SPDL仕様書 ver 2.1.

Yuichi Nakamura *

December 27, 2006

Contents

*himainu-ynakam@miomio.jp

これは、SPDL(Simplified Policy Description Language(単純化ポリシ設定言語)) の設定要素の解説書です。SPDL の設定要素を詳しく知らなくても、ポリシ自動生成ツール等により設定を行うことはできますが、生成された設定の意味を知る際に有効なドキュメントかと思います。

1 概要

seedit のポリシーファイル, Simplified Policy(/etc/seedit/policy) は、SPDL という書式で書かれています。SPDL コンパイラ (seedit-converter コマンド。seedit-load コマンドの中で呼び出されます) によって、SPDL は普通の SELinux のポリシー書式に変換され、設定が反映されます。

1.1 機能

SPDL の主な機能は、ラベルの隠蔽（パス名ベースの設定）と、パーミッションの絞りこみです。

- ラベルの隠蔽
  通常の SELinux では、ラベルをリソースに付与し、ラベルを使って設定する必要があります。ラベルの管理、ラベルとリソースの対応付けの把握が繁雑でした。一方、SPDL では、ラベルは隠蔽されています。ファイル名やポート番号などのリソース名を指定して設定できます。

- パーミッションの絞りこみ
  SELinux には、数百のパーミッションが存在していました。一方 SPDL では、パーミッションを除去したり、統合することによって、パーミッションの数を絞りこんでいます。セキュリティ上の影響が極力少なくなるよう配慮し、絞りこんでいます。
  「パーミッション除去」とは、そのパーミッションが全ドメインに許可されるようにし、実質そのパーミッションに関するアクセス制御が働かないようにすることです。
  「パーミッション統合」は、複数のパーミッションを一つのパーミッションとして扱うことです。どんなパーミッションが除去され、統合されているのかは、http://seedit.sourceforge.net/doc/permission_integrate/にあります。

1.2 SPDL の設定要素の概要

SPDL は、次のような設定を行うことができます。

- アプリケーションへのドメイン付与
  ドメインを割り当てるには、SELinux のドメイン遷移を設定する必要があります。ドメイン遷移の設定には、2 種類の要素が用意されています。
  domain_trans という諸式は普通のドメイン遷移の諸式です。
  program は、簡略版の書式です。

- RBAC
  RBAC をサポートしています。*role*, *user* 文を使うことで設定できます。

- ファイルアクセス制御
  allow/deny 文は、通常ファイルへのアクセス制御設定です。デバイスファイル関連としては、 *allowdev* 文があります。*allowtmp* は、一時ファイルの保護に使います。

- ネットワークアクセス制御
  allownet 文で設定できます。

- IPC(プロセス間通信) の制御
  allowcom 文で、IPC およびシグナルのアクセス制御設定ができます。

- Key のアクセス制御
  allowkey 文は、プロセス毎の鍵領域のアクセス制御に使います。

- その他の特権
  その他の、重大な OS の操作については、*allowpriv* 文で設定します。

1.3 デフォルト拒否

ドメインは、デフォルトでは、全て拒否されます。

1.4 用語

- ドメイン
  ドメインは、SELinux と同じです。ドメインは、プロセスにドメイン遷移で割り当てられます。

- ロール
  SPDL のロールは、単純化されています。ユーザシェルに付与されるドメインと同じ意味です。SPDL では、アクセス制御設定をロールに対して行いますが、実際は、ユーザシェルのドメインに対して権限が与えられています。例を見てみましょう。sysadm_r ロールに対して許可された権限は、ログイン後のユーザーシェルのドメイン sysadm_t に許可されます。
  SPDL から生成される SELinux のポリシでは, *role* 全てのロール 全てのタイプ*;* のような設定がされています。

- 非制限 (unconfined) ドメイン
  unconfined ドメインとは、全ての権限が許可されたドメインのことをいいます。つまり、このようなドメインが付与されたプロセスは実質 SELinux のアクセス制御を受けません。*allowpriv all;* と設定されたドメインは unconfined ドメインとなります。

2 SPDLによる設定の構造

設定は、「セクション」という単位で構成されます。セクションとは、{ }で囲まれた領域のことです。

一つのドメイン・ロールに対して、セクションが一つ存在し、セクションの中で、ドメイン，ロールに対する設定がなされます。

2.1 セクションの構造

{ (セクションの開始)
domain/*role* (ドメイン、ロールの宣言。一つのドメイン、ロールしか宣言できない。)
users (ロールの設定の時のみ有効。)
domain_trans—*program* (ドメイン付与の設定)
allow/*deny*—*allowxx* (ファイルやネットワークなどへのアクセスを許可)
} (セクション終わり)

3 他ファイルの設定を流用:include

*include*を使うことで、他ファイルの設定を流用することができます。書式は以下です。include *filename*; filenameの内容が、そのまま設定されます。デフォルトでは、filenameは、/etc/seedit/policy/includeからのパスです。seedit-converterの-Iオプションでパスを帰ることもできます。

4 ドメイン、ロールの宣言

4.1 ドメインの宣言:domain

1. 書式
   domain *domain name* ;

