftpd)

NFS ファイルシステムへのアクセスを許可するには、以下を実行します。

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

Samba ファイルシステムへのアクセスを許可するには、以下を実行します。

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

他のサービス

他のサービス用の NFS 関連のブール値を一覧表示するには、以下を実行します。

```
~]# semanage boolean -l | grep nfs
```

他のサービス用の Samba 関連のブール値を一覧表示するには、以下を実行します。

```
~]# semanage boolean -l | grep cifs
```

注記

これらのブール値は、Red Hat Enterprise Linux 6 と出荷されたSELinux ポリシー内に存在します。他のバージョンの Red Hat Enterprise Linux や別のオペレーティングシステムと出荷されたポリシーには含まれない場合もあります。

5.7. 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がセキュリティー関連の情報を表示する方法でラベル付けされます。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**)、レベル (**s0**) を提示しています。この情報は、アクセス制限の決定に使われます。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。

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。

5.7.1. 一時的な変更: chcon

chcon コマンドは、ファイルの SELinux コンテキストを変更します。ただし、**chcon** コマンドによる変更は、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。

- 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** などのタイプで、*directory-name* はディレクトリー名になります。

ファイルまたはディレクトリーのタイプ変更

以下の例では、SELinux コンテキストの属性のうち、タイプのみを変更する方法を示しています。

1. 引数なしで **cd** コマンドを実行して、ホームディレクトリーに移動します。

2. **touch file1** コマンドの実行で新規ファイルを作成します。**ls -Z file1** コマンドで **file1** の SELinux コンテキストを表示します。

```
~]$ 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、3章*SELinux コンテキスト*を参照してください。

3. **chcon -t samba_share_t file1** コマンドを実行して、タイプを**samba_share_t** に変更します。**-t** オプションのみがタイプを変更します。**ls -Z file1** で変更を表示します。

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

4. **restorecon -v file1** コマンドを実行して、**file1** ファイルの SELinux コンテキストを復元します。**-v** オプションで変更を表示します。

```
~]$ 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** は、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 6 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を持つべきかをチェックします。

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。

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. Linux root ユーザーで **mkdir /web** コマンドを実行し、新規ディレクトリーを作成します。次に **touch /web/file{1,2,3}** コマンドで 3 つの空ファイル (**file1**、**file2**、**file3**) を作成します。**/web/** ディレクトリーおよびその中のファイルには、**default_t** タイプのラベルが付けられます。

```
~]# 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
```

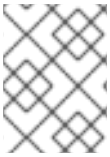
2. Linux root ユーザーで **chcon -R -t httpd_sys_content_t /web/** コマンドを実行し、**/web/** ディレクトリー (およびそのコンテンツ) のタイプを **httpd_sys_content_t** に変更します。

```
~]# 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. Linux root ユーザーで **restorecon -R -v /web/** コマンドを実行し、デフォルトの SELinux コンテキストを復元します。

```
~]# 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。

5.7.2. 永続的な変更: semanage fcontext

semanage fcontext コマンドは、ファイルの SELinux コンテキストを変更します。ターゲットポリシーを使用している場合、このコマンドによる変更が **file_contexts** にあるファイルになされると **/etc/selinux/targeted/contexts/files/file_contexts** ファイルに追加され、**/web/** ディレクトリー作成時など、新規ファイルやディレクトリーの変更の場合は **file_contexts.local** に追

加されます。ファイルシステムの再ラベリング時に使用される **setfiles** コマンドと、デフォルトの SELinux コンテキストを復元する **restorecon** コマンドは、これらのファイルを読み取ります。つまり、**semanage fcontext** コマンドによる変更は、ファイルシステムが再ラベル付けされても永続的なものとなります。SELinux ポリシーは、ユーザーが特定のファイルの 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

ファイルシステムが再ラベル付けされても SELinux コンテキストの変更が維持されるには、以下の手順を実行します。

1. **semanage fcontext -a *options file-name|directory-name*** コマンドを実行します。ファイルもしくはディレクトリーへのフルパスを使用します。
2. **restorecon -v *file-name|directory-name*** コマンドを実行し、コンテキストの変更を適用します。

ファイルのタイプの変更

以下の例では、SELinux コンテキストの属性のうち、ファイルのタイプのみを変更しています。

1. Linux root ユーザーで **touch /etc/file1** コマンドを実行し、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ファイルは **etc_t** タイプのラベルが付けられます。

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

2. Linux root ユーザーで **semanage fcontext -a -t samba_share_t /etc/file1** コマンドを実行し、**file1** 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。注記: このコマンドの実行は直接にはタイプを変更しません–**file1** は **etc_t** タイプのラベル付けがされたままです。

```
~]# 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. Linux root ユーザーで **restorecon -v /etc/file1** コマンドを実行し、タイプを変更します。**semanage** コマンドは **/etc/file1** のエントリーを **file.contexts.local** に追加するので、**restorecon** コマンドはタイプを **samba_share_t** に変更します。

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

4. Linux root ユーザーで **rm -i /etc/file1** コマンドを実行し、**file1** を削除します。

5. Linux root ユーザーで **semanage fcontext -d /etc/file1** コマンドを実行し、**/etc/file1** に追加されたコンテキストを削除します。

ディレクトリーのタイプ変更

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 Apache HTTP Server が使用するタイプに変更する方法を示します。

1. Linux root ユーザーで **mkdir /web** コマンドを実行し、新規ディレクトリーを作成します。このディレクトリーには **default_t** タイプのラベルが付けられます。

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

ls -d オプションは、**ls** にコンテンツではなくディレクトリーについての一覧情報を作成させ、**-Z** オプションは **ls** に SELinux コンテキスト **(**この例では **unconfined_u:object_r:default_t:s0)** を表示させます。

2. Linux root ユーザーで **semanage fcontext -a -t httpd_sys_content_t /web** コマンドを実行し、**/web/** 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。注記**:** このコマンドの実行は直接にはタイプを変更しません**–/web/** は **default_t** タイプのラベル付けがされたままです。

```
~]# 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
```

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    unconfined_u:object_r:httpd_sys_content_t:s0
```

3. Linux root ユーザーで **restorecon -v /web** コマンドを実行し、タイプを変更します。**semanage** コマンドは **/web** のエントリーを **file.contexts.local** に追加するので、**restorecon** コマンドはタイプを **httpd_sys_content_t** に変更します。

```
~]# restorecon -v /web
restorecon reset /web 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、親フォルダの SELinux タイプを引き継ぎます。この例では、**/web/** に追加された SELinux コンテキストを削除する前は、**/web/** ディレクトリーに作成されたファイルとディレクトリーは **httpd_sys_content_t** タイプのラベル付けがされています。

4. Linux root ユーザーで **semanage fcontext -d /web** コマンドを実行し、**/web/** に追加されたコンテキストを削除します。

5. Linux root ユーザーで **restorecon -v /web** コマンドを実行し、デフォルトの SELinux コンテキストを復元します。

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. Linux root ユーザーで **mkdir /web** コマンドを実行し、新規ディレクトリーを作成します。次に **touch /web/file{1,2,3}** コマンドで 3 つの空ファイル (**file1**、**file2**、**file3**) を作成します。**/web/** ディレクトリーおよびその中のファイルには、**default_t** タイプのラベルが付けられます。

