Red Hat Enterprise Linux 6

制限のあるサービスの管理

SELinux の制御下にあるサービスの設定について

エディッション 4

Last Updated: 2017-10-16

Red Hat Enterprise Linux 6 制限のあるサービスの管理

SELinux の制御下にあるサービスの設定について
エディッション 4

Red Hat Engineering Content Services

法律上の通知

概要

上級ユーザーや管理者の方などが Security-Enhanced Linux (SELinux) を使用したり、 また設定を行なったりする際に役立つガイドです。 SELinux のコンポーネントは、 上級ユーザーや管理者の方々が設定を行なわなければならない場合があるサービスに関連しているため、 本ガイドでは Red Hat Enterprise Linux に焦点を絞って SELinux のコンポーネントについて説明していきます。 また、 こうしたサービスを設定する場合の実例をあげながら、 SELinux でどのようにサービスの動作を補完しているのかについて見ていきます。

目次

第1章 はじめに

Security-Enhanced Linux (SELinux) とは、 Linux カーネル内の「*強制アクセス制御 (MAC)*」の実装であり、 標準の「*任意アクセス制御 (DAC)*」がチェックされた後、 許可される動作をチェックします。 SELinux は米国国家安全保障局 (National Security Agency) により開発され、 Linux システム内のファイルやプロセス、またその動作についても定義されたポリシーに応じてルールを強制することができます。

Security-Enhanced Linux (SELinux) では、 ディレクトリやデバイスなどのファイルをオブジェクト (客体) として参照します。 ユーザーによって実行されたコマンドや Mozilla Firefox アプリケーションなどのプロセスはサブジェクト (主体) 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、サブジェクトとオブジェクトの交信手段、 サブジェクト同士の交信手段などを制御する任意アクセス制御システム (DAC) が使用されています。 DAC を使用するオペレーティングシステムでは、 ユーザーが所有するファイル (オブジェクト) の権限はそのユーザーが制御するようになっています。 たとえば、 Linux オペレーティングシステムの場合、 ユーザーは自分のホームディレクトリを誰でも読み取れるようにすることができるため、 気付かずに機密の可能性がある情報へのアクセスを他のユーザーやプロセス (サブジェクト) に渡してしまう恐れがあります。

DAC によるアクセス決定はユーザー ID と所有権だけを基準とし、 ユーザーのロール、 プログラムの機能及び信頼性、データの機密性及び整合性など他のセキュリティ関連の情報は無視されます。 通常、各ユーザーが所有ファイルの完全決定権を持っているため、 システム全体にセキュリティポリシーを試行するのが困難になります。 さらに、 ユーザーによって実行されるプログラムはそのユーザーに与えられる権限を継承し、 そのユーザーのファイルへのアクセス権を自由に変更することができます。 このため、 悪意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ません。 多くのシステムサービスや特権を有するプログラムは、 特権付与の制御がおおまかで、その必要性をはるかに越える特権を持ったまま実行されます。 したがって、 こうしたプログラムのうちいずれかにでも不備があればそこを攻撃され、さらなるシステムアクセス権が奪われる恐れがあります。[1]

以下は、 SELinux を実行していない Linux オペレーティングシステムで使われているパーミッションの例です。 システムによっては、 パーミッションがこの例と多少異なる場合があります。 **ls -l** コマンドを使って、 ファイルのパーミッションを表示させます。

```
$ ls -l file1
-rwxrw-r-- 1 user1 group1 0 2010-01-29 09:17 file1
```

最初の 3 つのパーミッション **rwx** では、 Linux の **user1** ユーザー (この例では所有者) が **file1** に対して持っているアクセス権を制御しています。 次の 3 つのパーミッション **rw-** では、 Linux の **group1** グループが **file1** に対して持っているアクセス権を制御しています。 最後の 3 つのパーミッション **r--** では、 その他のユーザーが **file1** に対して持っているアクセス権を制御しています。 その他のユーザーには、 すべてのユーザーおよびプロセスが含まれます。

SELinux を使用すると、Linux カーネルに MAC (強制アクセス制御) が追加され、Red Hat Enterprise Linux ではデフォルトで有効になります。汎用の MAC アーキテクチャーには、各種のセキュリティ関連情報を含むラベルを決定基準とした管理用セキュリティポリシーを、 システム内の全プロセスおよびファイルに対して実施する能力が必要とされます。適切に実装することで、システム自体が的確に自己防御され、アプリケーションの改ざんを保護、回避することによりアプリケーションの安全性に必須のサポートを提供します。MAC ではアプリケーションどうしがしっかりと確実に分離されるため、信頼性の低いアプリケーションでも安全に実行することができます。プロセス実行に関する特権を制限する能力により、アプリケーションやシステムサービス内の脆弱性が悪用され発生する可能性のある被害範囲を限定することができます。限られた権限しか持たない正規ユーザーだけでなく、権限を与えられたユーザーが不正なアプリケーションを知らずに実行してしまった場合でも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、 ファイルなどに適用されるセキュリティ関連の情報を含むラベルの例です。 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 によりユーザー (**unconfined_u**)、 ロール (**object_r**)、 タイプ (**user_home_t**)、 レベル (**s0**) が **file1** ファイルに与えられています。 この情報がアクセス制御の決定に使用されます。 **ls -Z** コマンドでは SELinux コンテキストと共に DAC ルールも表示されます。 SELinux ポリシールールは、 DAC ルールの後にチェックされます。 このため、 DAC ルールで最初にアクセスが拒否されると、 SELinux ポリシールールは適用されません。

---

[1] 「Integrating Flexible Support for Security Policies into the Linux Operating System」、 Peter Loscocco および Stephen Smalley 著。 この論文は当初、 国家安全保障局向けに作成されましたが、 現在は公開されています。 詳細については オリジナル論文 を参照してください。

[2] 「Meeting Critical Security Objectives with Security-Enhanced Linux」、 Peter Loscocco および Stephen Smalley 著。 この論文は当初、 国家安全保障局向けに作成されましたが、その後公開されています。 詳細については、 オリジナル論文 を参照してください。

第2章 TARGETED ポリシー

Red Hat Enterprise Linux で使用されるデフォルトの SELinux ポリシーは Targeted ポリシーになります。Targeted ポリシーを使用すると、対象となるプロセスは制限ドメイン内で実行され、対象外のプロセスは未制限のドメイン内で実行されます。例えば、デフォルトではログインしたユーザーは **unconfined_t** ドメイン内で実行され、**init** で開始されたシステムプロセスは**initrc_t** ドメイン内で実行されます。 いずれも未制限のドメインです。

SELinux は、実行するサービスに必要な最小限レベルのアクセスに基づいています。サービスの実行手段は複数あるため、サービスをどのように実行するのかを SELinux に指示する必要があります。これを行なうため、Boolean を使用します。Boolean を使用すると、SELinux ポリシーの記述方法などの知識が全くなくても、ランタイム時の SELinux ポリシーを部分的に変更することができます。たとえば、サービスによる NFS ファイルシステムへのアクセスを許可するなど、Boolean を使用することで、SELinux ポリシーの再読み込みや再コンパイルを行なうことなく、各種の変更を行なうことができるようになります。Boolean 設定については事例をあげて詳細に説明していきます。

サービス用のファイル群の格納にデフォルト以外のディレクトリを使用する、 デフォルト以外のポート番号でサービスが実行するよう変更するなど、 その他の変更には、 policycoreutils-python パッケージで提供している **semanage** コマンドなどのツールを使ったポリシー設定の更新が必要になります。このコマンドについては設定事例をあげて詳細に説明していきます。

2.1. TYPE ENFORCEMENT

Type Enforcement が SELinux の targeted ポリシーで使用されるメインのパーミッション制御になります。全ファイルおよびプロセスにタイプのラベルが付けられます。ファイルの場合はタイプ、プロセスの場合はドメインを定義します。任意のタイプにアクセスするドメインなのか、 別のドメインにアクセスするドメインなのかなど、SELinux のポリシールールではタイプによって互いがアクセスしあう方法を定義します。アクセスを許可する特定の SELinux ポリシールールが存在する場合にのみ、そのアクセスが許可されます。

2.2. 制限のあるプロセス

Red Hat Enterprise Linux では、 ネットワークでリッスンするサービスはほぼ全て制限されています。また、 **root** ユーザーとして実行され **passwd** アプリケーションなどユーザーのタスクを行なうプロセスもほとんど制限されています。 プロセスが制限されている場合、 そのプロセスはそれ自体のドメイン内で実行されます。**httpd** プロセスなら**httpd_t** ドメイン内で実行されます。 制限のあるプロセスが攻撃を受けた場合、 SELinux ポリシー設定に応じて、 攻撃側がリソースにアクセスして加えることができる被害が限定されます。

次の例では、Samba での使用を目的としたファイルなど、正しくラベル付けが行なわれていないファイルは Apache HTTP サーバー (**httpd**) では読み込ませないよう SELinux で阻止する方法を示します。この例はサンプルのため、実稼働環境では使用しないようにしてください。ここでは、httpd、wget、setroubleshoot-server、audit などのパッケージがすでにインストールされていること、SELinux の targeted ポリシーが使用されていること、enforcing モードで実行し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 モードで実行していること、 targeted ポリシーが使われていることを確認します。

```
$ /usr/sbin/sestatus
SELinux status:                 enabled
SELinuxfs mount:                /selinux
Current mode:                   enforcing
```

```
Mode from config file:          enforcing
Policy version:                 24
Policy from config file:        targeted
```

SELinux が有効になっていると、 **SELinux status: enabled** が返されます。 SELinux が enforcing モードで実行されていると、 **Current mode: enforcing** が返されます。 SELinux targeted 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. root ユーザーになり、 **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** コマンドを実行して SELinux の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
```

この **testfile** ファイルには、SELinux の **unconfined_u** ユーザーラベルが付けられています。**unconfined_u** SELinux ユーザーにマッピングされた Linux ユーザーによってこのファイルが作成されたためです。ファイルではなくプロセスの場合には、ロールベースのアクセス制御 (RBAC) が使用されます。ファイルの場合はロールに意味はなく、汎用ロールとなる **object_r** ロールが使用されます (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上のファイル)。**/proc/** ディレクトリ配下では、プロセスに関連するファイルには**system_r** ロールが使用される場合があります。[3]**httpd_sys_content_t** タイプで **httpd** プロセスによるこのファイルへのアクセスを許可しています。

4. root ユーザーになり、 **service httpd start** コマンドを使って **httpd** プロセスを開始します。 **httpd** が正常に起動すると以下のような出力が表示されます。

```
# /sbin/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に移動してから、 **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。

```
--2009-12-01 11:40:28--  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

2009-12-01 11:40:28 (0.00 B/s) - `testfile' saved [0/0]
```

6. **chcon** コマンドでファイルのラベルを付け換えます。ただし、ファイルシステムのラベルが付け換えられると、この変更は失われます。ファイルシステムのラベルが付け換えられた場合でも、こうした変更を永続的に維持するには、**semanage** コマンドを使用します。このコマンド

についてはのちほど説明していきます。 root ユーザーになり、次のコマンドを実行してタイプを Samba で使用されるタイプに変更します。

chcon -t samba_share_t /var/www/html/testfile

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. 現在の DAC パーミッションでは、**httpd** プロセスによる **testfile** へのアクセスを許可している点に注意してください。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に移動し、**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。

```
--2009-12-01 11:43:18--  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
2009-12-01 11:43:18 ERROR 403: Forbidden.
```

8. root ユーザーになり、 **rm /var/www/html/testfile** コマンドを実行し **testfile** を削除します。

9. **httpd** を実行しておく必要がない場合は、 root ユーザーになり **service httpd stop** コマンドを実行し **httpd** を停止します。

```
# /sbin/service httpd stop
Stopping httpd:                                            [  OK
]
```

上記の例では、 SELinux によって追加される安全性を示しました。 ステップ 5 では、 DAC ルールによって **httpd** プロセスの **testfile** へのアクセスが許可されています。しかし、**httpd** プロセスによるアクセス権がないタイプにラベルが付け換えられたため、SELinux によってアクセスが拒否されるようになります。ステップ 7 のあと、setroubleshoot-server パッケージをインストールすると、次のようなエラーが **/var/log/messages** にログ記録されます。

```
setroubleshoot: SELinux is preventing httpd (httpd_t) "getattr" to
/var/www/html/testfile (samba_share_t). For complete SELinux messages run
sealert -l c05911d3-e680-4e42-8e36-fe2ab9f8e654
```

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。

```
[Tue Dec 01 11:43:18 2009] [error] [client 127.0.0.1] (13)Permission
denied: access to /testfile denied
```

2.3. 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** ドメインで実行され、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の場合でも SELinux ポリシールールは適用されます。ただし、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。このため、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 ルールに依存することになります。制限のないプロセスが攻撃された場合、攻撃者によってシステムリソースやデータへのアクセス権が奪われても SELinux ではそれを阻止しませんが、 DAC ルールは常に適用されます。SELinux は DAC ルールに加えて使用することで二重のセキュリティ強化を施行するものであり、DAC ルールの代替として使用するものではありません。

以下の例では、 Apache HTTP Server (**httpd**) を制限なしで実行している場合、 Samba 向けのデータにApache HTTP Server をどのようにしてアクセスさせることができるかを示します。Red Hat Enterprise Linux では、**httpd** プロセスはデフォルトで制限のある**httpd_t** ドメイン内で実行されます。次の例はあくまで例であり、実稼働環境では使用しないでください。httpd、 wget、dbus、audit パッケージがインストールされていること、 SELinux targeted ポリシーが使われていること、 またモードは enforcing 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 モードで実行していること、 targeted ポリシーが使われていることを確認します。

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

   SELinux が有効になっていると、 **SELinux status: enabled** が返されます。 SELinux が enforcing モードで実行されていると、 **Current mode: enforcing** が返されます。 SELinux targeted 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. root ユーザーになり、 **touch /var/www/html/test2file** 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/test2file** コマンドを実行して SELinux の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/test2file
```

   test2file には SELinux **unconfined_u** ユーザーのラベルが付けられています。 **unconfined_u** SELinux ユーザーにマッピングされた Linux ユーザーによってこのファイルが作成されたためです。 ファイルではなくプロセスの場合には、 ロールベースのアクセス制御 (RBAC) が使用されます。 ファイルの場合、 ロールの使用は意味がありません。 **object_r** ロールはファイルに使用される汎用ロールになります (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上)。 **/proc/** ディレクトリ配下では、 プロセスに関連するファイルは **system_r** ロールを使用することができます。[4]**httpd_sys_content_t** タイプを使用すると、 **httpd** プロセスにこのファイルへのアクセスを許可することになります。

4. **chcon** コマンドでファイルのラベルを付け換えます。ただし、ファイルシステムのラベルが付け換えられると、 この変更は失われます。ファイルシステムのラベルが付け換えられた場合でも、 こうした変更を永続的に維持するには、**semanage** コマンドを使用します。このコマンドについてはのちほど説明していきます。 root ユーザーになり、次のコマンドを実行してタイプを Samba で使用されるタイプに変更します。

   chcon -t samba_share_t /var/www/html/test2file

ls -Z /var/www/html/test2file コマンドを実行して変更を表示します。

```
-rw-r--r--  root root unconfined_u:object_r:samba_share_t:s0
/var/www/html/test2file
```

5. **service httpd status** コマンドを実行し、**httpd** プロセスが実行していないことを確認します。

```
$ /sbin/service httpd status
httpd is stopped
```

出力が異なる場合は、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。

```
# /sbin/service httpd stop
Stopping httpd:                                            [  OK
]
```

6. **httpd** プロセスを制限なしで実行する場合は、root ユーザーで以下のコマンドを実行し、**/usr/sbin/httpd** のタイプを制限のあるドメインに遷移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されていることを確認します。

```
-rwxr-xr-x  root root system_u:object_r:unconfined_exec_t
/usr/sbin/httpd
```

8. root ユーザーになり、**service httpd start** コマンドを使って **httpd** プロセスを開始します。**httpd** が正常に起動すると以下のような出力が表示されます。

```
# /sbin/service httpd start
Starting httpd:                                            [  OK
]
```

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

```
$ ps -eZ | grep httpd
unconfined_u:system_r:unconfined_t 7721 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 7723 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 7724 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 7725 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 7726 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 7727 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 7728 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 7729 ?      00:00:00 httpd
unconfined_u:system_r:unconfined_t 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に移動し、 **wget http://localhost/test2file** 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。

```
--2009-12-01 11:55:47--  http://localhost/test2file
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: `test2file.1'

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

2009-12-01 11:55:47 (0.00 B/s) - `test2file.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** ドメイン内で実行しているため、 DAC ルールの使用に依存しています。 したがって、 **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、 ファイルのデフォルト SELinux コンテキストを復元します。 root ユーザーになり、 **restorecon -v /usr/sbin/httpd** コマンドを実行して、 **/usr/sbin/httpd** のデフォルトの SELinux コンテキストを復元します。

```
# /sbin/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されていることを確認します。

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

12. root ユーザーになり、 **/sbin/service httpd restart** コマンドを実行し、 **httpd** を再起動します。 再起動したら、 **ps -eZ | grep httpd** コマンドを実行して **httpd** が制限のある **httpd_t** ドメイン内で実行していることを確認します。

```
# /sbin/service httpd restart
Stopping httpd:                                            [  OK
]
Starting httpd:                                            [  OK
]
# ps -eZ | grep httpd
unconfined_u:system_r:httpd_t    8880 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t    8882 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t    8883 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t    8884 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t    8885 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t    8886 ?        00:00:00 httpd
```

```
unconfined_u:system_r:httpd_t    8887 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t    8888 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t    8889 ?        00:00:00 httpd
```

13. root ユーザーになり、 **rm /var/www/html/test2file** コマンドを実行し **test2file** を削除します。

14. **httpd** を実行しておく必要がない場合は、 root ユーザーになり **service httpd stop** コマンドを実行し **httpd** を停止します。

```
# /sbin/service httpd stop
Stopping httpd:                                            [  OK
]
```

このセクションでは、 制限のあるプロセスが攻撃を受けた場合、データはどのように保護されるのか (SELinux で保護)、また制限のないプロセスが攻撃を受けた場合、攻撃者にとっていかにデータにアクセスしやすいか (SELinux で保護されていない) を例を使って説明しました。

---

[3] MLS など他のポリシーを使用する場合は **secadm_r** など別のロールを使用する場合があります。

[4] MLS など他のポリシーを使用する場合は **secadm_r** など別のロールを使用する場合があります。

第3章 APACHE HTTP SERVER

Apache HTTP Server Project ページより抜粋:

原文: "The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows NT. The goal of this project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards". (訳文: Apache HTTP Server Project は、 UNIX や Windows NT など新しいオペレーティングシステム向けのオープンソース HTTP サーバーの開発および保守に取り組んでいます。 現在の HTTP 標準に適合する HTTP サービスを提供し、 安全かつ効率的で拡張性のあるサーバーを実現していくことを目標としています。)[5]

Red Hat Enterprise Linux では、 Apache HTTP Server は httpd パッケージで提供されています。 httpd パッケージがインストールされているか確認する場合は、 **rpm -q httpd** を実行します。Apache HTTP Server を使用する予定にも関わらず、このパッケージがインストールされていない場合は、root ユーザーになり次のコマンドを実行してパッケージのインストールを行ないます。

```
yum install httpd
```

3.1. APACHE HTTP SERVER と SELINUX

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

1. **getenforce** を実行して SELinux が enforcing モードで実行しているか確認します。

   ```
   $ getenforce
   Enforcing
   ```

   SELinux が enforcing モードで実行している場合は、 **getenforce** コマンドを実行すると **Enforcing** が返されます。

2. root ユーザーになり **service httpd start** を実行し、 **httpd** を起動します。

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

3. **ps -eZ | grep httpd** を実行して **httpd** プロセスを表示します。

   ```
   $ ps -eZ | grep httpd
   unconfined_u:system_r:httpd_t:s0 2850 ?        00:00:00 httpd
   unconfined_u:system_r:httpd_t:s0 2852 ?        00:00:00 httpd
   unconfined_u:system_r:httpd_t:s0 2853 ?        00:00:00 httpd
   unconfined_u:system_r:httpd_t:s0 2854 ?        00:00:00 httpd
   unconfined_u:system_r:httpd_t:s0 2855 ?        00:00:00 httpd
   unconfined_u:system_r:httpd_t:s0 2856 ?        00:00:00 httpd
   ```

```
unconfined_u:system_r:httpd_t:s0 2857 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t:s0 2858 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t:s0 2859 ?        00:00:00 httpd
```

httpd プロセスに関連する SELinux コンテキストは **unconfined_u:system_r:httpd_t:s0** です。 コンテキストの末尾から 2 番目の部分、**httpd_t** がタイプになります。 プロセスのドメインやファイルのタイプを定義するのがタイプです。 この例の場合、 **httpd** プロセスは **httpd_t** ドメインで実行されています。

httpd_t などの制限ドメイン内で実行しているプロセスがファイルや他のプロセス、システムなどとどのように交信するのかをSELinux ポリシーで定義します。**httpd** からアクセスができるよう、ファイルには適切なラベルを付けなければなりません。たとえば、**httpd_sys_content_t** タイプのラベルが付いたファイルの場合、**httpd** からはこのファイルの読み取りは可能ですが書き込みはできません。この場合、Linux (DAC) のパーミッションで書き込みのアクセスが許可されていても書き込みを行なうことはできません。特定の動作を許可する場合、たとえば、スクリプトによるネットワークへのアクセスを許可する、**httpd** による NFS や CIFS ファイルシステムへのアクセスを許可する、**httpd** による CGI (Common Gateway Interface) スクリプトの実行を許可するなどの場合には、Boolean をオンにする必要があります。