2. 意味
   ドメインを宣言します。セクション中の設定は、ここで宣言したドメインに対してなされることになります。

3. 制約
   *domain name*は、必ず *_t*で終わる名前である必要があります。また、一つのセクション中で2つのドメインの宣言はできません。

4.2 ロールの宣言:role

1. 書式
   role *role name* ;

2. 意味
   ロールを宣言します。ロールを使えるユーザは次の *user* 文で行います。

3. 制約
   role name は、*_r* で終わる必要があります。一つのセクション中では 1 度しか記述できません。

5 RBAC の設定:user

1. 書式
   user *user name*;

2. 意味
   ロールを使えるユーザを列挙します。

3. 例
   {
   role user_r;
   user root;
   user ynakam;
   ....
   ユーザ root と ynakam が user_r ロールを使えることを意味します。

4. 制約
   ロールが宣言されたセクションでしか使えません。

6 ドメイン遷移

SELinux では、ドメインを付与するためにドメイン遷移を使います。SPDL は、このドメイン遷移を詳細に設定する domain_trans 文および単純設定する program 文をサポートしています。

6.1 domain_trans

1. 書式
   domain_trans *parent domain filename-of-entrypoint*;

2. 意味
   parent domain のドメインで動作しているプロセスが、*filename-of-entrypoint* 実行ファイルを実行すると、ドメインが割り当てられます。 *parent domain* および *filename-of-entrypoint* は、カンマ区切りで列挙可能です。

3. 例

```
{
   domain httpd_t;
   domain_trans initrc_t /sbin/httpd;
...
```

initrc_t ドメインで動作するプロセスが、/sbin/httpd を実行すると、/sbin/httpd には、httpd_t ドメインが割り当てられます。

4. 高度な設定
   Dynamic domain transition も設定可能です。 以下のように *filename-of-entrypoint* を省略します。

```
{
domain httpd_t;
domain_trans initrc_t;
```

initrc_t から httpd_t への dynamic domain transition が許可されます。

6.2 単純なドメイン付与:program

1. 書式
   program *path-to-program*;

2. 意味
   path-to-program が、コマンドラインから起動されたり、/etc/init.d から起動された時にドメインを割り当てる書式です。
   正確に言うと、unconfined ドメインから起動された時、ドメインが割り当てられます。
   しかし、認証を伴うプログラムからドメインを割り当てる設定は、これではできません (su,login,sshd など)。認証を伴うプログラムに使うドメイン一覧は、converter.conf の authentication_domain フィールドで列挙されています。

3. 例
   1)

```
{
domain httpd_t;
program /usr/sbin/httpd;
}
```

   コマンドラインまたはシステム起動スクリプトから`/usr/sbin/httpd` を起動した
   ら、`/usr/sbin/httpd` には `httpd\_t` ドメインが割り当てられます。

4. 注意
   これは、unconfinedドメインが存在するポリシーのために用意された書式です。unconfinedドメインが存在しない場合は、意味がなくなります。
   また、RBACが有効な場合は、sysadm_r ロールのシェルから起動した場合のみ、ドメインが割り当てられます。sysadm_r は unconfinedドメインだからです。

7 通常ファイルのアクセス制御:allow/deny

7.1 allow

1. 書式

   (a) allow *filename* | *label* [r],[w],[x],[s],[o],[t],[a],[c],[e],[dx];

2. 意味

   (a) ファイルへのアクセスを許可します

3. ファイル名の指定

   - ワイルドカード
     filename の部分には、 *directory*/* and *directory*** のような指定ができます。 例えば、/var/* は、/var 直下の全てのファイルを意味します。/var/** は、/var 直下の全てのファイルに加え、/var のサブディレクトリの以下にある、全てのファイル、ディレクトリを意味します。
   - ホームディレクトリ
     ~ から始まるファイル名は、ユーザーのホームディレクトリ（/root 以外）を表します。例えば、 /public_html means, /home/全てのユーザ/public_html を表します。
     ただし、ロールの設定の場合は、意味が異なります。そのロールを使うことができるユーザーのホームディレクトリだけを表します。例えば、

     ```
     {
     role staff_r;
     user ynakam;
     user himainu;
     allow ~/** r,w;
     }
     ```

     では、staff_r は、/home/ynakam,/home/himainu へのアクセスのみが許可されています。

4. パーミッションの意味

- r(Read)
  ファイルの読み込み
- w(Write)
  ファイル、ディレクトリの書き込み、生成、消去、追記。ただし、デバイスファイルに関する操作は許可されません。デバイスファイルに対してもアクセスを許可したい場合は、allowpriv devcreate(デバイスの生成のみが許可されます) または allowdev を使う必要があります。
- x(eXecute)
  ファイルの実行
- s(Search)
  ディレクトリの中身の閲覧。ファイルに対して設定しても何も起こりません。ディレクトリ以下のファイルの読み込みを許可したい場合は r と一緒に設定します。

5. 例
   {
   domain httpd_t;
   ...
   allow /var/www/** r,s;
   ....