```
~]# 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. Linux root ユーザーで **semanage fcontext -a -t httpd_sys_content_t "/web(/.*)?"** 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(/.*)?"** の正規表現は、**semanage** コマンドが変更を **/web/** ディレクトリーとその中のファイルに適用させるようにします。注記: このコマンドの実行は直接にはタイプを変更しません–**/web/** およびその中のファイルは **default_t** タイプのラベル付けがされたままです。

```
~]# 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. Linux root ユーザーで **restorecon -R -v /web** 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変更します。**-R** オプションは再帰的ということで、**/web/** ディレクトリー下の全ファイルおよびディレクトリーが **httpd_sys_content_t** タイプでラベル付けされることを意味します。**semanage** コマンドは **/web(/.*)?** のエントリーを **file.contexts.local** に追加したので、**restorecon** コマンドはタイプを **httpd_sys_content_t** に変更します。

```
~]# 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、親フォルダの SELinux タイプを引き継ぎます。この例では、**/web/** ディレクトリーに作成されたファイルとディレクトリーは **httpd_sys_content_t** タイプのラベルが付けられます。

4. Linux root ユーザーで **semanage fcontext -d "/web(/.*)?"** コマンドを実行し、**"/web(/.*)?"** に追加されたコンテキストを削除します。

5. Linux root ユーザーで **restorecon -R -v /web** コマンドを実行し、デフォルトの SELinux コンテキストを復元します。

追加されたコンテキストの削除

以下の例では、SELinux コンテキストを追加・削除する方法を示しています。

1. Linux root ユーザーで **semanage fcontext -a -t httpd_sys_content_t /test** コマンドを実行します。**/test/** ディレクトリーはなくても構いません。このコマンドは、以下のコンテキストを **/etc/selinux/targeted/contexts/files/file_contexts.local** に追加します。

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

2. コンテキストを削除するには、Linux root ユーザーで **semanage fcontext -d *file-name*|*directory-name*** 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。以下は **file_contexts.local** のコンテキストの例です。

```
/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が再ラベル付けされた後に **/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、Linux root ユーザーで **file_contexts.local** からコンテキストを削除します。

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

例えば **/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(/.*)?"
```

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フルパスを使用してください。

5.8. FILE_T および DEFAULT_T タイプ

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。このタイプを使用するのはこの場合に限るべきで、そうすることでディスク上でコンテキストのないファイルがポリシーで区別可能になり、通常、制限されたドメインにはアクセス不可能になります。SELinux 実行中のシステム上では、全ファイルに SELinux コンテキストがあり、**file_t** タイプはファイル-コンテキスト設定では決して使われないため[9] **file_t** は正しくラベル付けされたファイルシステムにはあってはなりません。

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に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、通常は制限されたドメインにはアクセス不可能になります。**/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** タイプのラベル付けがされます。このような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、このロケーションのファイル-コンテキスト設定を更新します。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。

5.9. ファイルシステムのマウント

デフォルトでは、拡張属性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 ファイルシステムなど、拡張属性をサポートしないファイルシステムのラベリングのサポートにも使用できます。**context** 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** オプション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リング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。

5.9.1. 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 Linux 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 ファイルシステ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リングされます。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経由で NFS ファイルシステムを共有することが妨げられる可能があります。以下の例では NFS ファイルシステ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、この状況のコンテキスト変更がディスクに書き込まれていないため、**context** option で指定されたコンテキストは、**context** オプションが次のマウントで使われ、同一のコンテキストが指定された場合にのみ保持されます。

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、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ー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** を実行すると、**Operation not supported** エラーが出ます。

5.9.2. 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」[10] 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。

5.9.3. NFS ファイルシステムのマウント

デフォルトでは、クライアント側の NFS マウントは、NFS ファイルシステ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。Apache HTTP Server や MySQL などポリシー設定やサービスなどに

よっては、**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。これによって、このタイプのラベル付けされたファイルシステムがマウントされ、他のサービスによる読み取りやエクスポートを防ぐことができます。

NFS ファイルシステムをマウントし、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 ファイルシステ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"
```

この状況のコンテキスト変更がディスクに書き込まれないため、**context** option で指定されたコンテキストは、**context** オプションが次のマウントで使われ、同一のコンテキストが指定された場合にのみ保持されます。

context オプションでのファイルシステムのマウントの代替方法として、ブール値をオンにして **nfs_t** タイプのラベルが付けられたファイルシステムへの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、「NFS および CIFS 用のブール値」 を参照してください。

5.9.4. 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)
```

各マウントが異なるコンテキストを持ってい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** オプションを使います。以下の例では、各マウントが異なるコンテキストを持ってい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 のアクセス許可

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、MySQL 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。

5.9.5. 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持続させるには、**/etc/fstab** 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
```

5.10. 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ビング時にコンテキストを維持する方法も説明します。

5.10.1. ファイルおよびディレクトリーのコピー

ファイルまたはディレクトリーのコピー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/** ディレクトリーのデフォルトのラベリング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リング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、コピー中にコンテキストの保持を妨げる可能性があります。

SELinux コンテキストを保持せずにコピーする

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します。

```
~]$ touch file1
~]$ ls -Z file1
-rw-rw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0 file1
~]$ ls -dZ /var/www/html/
drwxr-xr-x  root root system_u:object_r:httpd_sys_content_t:s0
/var/www/html/
~]# 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
```

この例では、**file1** がユーザーのホームディレクトリーに作成され、**user_home_t** タイプのラベル付けがされています。**/var/www/html/** ディレクトリーは **ls -dZ /var/www/html/** コマンドで表示されているように、**httpd_sys_content_t** タイプでラベル付けされています。**file1** を **/var/www/html/** にコピーすると、**ls -Z /var/www/html/file1** コマンドにあるように **httpd_sys_content_t** タイプを継承します。

SELinux コンテキストを保持してコピーする

コピー時に **cp --preserve=context** コマンドを使うと、コンテキストを保持します。