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

1. **service httpd status** を実行して **httpd** が実行中ではないことを確認します。

```
# service httpd status
httpd is stopped
```

出力が上記と異なる場合は、 **service httpd stop** を実行してプロセスを停止します。

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

2. **semanage port -l | grep -w http_port_t** を実行して、 SELinux で **httpd** にリッスンを許可しているポートを表示させます。

```
# semanage port -l | grep -w http_port_t
http_port_t                    tcp      80, 443, 488, 8008, 8009,
8443
```

3. root ユーザーで **/etc/httpd/conf/httpd.conf** を編集、**Listen** オプションを設定します。SELinux ポリシー設定では **httpd** 用にまだ設定が行なわれていないポートをこのオプションに記述します。この例では、 ポート 12345 でリッスンするよう **httpd** を設定しています。

```
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses (0.0.0.0)
#
```

```
#Listen 12.34.56.78:80
Listen 127.0.0.1:12345
```

4. **service httpd start** を実行して **httpd** を起動します。

```
# service httpd start
Starting httpd: (13)Permission denied: make_sock: could not bind to
address 127.0.0.1:12345
no listening sockets available, shutting down
Unable to open logs        [FAILED]
```

以下のような SELinux 拒否がログ記録されます。

```
setroubleshoot: SELinux is preventing the httpd (httpd_t) from
binding to port 12345. For complete SELinux messages. run sealert -l
f18bca99-db64-4c16-9719-1db89f0d8c77
```

5. **httpd** によるポート 12345 のリッスンを SELinux で許可させるには、 以下の例で使用している次のコマンドが必要になります。

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

6. もう一度 **service httpd start** を実行して、 **httpd** を起動させ新しいポートをリッスンするようにします。

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

7. **httpd** による非標準ポート (この例では TCP 12345) でのリッスンを許可する SELinux 設定が完了しました。これで、**httpd** がこのポートで正常に起動するようになります。

8. **httpd** が確かに TCP ポート 12345 でリッスンし通信を行なっているか確認するには、 そのポートに telnet 接続を開き HTTP GET コマンドを以下のように発行します。

```
# telnet localhost 12345
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 200 OK
Date: Wed, 02 Dec 2009 14:36:34 GMT
Server: Apache/2.2.13 (Red Hat)
Accept-Ranges: bytes
Content-Length: 3985
Content-Type: text/html; charset=UTF-8
[...continues...]
```

3.2. タイプ

Type Enforcement が SELinux の targeted ポリシーで使用されるメインのパーミッション制御になります。全ファイルおよびプロセスにタイプのラベルが付けられます。ファイルの場合はタイプ、プロセ

スの場合はドメインを定義します。任意のタイプにアクセスするドメインなのか、 別のドメインにアクセスするドメインなのかなど、SELinux のポリシールールではタイプによって互いがアクセスしあう方法を定義します。 アクセスを許可する特定の SELinux ポリシールールが存在する場合にのみ、 そのアクセスが許可されます。

以下の例では、 **/var/www/html/** ディレクトリに新しいファイルを作成し、 そのファイルが親ディレクトリ (**/var/www/html/**) の **httpd_sys_content_t** タイプを継承していることを示しています。

1. **ls -dZ /var/www/html** を実行して **/var/www/html/** の SELinux コンテキストを表示させます。

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

/var/www/html/ には **httpd_sys_content_t** タイプのラベルが付けられていることが分かります。

2. root ユーザーになり **touch /var/www/html/file1** を実行して新しいファイルを作成します。

3. **ls -Z /var/www/html/file1** を実行して SELinux コンテキストを表示させます。

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

ls -Z コマンドを使用すると **file1** には **httpd_sys_content_t** タイプのラベルが付けられていることが分かります。 SELinux では、 **httpd** がこのタイプのラベルが付いたファイルを読み込めるよう許可していますが、 書き込みは許可していません。 Linux のパーミッションが書き込みアクセスを許可していても変わりません。 SELinux ポリシーで **httpd_t** ドメイン (**httpd** が実行されるドメイン) で実行しているプロセスが読み取りと書き込みを行なうことができるタイプを定義しています。 これにより、 プロセスが別のプロセス用のファイルにアクセスしてしまわないよう保護しています。

たとえば、**httpd** は **httpd_sys_content_t** タイプ (Apache HTTP Server 用) のラベルが付いたファイルを読み込むことができますが、デフォルトでは **samba_share_t** タイプ (Samba 用) のラベルが付いたファイルにはアクセスできません。また、ユーザーのホームディレクトリにあるファイルには **user_home_t** タイプのラベルが付けられます。これにより、デフォルトで**httpd** がユーザーのホームディレクトリにあるファイルの読み取りや書き込みを行なわないよう保護しています。

httpd で使用されるタイプをいくつか以下に示します。 タイプを使い分けることで柔軟なアクセスを設定することができるようになります。

httpd_sys_content_t

このタイプは、静的な Web サイトで使用される **.html** ファイルなどの Web コンテンツに使用します。このタイプのラベルが付けられたファイルは、**httpd** および **httpd** で実行されるスクリプトに対してアクセス可能となります (読み取り専用)。デフォルトでは、このタイプのラベルが付けられたファイルおよびディレクトリへの **httpd** や他のプロセスによる書き込み、編集は行なえません。デフォルトでは、**/var/www/html/**内に作成またはコピーされたファイルには **httpd_sys_content_t** タイプのラベルが付けられます。

httpd_sys_script_exec_t

このタイプは、 **httpd** に実行させるスクリプトに対して使用します。 一般的には **/var/www/cgi-bin/** 内の CGI (Common Gateway Interface) スクリプトに使用されます。 デフォルトでは、 SELinux ポリシーにより **httpd** による CGI スクリプトの実行は阻止されます。 これを許可するため、 スクリプトに **httpd_sys_script_exec_t** タイプのラベル付けを行い、 **httpd_enable_cgi** Boolean をオンにします。 **httpd_sys_script_exec_t** のラベルが付けられたスクリプトは、 **httpd** で実行されると **httpd_sys_script_t** ドメインで実行されます。 **httpd_sys_script_t** ドメインには、 **postgresql_t** や **mysqld_t** など他のシステムドメインへのアクセスがあります。

httpd_sys_content_rw_t

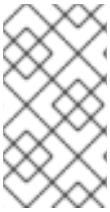
このタイプのラベルを使用したファイルは、 **httpd_sys_script_exec_t** タイプのラベルが付いたスクリプトによる書き込みが可能になります。 ただし、 これ以外のラベルタイプのスクリプトによる編集はできません。 **httpd_sys_script_exec_t** タイプのラベルが付いたスクリプトに読み込みや書き込みを行なわせるファイルには、 **httpd_sys_content_rw_t** タイプのラベルを使用してください。

httpd_sys_content_ra_t

このタイプのラベルを使用したファイルは、 **httpd_sys_script_exec_t** タイプのラベルが付いたスクリプトによる追加が可能になります。 ただし、 これ以外のラベルタイプのスクリプトによる編集はできません。 **httpd_sys_script_exec_t** タイプのラベルが付いたスクリプトに読み込みや追加を行なわせるファイルには、 **httpd_sys_content_ra_t** タイプのラベルを使用してください。

httpd_unconfined_script_exec_t

このタイプのラベルを付けたスクリプトは SELinux の保護なしで実行されます。 このタイプの使用は、 これ以外の手段を試したがいずれもうまくいかない複雑なスクリプトを使用する場合に限ってください。 **httpd** に対して SELinux の保護をオフにする、 またはシステム全体に対して SELinux の保護をオフにするよりは、 このタイプを使用した方がよいでしょう。



注記

httpd に使用できる他のタイプについては次のコマンドを実行すると確認できます。

```
grep httpd /etc/selinux/targeted/contexts/files/file_contexts
```

SELinux のコンテキストを変更する

ファイルやディレクトリのタイプは **chcon** コマンドを使用すると変更することができます。**chcon** を使って行なった変更は、 ファイルシステムの再ラベルや **restorecon** コマンドを実行すると失われます。 特定ファイルの SELinux コンテキストの変更をユーザーに許可するかどうかは SELinux ポリシーで制御します。 以下の例では、 **httpd** 用の **index.html** ファイルと新規ディレクトリを作成し、 そのファイルとディレクトリに **httpd** がアクセスできるようラベル付けを行ないます。

1. root ユーザーで **mkdir -p /my/website** を実行し、 **httpd** によって使用されるファイルを格納するディレクトリを最上位に作成します。

2. ファイルコンテキスト設定のパターンに合致しないファイルやディレクトリには **default_t** タイプのラベルが付けられる場合があります。 制限のあるサービスからはこのタイプのファイルやディレクトリにはアクセスできません。

```
$ ls -dZ /my
```

```
drwxr-xr-x  root root unconfined_u:object_r:default_t:s0 /my
```

3. root ユーザーで **chcon -R -t httpd_sys_content_t /my/** を実行し、**/my/** ディレクトリおよびサブディレクトリのタイプを、**httpd** からアクセスできるタイプに変更します。これで **/my/website/** の配下に作成されるファイルは、 **default_t** タイプではなく **httpd_sys_content_t** タイプを継承するようになるため、 httpd からアクセスできるようになります。

```
# chcon -R -t httpd_sys_content_t /my/
# touch /my/website/index.html
# ls -Z /my/website/index.html
-rw-r--r--  root root unconfined_u:object_r:httpd_sys_content_t:s0
/my/website/index.html
```

一時的な変更について: **chcon** についての詳細を Red Hat Enterprise Linux 6 SELinux ユーザーガイドの「chcon」セクションにて参照してください。

再ラベルや **restorecon** コマンドの実行後もこのラベルの変更を維持するために、**semanage fcontext** コマンド (**semanage** は policycoreutils-python パッケージで提供) を使用します。 このコマンドにより変更がファイルコンテキスト設定に追加されます。このあと、**restorecon** コマンドを実行すると、ファイルコンテキスト設定が読み込まれ、ラベルの変更が適用されます。次の例では、**httpd** に使用させる新規ディレクトリと **index.html** ファイルを作成し、**httpd** によるアクセスを許可するためラベルに永久的な変更を行ないます。

1. root ユーザーで **mkdir -p /my/website** を実行し、 **httpd** によって使用されるファイルを格納するディレクトリを最上位に作成します。

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

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

"/my(/.*)?" の式は、 **/my/** ディレクトリおよび配下の全ファイルとディレクトリにラベルの変更が適用されるという意味です。

3. root ユーザーで **touch /my/website/index.html** を実行し新しいファイルを作成します。

4. root ユーザーで **restorecon -R -v /my/** を実行しラベルの変更を適用します (ステップ 2 の **semanage** コマンドで変更されたファイルコンテキスト設定が**restorecon** により読み込まる)。

```
# restorecon -R -v /my/
restorecon reset /my context unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
restorecon reset /my/website context
unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
restorecon reset /my/website/index.html context
unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
```

一時的な変更について: **semanage** についての詳細を Red Hat Enterprise Linux SELinux ユーザーガイドの「semanage fcontext」セクションにて参照してください。

3.3. BOOLEAN

SELinux は実行するサービスに必要な最小限レベルのアクセスに基づいています。サービスの実行手段は複数あるため、サービスをどのように実行するのかを SELinux に指示する必要があります。 これを行なうため、 Boolean を使用します。 Boolean を使用すると、 SELinux ポリシーの記述方法などの知識が全くなくてもランタイム時の SELinux ポリシーの一部変更を行なうことができます。 SELinux ポリシーの再読み込みや再コンパイルを行なうことなく、サービスによる NFS ファイルシステムへのアクセスを許可するなどの変更を行なうことができるようになります。

Boolean の状態を変更するには、 **setsebool** コマンドを使用します。 たとえば、 **allow_httpd_anon_write** Boolean をオンにする場合は次のコマンドを root ユーザーで実行します。

```
# setsebool -P allow_httpd_anon_write on
```

同じ例を使って Boolean をオフにする場合は、 コマンドの **on** を **off** にします。 以下にそのコマンドを示します。

```
# setsebool -P allow_httpd_anon_write off
```

注記

再起動後、 **setsebool** による変更を維持したくない場合は **-P** オプションを使用しないでください。

httpd の動作方法を指定する一般的な Boolean について以下に説明します。

allow_httpd_anon_write

この Boolean を無効にした場合、**httpd** による **public_content_rw_t** タイプのラベルが付いたファイルへのアクセスを読み取り専用に限定します。有効にすると、パブリックファイル転送サービス用のファイルを含むパブリックディレクトリなど **public_content_rw_t** タイプのラベルが付いたファイルへの書き込みを許可するようになります。

allow_httpd_mod_auth_ntlm_winbind

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

allow_httpd_mod_auth_pam

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

allow_httpd_sys_script_anon_write

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

httpd_builtin_scripting

httpd スクリプト機能へのアクセスを定義する Boolean です。 PHP コンテンツの場合、 この Boolean の有効化が必要とされることが多くあります。

httpd_can_network_connect

この Boolean を無効にすると、 HTTP スクリプトやモジュールによるネットワークやリモートポートへの接続の開始が阻止されます。 接続の開始を許可する場合は Boolean をオンにします。

httpd_can_network_connect_db

この Boolean を無効にすると、 HTTP スクリプトやモジュールによるデータベースサーバーへの接続の開始が阻止されます。 接続の開始を許可する場合は Boolean をオンにします。

httpd_can_network_relay

httpd をフォワードプロキシまたはリバースプロキシとして使用する場合、 この Boolean をオンにします。

httpd_can_sendmail

この Boolean を無効にすると、 HTTP モジュールによるメール送信が阻止されます。 **httpd** に脆弱性が見つかった場合にスパム攻撃を阻止することができます。 HTTP モジュールにメールの送信を許可する場合は、 この Boolean をオンにします。

httpd_dbus_avahi

この Boolean をオフにすると、**httpd** による **D-Bus** 経由の **avahi** サービスへのアクセスが拒否されます。 このアクセスを許可する場合は、 この Boolean をオンにします。

httpd_enable_cgi

この Boolean を無効にすると、 **httpd** による CGI スクリプトの実行が阻止されます。**httpd** に CGI スクリプトの実行を許可する場合は、 この Boolean をオンにします (CGI スクリプトには **httpd_sys_script_exec_t** タイプのラベルを付けておく必要があります)。

httpd_enable_ftp_server

この Boolean をオンにすると、 **httpd** が FTP ポートでリッスンできるようになるため、 FTPサーバーとして動作できるようになります。

httpd_enable_homedirs

この Boolean を無効にすると、 **httpd** によるユーザーのホームディレクトリへのアクセスが阻止されます。 ユーザーのホームディレクトリ (**/home/*/** 内のコンテンツ) へのアクセスを許可する場合は、 この Boolean をオンにします。

httpd_execmem

この Boolean を有効にすると、 実行可能かつ書き込み可能なメモリーアドレスを必要とするプログラムの実行を **httpd** に許可します。 バッファのオーバーフローに対する保護が低下するため、 安全上、 この Boolean の有効化はお勧めできません。 ただし、 特定のモジュールやアプリケーションではこの特権を必要とするものがあります (Java や Mono アプリケーションなど)。

httpd_ssi_exec

Web ページ内の SSI (server side include) エレメントを実行可能にするかどうかを指定する Boolean です。

httpd_tty_comm

httpd による制御ターミナルへのアクセスを許可するかどうかを指定する Boolean です。 通常、このアクセスは必要とされませんが、 SSL 証明書ファイルを設定する場合などに、 パスワードのプロンプトを表示させ処理するためターミナルへのアクセスが必要になります。

httpd_unified

この Boolean を有効にすると、 **httpd_t** に対して **httpd** の全タイプへの完全アクセスが許可されます (つまり、 sys_content_t の実行、 読み込み、 書き込み)。 無効にすると、 読み取り専用 web コンテンツ、 書き込み可能 web コンテンツ、 実行可能 web コンテンツ間に分離が施行されます。 この Boolean を無効にすることで安全性は高くなりますが、 各ファイルに持たせるアクセス権に応じてスクリプトや他の web コンテンツを別々にラベル付けしなければならない管理作業コストが生じることになります。

httpd_use_cifs

Samba でマウントされるファイルシステムなど、 **cifs_t** タイプのラベルが付けられる CIFS ファイルシステム上にあるファイル群へのアクセスを **httpd** に許可する場合は、 この Boolean をオンにします。

httpd_use_nfs

NFS でマウントされるファイルシステムなど、 **nfs_t** タイプのラベルが付けられる NFS ファイルシステム上にあるファイル群へのアクセスを **httpd** に許可する場合は、 この Boolean をオンにします。

3.4. 設定の実例

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

3.4.1. 静的なサイトを実行する

静的な web サイトを作成する場合は、 その web サイトの **.html** ファイルに **httpd_sys_content_t** タイプのラベルを付けます。 デフォルトでは、**httpd_sys_content_t** タイプのラベルが付いたファイルには Apache HTTP Server による書き込みは行なえません。 次のように、読み取り専用 web サイト向けのファイルを格納する新しいディレクトリを作成します。

1. root ユーザーで **mkdir /mywebsite** を実行し最上位にディレクトリを作成します。

2. root ユーザーで **/mywebsite/index.html** ファイルを作成します。 以下のコンテンツを **/mywebsite/index.html** にコピーして貼り付けます。

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

3. **/mywebsite/** および配下のファイルやサブディレクトリへの Apache HTTP Server 読み取り専用アクセスを許可するため、**/mywebsite/** に **httpd_sys_content_t** タイプのラベルを付けます。**root** ユーザーで次のコマンドを実行してラベルの変更をファイルコンテキスト設定に追加します。

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

4. root ユーザーで **restorecon -R -v /mywebsite** を実行してラベルの変更を行ないます。

```
# restorecon -R -v /mywebsite
restorecon reset /mywebsite context
unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
restorecon reset /mywebsite/index.html context
unconfined_u:object_r:default_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
```

5. この例の場合、 root ユーザーで **/etc/httpd/conf/httpd.conf** を編集します。 既存の **DocumentRoot** オプションをコメントアウトします。**DocumentRoot "/mywebsite"** オプションを追加します。 編集結果は以下のようになるはずです。

```
#DocumentRoot "/var/www/html"
DocumentRoot "/mywebsite"
```

6. root ユーザーで **service httpd status** を実行し Apache HTTP Server の状態を確認します。 サーバーが停止している場合は、 root ユーザーで **service httpd start** を実行してサーバーを起動します。 サーバーが実行中の場合は、 root ユーザーで **service httpd restart** を実行しサーバーの再起動を行ないます (これにより **httpd.conf** への変更がすべて適用されます)。

7. web ブラウザで **http://localhost/index.html** に行きます。 次のように表示されます。

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

3.4.2. NFS と CIFS ファイルシステムを共有する

デフォルトでは、クライアント側の NFS マウントには 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がラベル付けされます。一般的なポリシーであれば、このデフォルトのコンテキストには **nfs_t** タイプが使用されます。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 共有にはポリシーで定義されたデフォルトのコンテキストがラベル付けされます。一般的なポリシーであれば、このデフォルトのコンテキストには **cifs_t** タイプが使用されます。

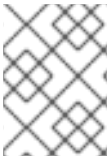
ポリシー設定によっては **nfs_t** または **cifs_t** タイプのラベルが付いたファイルはサービスからは読み取れない場合があります。 このため、このタイプのラベルが付いたファイルシステムは他のサービスによるマウント、読み込み、エクスポートなどが阻止される可能性があります。Boolean をオンまたはオフにすることで、**nfs_t** や **cifs_t** タイプへのアクセスを許可するサービスを管理することができるようになります。

httpd_use_nfs Boolean をオンにして、 **httpd** による NFS ファイルシステム (**nfs_t** タイプのラベルが付いたファイル) へのアクセスと共有を許可します。 root ユーザーで **setsebool** を実行して Boolean をオンにします。

```
setsebool -P httpd_use_nfs on
```

httpd_use_cifs Boolean をオンにして、 **httpd** による CIFS ファイルシステム (**cifs_t** タイプのラベルが付いたファイル) へのアクセスと共有を許可します。 root ユーザーで **setsebool** コマンドを実行します。

```
setsebool -P httpd_use_cifs on
```