   httpd_t は、/var/www 以下のファイル、ディレクトリをサブディレクトリを含めて全て読み込みを許可されています。

6. 詳細パーミッション
   s,r,x,w パーミッション以外にも、o,t,a,c,e というパーミッションを使うこともできます。これらは、w パーミッションを分割したものです。

   - o: Overwrite
     ファイルの上書き保存のみを許可します。ファイルの消去や生成は許可されません。
   - t: seTattr
     ファイルの属性（所有者など）を変更します。なお、SELinux のラベル情報は変更できません。
   - a: Append
     ファイルを追記モードでオープンすることを許可します。
   - c: Create
     ファイルの生成を許可します。Allow to create file.
   - e: Erase
     ファイルの消去を許可します。

7. ドメイン遷移パーミッション:dx
   dx という特殊なパーミッションが用意されています。もし、ドメインがそのプログラムに対して定義されていれば、そのプログラムに新たなドメインが付与されて動作します。

例:

```
        {
  domain httpd_t;
          program /usr/sbin/httpd;
          allow /var/www/cgi-bin/test.cgi r,s,dx;
        }
        {
  domain cgi_t;
          program /var/www/cgi-bin/test.cgi;
          allow ............
        }
```

この場合 httpd_t ドメインは、test.cgi に対して dx パーミッションが許可されています。test.cgi には、cgi_t というドメインを割り当てる設定がされているため、cgi_t ドメインが割り当てられます。
なお、通常の「x」パーミッションですと、test.cgi は、httpd_t のまま動作してしまいます。program /var/www/cgi-bin/test.cgi; という設定は、unconfined ドメインから起動した場合に cgi_t ドメインを割り当てる設定であるため、httpd_t から起動した場合には、cgi_t は割り当てられません。

8. ホームディレクトリについての制限事項
   個々のユーザーのホームディレクトリに deny を記述しても無視されます。
   例：

```
deny /home/ynakam/public_html;
```

7.2 deny

1. 書式
   deny *filename*;
2. 意味
   allow 文に対する制約を記述したり、allow をキャンセルするのに使われます。
3. 例
   (a) 例１ 1: 制約の記述

```
*constraints という名前のファイル名
deny /etc/shadow;

*httpd_t.sp というファイル
{
domain httpd_t;
include constraints;
allow /etc/* r,s;
}
```

*include constraints;*では、constraintsという名前のファイルで行った設定、つまり、deny /etc/shadowが設定されます。以下と同じ意味です。

```
{
domain httpd_t;
include constraints;
deny   /etc/shadow;
allow /etc/* r,s;
}
```

これの意味ですが、httpd_tドメインは、/etcの下にあるファイルに対して読み込みができます。しかし、/etc/shadowにはアクセスはできません。
/etc/shadowにアクセスするには、*allow /etc/shadow r,s;*と、明示的に書かねばなりません。denyは、設定ミスを防ぐのに役に立ちます。

(b) 例 2: allowのキャンセル

```
{
domain httpd_t;
allow /etc/* r,s;
deny /etc;
```

allow /etc/ r,s;*は、deny /etc;を記述することでキャンセルされます。

7.3 allowとdenyで衝突するルールが記述された時

1. allowがコンフリクトしたらORを取る

   - 例

   ```
   domain foo_t;
   allow /var/** r;
   allow /var/** s;
   ```

   foo_t は /var/**に、 r,sが許可されます。

   ```
   domain foo_t;
   allow /var/run/* r;
   allow /var/run/** w;
   ```

   foo_tは、/var/run直下のファイルにrパーミッションが許可されます。一方、/var/run以下のファイル・ディレクトリに対し、サブディレクトリ含めてwパーミッションが許可されます。

   - 親ディレクトリ、子ディレクトリで衝突した場合

```
domain foo_t;
allow /var/** r;
allow /var/run/** w;
```

foo_t は、/var 以下に r パーミッションが許可されますが、/var/run 以下についてだけは、w パーミッションが許可されます。

2. allow/deny が衝突したら、前にされていた設定を打ち消し

- 例：

```
domain foo_t;
allow /foo/* r,s;
deny  /foo/* ;
```

allow /foo/ r,s* が打ち消されます。

```
domain foo_t;
deny  /foo/* ;
allow /foo/* r,s;
```

*deny /foo/** が打ち消されます。

```
domain foo_t;
allow  /foo/bar/** r,s;
deny   /foo/** ;
```

*allow /foo/bar/** r,s* が打ち消されます。

- 例外

```
domain foo_t;
deny  /foo/bar/**;
allow /foo/** r,s;
```

*deny /foo/bar/*** は打ち消されません。 *not* cancelled. deny をキャンセルするには、常に deny されたディレクトリを直接指定した allow を書く必要があります。(今回の場合 *allow /foo/bar/** r,s;* のように書く必要があります)

7.4 特殊なファイル

以下のファイルに対するアクセス制御は特殊な扱いになっています。

1. /dev/tty* /dev/pts /dev/ptmx, /dev/vcs*,/dev/vcsa*
   これらは端末に関するファイルですが、allow を書いても何も起こりません。
   これらのファイルへのアクセスは、allowdev 文で行うようになっています。