```
~]$ touch file1
~]$ ls -Z file1
-rw-rw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0 file1
~]$ ls -dZ /var/www/html/
drwxr-xr-x  root root system_u:object_r:httpd_sys_content_t:s0
/var/www/html/
~]# 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
```

この例では、**file1** がユーザーのホームディレクトリーに作成され、**user_home_t** タイプのラベル付けがされています。**/var/www/html/** ディレクトリーは **ls -dZ /var/www/html/** コマンドで表示されているように、**httpd_sys_content_t** タイプでラベル付けされています。**--preserve=context** オプションを使うと、コピー作業中に SELinux コンテキストを保持します。**ls -Z /var/www/html/file1** コマンドにあるように、ファイルが **/var/www/html/** にコピーされた際に、**file1** **user_home_t** タイプは保持されています。

コピーおよびコンテキストの変更

cp -Z コマンドを使ってコピー先のコンテキストを変更します。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。

```
~]$ touch file1
~]$ cp -Z system_u:object_r:samba_share_t:s0 file1 file2
~]$ 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
~]$ rm file1 file2
```

この例では、コンテキストは **-Z** オプションで定義されています。**-Z** オプションなしだと、**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。

既存ファイル上へのファイルのコピー

既存ファイル上にファイルをコピーすると、(オプションを使って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保持されます。例えば、

```
~]# touch /etc/file1
~]# ls -Z /etc/file1
-rw-r--r--  root root unconfined_u:object_r:etc_t:s0   /etc/file1
~]# touch /tmp/file2
~]# ls -Z /tmp/file2
-rw-r--r--  root root unconfined_u:object_r:user_tmp_t:s0 /tmp/file2
~]# cp /tmp/file2 /etc/file1
~]# ls -Z /etc/file1
-rw-r--r--  root root unconfined_u:object_r:etc_t:s0   /etc/file1
```

この例では、**etc_t** タイプのラベル付けがされた **/etc/file1** と、**user_tmp_t** タイプのラベル付けがされた **/tmp/file2** の 2 つのファイルが作成されます。**cp /tmp/file2 /etc/file1** コマンドは、**file1** を **file2** で上書きします。コピー後に **ls -Z /etc/file1** コマンドは、**file1** が **/etc/file1** を置き換えた **/tmp/file2** からの **user_tmp_t** タイプではなく、**etc_t** タイプでラベル付けされていることを示しています。

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 コンテキストが誤っていると、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。

5.10.2. ファイルおよびディレクトリーの移動

ファイルおよびディレクトリーは、移動されると現行の 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誤ったものとなります。以下の例では、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/** 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。

1. 変更の引数なしで **cd** コマンドを実行し、ホームディレクトリーへ移動します。そして**touch file1** コマンドでファイルを作成します。このファイルは、**user_home_t** タイプでラベル付けされます。

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

2. **ls -dZ /var/www/html/** コマンドを実行し、**/var/www/html/** ディレクトリーの SELinux コンテキストを表示します。

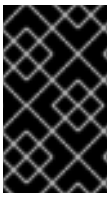
```
~]$ ls -dZ /var/www/html/
drwxr-xr-x  root root system_u:object_r:httpd_sys_content_t:s0
/var/www/html/
```

デフォルトでは、**/var/www/html/** ディレクトリーは **httpd_sys_content_t** タイプでラベル付けされます。**/var/www/html/** ディレクトリー下の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ので、それらのファイルおよびディレクトリーはこのタイプでラベル付けされます。

3. Linux root ユーザーで **mv file1 /var/www/html/** コマンドを実行し、**file1** を **/var/www/html/** ディレクトリーに移動します。このファイルは移動されたので、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 Firefox もしくはテキストベースの 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。

5.10.3. デフォルト SELinux コンテキストのチェック

matchpathcon コマンドでファイルやディレクトリーに正しい SELinux コンテキストがあるかどうかをチェックします。matchpathcon(8) man ページでが「**matchpathcon** が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ーコンテキストを出力します。」となっています。[11]。以下の例では、**matchpathcon** コマンドを使って **/var/www/html/** ディレクトリーが正しくラベル付けされているかを確認します。

1. Linux root ユーザーで **touch /var/www/html/file{1,2,3}** コマンドを実行し、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/** ディレクトリーからの **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. Linux root ユーザーで **chcon -t samba_share_t /var/www/html/file1** コマンドを実行し、**file1** タイプを **samba_share_t** に変更します。注記: 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ません。

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べます。**matchpathcon -V /var/www/html/*** コマンドを実行し、**/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、Linux root ユーザーで **restorecon -v /var/www/html/file1** コマンドを実行します。

```
~]# 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
```

5.10.4. tar を使ったファイルのアーカイビング

tar 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。**tar --selinux** を使ってコンテキストを維持するアーカイブを作成します。tar アーカイブが拡張属性のないファイルを含む場合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** をアーカイブで実行します。

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

注記: ディレクトリーによっては、Linux root ユーザーで **restorecon** コマンド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。

1. Linux root ユーザーで **touch /var/www/html/file{1,2,3}** コマンドを実行し、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/** ディレクトリーからの **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. **cd /var/www/html/** コマンドで **/var/www/html/** ディレクトリーに移動します。その後、Linux root ユーザーで **tar --selinux -cf test.tar file{1,2,3}** 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。

3. Linux root ユーザーで **mkdir /test** コマンドを実行し、新規ディレクトリーを作成します。そして **chmod 777 /test/** コマンドを実行し、全ユーザーに **/test/** ディレクトリーへの完全アクセスを可能にします。

4. **cp /var/www/html/test.tar /test/** コマンドを実行し、**test.tar** ファイルを **/test/** ディレクトリーにコピーします。

5. **cd /test/** コマンドを実行し、**/test/** ディレクトリーに切り替え、このディレクトリー内で **tar -xvf test.tar** コマンドを実行して tar アーカイブを抽出します。

6. **ls -lZ /test/** コマンドで SELinux コンテキストを表示させます。**httpd_sys_content_t** タイプが維持されます。**--selinux** が使われていなければ、**default_t** に変更されます。

```
~]$ 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
```

7. **/test/** ディレクトリーが不要であれば、Linux root ユーザーで **rm -ri /test/** 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を削除します。

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。

5.10.5. star を使ったファイルのアーカイビング

star 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。**star -xattr -H=exustar** を使ってコンテキストを維持するアーカイブを作成します。star パッケージはデフォルトで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 Linux 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。

1. Linux root ユーザーで **touch /var/www/html/file{1,2,3}** コマンドを実行し、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/** ディレクトリーからの **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. **cd /var/www/html/** コマンドで **/var/www/html/** ディレクトリーに移動します。その後、Linux root ユーザーで **star -xattr -H=exustar -c -f=test.star file{1,2,3}** コマンドを実行し、**test.star** という名前の Star アーカイブを作成します。

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

3. Linux root ユーザーで **mkdir /test** コマンドを実行し、新規ディレクトリーを作成します。そして **chmod 777 /test/** コマンドを実行し、全ユーザーに **/test/** ディレクトリーへの完全アクセスを可能にします。

4. **cp /var/www/html/test.star /test/** コマンドを実行し、**test.star** ファイルを **/test/** ディレクトリーにコピーします。

5. **cd /test/** コマンドを実行し、**/test/** ディレクトリーに切り替え、このディレクトリー内で **star -x -f=test.star** コマンドを実行して Star アーカイブを抽出します。

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

6. **ls -lZ /test/** コマンドで SELinux コンテキストを表示させます。**httpd_sys_content_t** タイプが維持されます。**-xattr -H=exustar** が使われていなければ、**default_t** に変更されます。