注記

再起動後、 **setsebool** による変更を維持したくない場合は **-P** オプションを使用しないでください。

3.4.3. サービス間でファイルを共有する

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

以下の例では、 ディレクトリとファイルを作成し、 Apache HTTP Server、 FTP、 rsync、 Samba 経由でそのディレクトリファイルを共有 (読み取り専用) できるようにしています。

1. root ユーザーで **mkdir /shares** を実行して、 複数のサービス間でファイルを共有できるよう最上位に新規のディレクトリを作成します。

2. ファイルコンテキスト設定のパターンに合致しないファイルやディレクトリには **default_t** タイプのラベルが付けられる場合があります。 制限のあるサービスからはこのタイプのファイルやディレクトリにはアクセスできません。

```
$ ls -dZ /shares
drwxr-xr-x  root root unconfined_u:object_r:default_t:s0 /shares
```

3. root ユーザーで **/shares/index.html** ファイルを作成します。 次のコンテンツをコピーして **/shares/index.html** に貼り付けます。

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

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

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

5. root ユーザーで **restorecon -R -v /shares/** を実行しラベルの変更を適用します。

```
# restorecon -R -v /shares/
restorecon reset /shares context unconfined_u:object_r:default_t:s0-
>system_u:object_r:public_content_t:s0
restorecon reset /shares/index.html context
unconfined_u:object_r:default_t:s0-
>system_u:object_r:public_content_t:s0
```

Samba で **/shares/** を共有する場合

1. **rpm -q samba samba-common samba-client** を実行して、 samba、 samba-common、 samba-client の各パッケージがインストールされているか確認します (バージョン番号は使用しているバージョンによって異なります)。

```
$ rpm -q samba samba-common samba-client
samba-3.4.0-0.41.el6.3.i686
samba-common-3.4.0-0.41.el6.3.i686
samba-client-3.4.0-0.41.el6.3.i686
```

   上記いずれのパッケージもインストールされていない場合は、 root ユーザーで **yum install *package-name*** を実行してインストールを行ないます。

2. root ユーザーで **/etc/samba/smb.conf** を編集します。 Samba で **/shares/** ディレクトリを共有するため、 次のエントリをこのファイルの末尾に追加します。

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

3. Samba ファイルシステムをマウントするには Samba アカウントが必要になります。 root ユーザーで **smbpasswd -a *username*** を実行し、 Samba アカウントを作成します。 *username* の部分は既存の Linux ユーザーにします。 たとえば、 **smbpasswd -a testuser** を実行すると、 Linux の testuser ユーザー用の Samba アカウントが作成されます。

```
# smbpasswd -a testuser
New SMB password: Enter a password
Retype new SMB password: Enter the same password again
Added user testuser.
```

   smbpasswd -a *username* を実行する際、 システムに存在しない Linux アカウントのユーザー名を *username* に使用すると、「**Cannot locate Unix account for '*username*'!**」というエラーが発生する原因になります。

4. root ユーザーで **service smb start** を実行し Samba サービスを起動します。

```
service smb start
Starting SMB services:                                     [  OK
]
```

5. **smbclient -U *username* -L localhost** を実行し、 使用できる共有を表示させます。 *username* はステップ 3 で追加した Samba アカウントにします。 パスワード入力を求められたら、 ステップ 3 で Samba アカウントに割り当てたパスワードを入力します (バージョン番号は使用しているバージョンによって異なります)。

```
$ smbclient -U username -L localhost
Enter username's password:
Domain=[HOSTNAME] OS=[Unix] Server=[Samba 3.4.0-0.41.el6]

Sharename       Type      Comment
---------       ----      -------
shares          Disk      Documents for Apache HTTP Server, FTP,
```

```
rsync, and Samba
IPC$            IPC       IPC Service (Samba Server Version 3.4.0-
0.41.el6)
username        Disk      Home Directories
Domain=[HOSTNAME] OS=[Unix] Server=[Samba 3.4.0-0.41.el6]

Server               Comment
---------            -------

Workgroup            Master
---------            -------
```

6. root ユーザーで **mkdir /test/** を実行し新規ディレクトリを作成します。 このディレクトリは Samba 共有の **shares**をマウントする際に使用します。

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

```
mount //localhost/shares /test/ -o user=username
```

   ステップ 3 で設定した *username* のパスワードを入力します。

8. **cat /test/index.html** を実行して Samba で共有しているファイルを表示させます。

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

Apache HTTP Server で **/shares/** を共有する場合

1. **rpm -q httpd** を実行して httpd パッケージがインストールされているか確認します (バージョン番号は使用しているバージョンによって異なります)。

```
$ rpm -q httpd
httpd-2.2.11-6.i386
```

   このパッケージがインストールされていない場合は、 root ユーザーで **yum install httpd** を実行してインストールします。

2. **/var/www/html/** ディレクトリに移動します。 root ユーザーで次のコマンドを実行して **/shares/** ディレクトリへのリンクを作成します。

```
ln -s /shares/ shares
```

3. root ユーザーで **service httpd start** を実行して Apache HTTP Server を起動します。

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

4. web ブラウザを使って **http://localhost/shares** に行きます。 **/shares/index.html** が表示されます。

デフォルトでは、**index.html** ファイルが存在していればそのファイルが Apache HTTP Server によって読み込まれます。**/shares/** に **file1**、**file2**、**file3** しかなく **index.html** がない場合、**http://localhost/shares** にアクセスするとディレクトリー覧が表示されます。

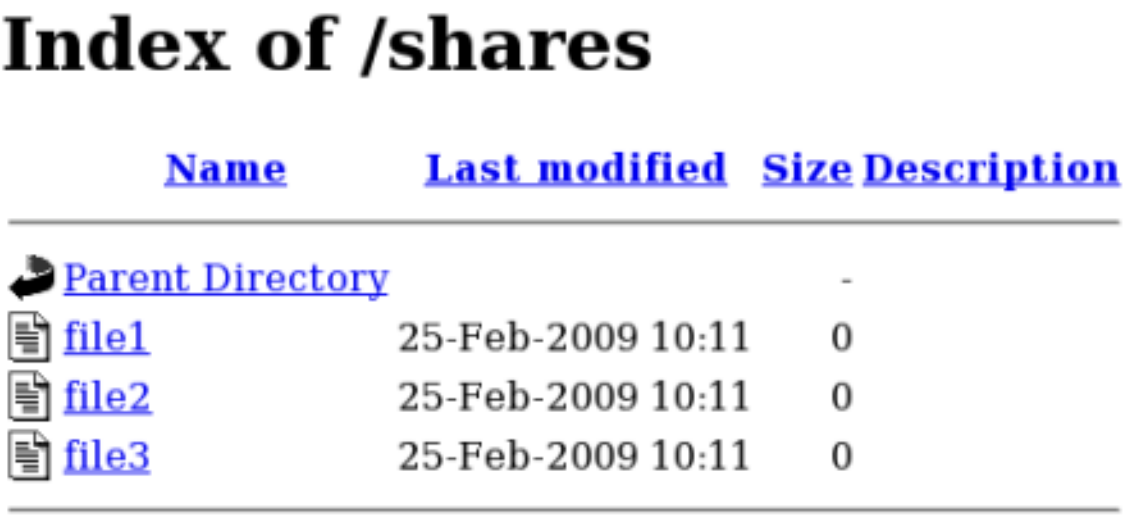
1. root ユーザーで **rm -i /shares/index.html** を実行して **index.html** ファイルを削除します。

2. root ユーザーで **touch /shares/file{1,2,3}** を実行し、 **/shares/** 内に 3 つのファイルを作成します。

```
# touch /shares/file{1,2,3}
# ls -Z /shares/
-rw-r--r--  root root system_u:object_r:public_content_t:s0 file1
-rw-r--r--  root root unconfined_u:object_r:public_content_t:s0
file2
-rw-r--r--  root root unconfined_u:object_r:public_content_t:s0
file3
```

3. root ユーザーで **service httpd status** を実行して Apache HTTP Server の状態を確認します。 サーバーが停止している場合は、 root ユーザーで **service httpd start** を実行し再起動を行ないます。

4. web ブラウザで **http://localhost/shares** に行きます。 ディレクトリー覧が表示されます。



3.4.4. ポート番号を変更する

ポリシー設定に応じて、サービスの実行が許可されるのは特定のポート番号に限られます。ポリシーを変更せず、サービスが実行されるポートを変えようとすると、サービスの起動に失敗する場合があります。root ユーザーで **semanage port -l | grep -w "http_port_t"** コマンドを実行し、SELinux で **http** によるリッスンを許可しているポートを表示させます。

```
# semanage port -l | grep -w http_port_t
http_port_t                    tcp      80, 443, 488, 8008, 8009, 8443
```

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

次のように、TCP ポート 80、443、488、8008、8009、8443 以外のポートで実行するよう **httpd** に設定を行ないます。

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

```
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses (0.0.0.0)
#
#Listen 12.34.56.78:80
Listen 10.0.0.1:12345
```

2. root ユーザーで **semanage port -a -t http_port_t -p tcp 12345**を実行して、 そのポートを SELinux ポリシー設定に追加します。

3. root ユーザーで **semanage port -l | grep -w http_port_t** を実行してポートが追加されたか確認します。

```
# semanage port -l | grep -w http_port_t
http_port_t                    tcp      12345, 80, 443, 488, 8008,
8009, 8443
```

ポート 12345 での **httpd** の実行が必要なくなった場合には、 root ユーザーで **semanage port -d -t http_port_t -p tcp 12345** を実行してポリシー設定からそのポートを削除します。

---

[5] Apache HTTP Server Project ページの "The Number One HTTP Server On The Internet" セクションより抜粋: http://httpd.apache.org/。 著作権 © 2009 The Apache Software Foundation (2010年 7月 7日に閲覧)

第4章 SAMBA

Samba の web サイトより抜粋:

原文: "Samba is an Open Source/Free Software suite that has, since 1992, provided file and print services to all manner of SMB/CIFS clients, including the numerous versions of Microsoft Windows operating systems. Samba is freely available under the GNU General Public License.". (訳文: Samba とは、 1992 年に設立され、 Microsoft Windows オペレーティングシステムの各種バージョンをはじめ、あらゆる SMB/CIFS クライアントにファイルやプリンタなどのサービスを提供している Open Source/Free Software スィートです。 Samba は GNU General Public License に基づき自由に入手が可能です。)[6]

Red Hat Enterprise Linux では、 Samba サーバーは samba パッケージにより提供されます。 **rpm -q samba** を実行して samba パッケージがインストールされているか確認します。Samba を使用する予定にも関わらず、このパッケージがインストールされていない場合は、root ユーザーになり次のコマンドを実行してパッケージのインストールを行ないます。

```
yum install samba
```

4.1. SAMBA と SELINUX

SELinux を有効にすると、 Samba サーバー (**smbd**) はデフォルトで制限のあるサービスとして実行されます。 制限のあるサービスはそのサービス自体のドメイン内で実行され、 他の制限のあるサービスとは分離されます。 次の例では、 サービス自体のドメイン内で実行している **smbd** プロセスを示しています。 samba パッケージがインストールされていることを前提としています。

1. **getenforce** を実行して、 SELinux が enforcing モードで実行していることを確認します。

   ```
   $ getenforce
   Enforcing
   ```

   SELinux が enforcing モードで実行されている場合は、 **getenforce** コマンドにより **Enforcing** が返されます。

2. root ユーザーで **service smbd start** を実行して **smbd** を起動します。

   ```
   service smb start
   Starting SMB services:                                     [  OK
   ]
   ```

3. **ps -eZ | grep smb** 実行して **smbd** プロセスを表示させます。

   ```
   $ ps -eZ | grep smb
   unconfined_u:system_r:smbd_t:s0 16420 ?        00:00:00 smbd
   unconfined_u:system_r:smbd_t:s0 16422 ?        00:00:00 smbd
   ```

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

smbd によるファイルのアクセスおよび共有を許可するため、適切なラベルを付ける必要があります。

たとえば、**smbd** では **samba_share_t** タイプのラベルが付いたファイルの読み込みと書き込みを行なうことができますが、デフォルトでは **httpd_sys_content_t** タイプのラベルが付いたファイルにはアクセスできません。このタイプは Apache HTTP Server での使用を目的としているためです。Samba によるホームディレクトリや NFS ファイルシステムのエクスポートを許可する、また Samba にドメインコントローラとしての動作を許可するなど、特定の動作を許可する場合は Boolean をオンにする必要があります。

4.2. タイプ

ファイルに **samba_share_t** タイプのラベルを付けて Samba によるファイル共有ができるようにします。 このタイプのラベル付けはユーザー作成のファイルに限ってください。システムファイルにはこのタイプのラベルは付けないよう注意してください。ラベル付けしたファイルやディレクトリを共有するため Boolean をオンにします。SELinux では、**/etc/samba/smb.conf** と Linux パーミッションが設定されていれば、**samba_share_t** タイプのラベルが付いたファイルへの Samba による書き込みは許可されます。

samba_etc_t タイプは、 **/etc/samba/** 内にある**smb.conf** などの特定ファイルに使用されます。**samba_etc_t** タイプのラベル付けは手作業では行なわないでください。**/etc/samba/** 内のファイルに適切なラベルが付けられていない場合、 root ユーザーで **restorecon -R -v /etc/samba** を実行してそのファイルをデフォルトのコンテキストに復元します。 **/etc/samba/smb.conf** に **samba_etc_t** タイプのラベルが付いていない場合、**service smb start** コマンドの実行が失敗し、 SELinux 拒否がログ記録される場合があります。 **/etc/samba/smb.conf** に **httpd_sys_content_t** タイプのラベルが付いていた場合に拒否となる例を示します。

```
setroubleshoot: SELinux is preventing smbd (smbd_t) "read" to ./smb.conf
(httpd_sys_content_t). For complete SELinux messages. run sealert -l
deb33473-1069-482b-bb50-e4cd05ab18af
```

4.3. BOOLEAN

SELinux は実行するサービスに必要な最小限レベルのアクセスに基づいています。サービスの実行手段は複数あるため、サービスをどのように実行するのかを SELinux に指示する必要があります。 次の Boolean を使って Samba の動作方法を SELinux に指示します。

allow_smbd_anon_write

この Boolean を有効にすると、 特殊なアクセス制限がなく共通ファイル用に予約されている領域など、 パブリックディレクトリへの **smbd** による書き込みが許可されます。

samba_create_home_dirs

この Boolean を有効にすると、 Samba が単独で新規ホームディレクトリを作成できるようになります。 PAM などのようなメカニズムで有効にされることが多くあります。

samba_domain_controller

この Boolean を有効にすると、 ドメインコントローラとして Samba を動作させ、 **useradd**、**groupadd**、 **passwd** などの関連コマンドの実行パーミッションを付与することができます。

samba_enable_home_dirs

この Boolean を有効にすると、 Samba によるユーザーのホームディレクトリ共有が可能になります。

samba_export_all_ro

あらゆるファイルやディレクトリをエクスポートし、読み取り専用のパーミッションを付与します。これにより、**samba_share_t** タイプのラベルが付いていないファイルやディレクトリを Samba で共有できるようにします。**samba_export_all_ro** Boolean はオンになっているが、**samba_export_all_rw** Boolean がオフになっている場合、**/etc/samba/smb.conf** で書き込みアクセスが設定され Linux パーミッションでも書き込みアクセスが許可されていても、Samba 共有への書き込みアクセスは拒否されます。

samba_export_all_rw

あらゆるファイルやディレクトリをエクスポートし、読み取りと書き込みのパーミッションを付与します。これにより、**samba_share_t** タイプのラベルが付いていないファイルやディレクトリを Samba でエクスポートできるようにします。**/etc/samba/smb.conf** のパーミッションおよび Linux パーミッションを設定して書き込みアクセスを許可する必要があります。

samba_run_unconfined

この Boolean を有効にすると、Samba による /var/lib/samba/scripts ディレクトリ内の制限のないスクリプトの実行を許可します。

samba_share_fusefs

Samba に fusefs ファイルシステムを共有させる場合は、この Boolean を有効にする必要があります。

samba_share_nfs

この Boolean を無効にすることで、Samba を介した NFS 共有への完全アクセスを **smbd** に与えないようにします。この Boolean を有効にすると、Samba による NFS ファイルシステムの共有が許可されます。

use_samba_home_dirs

Samba のホームディレクトリ用にリモートサーバーを使用する場合、この Boolean を有効にします。

virt_use_samba

仮想マシンによる CIFS ファイルへのアクセスを許可します。

4.4. 設定例

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

4.4.1. 作成したディレクトリを共有する

新規のディレクトリを作成し、そのディレクトリを Samba で共有します。

1. **rpm -q samba samba-common samba-client** を実行して、samba、samba-common、samba-client の各パッケージがインストールされているか確認します。いずれかのパッケージがインストールされていない場合は、root ユーザーで **yum install *package-name*** を実行してインストールを行ないます。

2. root ユーザーで **mkdir /myshare** を実行して Samba でファイルを共有するためのディレクトリを最上位に新規に作成します。

3. root ユーザーで **touch /myshare/file1** を実行して空のファイルを作成します。 このファイルは後で Samba 共有が正しくマウントされたか確認する際に使用します。

4. SELinux では、**/etc/samba/smb.conf** および Linux パーミッションが設定されていれば、**samba_share_t** タイプのラベルが付いたファイルへの Samba による読み取りおよび書き込みは許可されます。root ユーザーで次のコマンドを実行し、ファイルコンテキスト設定にラベルの変更を追加します。

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

5. root ユーザーで **restorecon -R -v /myshare** を実行しラベルの変更を適用します。

```
# restorecon -R -v /myshare
restorecon reset /myshare context
unconfined_u:object_r:default_t:s0-
>system_u:object_r:samba_share_t:s0
restorecon reset /myshare/file1 context
unconfined_u:object_r:default_t:s0-
>system_u:object_r:samba_share_t:s0
```

6. root ユーザーで **/etc/samba/smb.conf** を編集します。 Samba で /myshare/ ディレクトリを共有するため、 以下をこのファイルの末尾に追加します。

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

7. Samba ファイルシステムをマウントするには Samba アカウントが必要になります。 root ユーザーで **smbpasswd -a *username*** を実行し Samba アカウントを作成します。 *username* は既存の Linux ユーザーにします。 たとえば、 **smbpasswd -a testuser** にすると、 Linux testuser ユーザーの Samba アカウントが作成されます。

```
# smbpasswd -a testuser
New SMB password: (パスワードを入力)
Retype new SMB password: (もう一度同じパスワードを入力)
Added user testuser.
```

smbpasswd -a *username* を実行する際、 システム上に存在していない Linux アカウントをのユーザー名を *username* に入れると、「**Cannot locate Unix account for '*username*'!**」エラーが発生する原因になります。

8. root ユーザーで **service smb start** を実行して Samba サービスを起動します。

```
service smb start
Starting SMB services:                                     [  OK
]
```

9. **smbclient -U *username* -L localhost** を実行し、 使用できる共有を表示させます。 *username* はステップ **7** で追加した Samba アカウントにします。 パスワード入力を求められたら、 ステップ **7** で Samba アカウントに割り当てたパスワードを入力します (バージョン番号は使用しているバージョンによって異なります)。

```
$ smbclient -U username -L localhost
Enter username's password:
Domain=[HOSTNAME] OS=[Unix] Server=[Samba 3.4.0-0.41.el6]

Sharename       Type      Comment
---------       ----      -------
myshare         Disk      My share
IPC$            IPC       IPC Service (Samba Server Version 3.4.0-
0.41.el6)
username        Disk      Home Directories
Domain=[HOSTNAME] OS=[Unix] Server=[Samba 3.4.0-0.41.el6]

Server               Comment
---------            -------

Workgroup            Master
---------            -------
```

10. root ユーザーで **mkdir /test/** を実行し新規ディレクトリを作成します。 このディレクトリは Samba 共有の **myshare** をマウントする際に使用します。

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

    ```
    mount //localhost/myshare /test/ -o user=username
    ```

    ステップ **7** で設定した *username* のパスワードを入力します。

12. **ls /test/** を実行して ステップ **3** で作成した **file1** を表示させます。

    ```
    $ ls /test/
    file1
    ```

4.4.2. web サイトを共有する

/var/www/html/ で web サイトを共有したい場合など、ファイルに **samba_share_t** タイプのラベルが付けられない場合があります。 このような場合には、 すべてのファイルやディレクトリを共有するよう **samba_export_all_ro** Boolean を使用して読み取り専用パーミッションを付与するか (現在のラベルに関わらず)、 **samba_export_all_rw** Boolean を使用して読み取りと書き込みのパーミッションを付与します (現在のラベルに関わらず)。