2. /proc, /sysfs, /selinux, /dev/tmpfs
   これらのファイルに対する設定は、何も起こりません。内部的なことをいうと、SELinux は、これらのファイルがマウントされているファイルシステムをサポートしていないためです。allowfs を使うことで一部設定することはできます。また、/selinux 以下へのアクセス制御は allowpriv getsecurity, setsecurity で制御できます。

7.5 リンクに関する注意事項

7.5.1 シンボリックリンクの扱い

パスの途中にシンボリックリンクが含まれるようなファイル名は、無視されます。例えば、
allow /etc/init.d/httpd r;
は無視されます。(init.d は、rc.d/init.d へのシンボリックリンクだからです)。

7.5.2 ハード リンクの取扱い

Linux システムでは、ハード リンクを使うことで、ファイルの中身を複数のファイル名で参照することができます。ハード リンクは、デフォルトではほとんど使われていないため、以下の内容を気にする場面はほとんど現れませんが、セキュリティ上知っておいたほうがいいでしょう。
SPDL では、ハード リンクは以下のルールで処理されます。
ファイルの中身が複数のハード リンクで参照される場合、元々存在したファイル名を記述する必要がある。それ以外のファイル名が指定された場合は無視される。例えば、/etc/shadow と /var/chroot/etc/shadow がハード リンクされていたとします。/etc/shadowが元々存在していたとすると、/etc/shadow (と /var/chroot/etc/shadow) の中身を見るためには、allow /etc/shadow r と記述する必要があります。allow /var/chroot/etc/shadow r と記述しても無視されます。
ハード リンクが複数存在する場合、どのファイル名を「元々存在するファイル名」とするかの基準が気になるところです。以下の基準で「元々存在するファイル名」が判定されます。以下で出てくる例では、/etc/shadow, /var/shadow がハード リンクされたファイルだと仮定します。

1. 全ポリシ中で、一つのファイル名に対する設定しか書かれていない場合、そのファイル名が「元々存在するファイル名」になります
   例: allow /etc/shadow r がある場所で記述されているとします。そして、/var/shadow を使った設定はどこにも記述されていないとします。この場合は、/etc/shadow が、元々存在するファイル名として取り扱われます。

2. 複数のファイル名に対する設定が記述されていた場合、名前が一番若いものが、「元々存在するファイル名」になります。
   例: allow /etc/shadow r,allow /var/shadow r; という設定が記述されていたとします。この場合、「/etc/shadow」が、「元々存在するファイル名」になります。なぜなら、/etc/shadow のほうが名前が若いからです。

3. ハードリンクされたファイル名を使った設定がどこにも記述されてないときは、所属ディレクトリ名を比較して、所属ディレクトリ名が最も大きいものが「元々存在するファイル名」となります。
   例: /etc/shadow, /var/shadow を使った設定がどこにも記述されていない場合、/var/shadowが「元々あるファイル名」となります。なぜなら、/var/ > /etc だからです。

しかし、どのハードリンク名が「元々存在するファイル名」か分からない場合は、全ての名前を使う手もあります。例えば、

```
allow /etc/shadow r;
allow /var/shadow r;
```

のように記述した場合、どちらかの設定は無視されるだけで、無害です。
以上のハードリンクの取り扱いは、パス名ベースのセキュリティの「抜け穴」を防ぐために必要なものです。この取り扱いがなかったとすると、例えば、/etc/shadow のハードリンクが、なんらかの手で/var/www/html/shadow に作られてしまうと、Web サーバーから/etc/shadow の中身を覗けてしまうことになります。これを防ぐために、ハードリンクされたファイルの中身にアクセスするには、「一つのファイル名」しか使えないようにする必要があるわけです。パス名ベースのセキュリティの問題点については、
http://securityblog.org/brindle/2006/04/19
に詳しいです。

8 デバイスファイルのアクセス制御:allowdev

8.1 allowdev(1)

デバイスファイルの扱いには注意を要します。なぜなら、デバイスファイルは、カーネルのリソースへのインターフェースとなっているからです。デバイスファイルが、ハードディスクのデバイスドライバに関連付けられていたら、そのデバイスファイルへの読み書きは、ディスクの破壊や情報漏洩につながります。
ここで紹介する allowdev により、デバイスファイルへのアクセス制限をすることができます。

1. 書式
   (a) allowdev -root *directory*;
2. 意味
   デフォルトでは、*allow* を使ってファイルへのアクセスを許可したとしても、デバイスへのアクセスは許可されません。デバイスへのアクセスを許可するには、この書式を使って、デバイスの位置を明示的に指定しておく必要があります。*directory* 以下が、デバイスを格納するディレクトリとみなされ、そのディレクトリ以下に対する allow を記述すると、デバイスへのアクセスが許可されます。
3. 例

```
{
domain httpd_t;
allow /dev/* r,w;
```

httpd_t は、/dev 以下の普通のファイルには r,w アクセスできるものの、デバイスに対しては、アクセスできません。

```
{
domain httpd_t;
allowdev -root /dev;
allow /dev/* r,w;
```

httpd_t は、/dev 以下のファイルおよびデバイスファイルに r,w アクセスできます。
ただし、*w* を指定しても、デバイスの生成、消去は *allowpriv devcreate* が記述されない限り、許可されません。