```
~]$ 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であれば、Linux root ユーザーで **rm -ri /test/** 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を削除します。

8. **star** が不要であれば、Linux root ユーザーで**yum remove star** コマンドを実行し、パッケージを削除します。

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。

5.11. 情報収集ツール

これらのツールはコマンドラインツールで、フォーマット済みの出力を提供します。コマンドラインパイピングの一部としては使いづらいものですが、集中してうまくフォーマットされた情報をすばやく提供してくれます。

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短い出力を提供します。時間間隔を秒に指定することで、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** オプションで別のキャッシュファイルを指定できます。

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

seinfo

このユーティリティーは、クラスやタイプ、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んだ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、初期 SID はブート中にカーネルがポリシーをロードする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** コマンドは、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 コマンドを使うと、ポリシーで特定のタイプを検索できます。ポリシーソースファイルかバイナリーファイルの検索が可能です。例えば、

```
~]$ 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 コマンドは、**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
```

5.12. マルチレベルのセキュリティー (MLS)

マルチレベルのセキュリティー技術とは、Bell-La Padula Mandatory Access Model を強制するセキュリティースキームを指します。MLS では、ユーザーとプロセスは *サブジェクト (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ーレベルでラベル付けされ、これはサブジェクトのクリアランスとオブジェクトの分類を必要とします。各セキュリティーレベルは *sensitivity (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5.1「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獲得したユーザーのみとなります。しかし、部外秘クリアランスのみを持つユーザーは、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5.1 クリアランスレベル

図5.2「MLS を使ったデータフローの許可」は、「秘密」セキュリティー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ー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。

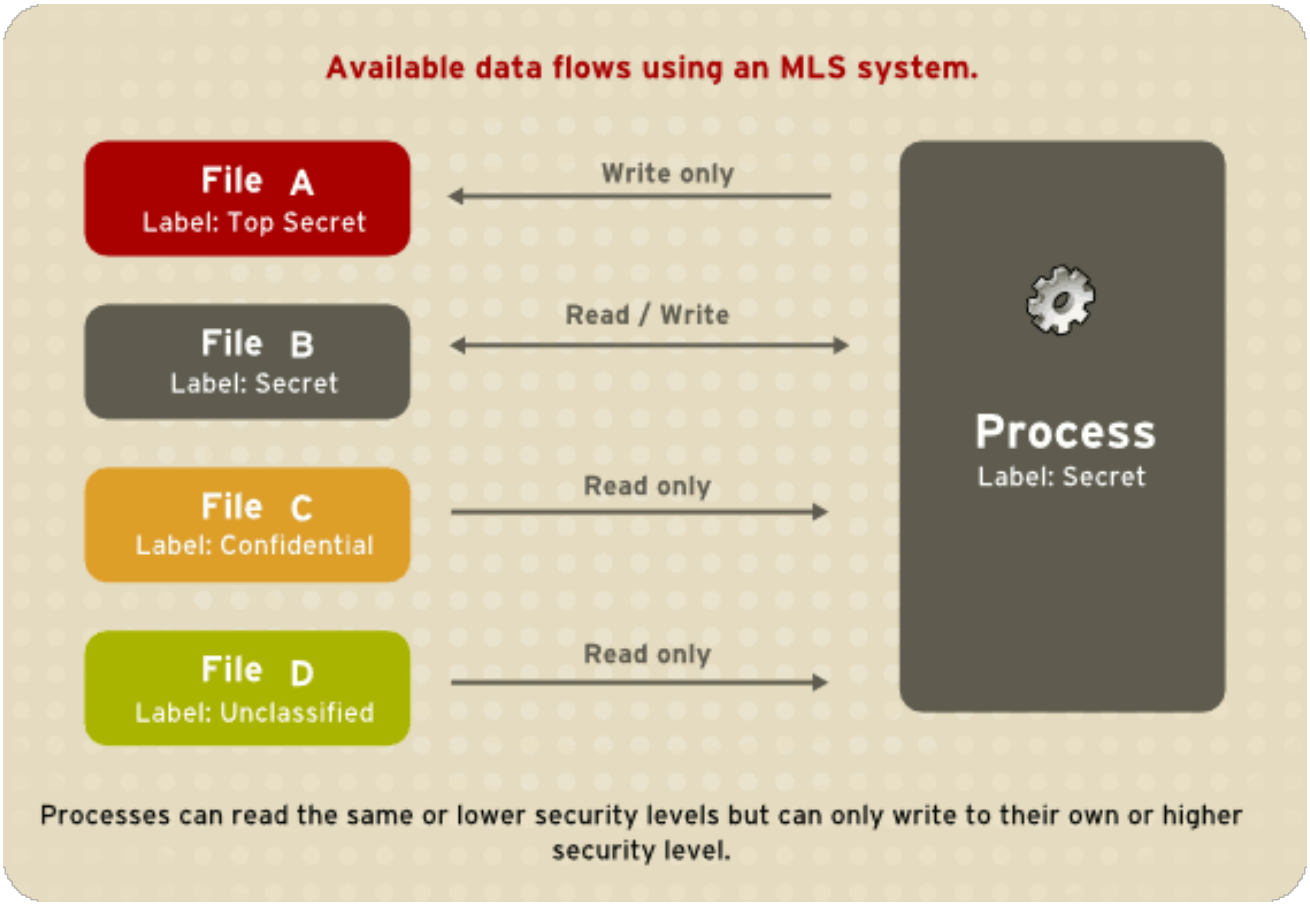


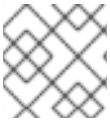
図5.2 MLS を使ったデータフローの許可

5.12.1. 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」のセキュリティー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」のセキュリティーレベルをもつユーザーによる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ークリアランスが、任意にファイルシステムを閲覧する許可を自動的に与え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。

5.12.2. 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。

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
```

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。enforcing モードに変更する前に、Linux root ユーザーで **grep "SELinux is preventing" /var/log/messages** 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、このコマンドに返される出力はありません。起動時に SELinux がアクセスを拒否していた場合は、トラブルシューティング情報を 8 章*トラブルシューティング*で参照してください。

7. **/var/log/messages** 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 permissive モードで稼働していることを確認します。