以下の例では、 **/var/www/html/** 内に web サイトのファイルを作成してから、 そのファイルを Samba で共有し読み取りと書き込みのパーミッションを与えています。 ここでは、 **httpd**、 **samba**、 **samba-common**、 **samba-client**、 **wget** のパッケージがインストールされていることを前提としています。

1. root ユーザーになり **/var/www/html/file1.html** ファイルを作成します。 次のコンテンツをコピーして **/var/www/html/file1.html** に貼り付けます。

   ```
   <html>
   <h2>File being shared through the Apache HTTP Server and Samba.</h2>
   </html>
   ```

2. **ls -Z /var/www/html/file1.html** を実行して **file1.html** の SELinux コンテキストを表示させます。

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

file1.index.html には **httpd_sys_content_t** タイプのラベルが付けられています。デフォルトでは、Apache HTTP Server によるアクセスはできますが、 Samba によるアクセスはできません。

3. root ユーザーで **service httpd start** を実行して Apache HTTP Server を起動します。

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

4. Linux ユーザーでの書き込みアクセスがあるディレクトリに移動し、 **wget http://localhost/file1.html** コマンドを実行します。 デフォルト設定に変更がなければ、 このコマンドは成功します。

```
$ wget http://localhost/file1.html
Resolving localhost... 127.0.0.1
Connecting to localhost|127.0.0.1|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 84 [text/html]
Saving to: `file1.html.1'

100%[======================>] 84          --.-K/s   in 0s

`file1.html.1' saved [84/84]
```

5. root ユーザーで **/etc/samba/smb.conf** を編集します。 Samba で **/var/www/html/** ディレクトリを共有するため、 以下をこのファイルの末尾に追加します。

```
[website]
comment = Sharing a website
path = /var/www/html/
public = no
writeable = no
```

6. **/var/www/html/** ディレクトリには **httpd_sys_content_t** タイプのラベルが付けられます。 デフォルトでは、 **httpd_sys_content_t** タイプのラベルの付いたファイルやディレクトリには、 Linux パーミッションが付与されていても Samba からはアクセスできません。 Samba によるアクセスを許可するため、 root ユーザーで次のコマンドを実行し、 **samba_export_all_ro** Boolean をオンにします。

```
setsebool -P samba_export_all_ro on
```

再起動後、この変更を維持したくない場合は **-P** を使用しないでください。 **samba_export_all_ro** Boolean をオンにすると、Samba からはいずれのタイプにもアクセスもできるようになるため注意してください。

7. root ユーザーで **service smb start** を実行して **smbd** を起動します。

```
service smb start
Starting SMB services:                                     [  OK
]
```

[6] Samba web サイトの冒頭部分より抜粋: http://samba.org (2009年1月 20日に閲覧)

第5章 ファイル転送プロトコル

ファイル転送プロトコル、 FTP は、 今日インターネット上で見られる、最も古く、一般的に使用されているプロトコルです。ユーザーはリモートホストに直接ログインする必要がなく、またリモートシステムの使い方を知らなくても、ネットワーク上の複数のコンピュータホスト間で確実にファイル転送を行なえるようにすることがその目的になります。FTP により、ユーザーはシンプルで標準的なコマンドセットを使用するだけでリモートシステム上のファイルにアクセスできるようになります。

Very Secure FTP Daemon (**vsftpd**) は、高速で安定性があり、かつ何よりも高い安全性を確保するため、その土台から設計が行なわれています。多数の接続を効率的かつ安全に処理できる能力があることから、Red Hat Enterprise Linux に同梱されている唯一の独立型 FTP サーバーとなります。

Red Hat Enterprise Linux では、Very Secure FTP デーモンは vsftpd パッケージで提供されます。 **rpm -q vsftpd** を実行して vsftpd がインストールされているか確認します。

```
$ rpm -q vsftpd
```

FTP サーバーを利用する方で vsftpd パッケージがインストールされていない場合には、 次のコマンドを root ユーザーで実行してインストールを行ないます。

```
yum install vsftpd
```

5.1. FTP と SELINUX

FTP デーモンの **vsftpd** はデフォルトで制限のあるサービスとして実行されます。**vsftpd** とファイル、 プロセスまたシステムとの交信方法は SELinux ポリシーで定義されます。たとえば、 認証ユーザーが FTP 経由でログインすると、そのユーザーは自分のホームディレクトリの読み取りや書き込みが行なえません。SELinux により、**vsftpd** によるユーザーのホームディレクトリへのアクセスはデフォルトで阻止されるためです。また、**vsftpd** には NFS や CIFS ファイルシステムへのアクセスもデフォルトではありません。 このため、**/etc/vsftpd/vsftpd.conf** 内で書き込みアクセスが設定されていても、anonymous ユーザーには書き込みアクセスが与えられません。Boolean をオンにすると、こうしたアクセスを許可することができるようになります。

認証ユーザーがログインして、自分のホームディレクトリを表示させようとすると SeLinux 拒否になる例を以下に示します。

1. **rpm -q ftp** を実行して ftp パッケージがインストールされているか確認します。 インストールされていない場合は root ユーザーで **yum install ftp** を実行しインストールを行ないます。

2. **rpm -q vsftpd** を実行して vsftpd パッケージがインストールされているか確認します。 インストールされていない場合は root ユーザーで **yum install vsftpd** を実行しインストールを行ないます。

3. Red Hat Enterprise Linux では、 **vsftpd** はデフォルトでは anonymous ユーザーによるログインしか許可していません。認証ユーザーによるログインを許可するため、root で **/etc/vsftpd/vsftpd.conf** を編集します。 **local_enable=YES** オプションを必ずコメントアウトしてください。

   ```
   # Uncomment this to allow local users to log in.
   local_enable=YES
   ```

4. root ユーザーで **service vsftpd start** を実行して **vsftpd** を起動します。 **vsftpd.conf** の編集前にこのサービスが実行中だった場合は、 root ユーザーで **service vsftpd restart** を実行し設定の変更を適用します。

```
service vsftpd start
Starting vsftpd for vsftpd:                                [  OK
]
```

5. 現在ログインしているユーザーのまま **ftp localhost** を実行します。 ユーザー名のプロンプトが表示されたら、 ログインしているユーザー名になっているか確認します。 ユーザー名が正しければ **Enter** を押します。 ユーザー名が違う場合は、 正しいユーザー名を入力します。

```
$ ftp localhost
Connected to localhost (127.0.0.1).
220 (vsFTPd 2.1.0)
Name (localhost:username):
331 Please specify the password.
Password: Enter your password
500 OOPS: cannot change directory:/home/username
Login failed.
ftp>
```

6. 次のような SELinux 拒否がログ記録されます。

```
setroubleshoot: SELinux is preventing the ftp daemon from reading
users home directories (username). For complete SELinux messages.
run sealert -l c366e889-2553-4c16-b73f-92f36a1730ce
```

7. ホームディレクトリへのアクセスが SELinux により拒否されています。 これを解決するには、 **ftp_home_dir** Boolean を有効にします。 root ユーザーで次のコマンドを実行して **ftp_home_dir** Boolean を有効にします。

```
# setsebool -P ftp_home_dir=1
```

注記

再起動後、 変更を維持したくない場合は **-P** オプションを使用しないでください。

もう一度ログインしてみます。 今度は **ftp_home_dir** Boolean でホームディレクトリへのアクセスが許可され、 ログインに成功します。

5.2. タイプ

デフォルトでは、 anonymous ユーザーは FTP でログインすると **/var/ftp/** 内のファイルへの読み取りアクセスが与えられます。 このディレクトリには **public_content_t** タイプのラベルが付いているため、 **/etc/vsftpd/vsftpd.conf** で書き込みが設定されていても、 許可されるのは読み取り専用アクセスのみになります。 **public_content_t** タイプへは、 Apache HTTP Server、 Samba、 NFS など他のサービスによるアクセスも可能です。

FTP 経由でファイルを共有する場合は、 次のいずれかのタイプを使用します。

public_content_t

ユーザーが作成したファイルやディレクトリを vsftpd 経由の読み取り専用で共有する場合に **public_content_t** タイプのラベルを付けます。このタイプのラベルが付いたファイルには、Apache HTTP Server、Samba、NFS など、他のサービスからもアクセスすることができます。**public_content_t** タイプのラベルが付いたファイルへの書き込みは、Linux パーミッションで書き込みが許可されていても行なえません。書き込みアクセスが必要な場合は、**public_content_rw_t** タイプを使用してください。

public_content_rw_t

ユーザーが作成したファイルやディレクトリを **vsftpd** 経由の読み取りおよび書き込みのパーミッションで共有する場合に **public_content_rw_t** タイプのラベルを付けます。このタイプのラベルが付いたファイルには、Apache HTTP Server、Samba、NFS など、他のサービスからもアクセスすることができます。このタイプのラベルが付いたファイルに書き込みを行う場合は、まず最初に各サービスの Boolean をオンにしておく必要がある点に注意してください。

5.3. BOOLEAN

SELinux は実行するサービスに必要な最小限レベルのアクセスに基づいています。サービスの実行手段は複数あるため、サービスをどのように実行するのかを SELinux に指示する必要があります。 次の Boolean を使って **vsftpd** の動作方法を SELinux に指示します。

allow_ftpd_anon_write

この Boolean を無効にした場合、 **vsftpd** による **public_content_rw_t** タイプのラベルが付いたファイルおよびディレクトリへの書き込みが阻止されます。 有効にすると、 ユーザーによる FTP 経由のファイルのアップロードが可能になります。 ファイルのアップロード先となるディレクトリには **public_content_rw_t** タイプのラベルを付け、 また Linux パーミッションも設定しておく必要があります。

allow_ftpd_full_access

この Boolean をオンにすると、 アクセス制御に Linux (DAC) のパーミッションしか使用されなくなるため、 認証ユーザーはファイルに **public_content_t** や **public_content_rw_t** のタイプのラベルが付いていなくてもファイルの読み取りおよび書き込みが可能になります。

allow_ftpd_use_cifs

この Boolean を有効にすると、 **vsftpd** による **cifs_t** タイプのラベルが付いたファイルやディレクトリへのアクセスを許可します。 したがって、 この Boolean を有効にすることで Samba でマウントしたファイルシステムを **vsftpd** で共有することができるようになります。

allow_ftpd_use_nfs

この Boolean を有効にすると、 **vsftpd** による **nfs_t** タイプのラベルが付いたファイルやディレクトリへのアクセスを許可します。 したがって、 この Boolean を有効にすることで NFS でマウントしたファイルシステムを **vsftpd** で共有することができるようになります。

ftp_home_dir

この Boolean を有効にすると、 認証ユーザーによるユーザーのホームディレクトリ内のファイルの読み取りと書き込みを許可します。 この Boolean をオフにした場合、 ホームディレクトリからファイルをダウンロードしようとすると **550 Failed to open file** などのエラーが発生します。 SELinux 拒否がログ記録されます。

ftpd_connect_db

FTP デーモンによるデータベースへの接続開始を許可します。

httpd_enable_ftp_server

httpd による FTP ポートでのリッスンおよび FTP サーバーとしての動作を許可します。

tftp_anon_write

この Boolean を有効にすると、 特殊なアクセス制限がなく共通ファイル用に予約されている領域など、パブリックディレクトリへの TFTP によるアクセスが許可されます。

5.4. 設定例

5.4.1. FTP サイトにアップロードする

特定のユーザーがファイルのアップロード専用として使用できる FTP サイトを作成している例を示します。 ディレクトリ構造を作成し、 必要となる SELinux 設定の変更を行なっています。

1. root ユーザーで **setsebool ftp_home_dir=1** を実行して FTP ホームディレクトリへのアクセスができるようにします。
2. root ユーザーで **mkdir -p /myftp/pub** を実行して新規ディレクトリを最上位に作成します。
3. Linux ユーザーの書き込みアクセスを許可するため、**/myftp/pub/** ディレクトリで Linux パーミッションの設定を行ないます。以下の例では、所有者とグループを root から所有者 user1 とグループ root に変更します。書き込みアクセスを与えたいユーザーを「user1」の部分に入れてください。

```
# chown user1:root /myftp/pub
# chmod 775 /myftp/pub
```

chown コマンドで所有者とグループのパーミッションを変更しています。**chmod** コマンドではモードを変更し、user1 ユーザーには読み取り、書き込み、実行のパーミッションを許可、root グループのメンバーには読み取り、書き込み、実行のパーミッションを許可しています。これ以外のユーザーには読み取りと実行のパーミッションを許可しています。このディレクトリ配下にあるファイルの読み込みを Apache HTTP Server に許可する必要があります。

4. SELinux を実行する場合は、ファイルやディレクトリにアクセス許可のラベルを適切に付ける必要があります。Linux パーミッションの設定だけでは不十分です。ファイルに **public_content_t** タイプのラベルが付いている場合は、FTP、Apache HTTP Server、Samba による読み込みおよび再同期が可能です。ファイルに **public_content_rw_t** タイプのラベルが付いている場合は、FTP による書き込みが可能です。 Samba など FTP 以外のサービスによる **public_content_rw_t** タイプのラベルが付いているファイルへの書き込みについては、書き込みを行なう前にまず Boolean を設定しておく必要があります。最上位のディレクトリ (**/myftp/**) に **public_content_t** タイプのラベルを付け、**/myftp/** 配下にコピーまたは新規作成されたファイルに対してサービスによる書き込みや変更が行なわれないようにします。root ユーザーで次のコマンドを実行し、ラベルの変更をファイルコンテキスト設定に追加します。

```
semanage fcontext -a -t public_content_t /myftp
```

5. **restorecon -R -v /myftp/** を実行してラベルの変更を適用します。

```
# restorecon -R -v /myftp/
restorecon reset /myftp context unconfined_u:object_r:default_t:s0-
>system_u:object_r:public_content_t:s0
```

6. **/myftp** には **public_content_t** タイプのラベル、**/myftp/pub/** には **default_t** タイプのラベルが付いているか確認します。

```
$ ls -dZ /myftp/
drwxr-xr-x. root root system_u:object_r:public_content_t:s0 /myftp/
$ ls -dZ /myftp/pub/
drwxrwxr-x. user1 root unconfined_u:object_r:default_t:s0
/myftp/pub/
```

7. ユーザーが FTP 経由でファイルをアップロードできるようにするため、 まず先に FTP にディレクトリへの書き込みを許可する必要があります。 SELinux で FTP に書き込みを許可しているのは、**public_content_rw_t** タイプのラベルが付いたディレクトリです。ここでは、FTP に書き込みを許可するディレクトリとして **/myftp/pub/** を使用しています。root ユーザーで次のコマンドを実行し、ラベルの変更をファイルコンテキスト設定に追加します。

```
semanage fcontext -a -t public_content_rw_t "/myftp/pub(/.*)?"
```

8. root ユーザーで **restorecon -R -v /myftp/pub** を実行してラベルの変更を適用します。

```
# restorecon -R -v /myftp/pub
restorecon reset /myftp/pub context system_u:object_r:default_t:s0-
>system_u:object_r:public_content_rw_t:s0
```

9. **allow_ftpd_anon_write** Boolean をオンにして、**vsftpd** による **public_content_rw_t** タイプのラベルが付いたファイルへの書き込みを許可する必要があります。root ユーザーで次のコマンドを実行し、この Boolean をオンにします。

```
setsebool -P allow_ftpd_anon_write on
```

   再起動後、 変更を維持したくない場合は **-P** オプションを使用しないでください。

FTP でログインしてからファイルをアップロードする例を以下に示します。 ユーザーは前述の例と同じ user1 ユーザーを使用しています。 **/myftp/pub/** ディレクトリは user1 が専用所有者となるディレクトリです。

1. **cd ~/** を実行してホームディレクトリに移動します。 次に、**mkdir myftp** を実行して FTP 経由でアップロードするファイルを格納するディレクトリを作成します。

2. **cd ~/myftp** を実行して **~/myftp/** ディレクトリに移動します。 このディレクトリ内に **ftpupload** ファイルを作成します。 以下の内容をこのファイルにコピーします。

```
File upload via FTP from a home directory.
```

3. **getsebool allow_ftpd_anon_write** を実行して、 **allow_ftpd_anon_write** Boolean がオンになっているか確認します。

```
$ getsebool allow_ftpd_anon_write
allow_ftpd_anon_write --> on
```

この Boolean がオフになっている場合は、 root ユーザーで **setsebool -P allow_ftpd_anon_write on** を実行し Boolean をオンにします。 再起動後、 変更を維持したくない場合は **-P** オプションを使用しないでください。

4. root ユーザーで **service vsftpd start** を実行し、 **vsftpd** を起動します。

```
# service vsftpd start
Starting vsftpd for vsftpd:                                [  OK
]
```

5. **ftp localhost** を実行します。 ユーザー名の入力が求められたら、 書き込みアクセスを持っているユーザーのユーザー名を入力し、 そのユーザーの正しいパスワードを入力します。

```
$ ftp localhost
Connected to localhost (127.0.0.1).
220 (vsFTPd 2.1.0)
Name (localhost:username):
331 Please specify the password.
Password: Enter the correct password
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd myftp
250 Directory successfully changed.
ftp> put ftpupload
local: ftpupload remote: ftpupload
227 Entering Passive Mode (127,0,0,1,241,41).
150 Ok to send data.
226 File receive OK.
ftp> 221 Goodbye.
```

allow_ftpd_anon_write Boolean が有効になっているためアップロードが成功します。

第6章 ネットワークファイルシステム

Red Hat Linux リファレンスガイド より抜粋:

NFS (Network File System) を利用すると、 ホストはリモートのシステム上にあるパーティションをマウントし、 ローカルのファイルシステムと同じように使用することができるようになります。 これにより、 管理者側はリソースをネットワーク上の一元的な場所に格納し、 認証ユーザーに安定したリソースアクセスを提供できるようになります。

Red Hat Enterprise Linux では、 NFS に完全対応させる場合は nfs-utils が必要になります。 **rpm -q nfs-utils** を実行して nfs-utils がインストールされているか確認してください。NFS を使用する予定にも関わらず、このパッケージがインストールされていない場合は、root ユーザーで次のコマンドを実行してパッケージのインストールを行ないます。

```
yum install nfs-utils
```

6.1. NFS と SELINUX

SELinux を実行すると、**NFS** デーモンはデフォルトで制限されます。SELinux ポリシーにより、NFS によるファイル共有はデフォルトで許可されます。

6.2. タイプ

デフォルトでは、 クライアント側にマウントした NFS ファイルシステムには、 ポリシーで定義された NFS ファイルシステム用デフォルトコンテキストのラベルが付けられます。一般的なポリシーであれば、このデフォルトのコンテキストには **nfs_t** タイプが使用されます。NFS では次のようなタイプが使用されます。タイプに応じて柔軟なアクセス設定ができます。

var_lib_nfs_t

このタイプは、 **/var/lib/nfs** ディレクトリ内の既存ファイルおよびこのディレクトリにコピーまたは作成される新規ファイルに対して使用されます。通常の操作では、このタイプは変更する必要はありません。加えられた変更をデフォルトの設定に復元する場合は、root ユーザーで **restorecon -R -v /var/lib/nfs** コマンドを実行します。

nfsd_exec_t

/usr/sbin/rpc.nfsd ファイルには、**nfsd_exec_t** のラベルが付けられます。これ以外、NFS 関連の実行可能なシステムファイルやライブラリも同様にこのタイプのラベルが付けられます。ユーザーがファイルのラベル付けを行なう際は、このタイプは使用しないでください。**nfsd_exec_t** は **nfsd_t** に遷移します。