8.2 allowdev(2)(RBAC 有効時のみ）)

/dev/tty*デバイスファイルは、端末デバイスです。/dev/pts 以下のデバイスは、疑似端末です。端末デバイスは、ローカルログインの端末を現し、疑似端末は、X や ssh ログインの端末を表します。これらのデバイスと端末の関連付けはログイン時に行われます。
もし、他のログインユーザの端末デバイスに書き込みが行えるとしたら、そのユーザの端末に余計な文字列を表示することで、他のユーザの作業を妨害できてしまいます。
SELinux では、端末や疑似端末デバイスにログインユーザのロールに応じたラベルが割り当てられます。ここで紹介する allowdev は、ラベルを隠蔽しつつ、端末のアクセス制御を実現します。

1. 書式
   (a) allowdev -pts—-tty—-allterm open;
   (b) allowdev -pts—-tty—-allterm *role* [r],[w];
   (c) allowdev -pts—-tty—-allterm *role* admin;
2. 意味
   -tty は端末デバイス、 -pts は疑似端末、-allterms は端末、疑似端末両方。
   (a) これは、通常はロールの設定時に使われます。ロールに対して、自分専用の端末を持つことを許可されます。
   内部的な動作ですが、これを設定しておくと、ログイン時にログインプログラムによって、デバイスにロールに応じたラベルが付与されるようになります。*role prefix*_tty_device_t という名前のラベルです。

(b) 他ロール専用の端末に、読み込みや書き込みのアクセス権限を与えます。

(c) 端末の管理権限を与えます。つまり、端末のラベルを変更したり、消去したりする権限です。

3. 特別なロール名
role 部分には、以下のような特別な名前が使えます。

- general
ログインプログラムによってラベル付けが行われる前の端末を指定します。ちなみに、ラベル付け前は、devtty_t tty_device_t, devpts_t, ptmx_t といったラベルがついています。これらの端末に対するアクセスを許可しても害はありません（admin パーミッションは除く）
- all
全てのロールの端末を指定します。
- vcs
allowdev -tty のみに使える指定です。これは、vcs ファイル (/dev/vcs*, /dev/vcsa*) を指定します。vcs ファイルとは、端末のスクリーンショットを提供するものです。

9 SELinuxがサポートしないファイルシステムへのアクセス制御: allowfs

SELinux は、拡張属性 (xattr) をサポートするファイルシステムに対しては、ファイル単位でアクセス制御を行うことが可能です。ext3,ext2,xfs などは xattr をサポートしていますが、サポートしないファイルシステムもあります。こういったファイルシステムに対しては、個々のファイル単位のアクセス制御はできません。「ファイルシステムのファイル全体」に対するアクセス制御になってしまいます (proc ファイルシステムについてはもう少し細かくできますが)。allowfs はそういったファイルシステム用の設定要素です。

- 書式

1. allowfs *name_of_filesystem* [s],[r],[x],[w],[o],[a],[t],[c],[e];
name_of_filesystem については、 *tmpfs sysfs autofs usbfs cdfs romfs ramfs dosfs smbfs nfs proc proc_kmsg proc_kcore xattrfs* という文字列を利用可能。

- 意味

1. 指定されたファイルシステムにあるファイルへのアクセスを許可します。例えば、 *allowfs sysfs s,r;* は、sysfs ファイルシステム上のファイル (普通は/sysfs 以下) に対して、s,r を許可します。アクセス拒否ログの中に、*filesystem* 名_t という名前のタイプを発見したら、allowfs を使う必要があるかもしれません。例えば、*sysfs_t* への読み込みが拒否

されたというログを発見したら、*allowfs sysfs s,r;* と設定する必要があるでしょう。もっとも、自動生成コマンドがやってくれる仕事ですが。

- *name_of_filesystem* についての注意点
  - proc ファイルシステム
    proc ファイルシステムについては、もう少し細かくできます。
    proc_kmsg とすると、/proc/kmsg を指定でき、proc_kcore とすると、/proc/kcore を指定できます。 proc_pid_self とすると、/proc/pid にある、自ドメインに属するプロセス情報です。proc_pid_other は他ドメインに属するプロセス情報全てです。
  - xattrfs
    これは、xattr をサポートするファイルシステムにも関わらず、SELinux のラベルを使うように設定されていないファイルシステムを意味します。例えば、USB メモリを非 SELinux マシンでフォーマットして、マウントした場合が相当します。
  - cdfs
    iso9660 および udf ファイルシステムを意味します。
  - dosfs
    vfat, fat, ntfs を意味します。
  - smbfs
    cifs, smbfs を意味します。

10 一時ファイルへのアクセス制御:allowtmp

10.1 なぜ allowtmp が必要?

allowtmp は、一時ファイルのアクセス制御のために用意された設定要素です。その前に、なぜそんなものが必要なのかを見ていきましょう。SELinux は、内部的にファイルをファイル名ではなく、i ノードで識別しています (正確には i ノードに関連づけられたタイプラベル)。seedit は、ファイル名を指定して設定を行いますが、i ノード番号が変わってしまう、もしくは設定時にファイルが存在しない場合 (i ノード番号が設定時に分からない) は、うまく動かなくなります。/var/run, /tmp, /var/tmp 等には、そのようなファイルがたくさんあります。
一つ例を見てみましょう。

```
domain httpd_t
allow /var/run/httpd.pid r,w;
```