```
~]$ getenforce
Enforcing
```

そして MLS ポリシーを有効にします。

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

5.12.3. 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。

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。

```
~]# passwd john
```

3. semanage login -l 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。

```
Login Name                SELinux User              MLS/MCS Range

__default__               user_u                    s0
john                      user_u                    s0
root                      root                      s0-
s15:c0.c1023
system_u                  system_u                  s0-
s15: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. semanage login -l 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。ユーザー john に特定 MLS 範囲が定義されていることに注意してください。

```
Login Name                SELinux User              MLS/MCS Range

__default__               user_u                    s0
john                      user_u                    s2:c100
root                      root                      s0-
s15:c0.c1023
system_u                  system_u                  s0-
s15: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
```

5.12.4. 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。

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。

---

[8] Brindle, Joshua. "Re: blurb for fedora setools packages" Email to Murray McAllister. 1 November 2008 年 11 月 1 日。このバージョンにおける編集および変更は、すべて Murray McAllister による。

[9] **/etc/selinux/targeted/contexts/files/** 内のファイルがファイルと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** に読み取られ、デフォルトのコンテキストにファイルおよびディレクトリーが復元されます。

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

[11] Red Hat Enterprise Linux の libselinux-utils パッケージと出荷されたmatchpathcon(8) man ページは、Daniel Walsh が作成したものです。編集および変更はすべて、Muray McAllister によるものです

第6章 ユーザーの制限

Red Hat Enterprise Linux 6 では、数多くの制限のある SELinux ユーザーが利用可能です。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、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。(ユーザーによりますが) 例えば、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、などの制限です。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

Linux root ユーザーで **semanage login -l** 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

__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 6 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** user 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、これが表示されます。

```
~]$ 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。制限のない Linux ユーザーが、**unconfined_t** ドメインから自身の制限のあるドメインへの移行が可能だと SELinux ポリシーが定義するアプリケーションを実行しても、制限のない Linux ユーザーはその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ーの利点は、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: これは、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 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** も実行できないので、これらのコマンドで Linux root ユーザーになることを防いでいます。

1. Linux root ユーザーで **useradd -Z user_u useruuser** コマンドを実行し、SELinux **user_u** ユーザーにマッピングされる新規の Linux ユーザー (useruuser) を作成します。

2. Linux root ユーザーで **semanage login -l** コマンドを実行し、Linux **useruuser** ユーザーと **user_u** の間のマッピングを表示します。

```
~]# semanage login -l

Login Name                SELinux User              MLS/MCS Range

__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. Linux root ユーザーで **passwd useruuser** コマンドを実行し、Linux useruuser ユーザーにパスワードを割り当てます。

```
~]# passwd useruuser
Changing password for user useruuser.
New UNIX password: Enter a password
Retype new UNIX 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 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。**id -Z** コマンドを実行し、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、Linux root ユーザーで **userdel -r useruuser** 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。

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。

1. Linux root ユーザーで **useradd newuser** 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。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

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

2. この Linux newuser ユーザーを SELinux **user_u** ユーザーにマッピングするには、Linux root ユーザー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 ユーザーと **user_u** 間のマッピングを表示するには、**semanage login -l** コマンドを Linux root ユーザーで実行します。

```
~]# semanage login -l

Login Name                SELinux User              MLS/MCS Range

__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. Linux root ユーザーで **passwd newuser** コマンドを実行し、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。**id -Z** 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、Linux root ユーザーで **userdel -r newuser** 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。**semanage login -d newuser** コマンドを実行し、Linux newuser ユーザーと **user_u** の間のマッピングを削除します。

```
~]# userdel -r newuser
~]# semanage login -d newuser
~]# semanage login -l
```

```
Login Name                SELinux User                MLS/MCS Range

__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 6 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。

例えば以下のように、Linux root ユーザーで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__
```

Linux root で **semanage login -l** コマンドを実行し、**__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

__default__               user_u                      s0
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、Linux root ユーザー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. Linux root ユーザーで **yum install xguest** コマンドを実行して 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。

2. 誰もがキオスクアカウントを使えるようにするためにアカウントはパスワード保護されないので、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** コマンドを使って 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 モード」 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 6 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。

この動作の変更のためにブール値が利用でき、 **setsebool** コマンドで設定します。**setsebool** コマンドは、Linux 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。リブート後には変更を維持したくない場合は、**-P** オプションを使わないでください。