6.3. BOOLEAN

SELinux は実行するサービスに必要な最小限レベルのアクセスに基づいています。サービスの実行手段は複数あるため、サービスをどのように実行するのかを SELinux に指示する必要があります。 次の Boolean を使って NFS の動作方法を SELinux に指示します。

allow_ftpd_use_nfs

この Boolean を有効にすると、**ftpd** による NFS マウントへのアクセスを許可します。

allow_nfsd_anon_write

この Boolean を有効にすると、特殊なアクセス制限がなく共通ファイル用に予約されている領域など、パブリックディレクトリへの **nfsd** による anonymous の書き込みが許可されます。

httpd_use_nfs

この Boolean を有効にすると、NFS ファイルシステム上に格納されたファイルへの **httpd** によるアクセスを許可します。

nfs_export_all_ro

ファイルやディレクトリのエクスポートはすべて NFS 経由で行い、読み取り専用パーミッションを与えます。

nfs_export_all_rw

ファイルやディレクトリのエクスポートはすべて NFS 経由で行い、読み取りと書き込みのパーミッションを与えます。

qemu_use_nfs

qemu による NFS ファイルシステムの使用を許可します。

samba_share_nfs

この Boolean を無効にすることで、**smbd** には Samba を介した NFS 共有へのフルアクセスを与えないようにします。この Boolean を有効にすると、Samba による NFS ファイルシステムの共有が許可されます。

use_nfs_home_dirs

この Boolean を有効にすると、NFS ホームディレクトリのサポートが有効になります。

virt_use_nfs

仮想マシンによる NFS ファイルへのアクセスを許可します。

xen_use_nfs

Xen による NFS ファイルの使用を許可します。

6.4. 設定例

6.4.1. NFS を使ってディレクトリを共有する

NFS と SELinux を使ったディレクトリの作成と共有の例を示します。 2 台のホストを使用しています。 ホスト名が **nfs-srv** で IP アドレスが **192.168.1.1**の NFS サーバーと、 ホスト名が **nfs-client** で IP アドレスが **192.168.1.100** のクライアントです。 いずれのホストも同じサブネット (192.168.1.0/24) 上にあります。これは一例に過ぎません。 また、 nfs-utils パッケージがインストールされていること、 SELinux targeted ポリシーを使用していること、 SELinux は enforced モードで実行していることを前提としています。

NFS を介した全ユーザーへのアクセスが Linux ファイルパーミッションで与えられ、 ネットワークへのアクセスにも制限がない状態であっても、 SELinux の Boolean を使って適切なパーミッションを与えない限り、 SELinux で NFS ファイルシステムがマウントされないよう防ぐことができます。

6.4.1.1. サーバーのセットアップ

次のステップ1からステップ10までは NFS サーバー **nfs-srv** で行なってください。

1. **setsebool** コマンドを実行し、 NFS ファイルシステムの読み取りと書き込みでのマウントを無効にします。

```
setsebool -P nfs_export_all_rw off
```

注記

再起動後、 **setsebool** による変更を維持したくない場合は **-P** オプションを使用しないでください。

2. **rpm -q nfs-utils** を実行して、 nfs-utils パッケージがインストールされているか確認します。 nfs-utils パッケージにより NFSを使用するためのサポートプログラムが提供されるため、このパッケージは NFS サーバーおよび使用中のすべてのクライアントにインストールしてください。 このパッケージがインストールされていない場合は、 root ユーザーで **yum install *nfs-utils*** を実行してインストールしてください。

3. root で **mkdir /myshare** を実行して、 NFS を使って共有するディレクトリを最上位に新規作成します。

4. root ユーザーで **touch /myshare/file1** を実行して共有エリア内に新規で空のファイルを作成します。 このファイルがクライアントによってアクセスされることになるファイルになります。

5. 全ユーザーへのアクセスが Linux パーミッションにより完全に与えられていても、 SELinux でアクセスをブロックすることができることを確認するため、 **/myshare** ディレクトリに誰でもアクセスできる Linux のフルアクセス権限を与えます。

```
# chmod -R 777 /myshare
```

警告

これは説明を目的とした用例に過ぎません。 実稼働のシステムにはここで示すパーミッションを使用しないでください。

6. **/etc/exports** ファイルを編集して、 次の行をファイルの先頭に追加します。

```
/myshare  192.168.1.100(rw)
```

このエントリでは、サーバー上の共有フォルダ **/myshare** へのフルパス、**nfs-srv** が共有するホストやネットワークの範囲 (この例の場合は単一ホスト **nfs-client** の IP アドレス **192.168.1.100**)、共有パーミッションを示しています。ここでは **(rw)** でわかるように、読み取りと書き込みのパーミッションが与えられています。

7. NFS に使用する TCP と UDP のポートが rpcbind により動的に割り当てられますが、ファイアウォールルールを作成する際、これが問題となる場合があります。この例では、 NFS トラ

フィックがファイアウォールを通過できるようにするプロセスを簡略化するため、**/etc/sysconfig/nfs** ファイルを編集して、**MOUNTD_PORT**、**STATD_PORT**、**LOCKD_TCPPORT**、**LOCKD_UDPPORT** の変数をすべてアンコメントします。ここではファイル内のポート番号の変更は必要ありません。

着信接続がサーバーのファイアウォールを必ず通過できるようにしておきます。**system-config-firewall** ツールを使って行ないます。

- NFS 用 TCP および UDP ポートの 2049
- TCP および UDP ポートの 111 (rpcbind/sunrpc)
- **MOUNTD_PORT="port"** オプションで指定された TCP および UDP ポート
- **STATD_PORT="port"** オプションで指定された TCP および UDP ポート
- **LOCKD_TCPPORT="port"** オプションで指定された TCP ポート
- **LOCKD_UDPPORT="port"** オプションで指定された UDP ポート

8. root ユーザーで **service nfs start** を実行して、 NFS および関連サービスを起動させます。

```
# service nfs start
Starting NFS services:  [  OK  ]
Starting NFS quotas:  [  OK  ]
Starting NFS daemon:  [  OK  ]
Starting NFS mountd:  [  OK  ]
```

9. root で **exportfs -rv** を実行し、 NFS サブシステムのエクスポートテーブルを必ず更新させます。

```
# exportfs -rv
exporting 192.168.1.100:/myshare
```

10. root ユーザーで **showmount -e** を実行し、 エクスポートされたファイルシステムをすべて表示させます。

```
# showmount -e
Export list for nfs-srv:
/myshare 192.168.1.100
```

これで、 サーバー **nfs-srv** は **192.168.1.100** での **nfs-client** との NFS 通信が許可されるよう設定され、 Linux ファイルシステムの全権限が有効にされました。 SELinux を無効にすると、 クライアントによるこの共有のマウントおよびフルアクセスが可能になります。 しかし、 **nfs_export_all_rw** Boolean を無効にしているため、 以下の出力で示すように、 クライアントは現在、 このファイルシステムをマウントすることができません。 このステップはクライアント側 (**nfs-client**) で行なってください。

```
[nfs-client]# mkdir /myshare
[nfs-client]# mount.nfs 192.168.1.1:/myshare /myshare
mount.nfs: access denied by server while mounting 192.168.1.1:/myshare/
```

前述のステップ1で無効にしたSELinux Booleanを有効にすると、クライアント側で共有ファイルシステムのマウントが行なえるようになります。このステップはNFSサーバー**nfs-srv**で行なってください。

```
[nfs-srv]# setsebool -P nfs_export_all_rw on
```

NFSデーモンを再起動します。

```
[nfs-srv]# service nfs restart
```

ここでもう一度NFSファイルシステムをマウントしてみます。このステップはNFSクライアント**nfs-client**で行なってください。

```
[nfs-client]# mount.nfs 192.168.1.1:/myshare /myshare
[nfs-client]#
[nfs-client]# ls /myshare
total 0
-rwxrwxrwx.  1 root root 0 2009-04-16 12:07 file1
[nfs-client]#
```

クライアント側でファイルシステムが正しくマウントされました。この例でわかるように、SELinuxで防御性を高めることにより、Linuxパーミッションでは全ユーザーに完全アクセスが与えられるよう設定されていても、SELinuxのパーミッションを強制することができます。

第7章 BIND (BERKELEY INTERNET NAME DOMAIN)

BIND では **named** デーモンを使って名前解決サービスを行ないます。 BIND のおかけで、 ユーザーは数値アドレスではなく名前でコンピューターリソースやサービスを検索することができます。

Red Hat Enterprise Linux では、 DNS サーバーは bind パッケージで提供されます。 **rpm -q bind** を実行して、 bind パッケージがインストールされているか確認します。 インストールされていない場合は、 root ユーザーで次のコマンドを実行してインストールしてください。

```
yum install bind
```

7.1. BIND と SELINUX

/var/named/slaves、**/var/named/dynamic**、 **/var/named/data** ディレクトリのデフォルトパーミッションでは、 ゾーン転送およびダイナミック DNS 更新によるゾーンファイルの更新が許可されます。 **/var/named** 内のファイルには **named_zone_t** タイプのラベルが付けられ、 マスターゾーンファイルに使用されます。

スレーブサーバーの場合、 **/etc/named.conf** でスレーブゾーンを **/var/named/slaves** に配置するよう設定します。 以下に、 スレーブ DNS サーバーの **/etc/named.conf** 内にあるドメインエントリの例を示します。 このスレーブ DNS サーバーは、 **/var/named/slaves** 内に **testdomain.com** 用のゾーンファイルを格納しています。

```
zone "testdomain.com" {
   type slave;
   masters { IP-address; };
   file "/var/named/slaves/db.testdomain.com";
         };
```

ゾーンファイルに **named_zone_t** のラベルが付けられている場合は、 **named_write_master_zones** Boolean を有効にして、 ゾーンファイル更新のためゾーン転送とダイナミック DNS を許可する必要があります。 また、 親ディレクトリのモードを変更し、 **named** ユーザーまたはグループに読み取り、 書き込み、 実行のアクセスを許可しなければなりません。

/var/named/ 内のゾーンファイルで **named_cache_t** タイプのラベルが付いているファイルは、 ファイルシステムの再ラベル付けや **restorecon -R /var/** の実行が行なわれると **named_zone_t** タイプにラベルが変更されます。

7.2. タイプ

BIND で使用されるタイプを以下に示します。 タイプに応じて柔軟なアクセス設定ができます。

named_zone_t

マスターゾーンファイルに使用されます。 他のサービスでは、 このタイプのファイルを変更することはできません。このタイプのファイルを変更できるのは **named** のみになります。この場合、**named_write_master_zones** Boolean をオンにする必要があります。

named_cache_t

このタイプのラベルが付いたファイルの場合、特にBoolean を設定しなくてもデフォルトで **named** による書き込みが可能です。**/var/named/slaves**、**/var/named/dynamic**、**/var/named/data** ディレクトリ内にコピーまたは作成されるファイルには **named_cache_t** タイプのラベルが自動的に付けられます。

named_var_run_t

/var/run/bind/、 **/var/run/named/**、 **/var/run/unbound/** ディレクトリ内にコピーまたは作成されるファイルには、 **named_var_run_t** タイプのラベルが自動的に付けられます。

named_conf_t

BIND 関連の設定ファイル (一般的には **/etc/** ディレクトリに格納される) には、 **named_conf_t** タイプのラベルが自動的に付けられます。

named_exec_t

BIND 関連の実行可能ファイル (一般的には **/usr/sbin/** ディレクトリに格納される) には、 **named_exec_t** タイプのラベルが自動的に付けられます。

named_log_t

BIND 関連のログファイル (一般的には **/var/log/** ディレクトリに格納される) には、 **named_log_t** タイプのラベルが自動的に付けられます。

named_initrc_exec_t

/etc/rc.d/init.d/ ディレクトリ内にある実行可能な BIND 関連のファイルには、 **named_initrc_exec_t** タイプのラベルが自動的に付けられます。

7.3. BOOLEAN

SELinux は実行するサービスに必要な最小限レベルのアクセスに基づいています。サービスの実行手段は複数あるため、サービスをどのように実行するのかを SELinux に指示する必要があります。 次の Boolean を使って BIND の動作方法を SELinux に指示します。

named_write_master_zones

この Boolean を無効にすると、 **named** による **named_zone_t** タイプのラベルが付いたゾーンファイルやディレクトリへの書き込みが阻止されます。 一般的には、 **named** はゾーンファイルへの書き込みを必要としません。 ただし、 第 2 サーバーなどがゾーンファイルへの書き込みを必要とする場合には、 この Boolean を有効にして書き込み動作を許可します。

7.4. 設定例

7.4.1. ダイナミック DNS

BIND を使用すると、 ホストがゾーンファイルや DNS 内の記録を動的に更新することができるようになります。 ホストコンピューターの IP アドレスが頻繁に変更され、 DNS レコードでリアルタイムの修正が必要となる場合に BIND を使用します。

ダイナミック DNS で更新させるゾーンファイル用に **/var/named/dynamic** ディレクトリを使用します。 **/var/named/dynamic** に作成またはコピーされるファイルは、 **named** による書き込みを許可する Linux パーミッションを継承します。また、こうしたファイルには **named_cache_t** タイプのラベルが付けられるため、 **named** による書き込み許可が SELinux により与えられます。

/var/named/dynamic 内のゾーンファイルに **named_zone_t** タイプのラベルが付けられている場合、 DNS 更新がマージされる前にまずジャーナルに書き込まれるため、一定の期間 DNS の更新に失敗することがあります。このタイプのラベルが付けられているゾーンファイルでジャーナルに書き込みが

行なわれ、そのジャーナルのマージが試行されていると、次のようなエラーがログ記録されます。

```
named[PID]: dumping master file: rename: /var/named/dynamic/zone-name:
permission denied
```

また、 次のような **SELinux** 拒否もログ記録されます。

```
setroubleshoot: SELinux is preventing named (named_t) "unlink" to zone-
name (named_zone_t)
```

このラベル付けに関する問題を解決するには、 **Linux root** ユーザーで **restorecon -R -v /var/named/dynamic** コマンドを実行します。

第8章 CVS (CONCURRENT VERSIONING SYSTEM)

CVS (Concurrent Versioning System) は、フリーのリビジョンコントロールシステムです。 中央に置かれた複数ファイルのセットに対する変更の監視および追跡に使用します。 一般的に複数のユーザーによってアクセスされます。 ソースコードリポジトリの管理などによく使用され、 オープンソースのプログラマー間では幅広く使用されています。

Red Hat Enterprise Linux では、 CVS は cvs パッケージにより提供されます。 **rpm -q cvs** を実行して cvs パッケージがインストールされているか確認します。CVS を使用する予定にも関わらず、このパッケージがインストールされていない場合は、root ユーザーで次のコマンドを実行してパッケージのインストールを行ないます。

```
yum install cvs
```

8.1. CVS と SELINUX

cvs は **cvs_t** として実行します。 Red Hat Enterprise Linux では、 CVS による読み取りと書き込みが許可されるのは特定のディレクトリに限られます。 **cvs** デーモンによる読み取りと書き込みのアクセスが与えられる領域は **cvs_data_t** ラベルで定義されます。 SELinux で CVS を使用する場合、 CVS データ用に予約されている領域へのフルアクセスがクライアントに与えられるため、 適切なラベル割り当てが重要になります。

8.2. タイプ

CVS で使用されるタイプを以下に示します。 タイプに応じて柔軟なアクセス設定ができます。

cvs_data_t

このタイプは CVS リポジトリ内のデータに対して使用されます。 CVS がフルアクセスできるのはこのタイプのデータのみです。

cvs_exec_t

このタイプは **/usr/bin/cvs** バイナリに対して使用されます。

8.3. BOOLEAN

SELinux は実行するサービスに必要な最小限レベルのアクセスに基づいています。サービスの実行手段は複数あるため、サービスをどのように実行するのかを SELinux に指示する必要があります。 次の Boolean を使って CVS の動作方法を SELinux に指示します。

allow_cvs_read_shadow

この Boolean を使って **cvs** デーモンによるユーザー認証用 **/etc/shadow** ファイルへのアクセスを許可します。

8.4. 設定例

8.4.1. CVS のセットアップ

リモートアクセスを許可する SELinux 設定と簡単な CVS セットアップの例を以下に示します。 2 台のホストを使用しています。 ホスト名が **cvs-srv** で IP アドレスが **192.168.1.1** の CVS サーバーと、

ホスト名が **cvs-client** で IP アドレスが **192.168.1.100** のクライアントです。 いずれのホストも同じサブネット上にあります (192.168.1.0/24)。 これは一例に過ぎません。 また、 cvs と xinetd パッケージがインストールされていること、 SELinux targeted ポリシーを使用していること、 SELinux は enforced モードで実行していることを前提としています。

DAC の全パーミッションが許可されている場合であっても、 SELinux ではファイルラベルに応じたポリシールールを施行し、 CVS アクセス用のラベルが明確に付けられている特定領域にしかアクセスを許可しないようにすることができます。

注記

ステップ 1 から 9 は CVS サーバー **cvs-srv** で行なってください。

1. cvs と xinetd のパッケージが必要になります。 **rpm -q cvs** を実行して、 cvs パッケージがインストールされているか確認してください。 インストールされていない場合は、 root ユーザーで次のコマンドを実行し、 cvs をインストールします。

   ```
   # yum install cvs
   ```

   rpm -q xinetd を実行して、 xinetd パッケージがインストールされているか確認してください。インストールされていない場合は、 root ユーザーで次のコマンドを実行し xinetd をインストールします。

   ```
   # yum install xinetd
   ```

2. **CVS** と言う名前のグループを作成します。 root ユーザーで**groupadd CVS** コマンドを実行するか、 **system-config-users** ツールを使って行ないます。

3. **cvsuser** と言うユーザー名のユーザーを作成し、 このユーザーを CVS グループのメンバーにします。 **system-config-users** ツールを使って行ないます。

4. **/etc/services** ファイルを編集し、 以下のように CVS サーバーのエントリをアンコメントします。

   ```
   cvspserver 2401/tcp   # CVS client/server operations
   cvspserver 2401/udp   # CVS client/server operations
   ```

5. CVS リポジトリをファイルシステムの root 領域に作成します。 SELinux を使用する場合、 リポジトリは root ファイルシステムに配置するのが最適です。 他のサブディレクトリに影響を与えることなく、 再帰的なラベルを与えることができます。 たとえば、 root ユーザーでリポジトリを格納する **/cvs** ディレクトリを作成します。

   ```
   [root@cvs-srv]# mkdir /cvs
   ```

6. 誰でもアクセスできるよう **/cvs** ディレクトリに全パーミッションを与えます。

   ```
   [root@cvs-srv]# chmod -R 777 /cvs
   ```

警告

これは説明を目的とした用例に過ぎません。 実稼働のシステムにはここで示すパーミッションを使用しないでください。

7. **/etc/xinetd.d/cvs** ファイルを編集し、 CVS セクションをアンコメントして **/cvs** ディレクトリを使用するよう設定します。 以下のようになるはずです。

```
service cvspserver
{
 disable = no
 port    = 2401
 socket_type  = stream
 protocol  = tcp
 wait    = no
 user    = root
 passenv    = PATH
 server    = /usr/bin/cvs
 env    = HOME=/cvs
 server_args  = -f --allow-root=/cvs pserver
# bind    = 127.0.0.1
```

8. root ユーザーで **service xinetd start** を実行し、 **xinetd** デーモンを起動します。

9. **system-config-firewall** ツールを使って、 ポート 2401 上で TCP を使用した着信接続を許可するルールを追加します。

10. **cvsuser** ユーザーになり、 次のコマンドを実行します。

```
[cvsuser@cvs-client]$ cvs -d /cvs init
```

11. これで CVS は設定されましたが、 SELinux ではログインおよびファイルのアクセスが拒否されます。これを確認するため、**cvs-client** に $CVSROOT 変数をセットして、リモートによるログインを試行してみます。次のステップは **cvs-client** で行なってください。

```
[cvsuser@cvs-client]$ export
CVSROOT=:pserver:cvsuser@192.168.1.1:/cvs
[cvsuser@cvs-client]$
[cvsuser@cvs-client]$ cvs login
Logging in to :pserver:cvsuser@192.168.1.1:2401/cvs
CVS password: ********
cvs [login aborted]: unrecognized auth response from 192.168.100.1:
cvs pserver: cannot open /cvs/CVSROOT/config: Permission denied
```

SELinux によりアクセスがブロックされました。 SELinux でこのアクセスを許可させるため、次のステップを **cvs-srv** で行なってください。

12. 既存のデータおよび新規のデータすべてに再帰的にラベル付けが行なわれるよう、 root ユーザーで **/cvs** ディレクトリのコンテキストを変更し **cvs_data_t** タイプを与えます。

```
[root@cvs-srv]# semanage fcontext -a -t cvs_data_t '/cvs(/.*)?'
[root@cvs-srv]# restorecon -R -v /cvs
```

13. これで、クライアント **cvs-client** はログインして、このリポジトリ内のすべての CVS リソースにアクセスできるようになったはずです。

```
[cvsuser@cvs-client]$ export
CVSROOT=:pserver:cvsuser@192.168.1.1:/cvs
[cvsuser@cvs-client]$
[cvsuser@cvs-client]$ cvs login
Logging in to :pserver:cvsuser@192.168.1.1:2401/cvs
CVS password: ********
[cvsuser@cvs-client]$
```

第9章 SQUID キャッシングプロキシ

Squid Caching Proxy プロジェクトページより抜粋:

原文: "Squid is a caching proxy for the Web supporting HTTP, HTTPS, FTP, and more. It reduces bandwidth and improves response times by caching and reusing frequently-requested web pages. Squid has extensive access controls and makes a great server accelerator." (訳文: Squid とは HTTP、HTTPS、 FTP などに対応する Web 用キャッシングプロキシです。 頻繁に要求される Web ページをキャッシングして再利用することで帯域幅の使用を抑えます。 Squid は豊富なアクセス制御機能が備わっている強力なサーバーアクセラレーターになります。)

Red Hat Enterprise Linux では、 Squid キャッシングプロキシは squid パッケージにより提供されます。 **rpm -q squid** を実行して squid パッケージがインストールされているか確認します。squid を使用する予定にも関わらず、このパッケージがインストールされていない場合は、root ユーザーで次のコマンドを実行してパッケージのインストールを行ないます。

```
# yum install squid
```

9.1. SQUID キャッシングプロキシと SELINUX

SELinux を有効にすると、 squid はデフォルトで制限のあるサービスとして実行されます。 制限のあるプロセスはそれ自体のドメイン内で実行され、 他の制限のあるプロセスとは分離されます。 制限のあるプロセスが攻撃を受けると、 SELinux ポリシー設定に応じて、 攻撃側がリソースにアクセスして加えることができる被害は限定されます。 以下に、 squid 自体のドメイン内で実行している squid プロセスの例を示します。 ここでは squid パッケージがインストールされていることを前提としています。

1. **getenforce** を実行して、 SELinux が enforcing モードで実行しているか確認します。

   ```
   $ getenforce
   Enforcing
   ```

   SELinux が enforcing モードで実行している場合は、 **getenforce** コマンドを実行すると **Enforcing** が返されます。

2. root ユーザーで **service squid start** を実行し、 **squid** を起動します。

   ```
   # service squid start
   Starting squid:                                            [  OK
   ]
   ```

3. **ps -eZ | grep squid** を実行し、 **squid** プロセスを表示させます。

   ```
   $ ps -eZ | grep squid
   unconfined_u:system_r:squid_t:s0 2522 ?        00:00:00 squid
   unconfined_u:system_r:squid_t:s0 2524 ?        00:00:00 squid
   unconfined_u:system_r:squid_t:s0 2526 ?        00:00:00 ncsa_auth
   unconfined_u:system_r:squid_t:s0 2527 ?        00:00:00 ncsa_auth
   unconfined_u:system_r:squid_t:s0 2528 ?        00:00:00 ncsa_auth
   unconfined_u:system_r:squid_t:s0 2529 ?        00:00:00 ncsa_auth
   unconfined_u:system_r:squid_t:s0 2530 ?        00:00:00 ncsa_auth
   unconfined_u:system_r:squid_t:s0 2531 ?        00:00:00 unlinkd
   ```

   squid プロセスに関連する SELinux コンテキストは

unconfined_u:system_r:squid_t:s0 です。 コンテキストの最後から 2 番目の部分、**squid_t** がタイプになります。 プロセスのドメインやファイルのタイプを定義するのがタイプです。 この例の場合、 **squid** プロセスは **squid_t** ドメイン内で実行しています。

squid_t などの制限ドメイン内で実行しているプロセスがファイルや他のプロセス、 システムなどとどのように交信するのかを SELinux ポリシーで定義します。squid によるアクセスを許可するには、ファイルに適切なラベルを付けを行なう必要があります。

/etc/squid/squid.conf を設定して、**squid** にデフォルトの TCP ポート 3128、3401、4827 以外のポートでリッスンさせる場合は、**semanage port** コマンドを使って SELinux ポリシー設定にそのポート番号を追加する必要があります。以下では、SELinux ポリシー設定では最初 **squid** 用には定義されていなかったポートでリッスンするよう **squid** を設定したため、**squid** の起動に失敗する例を示します。また、SELinux システムを設定し、ポリシーではまだ定義されていなかった非標準のポートで **squid** によるリッスンを許可する方法についても示します。ここでは、squid パッケージがインストールされていることを前提としています。各コマンドは root ユーザーで実行してください。

1. **service squid status** を実行し、 **squid** が実行中ではないことを確認します。

```
# service squid status
squid is stopped
```

出力が上記と異なる場合は、 **service squid stop** を実行してプロセスを停止します。

```
# service squid stop
Stopping squid:                                            [  OK
]
```

2. **semanage port -l | grep -w squid_port_t** を実行して、 SELinux で **squid** にリッスンを許可しているポートを表示させます。

```
semanage port -l | grep -w -i squid_port_t
squid_port_t                   tcp      3401, 4827
squid_port_t                   udp      3401, 4827
```

3. root ユーザーで **/etc/squid/squid.conf** を編集します。SELinux ポリシー設定では **squid** 用に設定していないポートをリッスンするよう **http_port** オプションを設定します。この例では、ポート 10000 でリッスンするよう **squid** を設定しています。

```
# Squid normally listens to port 3128
http_port 10000
```

4. **setsebool** コマンドを実行し、 **squid_connect_any** Boolean をオフに設定します。 これで、 squid の動作は特定ポート上に限られることになります。

```
setsebool -P squid_connect_any 0
```

5. **service squid start** を実行し、 **squid** を起動します。

```
# service squid start
Starting squid: ...................                        [FAILED]
```

以下のような SELinux 拒否がログ記録されます。

```
localhost setroubleshoot: SELinux is preventing the squid (squid_t)
from binding to port 10000. For complete SELinux messages. run
sealert -l 97136444-4497-4fff-a7a7-c4d8442db982
```

6. **squid** によるポート 10000 のリッスンを SELinux で許可するには、例で使用しているものと同じ次のコマンドが必要になります。

```
# semanage port -a -t squid_port_t -p tcp 10000
```

7. もう一度 **service squid start** を実行して、 **squid** を起動させ新しいポートをリッスンするようにします。

```
# service squid start
Starting squid:         [  OK  ]
```

8. これで、 **squid** による非標準のポート (この例では TCP 10000) でのリッスンを許可する SELinux 設定が完了しました。 **squid** はこのポートで正常に起動するようになります。

9.2. タイプ

Type Enforcement が SELinux の targeted ポリシーで使用されるメインのパーミッション制御になります。全ファイルおよびプロセスにタイプのラベルが付けられます。ファイルの場合はタイプ、プロセスの場合はドメインを定義します。任意のタイプにアクセスするドメインなのか、 別のドメインにアクセスするドメインなのかなど、SELinux のポリシールールではタイプによって互いがアクセスしあう方法を定義します。 アクセスを許可する特定の SELinux ポリシールールが存在する場合にのみ、 そのアクセスが許可されます。

squid で使用されるタイプを以下に示します。 タイプに応じて柔軟なアクセス設定ができます。

httpd_squid_script_exec_t

このタイプは、 **cachemgr.cgi** などのユーティリティに使用されます。 squid とその設定に関するさまざまな統計値を提供します。

squid_cache_t

このタイプは、 **/etc/squid/squid.conf** 内の **cache_dir** ディレクティブで定義されているように、 squid によってキャッシュされるデータに使用されます。 デフォルトでは、 **/var/cache/squid** と **/var/spool/squid** にコピーまたは作成されるファイルには **squid_cache_t** タイプのラベルが付けられます。 また、**/var/squidGuard** にコピーまたは作成される squidGuard URL リダイレクト用のファイルや **squid** 用のプラグインにも **squid_cache_t** タイプのラベルが付けられます。 squid のキャッシュデータ用として使用できるのは、このラベルが付いたファイルやディレクトリのみなります。

squid_conf_t

squid の設定用に使用されるディレクトリおよびファイルに対して使用されます。エラーメッセージやアイコンなどを含め、**/etc/squid** と **/usr/share/squid** 内に既存するファイルや、 ここに作成またはコピーされるファイルにはこのタイプのラベルが付けられます。

squid_exec_t

このタイプは squid バイナリの **/usr/sbin/squid** に使用されます。

squid_log_t

このタイプはログに使用されます。 **/var/log/squid** または **/var/log/squidGuard** 内に既存するファイル、 ここに作成またはコピーされるファイルにはこのタイプのラベルを付けなければなりません。

squid_initrc_exec_t

このタイプは、 **squid** の起動に必要となる初期設定ファイルに使用します。初期設定ファイルは **/etc/rc.d/init.d/squid** にあります。

squid_var_run_t

このタイプは **/var/run** 内のファイルに使用されます。 特に、 squid の実行時に作成される **/var/run/squid.pid** という名前のプロセス ID (PID) にはこのタイプが付けられます。

9.3. BOOLEAN

SELinux は実行するサービスに必要な最小限レベルのアクセスに基づいています。サービスの実行手段は複数あるため、サービスをどのように実行するのかを SELinux に指示する必要があります。 次の Boolean を使って Squid の動作方法を SELinux に指示します。

squid_connect_any

この Boolean を有効にすると、 ポートを問わず squid によるリモートホストへの接続開始を許可します。

9.4. 設定例

9.4.1. Squid を非標準のポートに接続させる

前述の Boolean を施行し、特定のポートに限ったアクセスをデフォルトで許可することで SELinux が Squid を補完している実践的な例を以下に示します。また、Boolean を変更する方法、およびその変更により許可されるアクセスについても示します。

以下に示す例は、 シンプルな Squid 設定に対してどのように SELinux が影響を与えることができるのかを示す一例に過ぎません。 Squid に関する総合的な説明は本ガイドの範疇を越えてしまいますので、詳細については、 公式の Squid ドキュメント を参照してください。 ここでは、 Squid ホストにはインターネットアクセスがあり、2 種類のネットワークインターフェースが備わっていることを前提としています。 また、ファイアウォールでは、Squid がリッスンするデフォルトの TCP ポート (TCP 3128) を使った内部インターフェース上のアクセスを許可するよう設定されていることを前提としています。

1. root ユーザーになり squid パッケージをインストールします。 **rpm -q squid** を実行して squid パッケージがインストールされているか確認します。 インストールされていない場合は、 root ユーザーで **yum install squid** を実行しインストールを行ないます。

2. メインの設定ファイル **/etc/squid/squid.conf** を編集し、 **cache_dir** ディレクティブが以下のようにアンコメントされているか確認します。

```
cache_dir ufs /var/spool/squid 100 16 256
```

上記では、 この設定例で使用する **cache_dir** ディレクティブのデフォルト設定を定義しています。 Squid ストレージフォーマット (ufs)、 キャッシュを配置するシステム上のディレクトリ (/var/spool/squid)、 キャッシュに使用するメガバイト単位のディスク領域 (100)、 第一レ

ベルのキャッシュディレクトリ数と第二レベルのキャッシュディレクトリ数 (16 と 256) の設定情報で構成されています。

3. 同じ設定ファイル内の **http_access allow localnet** ディレクティブもアンコメントされているか確認してください。 Red Hat Enterprise Linux では、 Squid のデフォルトインストールで自動的に設定される **localnet** ACL からのトラフィックを許可します。 既存する RFC1918 ネットワーク上のクライアントマシンにプロキシを通過できるアクセスを許可します (この設定例ではこれで充分です)。

4. 同じ設定ファイル内の **visible_hostname** ディレクティブもアンコメントされ、 マシンのホスト名が設定されているか確認してください。 値はホストの完全修飾ドメイン名にします。

```
visible_hostname squid.example.com
```

5. root ユーザーで **service squid start** を実行し、 **squid** を起動します。 はじめて **squid** を起動すると、 前のステップで **cache_dir** ディレクティブに指定したキャッシュディレクトリがこのコマンドにより初期化されてから **squid** デーモンが起動されます。 **squid** の起動に成功すると、 以下のような出力となります。

```
# /sbin/service squid start
init_cache_dir /var/spool/squid... Starting squid: .       [  OK  ]
```

6. **squid** プロセス ID (PID) が制限のあるサービスとして起動されているか確認します。この例では **squid_var_run_t** の値で確認します。

```
# ls -lZ /var/run/squid.pid
-rw-r--r--. root squid unconfined_u:object_r:squid_var_run_t:s0
/var/run/squid.pid
```

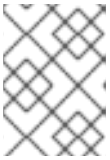
7. この時点で、前に設定していた **localnet** ACL に接続しているクライアントマシンは、そのプロキシとしてこのホストの内部インターフェースを使用できるようになります。これはシステム全体または一般的な全 Web ブラウザのセッティングで設定することができます。これで Squid では目的のマシンのデフォルトポートでリッスンするようになりますが (TCP 3128)、目的のマシンで許可されるのは、一般的なポートを介したインターネット上の他のサービスへの発信接続のみになります。これが SELinux 自体で定義されているポリシーになります。 SELinux では、次のステップで示すように非標準のポートへのアクセスは拒否されます。

8. TCP ポート 10000 での web サイトのリスニングなど、 Squid プロキシを介し非標準のポートを使った要求がクライアントによって行なわれると、 次のような拒否がログ記録されます。

```
SELinux is preventing the squid daemon from connecting to network
port 10000
```

9. このアクセスを許可する場合は、 デフォルトでは無効になっている **squid_connect_any** Boolean を変更する必要があります。 **squid_connect_any** Boolean をオンにするには、 root ユーザーで次のコマンドを実行します。

```
# setsebool -P squid_connect_any on
```



注記

再起動後、 **setsebool** による変更を維持したくない場合は **-P** オプションを使用しないでください。

10. クライアントの代わりとして、いずれのポートでも Squid による接続の開始が許可されるようになるため、クライアントはインターネット上の非標準のポートにアクセスできるようになります。

第10章 MYSQL

MySQL プロジェクトページより抜粋:

原文: "The MySQL® database has become the world's most popular open source database because of its consistent fast performance, high reliability and ease of use. It's used on every continent -- Yes, even Antarctica! -- by individual Web developers as well as many of the world's largest and fastest-growing organizations to save time and money powering their high-volume Web sites, business-critical systems and packaged software -- including industry leaders such as Yahoo!, Alcatel-Lucent, Google, Nokia, YouTube, and Zappos.com." (訳文: 安定した高速パフォーマンス、高い信頼性、 そしてその使いやすさで、 MySQL® データベースは世界でもっともポピュラーなオープンソースのデータベースとなりました。すべての大陸で (南極大陸をも含め!)、web 開発者そして急速な成長を遂げている世界でも有数の企業の多くにより (Yahoo、Alcatel-Lucent、Google、Nokia、YouTube、Zappos.com など) 、時間的、金銭的な効率性を高めながら大量の web サイト、ビジネスに不可欠なシステム、パッケージ化されたソフトウェアなどを供給するために利用されています。)

Red Hat Enterprise Linux では、 MySQL は mysql-server パッケージで提供されます。 **rpm -q mysql-server** を実行して、 mysql-server パッケージがインストールされているか確認してください。 インストールされていない場合は、 root ユーザーで次のコマンドを実行してインストールを行ないます。

```
yum install mysql-server
```

10.1. MYSQL と SELINUX

MySQL を有効にすると、 デフォルトで制限のあるサービスとして実行されます。 制限のあるプロセスはそれ自体のドメイン内で実行され、 他の制限のあるプロセスとは分離されます。 制限のあるプロセスが攻撃を受けると、 SELinux ポリシー設定に応じて、 攻撃側がリソースにアクセスして加えることができる被害は限定されます。 以下に、 MySQL 自体のドメイン内で実行している MySQL プロセスの例を示します。 ここでは mysql パッケージがインストールされていることを前提としています。

1. **getenforce** を実行して、 SELinux が enforcing モードで実行していることを確認します。

   ```
   $ getenforce
   Enforcing
   ```

   SELinux が enforcing モードで実行している場合は、 **getenforce** コマンドを実行すると **Enforcing** が返されます。

2. root ユーザーで **service mysqld start** を実行し、 **mysqld** を起動します。

   ```
   # service mysqld start
   Initializing MySQL database:  Installing MySQL system tables... [
   OK  ]
   Starting MySQL:                                            [  OK
   ]
   ```

3. **ps -eZ | grep mysqld** を実行し、 **mysqld** プロセスを表示させます。

   ```
   $ ps -eZ | grep mysqld
   unconfined_u:system_r:mysqld_safe_t:s0 6035 pts/1 00:00:00
   mysqld_safe
   unconfined_u:system_r:mysqld_t:s0 6123 pts/1   00:00:00 mysqld
   ```

mysqld プロセスに関連する SELinux コンテキストは **unconfined_u:system_r:mysqld_t:s0** です。 コンテキストの最後から 2 番目の部分、**mysqld_t** がタイプになります。 プロセスのドメインやファイルのタイプを定義するのがタイプです。 この例の場合、 **mysqld** プロセスは **mysqld_t** ドメイン内で実行しています。

10.2. タイプ

Type Enforcement が SELinux の targeted ポリシーで使用されるメインのパーミッション制御になります。全ファイルおよびプロセスにタイプのラベルが付けられます。ファイルの場合はタイプ、プロセスの場合はドメインを定義します。任意のタイプにアクセスするドメインなのか、 別のドメインにアクセスするドメインなのかなど、SELinux のポリシールールではタイプによって互いがアクセスしあう方法を定義します。 アクセスを許可する特定の SELinux ポリシールールが存在する場合にのみ、 そのアクセスが許可されます。

mysqld で使用されるタイプを以下に示します。 タイプに応じて柔軟なアクセス設定ができます。

mysqld_db_t

このタイプは MySQL データベースの場所に使用します。 Red Hat Enterprise Linux では、 データベースのデフォルトの場所は **/var/lib/mysql** になりますが、 この場所は変更することができます。 MySQL データベース用の場所を変更する場合は、 新しい場所にこのタイプのラベルを付ける必要があります。 デフォルトのデータベースの場所を変更する方法、 新しいセクションに適切にラベル付けを行なう方法について以下の例で説明していきます。

mysqld_etc_t

このタイプは、 MySQL のメイン設定ファイル **/etc/my.cnf** および **/etc/mysql** ディレクトリにある他の設定ファイルすべてに使用されます。

mysqld_exec_t

このタイプは **/usr/libexec/mysqld** にある **mysqld** バイナリに使用されます。 Red Hat Enterprise Linux では MySQL バイナリのデフォルトの場所になります。 他のシステムでは、 このバイナリは **/usr/sbin/mysqld** に配置されることがあります。 この場合にもこのタイプのラベルを付けるようにしてください。

mysqld_initrc_exec_t

このタイプは、 Red Hat Enterprise Linux ではデフォルトで **/etc/rc.d/init.d/mysqld** に配置されている MySQL の初期設定ファイルに使用されます。

mysqld_log_t

正常に動作させるため、 MySQL のログにはこのタイプのラベルを付ける必要があります。 **/var/log/** 内にあるログファイルで、 **mysql.*** のワイルドカードに一致するログファイルはすべて、このタイプのラベルを付ける必要があります。

mysqld_var_run_t

このタイプは **/var/run/mysqld** 内のファイルで使用されます。特に、**mysqld** デーモンの実行時に作成される**/var/run/mysqld/mysqld.pid** と言う名前のプロセス ID (PID) に使用されます。また、**/var/lib/mysql/mysql.sock** などの関連ソケットファイルにも使用されます。このようなファイルには、 制限のあるサービスとして正常に動作させるため適切なラベル付けが必要になります。

10.3. BOOLEAN

SELinux は実行するサービスに必要な最小限レベルのアクセスに基づいています。サービスの実行手段は複数あるため、サービスをどのように実行するのかを SELinux に指示する必要があります。 次の Boolean を使って MySQL の動作方法を SELinux に指示します。

`allow_user_mysql_connect`

この Boolean を有効にすると、 ユーザーによる MySQL への接続が可能になります。

`exim_can_connect_db`

この Boolean を有効にすると、 **`exim`** メーラーによるデータベースサーバーへの接続開始を許可します。

`ftpd_connect_db`

この Boolean を有効にすると、 **`ftp`** デーモンによるデータベースサーバーへの接続開始を許可します。

`httpd_can_network_connect_db`

この Boolean を有効にすると、 データベースサーバーとの通信に web サーバーが必要になります。

10.4. 設定例

10.4.1. MySQL のデータベース格納場所を変更する

Red Hat Enterprise Linux 6 を使用する場合、MySQL のデフォルトのデータベース格納場所は **`/var/lib/mysql`** になります。この場所は SELinux でデータベースが配置される場所として期待される場所となるため、この領域にはすでに **`mysqld_db_t`** タイプを使った適切なラベル付けが行なわれています。

データベースを格納する場所は、それぞれの環境要件や設定に応じて変更することもできますが、SELinux に変更後の新しい場所を認識させる、つまりラベル付けを行なうことが重要となります。MySQL データベースの格納場所を変更する方法、また SELinux によるコンテンツに応じた保護メカニズムを新しい格納場所にも適用できるようラベル付けを行なう方法を以下の例で説明していきます。