設定時には、httpd_t は、/var/run/httpd.pid に対する r,w パーミッションが許可されています。しかし、httpd を再起動すると、httpd.pid ファイルは消去され、また同じ名前のファイルが生成されます。i ノード番号も変わってしまっています。そうなると、seedit 側では、新しく生成されたファイルを認識できなくなり、httpd_t は httpd.pid にアクセスできなくなってしまいます。
次の例です。/tmp にランダムにファイルを生成する場合問題になります。プログラム A（a_t ドメインで動作する)、プログラム B（b_t ドメインで動作する）が

あるとします。そして、これらが、/tmp 以下にランダムな名前のファイルを作成するとします（tmpfs システムコールで一時ファイルを生成した場合に相当します)。すると、以下のような設定を書かざるをえません。

```
{
domain a_t;
allow /tmp/** r,w;
}
{
domain b_t;
allow /tmp/** r,w;
}
```

これは、プログラム A、B ともお互いの一時ファイルにアクセスできることを意味します。一時ファイルに重要な情報が書かれていたら情報漏洩に繋がります。

10.2 allowtmp とは

こういった問題を解決するため、SELinux はファイルタイプ遷移という機能を持っています。ファイルタイプ遷移を分かりやすく設定するのが allowtmp です。ファイルタイプ遷移を使うと、「ドメイン A がディレクトリ B に生成したファイル」のようなラベルを付与し、そのラベルを使ってアクセス制御できます。
例です。

```
domain httpd_t;
allow /var/run r,s;
allowtmp -dir /var/run -name httpd_var_run_t; -(a)
allow httpd_var_run_t r,w; -(b)
```

(a) では、httpd_t ドメインのプログラムが/var/run にファイルを生成したとき、そのファイルに httpd_var_run_t のようなラベルを付与します。(b) では、そのラベルが付与されたファイルに読み書きできるようにしています。以下のような省略記法もあります。

```
domain httpd_t;
allow /var/run r,s;
allowtmp -dir /var/run -name auto r,w;
```

「/var/run に生成するファイルに読み書き許可」と設定しています。内部的には先ほどの (a)(b) を設定しています。

10.3 allowtmp の書式と意味

1. 書式
   (a) allowtmp -dir *directory* -name *label permission*;
   (b) allowtmp -fs *file system name* -name *label permission*; *permission* は、ファイルのものと同じです。省略して、ラベル付け設定のみを行うこともできます。

2. 意味

   (a) ドメインが、*directory*ディレクトリにファイルを生成すると、*label*という名前のラベルが付与されます。そして、そのラベルに対するパーミッションを *permission*で指定します。 *permission*は省略可能です。省略すると、パーミッション設定は別途 allow 文を使って行うことになります。

   (b) これは、xattr をサポートしないファイルシステム上のファイルに対する設定です。今のところ、tmpfs のみのサポートとなります。

   (c) *label*の指定について

   - *label*が *auto* のときは、ラベルの名前は、自動的に名付けられます（ドメインとディレクトリ名から生成されます）。例えば、hoge_t ドメイン、 *directory*が /var/の場合は、hoge_var_t という名前のラベルが生成されます。
   - *label*が *all* または *の時は、*directory* ディレクトリ以下の、allowtmp 設定によりラベル付けされた全てのファイルを意味します。

3. 例

```
domain httpd_t ;
allowtmp -dir /var/run -name auto r,w;
```

   httpd_t が、/var/run にファイルを生成すると、httpd_var_run_t というラベルが付与され、それに対して r,w アクセスができます。

```
domain httpd_t
allowtmp -dir /var/run -name auto r,w;
domain named_t
allowtmp -dir /var/run -name auto r,w;
domain initrc_t;
allowtmp -dir /var/run -name all r,w;
```

   httpd_t が、/var/run にファイルを生成すると、httpd_var_run_t というラベルが付与され、それに対して r,w アクセスができます (named_t などはアクセス不可)。named_t が、/var/run にファイルを生成すると、named_var_run_t というラベルが付与され、それに対して r,w アクセスができます (httpd_t などはアクセス不可)。initrc_t は、上記のファイルに r,w アクセスができます。なぜなら、-name all が指定されているからです。

11 ネットワークアクセス制御:allownet

*allownet*は、ソケット、ポート番号、NIC, IP アドレスなど、ネットワークリソースへのアクセス制御を設定するものです。

11.1 ポートのアクセス制御

1. 書式
   allownet -protocol *protocol* -port *port number permission*;
   protocol: tcp,udp が指定できます。カンマ区切りで tcp,udp のように書くこともできます。
   port number: ポート番号を指定。数字および *-1023,1024-* , および* が指定できます。カンマ区切りで数字を列挙できます。*permission*: *client* または *server* を指定可能。カンマ区切りで列挙可能。