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 allow_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 allow_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 allow_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 allow_staff_exec_content off
```

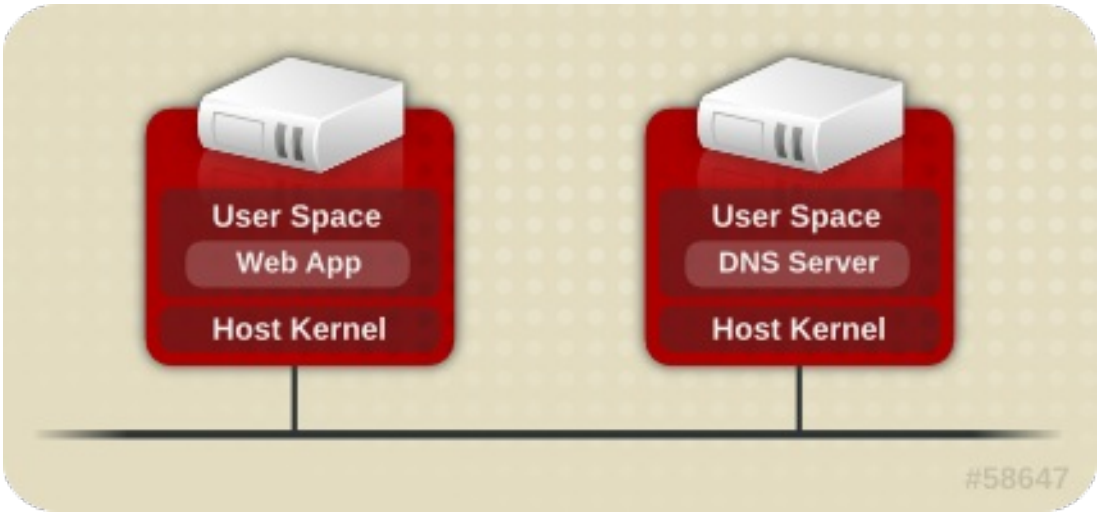
第7章 SVIRT

sVirt は Red Hat Enterprise Linux 6 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ーを改善します。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、セキュリティーを高めるためです。

本章では、Red Hat Enterprise Linux 6 での sVirt による仮想化技術の統合について説明します。

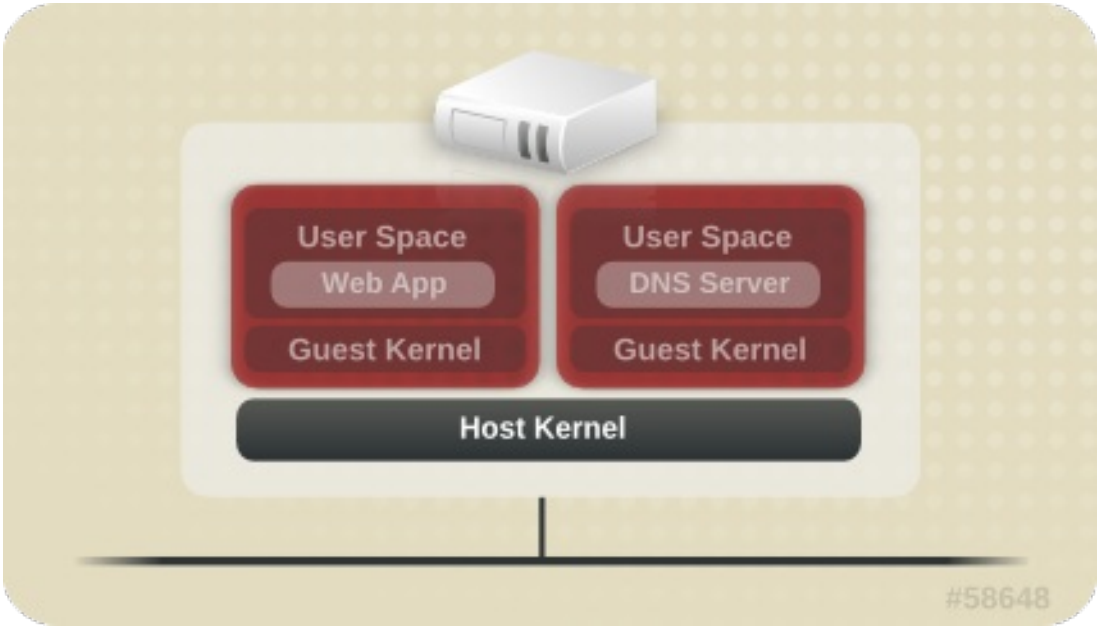
非仮想化環境

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

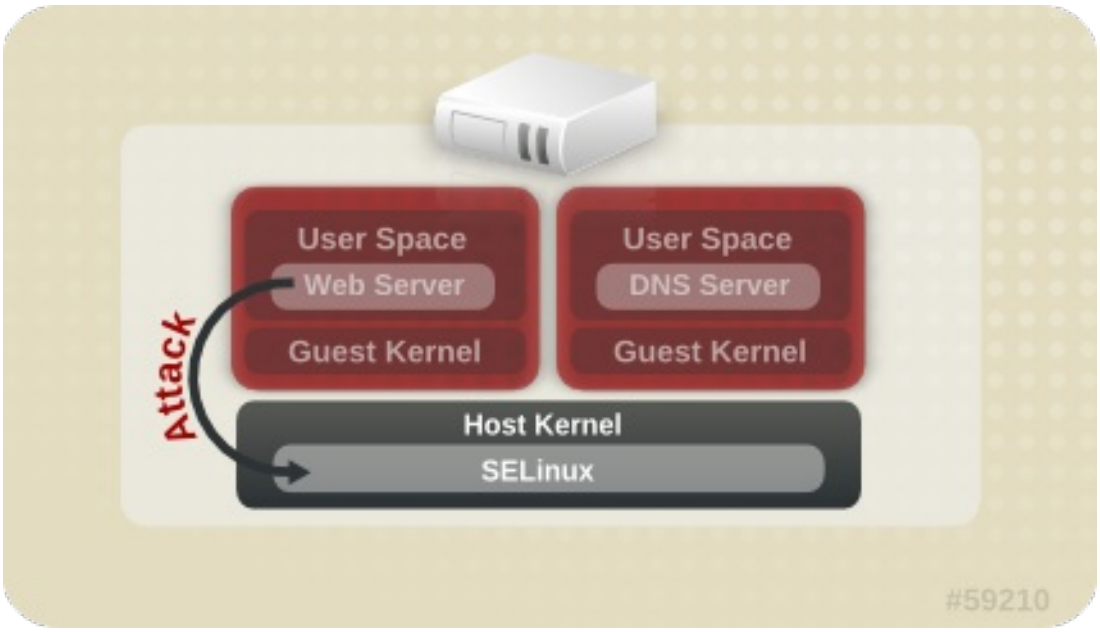
仮想化環境では、複数のオペレーティングシステムを (「ゲスト」として) 単一のホストカーネルおよび物理ホストに格納することができます。下の図は仮想化環境を示したものです。



7.1. セキュリティーと仮想化

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ーに不備があって、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、そのゲストはホストのみならず、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。ラベルが付くと、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ます。

7.2. SVIRT のラベリング

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ー層を提供します。通常の使用では、sVirt が背後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付けが行なわれ、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。現在は、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 フィールドが付いた **svirt_t** 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** 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することができます。
仮想マシンのイメージ	system_u:object_r:virt_content_t:s0	イメージが存在する場合に使用されるシステムのデフォルトラベル。**svirt_t** 仮想プロセスは、このラベルの付いたファイル/デバイスの読み取りはできません。

sVirt 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殊な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 トラブルシューティング

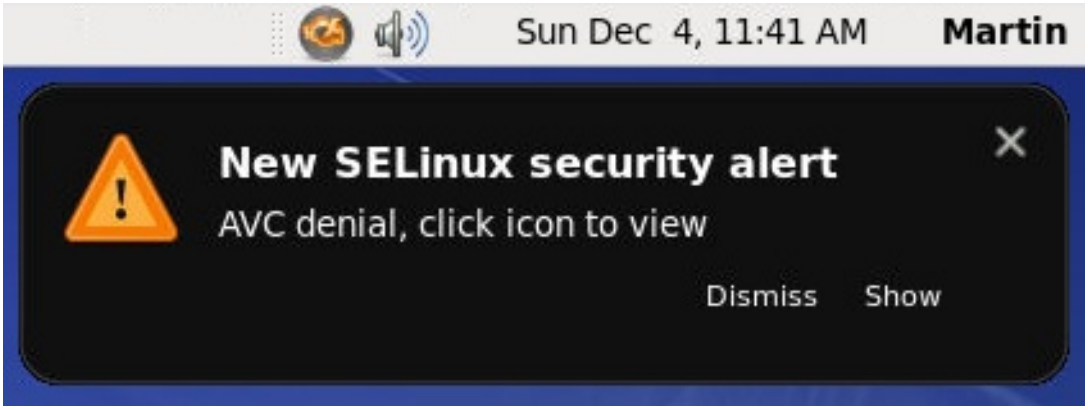
本章では、SELinux がアクセスを拒否した場合に何が起こるかと以下の点を説明します。問題の 3 つの主要原因。正しいラベリングについての情報の場所。SELinux 拒否の分析。**audit2allow** を使ったカスタムポリシーモジュールの作成。

8.1. アクセス拒否の場合

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。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**. Easier-to-read denial messages also sent to **/var/log/messages**

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 によってアクセスが拒否され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 Linux 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
```

