明日はじめる SELinux

レッドハット株式会社
ソリューションアーキテクト 森若和雄
2017.11.28

このスライドの目的

- SELinux を使ったことがない、無条件反射で disable している人に「ちょっと試してみようかな」と思ってもらうこと
- ちょっと使いはじめるときに必要な最低限の話だけをします

概要

- SELinux って何？
- SELinux の概要
  - パーミッションと比較、 type 、ポリシー
- SELinux の設定
  - permissive ではじめる
  - モジュール、 boolean 、 TCP/UDP ポート設定 , file context
  - ひっかかったらどうするか？
  - enforcing
- Appendix

SELinux って何？

- 権限管理を拡張して、通常のパーミッションの仕組みで許可された操作をさらに追加でチェックする仕組み
- 通常のパーミッションよりはるかに細かくポリシーを定義
- kernel が行う他の権限管理よりも詳細なログを出力
  - 「“ Permission denied”で失敗してるんだけど何にひっかかったんだろう？」とはならない
  - ログを元に失敗した動作を許可するための自動ガイドをする仕組み (setroubleshoot) がある

SELinux って誰が使ってるの？

- Android 4.4 以降
  - 約 20 億台 (?)
  - 最近は Android 由来の拡張 (ioctl の制限など) も含まれています
- oVirt, OpenStack などの KVM による仮想化環境
  - 仮想マシン間の分離やホストの保護に利用
- OpenShift などのコンテナ環境
  - コンテナ間の分離やホストの保護に利用
- アメリカ政府機関やアウトソース先等で機密性が高い情報を扱う Linux システム
  - 法律で必須になっているので

Why SELinux?

目的 :

脆弱性があってプロセスがのっとられても、普段しないことをできないように権限を制限する

すぐ使える :

- Red Hat の製品では RHEL だけでなく仮想化環境やコンテナ基盤も SELinux で保護されている
- RHEL の Quality Engineering は基本的に全部 SELinux enforcing で実施