2. 意味
   TCP、UDP のサーバーまたはクライアントとして、指定されたポートで通信することを許可。*-1023* というポート番号は、allownet で指定されていない 1023 以下のポート番号全て。*1024-*は、allownet で指定されていない 1024 以上のポート全て。 *は、全てのポートを意味します。

3. UDP サーバの設定について注意
   52 allownet -protocol udp -port xxx server; のように設定すると、そのドメインは、1024 以上のポートを使ってクライアントとして振る舞うことも同時に許可されます。

4. 例

```
domain httpd_t;
# TCP80,443 を使ってサーバとして振る舞う
allownet -protocol tcp -port 80,443 server;
# TCP,UDP3306 を使ってクライアントとして振る舞う
allownet -protocol tcp,udp 3306 client;
```

11.2 RAW ソケットの利用

RAW ソケットは、通信の盗聴などに利用されるため、利用を制限しなくてはなりません。

1. 書式
   allownet -protocol raw use;
   permission: *client* 、*server* 、 * がコンマ区切り指定可能。

2. 意味
   RAW ソケットを使ったデータの受信（server)、送信 (client) が許可されます。

11.3 NIC(netif) または IP アドレス (node) の利用制限

netif や node の利用制限ができます。なお、デフォルトで用意されているポリシでは、common-relaxed.sp ファイルにより、全ドメインに許可されてます。

1. 書式

(a) allownet -protocol *protocol* -netif *name of NIC permission*;
protocol: tcp,udp,raw、 * がカンマ区切りで指定可能。
name of NIC: NIC の名前 (lo,eth0,eth1 等) がカンマ区切りで指定可能
permission: *send*、*recv*がカンマ区切りで指定可能

(b) allownet -protocol *protocol* -node *address permission*;
protocol: tcp,udp,raw、 * がカンマ区切りで指定可能。
address: *ipv4address/netmask* または * がカンマ区切りで指定可能
例： 192.168.0.1/255.255.255.0。なお*は、全てのアドレスを意味します.
permission: *send* または *recv* をカンマ区切りで指定可能。

2. 意味

(a) NIC を通してデータを送信、受信することを許可。

(b) IP アドレス宛にデータを送信、IP アドレスからデータを受信することを許可。

3. 例

```
{
domain httpd_t;
allownet -protocol tcp  use;
allownet -protocol tcp -port 80 server;
allownet -netif eth0 send,recv;
}
--> httpd_t は 80 番ポートを通してサーバとして振る舞える。
eth0 を通してパケットを送受信できる。
(lo0,eth1 は使えない）
```

11.4 他ド メインからのソケット継承

他ド メインで動作するプロセスからソケットを継承して使う権限を与えます（滅多にないことですが。。）

1. 書式

(a) allownet -protocol *protocol* -domain *domain* use; *protocol*, tcp,udp をカンマ区切りで指定可能。

2. 意味

(a) 他ド メインで動作するプロセスからソケットを継承することを許可します。どのド メインから継承可能かを *domain* で指定します。*domain* として *self* を指定することで、自分自身が作成したソケットの利用権

限を与えます (こちらはデフォルトのポリシの common-relaxed.sp で許可されてます)

3. 例

```
domain foo_t;
# foo_t は UDP ソケットを bar_t から継承可能
allownet -protocol udp -domain bar_t;
```

12 プロセス間通信のアクセス制御:allowcom

12.1 allowcom (IPC)

1. 書式
   allowcom -ipc|-unix|-sem|-msg|-msgq|-shm|-pipe *to domain* [r],[w];

2. 意味
   指定された IPC で *to domain* と通信することを許可します。*to domain* が *self* の場合は、自ドメインないの通信を意味します。*to domain* として * とすると、全ドメインを指定します。-ipc は、全種類の IPC をまとめて許可しています。
   個々の IPC を制限したい場合は、以下を使えます。
   -unix は、UNIX ドメインソケットです。-sem はセマフォです。-msg はメッセージ、-msgq はメッセージキュー、 -shm は共有メモリ、 -pipe はパイプです。

12.2 allowcom(シグナルの制限)

1. 書式
   allowcom -sig *to domain* [c],[k],[s],[n],[o];

2. 意味
   to domain に指定されたシグナルを送信することを許可します。
   c は sigchld、 k は sigkill, s は sigstop, n は signull, o はその他のシグナルです。

13 他の特権のアクセス制御:allowpriv

これまで出てきた以外にも、OS にはセキュリティ上重要な操作（攻撃者にとられたら危ない権限）があります。これらを制限するのが allowpriv です。

- 書式
  allowpriv *string*;
  string に権限名を書いていきます。どんな名前が指定可能かは、後述します。

- 意味
  string で指定した権限を許可。

さて、*string* に記述可能な権限は、以下のように分類できます。この分類にそって紹介していきます。

- POSIX capability
- カーネル関連
- SELinux 管理操作
- その他

13.1 POSIX capability

*cap_*で始まる文字列は、POSIX capability(ケーパビリティ) です。POSIX Capability は、特権を分割して与えるものです。POSIX capability の詳細は、man capabilities を参考にしてください。