以下に示す例は、MySQL に対してどのように SELinux が影響を与えることができるのかを示す一例に過ぎません。MySQL に関する総合的な説明は本ガイドの範疇を越えてしまいますので、詳細については、公式の MySQL ドキュメント を参照してください。ここでは、mysql-server パッケージと setroubleshoot-server パッケージがインストールされていること、**`auditd`** サービスが実行されていること、有効なデータベースがデフォルトの場所である **`/var/lib/mysql`** にあることを前提としています。

1. **`ls -lZ /var/lib/mysql`** を実行し、**`mysql`** デフォルトデータベース格納場所の SELinux コンテキストを表示させます。

```
# ls -lZ /var/lib/mysql
drwx------. mysql mysql unconfined_u:object_r:mysqld_db_t:s0 mysql
```

データベースファイルの格納場所にデフォルトで付けられるコンテキストエレメントの **`mysqld_db_t`** が表示されています。本例で使用する新しいデータベース格納場所が期待通り正常に動作するよう、このコンテキストをその新しい場所に手作業で適用する必要があります。

2. **mysqlshow -u root -p** を入力し、使用できるデータベースを表示させるため**mysqld** の root パスワードを入力します。

```
# mysqlshow -u root -p
Enter password: *******
+--------------------+
|     Databases      |
+--------------------+
| information_schema |
| mysql              |
| test               |
| wikidb             |
+--------------------+
```

3. root ユーザーで **service mysqld stop** を実行し、**mysqld** デーモンをシャットダウンさせます。

```
# service mysqld stop
Stopping MySQL:                                            [  OK
]
```

4. データベース格納場所となるディレクトリを新規作成します。この例では **/mysql** を使用しています。

```
# mkdir -p /mysql
```

5. 古い場所にあるデータベースファイルを新しい場所にコピーします。

```
# cp -R /var/lib/mysql/* /mysql/
```

6. mysql ユーザーおよび mysql グループによるアクセスを許可するため所有権を変更します。この変更は従来の Unix パーミッションに対する変更であり、SELinux による制限はそのまま変更されていません。

```
# chown -R mysql:mysql /mysql
```

7. **ls -lZ /opt** を実行し、新規ディレクトリの初期コンテキストを表示してみます。

```
# ls -lZ /opt
drwxr-xr-x. mysql mysql unconfined_u:object_r:usr_t:s0   mysql
```

コンテキスト **usr_t** は、現在、MySQL データベースファイルの格納場所として適したタイプではありません。コンテキストを変更すると、MySQL がこの場所で正しく動作できるようになります。

8. MySQL のメインとなる設定ファイル **/etc/my.cnf** をテキストエディターで開き、新しい格納場所を参照するよう **datadir** オプションを編集します。この例の場合なら、新しい格納場所として入力するのは **/mysql** になります。

```
[mysqld]
datadir=/mysql
```

このファイルを保存してから終了します。

9. root ユーザーで **service mysqld start** を実行し、**mysqld** を起動します。サービスの起動は失敗し、**/var/log/messages** ファイルに拒否のログが記録されます。ただし、**audit** デーモンおよび **setroubleshoot** サービスが実行されている場合は、**/var/log/audit/audit.log** ファイルの方に拒否のログが記録されます。

```
SELinux is preventing /usr/libexec/mysqld "write" access on /mysql.
For complete SELinux messages. run sealert -l b3f01aff-7fa6-4ebe-
ad46-abaef6f8ad71
```

拒否の理由は、**/mysql** が MySQL データファイル用として適切なラベルが付けられていないためです。SELinux で、MySQL による **usr_t** タイプのラベルが付いたコンテンツへのアクセスが阻止されています。この問題を解決するため次の手順を行なってください。

10. 次の **semanage** コマンドを実行し、**/mysql** のコンテキストマッピングを追加します。**semanage** はデフォルトではインストールされていないため注意してください。インストールされていない場合は、policycoreutils-python パッケージをインストールします。

```
semanage fcontext -a -t mysqld_db_t "/mysql(/.*)?"
```

11. このマッピングは **/etc/selinux/targeted/contexts/files/file_contexts.local** ファイルに書き込まれます。

```
# grep -i mysql
/etc/selinux/targeted/contexts/files/file_contexts.local

/mysql(/.*)?    system_u:object_r:mysqld_db_t:s0
```

12. **restorecon** コマンドを使ってこのコンテキストマッピングを実行中のシステムに適用します。

```
restorecon -R -v /mysql
```

13. これで **/mysql** のデータ格納場所に MySQL 用の正しいコンテキストがラベル付けされました。**mysqld** デーモンを起動します。

```
# service mysqld start
Starting MySQL:                                            [  OK
]
```

14. コンテキストが確かに **/mysql** 用に正しく変更されているか確認します。

```
ls -lZ /opt
drwxr-xr-x. mysql mysql system_u:object_r:mysqld_db_t:s0 mysql
```

15. データ格納場所の変更そしてラベル付けが正しく行なわれたため、**mysqld** デーモンが正常に起動するようになりました。ここまでの設定が完了したら、実行中の全サービスが正常に動作しているか確認テストを行なってください。

第11章 POSTGRESQL

PostgreSQL プロジェクトページより抜粋:

原文: "PostgreSQL is a powerful, open source object-relational database system. It has more than 15 years of active development and a proven architecture that has earned it a strong reputation for reliability, data integrity, and correctness." (訳文: PostgreSQL はオープンソースでありパワフルなオブジェクト関係データベースシステムになります。15年以上に渡り活発な開発が行なわれ、信頼性、データ整合性、正確性に強化な評判を得ている実績のあるアーキテクチャを提供しています。)

Red Hat Enterprise Linux 6 では、PostgreSQL は postgresql-server パッケージで提供されます。**rpm -q postgresql-server**を実行して、postgresql-server パッケージがインストールされているか確認してください。インストールされていない場合は、root ユーザーで次のコマンドを実行してインストールを行ないます。

```
yum install postgresql-server
```

11.1. POSTGRESQL と SELINUX

PostgreSQL を有効にすると、デフォルトで制限のあるサービスとして実行されます。制限のあるプロセスはそれ自体のドメイン内で実行され、他の制限のあるプロセスとは分離されます。制限のあるプロセスが攻撃を受けると、SELinux ポリシー設定に応じて、攻撃側がリソースにアクセスして加えることができる被害は限定されます。以下に、PostgreSQL 自体のドメイン内で実行している PostgreSQL プロセスの例を示します。 ここでは postgresql-server パッケージがインストールされていることを前提としています。

1. **getenforce** を実行して、SELinux が enforcing モードで実行しているか確認します。

   ```
   $ getenforce
   Enforcing
   ```

   SELinux が enforcing モードで実行している場合は、**getenforce** コマンドを実行すると **Enforcing** が返されます。

2. root ユーザーで **service postgresql start** を実行し、**postgresql** を起動します。

   ```
   service postgresql start
   Starting postgresql service:                               [  OK
   ]
   ```

3. **ps -eZ | grep postgres** を実行して、**postgresql** プロセスを表示させます。

   ```
   ps -eZ | grep postgres
   unconfined_u:system_r:postgresql_t:s0 395 ?    00:00:00 postmaster
   unconfined_u:system_r:postgresql_t:s0 397 ?    00:00:00 postmaster
   unconfined_u:system_r:postgresql_t:s0 399 ?    00:00:00 postmaster
   unconfined_u:system_r:postgresql_t:s0 400 ?    00:00:00 postmaster
   unconfined_u:system_r:postgresql_t:s0 401 ?    00:00:00 postmaster
   unconfined_u:system_r:postgresql_t:s0 402 ?    00:00:00 postmaster
   ```

   postgresql プロセスに関連する SELinux コンテキストは **unconfined_u:system_r:postgresql_t:s0** です。コンテキストの最後から 2 番目の部分、 **postgresql_t** がタイプになります。プロセスのドメインやファイルのタイプを定義す

るのがタイプです。この例の場合、 **postgresql** プロセスは **postgresql_t** ドメイン内で実行しています。

11.2. タイプ

Type Enforcement が SELinux の targeted ポリシーで使用されるメインのパーミッション制御になります。全ファイルおよびプロセスにタイプのラベルが付けられます。ファイルの場合はタイプ、プロセスの場合はドメインを定義します。任意のタイプにアクセスするドメインなのか、別のドメインにアクセスするドメインなのかなど、SELinux のポリシールールではタイプによって互いがアクセスしあう方法を定義します。アクセスを許可する特定の SELinux ポリシールールが存在する場合にのみ、そのアクセスが許可されます。

postgresql で使用されるタイプを以下に示します。タイプに応じて柔軟なアクセス設定ができます。

postgresql_db_t

このタイプは複数の場所で使用されます。このタイプでラベル付けされた場所は PostgreSQL のデータファイル用に使用されます。

- **/usr/lib/pgsql/test/regres**
- **/usr/share/jonas/pgsql**
- **/var/lib/pgsql/data**
- **/var/lib/postgres(ql)?**

postgresql_etc_t

このタイプは **/etc/postgresql** 内の設定ファイルに使用されます。

postgresql_exec_t

このタイプは複数の場所で使用されます。このタイプでラベル付けされた場所は PostgreSQL のバイナリに使用されます。

- **/usr/bin/initdb(.sepgsql)?**
- **/usr/bin/(se)?postgres**
- **/usr/lib(64)?/postgresql/bin/.***
- **/usr/lib/phsql/test/regress/pg_regress**

postgresql_initrc_exec_t

このタイプは **/etc/rc.d/init.d/postgresql** にある PostgreSQL 初期設定ファイルに使用されます。

postgresql_log_t

このタイプは複数の場所で使用されます。このタイプでラベル付けされた場所はログファイルに使用されます。

- **/var/lib/pgsql/logfile**
- **/var/lib/pgsql/pgstartup.log**

- **/var/lib/sepgsql/pgstartup.log**
- **/var/log/postgresql**
- **/var/log/postgres.log.***
- **/var/log/rhdb/rhdb**
- **/var/log/sepostgresql.log.***

postgresql_var_run_t

このタイプは、**/var/run/postgresql** 内のプロセス ID (PID) など PostgreSQL のランタイムファイルに使用されます。

11.3. BOOLEAN

SELinux は実行するサービスに必要な最小限レベルのアクセスに基づいています。サービスの実行手段は複数あるため、サービスをどのように実行するのかを SELinux に指示する必要があります。 次の Boolean を使って PostgreSQL の動作方法を SELinux に指示します。

allow_user_postgresql_connect

この Boolean を有効にすると、いずれのユーザードメイン (PostgreSQL の定義) からのデータサーバーへの接続も許可します。

11.4. 設定例

11.4.1. PostgreSQL のデータベース格納場所を変更する

Red Hat Enterprise Linux 6 を使用する場合、PostgreSQL のデフォルトのデータベース格納場所は **/var/lib/pgsql/data** になります。この場所は SELinux でデータベースが配置される場所として期待される場所となるため、この領域にはすでに **postgresql_db_t** タイプを使った適切なラベル付けが行なわれています。

データベースを格納する場所は、それぞれの環境要件や設定に応じて変更することもできますが、SELinux に変更後の新しい場所を認識させる、つまりラベル付けを行なうことが重要となります。PostgreSQL データベースの格納場所を変更する方法、また SELinux によるコンテンツに応じた保護メカニズムを新しい格納場所にも適用できるようラベル付けを行なう方法を以下の例で説明していきます。

以下に示す例は、PostgreSQL に対してどのように SELinux が影響を与えることができるのかを示す一例に過ぎません。PostgreSQL に関する総合的な説明は本ガイドの範疇を越えてしまいますので、詳細については、公式の PostgreSQL ドキュメント を参照してください。ここでは、postgresql-server パッケージがインストールされていることを前提としています。

1. **ls -lZ /var/lib/pgsql** を実行し、**postgresql** デフォルトデータベース格納場所の SELinux コンテキストを表示させます。

```
# ls -lZ /var/lib/pgsql
drwx------. postgres postgres system_u:object_r:postgresql_db_t:s0
data
```

データベースファイルの格納場所にデフォルトで付けられるコンテキストエレメントの **postgresql_db_t** が表示されています。本例で使用する新しいデータベース格納場所が期待通り正常に動作するよう、このコンテキストをその新しい場所に手作業で適用する必要があります。

2. データベース格納場所となるディレクトリを新規作成します。この例では **/opt/postgresql/data** を使用しています。別の場所を使用する場合は、次のコマンドを使用する際に置換してください。

```
# mkdir -p /opt/postgresql/data
```

3. 新規に作成したディレクトリを表示させます。このディレクトリの初期コンテキストは **usr_t** になっている点に注意してください。このコンテキストでは、SELinux による PostgreSQL への保護メカニズムを実施するには不十分です。コンテキストを変更することにより、新規に作成したディレクトリをデータ格納場所として適切に動作させることができるようになります。

```
# ls -lZ /opt/postgresql/
drwxr-xr-x. root root unconfined_u:object_r:usr_t:s0   data
```

4. postgres ユーザーおよび postgres グループによるアクセスを許可するため所有権を変更します。この変更は従来の Unix パーミッションに対する変更であり、SELinux による制限はそのまま変更されていません。

```
# chown -R postgres:postgres /opt/postgresql
```

5. テキストエディターで PostgreSQL の初期設定ファイル **/etc/rc.d/init.d/postgresql** を開き、新しい場所をポイントするよう **PGDATA** と **PGLOG** を変更します。

```
# vi /etc/rc.d/init.d/postgresql
PGDATA=/opt/postgresql/data
PGLOG=/opt/postgresql/data/pgstartup.log
```

このファイルを保存してからテキストエディターを終了します。

6. 新しい場所にあるデータベースを初期化します。

```
su - postgres -c "initdb -D /opt/postgresql/data"
```

7. データベースの場所を変更したことにより、サービスの起動に失敗します。

```
# service postgresql start
Starting postgresql service:    [FAILED]
```

サービスが起動しない原因は SELinux にあります。新しい場所に適切なラベル付けが行なわれていないためです。以下の手順で、新しい場所 **(/opt/postgresql)** にラベルを付け、postgresql サービスを正常に起動させます。

8. **semanage** コマンドを実行し、**/opt/postgresql** および配下にあるすべてのディレクトリとファイルに対するコンテキストマッピングを追加します。

```
semanage fcontext -a -t postgresql_db_t "/opt/postgresql(/.*)?"
```

9. このマッピングは **/etc/selinux/targeted/contexts/files/file_contexts.local** ファイルに書き込まれます。

```
# grep -i postgresql
/etc/selinux/targeted/contexts/files/file_contexts.local

/opt/postgresql(/.*)?    system_u:object_r:postgresql_db_t:s0
```

10. **restorecon** コマンドを使ってこのコンテキストマッピングを実行中のシステムに適用します。

```
restorecon -R -v /opt/postgresql
```

11. これで **/opt/postgresql** のデータ格納場所に PostgreSQL 用の正しいコンテキストがラベル付けされました。**postgresql** サービスが正常に起動するようになります。

```
# service postgresql start
Starting postgreSQL service:
[  OK  ]
```

12. コンテキストが確かに **/opt/postgresql** 用に正しく変更されているか確認します。

```
ls -lZ /opt
drwxr-xr-x. root root system_u:object_r:postgresql_db_t:s0
postgresql
```

13. **ps** コマンドを使って、**postgresql** プロセスで新しい場所が表示されるか確認します。

```
# ps aux | grep -i postmaster

postgres 21564  0.3  0.3  42308  4032 ?        S    10:13   0:00
/usr/bin/postmaster -p 5432 -D /opt/postgresql/data
```

14. データ格納場所の変更そしてラベル付けが正しく行なわれたため、**postgresql** デーモンが正常に起動するようになりました。ここまでの設定が完了したら、実行中の全サービスが正常に動作しているか確認テストを行なってください。

第12章 RSYNC

Rsync プロジェクトページより抜粋:

原文: "rsync is an open source utility that provides fast incremental file transfer." (訳文: rsync は、高速な増分ファイル転送を実現するオープンソースのユーティリティです。)

Red Hat Enterprise Linux を使用する場合、rsync は rsync パッケージで提供されます。**rpm -q rsync** を実行して、rsync パッケージがインストールされているか確認してください。インストールされていない場合は、root ユーザーで次のコマンドを実行してインストールを行ないます。

```
yum install rsync
```

12.1. RSYNC と SELINUX

Red Hat Enterprise Linux 6 SELinux **rsync_selinux(8)** man ページより抜粋: "SELinux requires files to have an extended attribute to define the file type. Policy governs the access daemons have to these files. If you want to share files using the rsync daemon, you must label the files and directories **public_content_t**." (訳文: SELinux にはファイルタイプ定義の拡張属性を持たせるためのファイルが必要になります。ポリシーでは、こうしたファイルに対してデーモンに持たせるアクセス権を管理します。rsync デーモンを使ってファイルの共有をする場合は、ファイルやディレクトリに**public_content_t** タイプのラベルをつける必要があります。)

ほとんどのサービスと同様、 **rsync** に対して SELinux による保護メカニズムを適用させるには、適切なラベル付けが必要になります。

12.2. タイプ

Type Enforcement が SELinux の targeted ポリシーで使用されるメインのパーミッション制御になります。全ファイルおよびプロセスにタイプのラベルが付けられます。ファイルの場合はタイプ、プロセスの場合はドメインを定義します。任意のタイプにアクセスするドメインなのか、 別のドメインにアクセスするドメインなのかなど、SELinux のポリシールールではタイプによって互いがアクセスしあう方法を定義します。 アクセスを許可する特定の SELinux ポリシールールが存在する場合にのみ、 そのアクセスが許可されます。

rsync で使用されるタイプを以下に示します。 タイプに応じて柔軟なアクセス設定ができます。

public_content_t

rsync で共有するファイルの場所に使用する汎用タイプになります。**rsync** で共有するファイルの格納用に特殊なディレクトリを作成する場合は、そのディレクトリおよびコンテンツにはこのラベルを適用する必要があります。

rsync_exec_t

/usr/bin/rsync システムバイナリに使用されるタイプです。

rsync_log_t

デフォルトで **/var/log/rsync.log** にある **rsync** ログファイルに使用されます。rsync によりログが記録されるファイルの場所を変更する場合は、ランタイム時に **rsync** コマンドに対して **--log-file=FILE** オプションを使用します。

rsync_var_run_t

/var/run/rsyncd.lock にある **rsyncd** ロックファイルに使用されるタイプです。このロックファイルは **rsync** サーバーで接続関連の制限を管理する際に使用されます。

rsync_data_t

ファイルやディレクトリを rsync ドメインとして使用し、他のサービスのアクセス範囲とは分離させたい場合、このタイプを使用します。また、**public_content_t** は汎用の SELinux コンテキストになります。複数のサービスとやりとりを行なうファイルやディレクトリに使用します (rsync ドメインとしての FTP ディレクトリおよび NFS ディレクトリ)。

rsync_etc_t

/etc/ ディレクトリ内にある rsync 関連のファイルに使用されます。

12.3. BOOLEAN

SELinux は実行するサービスに必要な最小限レベルのアクセスに基づいています。サービスの実行手段は複数あるため、サービスをどのように実行するのかを SELinux に指示する必要があります。次の Boolean を使って rsync の動作方法を SELinux に指示します。

allow_rsync_anon_write

この Boolean を有効にすると、**rsync_t** ドメイン内の **rsync** による **public_content_rw_t** タイプのファイル、リンク、ディレクトリなどの管理を許可します。多くの場合、パブリック転送サービスに使用されるパブリックファイルになります。ファイルおよびディレクトリには **public_content_rw_t** のラベルを付ける必要があります。

rsync_client

この Boolean を有効にすると、**rsync_port_t** で定義されるポートへの **rsync** による接続開始を許可し、また **rsync_data_t** タイプのファイル、リンク、ディレクトリの管理も許可されます。**rsync** を SELinux の管理下に置くため、**rsync** デーモンは **rsync_t** ドメイン内になければならない点に注意してください。本章では、**rsync_t** ドメインで実行している **rsync** の設定例を示します。

rsync_export_all_ro

この Boolean を有効にすると、**rsync_t** ドメイン内の **rsync** による NFS および CIFS のエクスポートを許可します。クライアントに付与するアクセス権は読み取り専用になります。

12.4. 設定例

12.4.1. デーモンとして rsync を使用する

Red Hat Enterprise Linux を使用する場合、rsync をデーモンとして使用し、一元的にファイルを格納、継続的に同期しておくためのセントラルサーバーとして複数のクライアントが直接通信を行なえるようにすることができます。以下では、rsync を適切なドメイン内のネットワークソケット全体でデーモンとして実行させようした場合、事前定義された TCP ポート (SELinux ポリシー内) での実行を期待している SELinux がどのような反応を示すかについて見ていきます。次に、非標準のポートでの **rsync** デーモンによる正常な実行を許可するため SELinux を編集する方法について説明していきます。