8.2. 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、ラベリングの問題、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、です。

8.2.1. ラベリングの問題

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ー関連の情報を含むラベル付けがされます。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リングが間違っていると、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、間違ったラベルのファイルを作成することにもなります。

一般的なラベリングの問題は、非標準のディレクトリーをサービスに使う場合に起こります。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 6 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も **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できないようにします。アクセスを許可するには、**/srv/myweb/** にあるファイルが **httpd** 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[12]。**semanage** コマンドはコンテキストを変更しません。Linux root ユーザーで **restorecon** コマンドを実行し、変更を適用します。

```
~]# 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。

8.2.1.1. 正しいコンテキストとは？

matchpathcon コマンドは、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上での **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** コマンドを使って、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
```

ある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。

8.2.2. 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です。このため、SELinux にサービスの実行方法を指示する必要があります。これは、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できます。これは、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 ファイルシステ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 MySQL との通信を許可するには、**httpd_can_network_connect_db** のブール値をオンにします。

```
~]# 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** コマンドを使って、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
allow_ftpd_anon_write --> off
allow_ftpd_full_access --> off
allow_ftpd_use_cifs --> off
allow_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、Linux 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、Linux root ユーザーで **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、**service httpd start** コマンドは失敗します。

```
~]# service httpd start
Starting httpd: (13)Permission denied: make_sock: could not bind to
address [::]:9876
(13)Permission denied: make_sock: could not bind to address 0.0.0.0:9876
no listening sockets available, shutting down
Unable to open logs
                   [FAILED]
```

以下のような SELinux 拒否は、**/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[13]。

```
~]# 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。

8.2.3. 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** 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、「Allowing Access: audit2allow」を参照してください。

8.3. 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 - これはラベリングとブール値の情報を含んでいます。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。

8.3.1. 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。2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、**ls -l** 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、Linux root ユーザー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" の草案を参照してください。

8.3.2. 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 Linux root ユーザーで実行します。

```
~]# 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、Linux root ユーザー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 監査メッセージ」 と 「sealert メッセージ」 を参照してください。

8.3.3. 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 ファイルシステムにアクセスする **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 **selinux** が付加もしくは追加された man ページにあります。

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。サービスの SELinux 情報がある他の man ページには、以下のものが含まれます。

- Samba: samba_selinux(8) man ページでは、Samba でエクスポートされるファイルおよびディレクトリーが **samba_share_t** タイプとブール値でラベル付けされることで、**samba_share_t** タイプ以外のラベル付けがされたファイルを Samba でエクスポートすることが可能になることを説明しています。

- NFS: nfs_selinux(8) man ページでは、デフォルトではファイルシステムは NFS ではエクスポートできず、これを可能にするには **nfs_export_all_ro** や **nfs_export_all_rw** といったブール値をオンにする必要があることを説明しています。

- Berkeley Internet Name Domain (BIND): named(8) man ページは、特定の状況で使うファイルを説明しています (**Red Hat SELinux BIND Security Profile** セクションを参照)。named_selinux(8) man ページでは、デフォルトでは **named** はマスターゾーンファイルに書き込みができず、このアクセスを許可するには、**named_write_master_zones** ブール値をオンにする必要があ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。

8.3.4. 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 permissive の作成はできませんでした (ドメインのプロセス実行)。特定の状況ではこの結果、システム全体の permissive が解決が必要な問題となっていました。

Red Hat Enterprise Linux 4 および 5 では、***domain*_disable_trans** ブール値はアプリケーションが制限のあるドメインに移行することを防ぐために利用可能で、このため **initrc_t** といった制御のないドメインでプロセスが実行されます。その結果、このようなブール値をオンにすると大きな問題が発生する可能性があります。例えば、 **httpd_disable_trans** ブール値をオンにすると、以下のようなことが発生します。

- **httpd** サービスが制限のない **initrc_t** ドメインで実行されます。**initrc_t** ドメインで実行中のプロセスが作成したファイルは、**httpd_t** ドメインで実行中のプロセスが作成したファイルに適用されるラベリングルールとは違うラベリングルールが適用され、プロセスは間違ったラベルのファイルの作成を許可される可能性があります。これは後で、アクセス問題を引き起こします。

- **httpd_t** との通信が許可されている制限のあるドメインは、**initrc_t** とつ通信できず、新たな失敗を引き起こす可能性があります。

この問題に対処するために、Red Hat Enterprise Linux 6 では *permissive ドメイン*が導入されます。これは、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。

8.3.4.1. 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、Linux root ユーザー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、Linux root ユーザーで **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 Linux root ユーザーで実行します。以下のようになります。

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

8.3.4.2. 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** メッセージには **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 ドメインに関するさらなる情報は、Dan Walsh のブログ記事 "Permissive Domains" を参照してください。

8.3.5. 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 拒否の検索および表示には、**ausearch**、**aureport**、**sealert** などの数多くのツールが利用できます。

ausearch

audit パッケージは **ausearch** を提供します。ausearch(8) man ページでは、「**ausearch** は、異なる

検索基準に基づいたイベントで監査デーモンログにクエリを行えるツールです」となっています[14]。**ausearch** ツールは **/var/log/audit/audit.log** にアクセスするので、Linux 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**

特定のサービスでの SELinux 拒否を検索するには、**-c *comm-name*** オプションを使います。ここでの *comm-name* は「実行可能な名前です」[15]。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 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** を提供します。aureport(8) man ページでは、「**aureport** は、監査システムログのサマリーレポートを作成するツールです」となっています[16]。**aureport** ツールは **/var/log/audit/audit.log** にアクセスするので、Linux root ユーザーで実行する必要があります。SELinux 拒否の一覧を表示し、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
```