13.1.1 設定不能な POSIX capability

以下のものは、設定ができないようになっています (デフォルトで許可されます)。なぜなら、他の allow 文と完全に重なるからです。

- CAP_NET_BIND_SERVICE
  wellknown ポートの利用制限ですが、allownet でもっと細かく制限できるので省略します。
- CAP_MKNOD
  allowpriv devcreate と重なるので省略。
- CAP_AUDIT_WRITE
  allowpriv audit_write と重なるので省略。
- CAP_AUDIT_CONTROL
  allowpriv audit_control と重なるので省略。

13.1.2 設定可能な capability

詳細は、man capabilities を参照してください。

- cap_sys_pacct
  Configures kernel accounting(see acct(2)).
- cap_sys_module
  Allows to install kernel module.
- cap_net_admin
  Allow capability *CAP_NET_ADMIN*(Such as administrate NIC, route table).

- cap_sys_boot
  Allow capability *CAP_SYS_BOOT*. This means allow the usage of reboot system call.
- cap_sys_rawio
  Allow capability *CAP_SYS_RAWIO*.This means usage of ioperm, iopl system call and access to /dev/mem.
- cap_sys_chroot
  Allow to use chroot.
- cap_sys_nice
  Allow capability *CAP_SYS_NICE*. This means process scheduling.
- cap_sys_resource
  Allow capability *CAP_SYS_RESOURCE*. This means usage of rlimit etc.
- cap_sys_time
  Allow capability *CAP_SYS_TIME*. Thie means modify system clock.
- cap_sys_admin
  The same as POSIX capability *CAP_SYS_ADMIN*. This permissions overlaps other permissions, so if you allow this, not so serious problem. By denying this, it can restrict sethostname and some ioctl operations.
- cap_sys_tty_config
  The same as capability *CAP_TTY_CONFIG*. Change keyboard configuration, and usage of vhangup call.
- cap_ipc_lock
  Allow capability *CAP_IPC_LOCK*. This means to lock memory.
- cap_dac_override
- cap_dac_read_search
- cap_setuid
- cap_setgid
- cap_chown
- cap_setpcap
- cap_fowner

- cap_fsetid
- cap_linux_immutable
- cap_sys_ptrace
- cap_lease
- cap_ipc_owner
- cap_kill

13.2 カーネル関連操作

カーネルとの通信、管理する特権を許可します。

1. netlink
   netlink ソケットを使ってカーネルと通信することを許可
2. klog_read
   syslog(2) システムコールを使ってカーネルメッセージを読むことを許可。
3. klog_adm
   カーネルメッセージ出力設定の変更を許可
4. audit_read
   カーネル内の audit サブシステムのステータス読み出しを許可。
5. audit_write
   audit サブシステムにメッセージを送ることを許可。
6. audit_adm
   audit サブシステムの設定変更を許可。

13.3 SELinux 管理操作

Allow privileges to administrate SELinux.

1. relabel
   Allow to relabel all files. You must also allow getsecurity and allowpriv search.
2. part_relabel
   Allow to relabel files that the domain can write. You must also allow getsecurity.

3. setfscreate
   This is necessary only applications that use SELinux API(setfscreatecon).

4. getsecurity
   Allow to get security policy decisions, by accessing /selinux.

5. setenforce
   Allow to toggle enforcing/permissive mode.

6. load_policy
   Allow to load policy to kernel.

7. setsecparam
   Change performance parameter of SELinux via /selinux/avc

8. getsecattr
   Get security information(such as domain, stored in /proc/pid/attr) of other processes.

13.4 その他

Allow other privileges.

1. quotaon
   Allow to quotaon.

2. mount
   Allow to mount device.

3. unlabel
   Allow full access to unlabeled files(Files labeled as unlabeled_t).

4. devcreate
   Allow to create device files in directory that the domain can write. Without this, a process can not create device file on a directory even it is configured writable.

5. setattr
   Allow to setattr to files that the domain can s access. Without this setattr permission is granted in w permission.

6. search
   Allow s permission to all files.

7. read
   Allow r permission to all files.

8. write
   Allow w permission to all files.

9. all

13.5 denypriv

This can be used to cancel allowpriv configuration.

14 Access control of kernel key retention service:allowkey

This feature is included at version 2.1 or later.

After Linux 2.6.18, new feature *kernel key retention service)* is included. By the feature, each process can obtain key. For detail of key retention service, please refer to kernel document Document/keys.txt (You can look at the copy at http://free-electrons.com/kerneldoc/latest/keys.txt). allowkey controls access to key. This feature is effective only for FC5 or later. Cent OS does not have kernel key subsystem, so allowkey means nothing.

1. Syntax
   allowkey *domain permissions*;
   For permissions, you can use following.
   v: View. Look attribute of key.
   r: Read. Read contents of key.
   W: Write. Write contents of key.
   s: Search. Search keyrings.
   l: Link. Permits key or keyrings to be linked to.
   t: Set Attribute: Set attribute of key.
   For detail of permission, see Document/keys.txt.

2. Meaning
   Allow access to keys retained by *domain*.
   For example,
   allowkey login_t v,r;
   means, allow view and read access to keys, obtained by process whose domain is login_t.