SELinux ポリシーとローカルのデーモンおよびプロセスに対する制御力を示すため、本例は単一のシステム上で行います。以下に示す例は、rsync に対してどのように SELinux が影響を与えることができるのかを示す一例に過ぎません。rsync に関する総合的な説明は本ガイドの範疇を越えてしまいますの

で、詳細については、公式の rsync ドキュメント を参照してください。ここでは、rsync パッケージ、setroubleshoot-server パッケージ、audit パッケージがインストールされていること、SELinux targeted ポリシーを使用していること、SELinux が enforcing モードで実行されていることを前提としています。

rsync を rsync_t として起動させる

1. **getenforce** を実行して、 SELinux が enforcing モードで実行していることを確認します。

```
$ getenforce
Enforcing
```

SELinux が enforcing モードで実行している場合は、 **getenforce** コマンドを実行すると **Enforcing** が返されます。

2. **which** コマンドを実行し、rsync バイナリがシステムパス内にあるか確認します。

```
$ which rsync
/usr/bin/rsync
```

3. **rsync** をデーモンとして実行する場合、**/etc/rsyncd.conf** と言う名前を付けた設定ファイルを使用する必要があります。ここで使用している設定ファイルは非常に簡潔なファイルになっているため、利用できるすべてのオプションが表示されているわけではありません。**rsync** デーモンの事例として必要なものを備えているだけです。

```
log file = /var/log/rsync.log
pid file = /var/run/rsyncd.pid
lock file = /var/run/rsync.lock
[files]
        path = /srv/files
        comment = file area
        read only = false
 timeout = 300
```

4. これで、デーモンモードで動作させる rsync 用の簡単な設定ファイルができました。このステップでは、SELinux による保護メカニズムを rsync に適用させるには、 **rsync --daemon** を実行するだけでは不十分であることを確認します。次の出力を見てみてください。

```
# rsync --daemon

# ps x | grep rsync
 8231 ?        Ss     0:00 rsync --daemon
 8233 pts/3    S+     0:00 grep rsync

# ps -eZ | grep rsync
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 8231 ?
00:00:00 rsync
```

最後の **ps** コマンドからの出力に注目してください。コンテキストでは**rsync**デーモンは **unconfined_t** デーモン内で実行していることを示しています。つまり、rsync が**rsync --daemon** コマンドで起動されたため、**rsync_t** には遷移していないということです。この状態では、SELinux はこのデーモンに対してルールとポリシーを施行することができません。この問題を解決するため次のステップを見てみます。次のステップでは、**rsync** を適切にラベル付

けした init スクリプトから起動させるため、**rsync_t** に遷移するようになります。これではじめて SELinux とその保護メカニズムを **rsync** に適用できるようになります。**rsync** プロセスを終了してから、次のステップに進みます。

5. rsync 用のカスタムの init スクリプトには次のステップが必要になります。次を **/etc/rc.d/init.d/rsyncd** に保存します。

```
#!/bin/bash

# Source function library.
. /etc/rc.d/init.d/functions

[ -f /usr/bin/rsync ] || exit 0

case "$1" in
start)
action "Starting rsyncd: " /usr/bin/rsync --daemon
;;
stop)
action "Stopping rsyncd: " killall rsync
;;
*)
echo "Usage: rsyncd {start|stop}"
exit 1
esac
exit 0
```

以下のようにして、このスクリプトに **initrc_exec_t** タイプのラベルを付けます。

6. **semanage** コマンドを実行し、**/etc/rc.d/init.d/rsyncd** のコンテキストマッピングを追加します。

```
semanage fcontext -a -t initrc_exec_t "/etc/rc.d/init.d/rsyncd"
```

7. このマッピングは **/etc/selinux/targeted/contexts/files/file_contexts.local** ファイルに書き込まれます。

```
# grep rsync
/etc/selinux/targeted/contexts/files/file_contexts.local

/etc/rc.d/init.d/rsyncd    system_u:object_r:initrc_exec_t:s0
```

8. **restorecon** コマンドを使ってこのコンテキストマッピングを実行中のシステムに適用します。

```
restorecon -R -v /etc/rc.d/init.d/rsyncd
```

9. **ls -lZ** コマンドを実行して、確かにスクリプトに適切なタイプのラベルが付けられているかを確認します。以下の出力では、スクリプトには **initrc_exec_t** タイプのラベルが付けられています。

```
 ls -lZ /etc/rc.d/init.d/rsyncd
-rwxr-xr-x. root root system_u:object_r:initrc_exec_t:s0
/etc/rc.d/init.d/rsyncd
```

10. 新しいスクリプトで **rsyncd** を起動します。rsync が適切にラベル付けした init スクリプトから起動されるようになりました。プロセスは **rsync_t** として開始されるようになります。

```
# service rsyncd start
Starting rsyncd:                                           [  OK
]

ps -eZ | grep rsync
unconfined_u:system_r:rsync_t:s0 9794 ?        00:00:00 rsync
```

これで、rsync が **rsync_t** ドメイン内で実行するようになったため、SELinux ではその保護メカニズムを **rsync** デーモンに適用できるようになります。

rsyncd を **rsync_t** ドメイン内で実行させる方法について説明してきました。次に、このデーモンをデフォルト以外のポートで適切に実行させる方法について見ていきます。ここでは TCP ポート 10000 を使用します。

デフォルト以外のポートで rsync デーモンを実行する

1. **/etc/rsyncd.conf** ファイルを変更して、**port = 10000** の行をグローバル設定エリア内にあるファイルの冒頭に追加します (file エリアが定義される直前)。新しい設定ファイルは次のようになります。

```
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
lock file = /var/run/rsync.lock
port = 10000
[files]
        path = /srv/files
        comment = file area
        read only = false
 timeout = 300
```

2. この新しい設定の init スクリプトから rsync を起動すると、次のような拒否が SELinux によりログ記録されます。

```
Jul 22 10:46:59 localhost setroubleshoot: SELinux is preventing the
rsync (rsync_t) from binding to port 10000. For complete SELinux
messages. run sealert -l c371ab34-639e-45ae-9e42-18855b5c2de8
```

3. **semanage** コマンドを実行して、TCP ポート 10000 を **rsync_port_t** の SELinux ポリシーに追加します。

```
# semanage port -a -t rsync_port_t -p tcp 10000
```

4. これで TCP ポート 10000 が **rsync_port_t** の SELinux ポリシーに追加されました。**rsyncd** がこのポートで正常に起動し動作するようになります。

```
# service rsyncd start
```

```
Starting rsyncd:                                           [  OK
]
```

```
# netstat -lnp | grep 10000
tcp        0      0 0.0.0.0:10000   0.0.0.0:*       LISTEN
9910/rsync
```

SELinux のポリシーが修正されたため、**rsyncd** による TCP ポート 10000 での動作が許可されるようになりました。

第13章 POSTFIX

Postfix プロジェクトページより抜粋:

原文: "What is Postfix? It is Wietse Venema's mailer that started life at IBM research as an alternative to the widely-used Sendmail program. Postfix attempts to be fast, easy to administer, and secure. The outside has a definite Sendmail-ish flavor, but the inside is completely different." (訳文: Postfix とは? 幅広く利用されていた Sendmail プログラムの代替となるプログラムの調査が IBM によって行なわれたのが始まりとなる Wietse Venema のメーラーです。Postfix では高速で管理が容易でありながら安全なメーラーを目指しています。外観は Sendmail 色を色濃く残していますが、プログラム自体はまったく異なる仕様になっています。)

Red Hat Enterprise Linux では、 postfix は postfix パッケージで提供されます。 **rpm -q postfix** を実行して、 postfix パッケージがインストールされているか確認してください。 インストールされていない場合は、 root ユーザーで次のコマンドを実行してインストールを行ないます。

```
yum install postfix
```

13.1. POSTFIX と SELINUX

Postfix を有効にすると、デフォルトで制限のあるサービスとして実行されます。制限のあるプロセスはそれ自体のドメイン内で実行され、他の制限のあるプロセスとは分離されます。制限のあるプロセスが攻撃を受けると、SELinux ポリシー設定に応じて、攻撃側がリソースにアクセスして加えることができる被害は限定されます。以下に、Postfix 自体のドメイン内で実行している Postfix プロセスの例を示します。 ここでは postfix パッケージがインストールされていること、また Postfix サービスが起動されていることを前提としています。

1. **getenforce** を実行して SELinux が enforcing モードで実行しているか確認します。

   ```
   $ getenforce
   Enforcing
   ```

   SELinux が enforcing モードで実行している場合は、 **getenforce** コマンドを実行すると **Enforcing** が返されます。

2. root ユーザーで **service postfix start** を実行し、 **postfix** を起動します。

   ```
   service postfix start
   Starting postfix:                                  [  OK  ]
   ```

3. **ps -eZ | grep postfix** を実行し、 **postfix** プロセスを表示させます。

   ```
   ps -eZ | grep postfix
   system_u:system_r:postfix_master_t:s0 1651 ?   00:00:00 master
   system_u:system_r:postfix_pickup_t:s0 1662 ?   00:00:00 pickup
   system_u:system_r:postfix_qmgr_t:s0 1663 ?     00:00:00 qmgr
   ```

   たとえば、Postfix **master** プロセスに関連する SELinux コンテキストは **unconfined_u:system_r:postfix_master_t:s0** です。 コンテキストの最後から 2 番目の部分、 **postfix_master_t** がこのプロセスのタイプになります。 プロセスのドメインやファイルのタイプを定義するのがタイプです。 この例の場合、 **master** プロセスは **postfix_master_t** ドメイン内で実行しています。

13.2. タイプ

Type Enforcement が SELinux の targeted ポリシーで使用されるメインのパーミッション制御になります。全ファイルおよびプロセスにタイプのラベルが付けられます。ファイルの場合はタイプ、プロセスの場合はドメインを定義します。任意のタイプにアクセスするドメインなのか、 別のドメインにアクセスするドメインなのかなど、SELinux のポリシールールではタイプによって互いがアクセスしあう方法を定義します。アクセスを許可する特定の SELinux ポリシールールが存在する場合にのみ、そのアクセスが許可されます。

Postfix で使用されるタイプを以下に示します。 タイプに応じて柔軟なアクセス設定ができます。

postfix_etc_t

/etc/postfix/ ディレクトリ内にある Postfix 用の設定ファイルに使用されるタイプです。

postfix_data_t

/var/lib/postfix/ ディレクトリ内にある Postfix データファイル用に使用されるタイプです。

postfix_var_run_t

/run/ ディレクトリ内に格納される Postfix ファイルに使用されるタイプです。

postfix_initrc_exec_t

Postfix 実行可能ファイルの **postfix_initrc_t** ドメインへの遷移に使用されるタイプです。

postfix_spool_t

/var/spool/ ディレクトリ内に格納される Postfix ファイルに使用されるタイプです。

注記

次のコマンドを実行すると、Postfix 用のタイプとファイルの全一覧を表示させることができます。

```
$ grep postfix
/etc/selinux/targeted/contexts/files/file_contexts
```

13.3. BOOLEAN

SELinux は実行するサービスに必要な最小限レベルのアクセスに基づいています。サービスの実行手段は複数あるため、サービスをどのように実行するのかを SELinux に指示する必要があります。 次の Boolean を使って Postfix の動作方法を SELinux に指示します。

allow_postfix_local_write_mail_spool

この Boolean を有効にすると、Postfix によるシステム上のローカルメールスプールへの書き込みを許可します。ローカルスプールを使用する際、Postfix を正常に動作させるためにはこの Boolean を有効にする必要があります。

13.4. 設定例

13.4.1. SpamAssassin と Postfix

SpamAssassin プロジェクトページより抜粋:

原文: "Open Source mail filter, written in Perl, to identify spam using a wide range of heuristic tests on mail headers and body text. Free software." (訳文: Perl で記述されたオープンソースのメールフィルターです。メールのヘッダーおよび本文で幅広い範囲の発見的テストを使用してスパムを識別するフリーソフトウェアです。)

Red Hat Enterprise Linux を使用する場合、SpamAssassin は spamassassin パッケージで提供されます。**rpm -q spamassassin** を実行して、spamassassin パッケージがインストールされているか確認してください。インストールされていない場合は、root ユーザーで次のコマンドを実行してインストールを行ないます。

```
yum install spamassassin
```

SpamAssassin は Postfix などのメーラーと連携してスパムフィルタリング機能を提供します。メールの効果的な遮断、分析、フィルタリングが行なわれるためには、SpamAssassin はネットワークインターフェース上でリッスンを行なう必要があります。SpamAssassin のデフォルトポートは TCP/783 ですが、変更することもできます。SELinux で特定のポートに限ったアクセスをデフォルトで許可することにより SpamAssassin を補完している実践的な例を以下に示します。次に、ポートを変更する方法およびデフォルト以外のポートで SpamAssassin を正常に動作させる方法について説明していきます。

以下に示す例は、 シンプルな SpamAssassin 設定に対してどのように SELinux が影響を与えることができるのかを示す一例に過ぎません。 SpamAssassin に関する総合的な説明は本ガイドの範疇を越えてしまいますので、 詳細については、 公式の SpamAssassin ドキュメント を参照してください。 ここでは、spamassassin がインストールされていること、使用しているポートでのアクセス許可がファイアウォールで設定されていること、SELinux が enforcing モードで実行されていることを前提としています。

デフォルト以外のポートで SpamAssassin を実行する

1. **semanage** コマンドを実行して、SELinux で **spamd** によるリッスンをデフォルトで許可しているポートを表示させます。

   ```
   # semanage port -l | grep spamd
   spamd_port_t  tcp 783
   ```

   上記の出力では、SpamAssassin が動作するポートとして TCP/783 が **spamd_port_t** で定義されていることを示しています。

2. **/etc/sysconfig/spamassassin** 設定ファイルを編集し、SpamAssassin が TCP/10000 で起動するよう変更します。

   ```
   # Options to spamd
   SPAMDOPTIONS="-d -p 10000 -c m5 -H"
   ```

   上記の行は、SpamAssassin がポート 10000 で動作するよう指定しています。ここからは、このソケットを開くよう SELinux ポリシーを変更する方法を見ていきます。

3. SpamAssassin を起動すると、次のようなエラーメッセージが表示されます。

   ```
   # service spamassassin start
   Starting spamd: [2203] warn: server socket setup failed, retry 1:
   spamd: could not create INET socket on 127.0.0.1:10000: Permission
   ```

```
denied
[2203] warn: server socket setup failed, retry 2: spamd: could not
create INET socket on 127.0.0.1:10000: Permission denied
[2203] error: spamd: could not create INET socket on
127.0.0.1:10000: Permission denied
spamd: could not create INET socket on 127.0.0.1:10000: Permission
denied

[FAILED]
```

上記の出力は、このポートへのアクセスが SELinux によってブロックされたことを表しています。

4. SELinux により次のような拒否がログ記録されます。

```
SELinux is preventing the spamd (spamd_t) from binding to port
10000.
```

5. root ユーザーで **semanage** コマンドを実行し、SpamAssassin のサンプルポート (TCP/10000) での動作を許可するよう SELinux ポリシーを変更します。

```
semanage port -a -t spamd_port_t -p tcp 10000
```

6. SpamAssassin が起動し、TCP ポート 10000 で動作していることを確認します。

```
# service spamassassin start
Starting spamd:     [ OK ]

# netstat -lnp | grep 10000
tcp 0 0 127.0.0.1:10000 0.0.0.0:* LISTEN 2224/spamd.pid
```

7. SELinux ポリシーで **spamd** による TCP ポート 10000 へのアクセスが許可されたため、SpamAssassin がこのポートで正常に動作するようになりました。

第14章 DHCP

クライアントに第 3 層 TCP/IP を動的に提供、詳細を設定するため Red Hat Enterprise Linux で使用されるデーモンが DHCPD になります。

DHCP サーバーの **dhcpd** は、 **dhcp** パッケージで提供されます。**rpm -q dhcp** を実行して、 **dhcp** パッケージがインストールされているか確認します。 インストールされていない場合は、 **root** ユーザーで次のコマンドを実行してインストールしてください。

```
yum install dhcp
```

14.1. DHCP と SELINUX

DHCP を有効にすると、デフォルトで制限のあるサービスとして実行されます。制限のあるプロセスはそれ自体のドメイン内で実行され、他の制限のあるプロセスとは分離されます。制限のあるプロセスが攻撃を受けると、SELinux ポリシー設定に応じて、攻撃側がリソースにアクセスして加えることができる被害は限定されます。以下に、DHCPD 自体のドメイン内で実行している DHCPD と関連プロセスの例を示します。ここでは **dhcp** パッケージがインストールされていること、また DHCPD サービスが起動されていることを前提としています。

1. **getenforce** を実行して、 SELinux が enforcing モードで実行しているか確認します。

   ```
   $ getenforce
   Enforcing
   ```

   SELinux が enforcing モードで実行されている場合は、 **getenforce** コマンドにより **Enforcing** が返されます。

2. root ユーザーで **service dhcpd start** を実行し、 **DHCPD** を起動します。

   ```
   service dhcpd start
   Starting dhcpd:                                    [  OK  ]
   ```

3. **ps -eZ | grep dhcpd** を実行し、 **dhcpd** プロセスを表示させます。

   ```
   ps -eZ | grep dhcpd
   unconfined_u:system_r:dhcpd_t:s0 5483 ?        00:00:00 dhcpd
   ```

   dhcpd プロセスに関連する SELinux コンテキストは **unconfined_u:system_r:dhcpd_t:s0** です。

14.2. タイプ

dhcpd で使用されるタイプを以下に示します。

dhcp_etc_t

設定ファイルなど、**/etc** 内のファイルに主に使用されるタイプです。

dhcpd_var_run_t

/var/run 内にある dhcpd 用の PID ファイルに使用されるタイプです。

dhcpd_exec_t

DHCP 実行可能ファイルの **dhcpd_t** ドメインへの遷移に使用されるタイプです。

dhcpd_initrc_exec_t

DHCP 実行可能ファイルの **dhcpd_initrc_t** ドメインへの遷移に使用されるタイプです。

注記

次のコマンドを実行すると、**dhcp** 用のタイプとファイルの全一覧を表示させることができます。

```
$ grep dhcp /etc/selinux/targeted/contexts/files/file_contexts
```

第15章 参考文献

本ガイドではその範疇を越えてしまう SELinux 関連の詳細が記述されている書籍および資料などを以下にいくつか挙げておきます。SELinux は急速なスピードで開発されているため、記述の一部は Red Hat Enterprise Linux の特定リリースにしか適用できない場合があるため注意してください。

書籍

SELinux by Example

Mayer、MacMillan、Caplan 著

2007年、Prentice Hall 出版

SELinux: NSA's Open Source Security Enhanced Linux

Bill McCarty 著

2004年、O'Reilly Media Inc. 出版

チュートリアルとヘルプ

Russell Coker 氏によるチュートリアルとトーク

http://www.coker.com.au/selinux/talks/ibmtu-2004/

Dan Walsh 氏のジャーナル

http://danwalsh.livejournal.com/

Red Hat ナレッジベース

http://kbase.redhat.com/

全般情報

NSA SELinux メイン web サイト

http://www.nsa.gov/research/selinux/index.shtml

NSA SELinux FAQ

http://www.nsa.gov/research/selinux/faqs.shtml

メーリングリスト

NSA SELinux メーリングリスト

http://www.nsa.gov/research/selinux/list.shtml

Fedora SELinux メーリングリスト

http://www.redhat.com/mailman/listinfo/fedora-selinux-list

コミュニティ

SELinux プロジェクト Wiki

http://selinuxproject.org/page/Main_Page

SELinux コミュニティページ

http://selinux.sourceforge.net/

IRC

irc.freenode.net, #selinux

付録A 改訂履歴

改訂 4-2.2.400 **2013-10-31** **Rüdiger Landmann**
Rebuild with publican 4.0.0

改訂 4-2.2 **Fri Aug 9 2013** **Noriko Mizumoto**
翻訳および査読の完了

改訂 4-2.1 **Wed Jul 10 2013** **Chester Cheng**
翻訳ファイルを XML ソースバージョン 4-2 と同期

改訂 4-2 **Fri Feb 22 2013** **Tomáš Čapek**
6.4 GA リリース向けのバージョン

改訂 3-0 **Wed Jun 20 2012** **Martin Prpič**
Red Hat Enterprise Linux 6.3 GA 向け「制限のあるサービスの管理」ガイドをリリース

改訂 2-0 **Tue Dec 6 2011** **Martin Prpič**
Red Hat Enterprise Linux 6.2 GA向け「制限のあるサービスの管理」ガイドをリリース

改訂 1-0 **Thu May 19 2011** **Martin Prpič**
Red Hat Enterprise Linux 6.1 GA向け「制限のあるサービスの管理」ガイドをリリース

改訂 0-0 **Tue Nov 9 2010** **Scott Radvan**
6.0 GA リリース