aureport オプションの詳細については、aureport(8) man ページを参照してください。

sealert

setroubleshoot-server パッケージは **sealert** 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。**/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 httpd (httpd_t) "getattr" to
/var/www/html/file1 (samba_share_t). For complete SELinux messages. run
sealert -l 84e0b04d-d0ad-4347-8317-22e74f6cd020
```

この例の拒否 ID は、**84e0b04d-d0ad-4347-8317-22e74f6cd020** です。**-l** オプションは、ID を引数 として受け取ります。**sealert -l 84e0b04d-d0ad-4347-8317-22e74f6cd020** 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** と **dbus**、**auditd** デーモンが稼働している場合、SELinux によってアクセスが拒否されると警告が表示されます。「表示する」をクリックすると、**sealert** GUI が開始され、HTML 出力で拒否を表示します。

- **sealert -b** コマンドを実行し、**sealert** GUI を開始します。
- **sealert -l *** コマンドを実行し、すべての拒否の詳細な分析を表示します。
- Linux root ユーザーで **sealert -a /var/log/audit/audit.log -H > audit.html** コマンドを実行し、**sealert** GUI で見られたように **sealert** 分析の HTML バージョンを作成します。

sealert オプションの詳細については、sealert(8) man ページを参照してください。

8.3.6. Raw 監査メッセージ

Raw 監査メッセージは **/var/log/audit/audit.log** に記録されます。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 (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 }

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** 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起こります。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付けがされていたためです。よく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。注記:**samba_share_t** タイプは、**httpd_t** ドメインで実行中のプロセスにはアクセスできません。

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リソース (メモリーなど) 以上のものを使おうをして、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ー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、**initrc_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場合のよくある原因は、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。

8.3.7. 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** タイプのラベル付けがされたファイル (**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** が **samba_share_t** タイプのラベル付けをされています。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オンにする、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 の両方にアクセス可能な **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 監査メッセージ

/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 監査メッセージ」 を参照してください。

8.3.8. Allowing Access: audit2allow

本番環境では、このセクションの例を使用しないでください。これは、**audit2allow** の使用を説明する目的でのみ、使われています。

audit2allow(1) man ページでは、「**audit2allow** は、拒否操作のログから SELinux ポリシー allow ルールを生成する」となっています[17]。「sealert メッセージ」 にあるように拒否を分析し、ラベル変更がないもしくはアクセスを許可したブール値がない場合は、**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後に、**audit2allow** コマンド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 ルールが提示されます。

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. 拒否および関連するシステムコールは、**/var/log/audit/audit.log** 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 (comm="certwatch")** は **var_t** タイプ **(tcontext=system_u:object_r:var_t:s0)** のラベル付けがされたディレクトリーへの書き込みアクセス **({ write })** が拒否されました。「sealert メッセージ」 にあるように拒否を分析します。ラベル変更がないもしくはアクセスを許可したブール値がない場合は、**audit2allow** を使ってローカルポリシーモジュールを作成します。

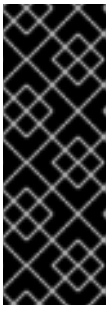
2. ステップ 1 の **certwatch** 拒否のように拒否がログ記録されている場合、**audit2allow -w -a** コマンドを実行してヒューマンリーダブルな記述でアクセスが拒否された理由を作成します。**-a** オプションでは、すべての監査ログが読み取られます。**-w** オプションは、ヒューマンリーダブルな記述を作成します。**audit2allow** ツールは **/var/log/audit/audit.log** にアクセスするので、Linux root ユーザーで実行する必要があります。

```
~]# 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.
```

Type Enforcement ルールがないのでアクセスが拒否されました。

3. **audit2allow -a** コマンドを実行して、拒否されたアクセスを可能にする 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、Linux root ユーザーで **audit2allow -a -M *mycertwatch*** 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

~]# ls
mycertwatch.pp  mycertwatch.te
```

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。モジュールをインストールするには、Linux root ユーザーで **semodule -i *mycertwatch.pp*** を実行します。

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、fedora-selinux-list などの SELinux リストに公表してレビューすることが推奨されます。ポリシーにバグがあると思われる場合は、Red Hat Bugzilla でバグを作成してください。

複数のプロセスから複数の拒否があって、そのうちの一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** コマンドを使って **audit2allow** の入力を絞り込みます。以下の例では、**grep** を使って **certwatch** に関連した拒否のみを **audit2allow** で送信する方法を示しています。

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

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

~]# semodule -i mycertwatch2.pp
```

audit2allow を使ってポリシーモジュールを作成する方法の詳細は、Dan Walsh のブログ記事 "Using audit2allow to build policy modules. Revisited." を参照してください。

---

[12] **/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** と **setfiles** 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。

[13] **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。注記: デフォルトでは、このファイルは Linux root ユーザーのみが読み取れます。

[14] Red Hat Enterprise Linux 6 の audit パッケージとして出荷されたausearch(8) man ページより

[15] Red Hat Enterprise Linux 6 の audit パッケージとして出荷されたausearch(8) man ページより

[16] Red Hat Enterprise Linux 6 の audit パッケージとして出荷されたaureport(8) man ページより

[17] Red Hat Enterprise Linux 6 で policycoreutils-sandbox パッケージがインストールされている場合に利用可能な audit2allow(1) man ページより

第9章 追加情報

9.1. 貢献者

- Domingo Becker - 翻訳 - スペイン語
- Dominick Grift - 技術編集者
- Daniel Cabrera - 翻訳 - スペイン語
- Murray McAllister - Red Hat エンジニアリングコンテンツサービス
- James Morris - 技術編集者
- Eric Paris - 技術編集者
- Scott Radvan - Red Hat エンジニアリングコンテンツサービス
- Daniel Walsh - Red Hat セキュリティーエンジニアリング
- Geert Warrink - 翻訳 - オランダ語

9.2. その他のリソース

米国国家安全保障局 (NSA)

以下は、NSA Contributors to SELinux ページからのものです。

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。セキュリティー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。

- 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 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 SELinux FAQ: http://docs.fedoraproject.org/.
- SELinux 制限のあるサービスの管理ガイド: http://docs.fedoraproject.org/

非公式の SELinux FAQ

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

IRC

Freenode について:

- #selinux
- #fedora-selinux
- #security

付録A 改訂履歴

改訂 3-3.2.400 **2013-10-31** **Rüdiger Landmann**
Rebuild with publican 4.0.0

改訂 3-3.2 **Wed May 22 2013** **Translator's Credit**
翻訳完了

改訂 3-3.1 **Wed May 22 2013** **Translator's Credit**
翻訳ファイルを XML ソースバージョン 3-3 と同期

改訂 3-3 **Fri Feb 22 2013** **Tomáš Čapek**
6.4 GA リリース向けバージョン

改訂 3-0 **Wed Jun 20 2012** **Martin Prpič**
Red Hat Enterprise Linux 6.3 向け SELinux ガイドのリリース

改訂 2-0 **Tue Dec 6 2011** **Martin Prpič**
Red Hat Enterprise Linux 6.2 向け SELinux ガイドのリリース

改訂 1.9-0 **Wed Mar 3 2010** **Scott Radvan**
Red Hat Enterprise Linux 6 向けの改訂