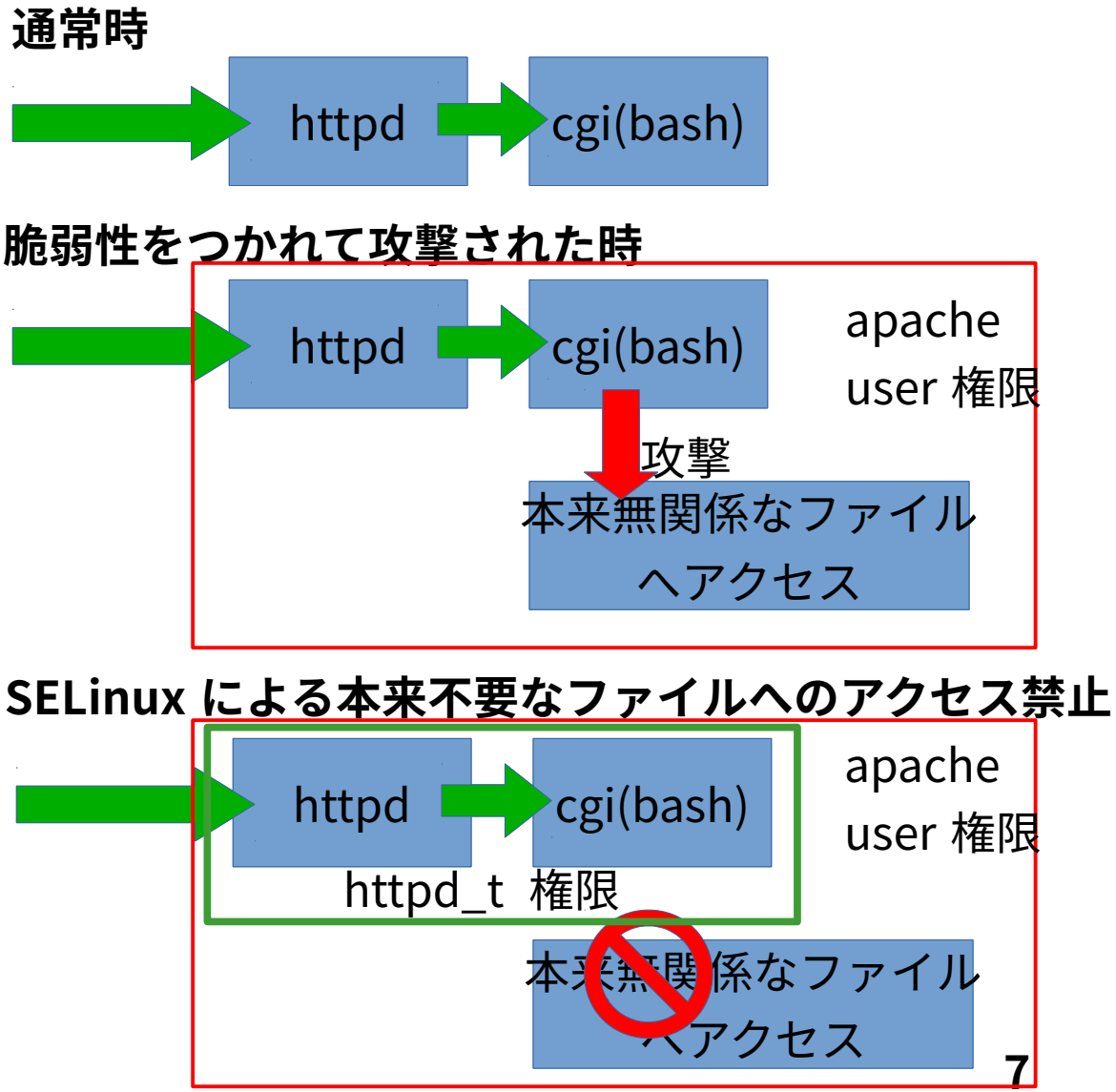
↓

まずは簡単なところからはじめる

- RHEL で ISV 製品なし、独自サーバ等なしで**既存のポリシーを使う**
- Boolean 設定して、ポートを指定、ファイルコンテキストの指定

SELinux の効果イメージ

- **のっとられたあと何ができるか？ を制限する**
  - ファイルアクセス
  - 他プロセスとの通信
  - execve
- **防げない攻撃もある**
  - SQL インジェクションのようなアクセス権限内の情報窃取
  - プロセス終了や無限ループ発生による DoS など



概要

- SELinux って何？
- SELinux の概要
  - パーミッションと比較、 type 、ポリシー
- SELinux の設定
  - permissive ではじめる
  - モジュール、 boolean 、 TCP/UDP ポート設定 , file context
  - ひっかかったらどうするか？
  - enforcing
- Appendix

普通のパーミッションと比べると？

- **パーミッション**
  - 権限管理の対象： ファイルとプロセス、 SysV IPC など
  - 識別方法 : uid, gid
  - 操作の分類 : r (read), w (write), x(execute) の 3 種 + sticky bit
- **SELinux**
  - 権限管理の対象： ファイル、プロセス、ソケット、 TCP/UDP ポート、共有メモリ、パイプ等、事実上 **OS が提供するリソース**全て + **SELinux 対応アプリケーションが管理するリソース** (dbus, systemd のサービス等)
  - 識別方法 : uid, gid とは独立した「 type 」
  - 操作は対象 class 毎に細かく分類

例 : file クラスに対して定義されている操作

append, execmod, ioctl, open, relabelto, unlink, audit_access, execute, link, quotaon, rename, write, create, execute_no_trans, lock, read, setattr, entrypoint, getattr, mounton, relabelfrom, swapon

SELinux の type

- SELinux ではあらゆるものに type を付与する
  - /usr/bin/httpd の type を確認する例：

```
$ ls -Z httpd
system_u:object_r:httpd_exec_t:s0 httpd
```

- SELinux の中ではあらゆるものを「type と操作」の組み合わせで判断する

※ 操作しようとするプロセスの type は正確には「ドメイン」と呼ばれますが、ここでは type と言います

type を確認する

- ユーザーの type を id コマンドで確認

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

- プロセスの type を ps コマンドで確認

```
$ ps auxcZ|grep sshd
system_u:system_r:sshd_t:s0-s0:c0.c1023 root 1156 0.0  0.0 97852
7188 ?        Ss   Aug18   0:00 sshd
```

- ファイルの type を ls コマンドで確認

```
# ls -Z /etc/passwd
system_u:object_r:passwd_file_t:s0 /etc/passwd
```

type によるポリシー

- 「操作元 (source type) が、
  操作対象 (target type) に対して、
  何をしてよい (class と permission) 」という形で許可する
- 操作の許可以外にも以下のものを指定する
  - 他の type に変更 (type_transition)
    - execve のタイミングで type を変更する
    - 元の type 、実行ファイルの type 、変更先の type の組み合わせで指定
  - その他数種類

ポリシーの例

- user_t のプロセスが passwd_exec_t のファイルを実行できる

```
allow user_t passwd_exec_t:file { execute getattr open read };
```

- ntpd_t のプロセスが dhcpc_t に SIGCHLD を送信できる

```
allow ntpd_t dhcpc_t:process sigchld;
```

- ping_t が rawip_socket を作って書き込みできる

```
allow ping_t ping_t:rawip_socket { append bind connect create
getattr getopt ioctl lock read setattr setopt shutdown
write };
```

「sesearch --allow -s SOURCE -t TARGET」で SOURCE が TARGET に何をできるポリシーになっているか確認できる

ポリシー

- SELinux では標準的な「 reference policy 」とよばれるポリシー。 RHEL ではその中の targeted をサポート。
  - https://github.com/TresysTechnology/refpolicy
- Fedora 同梱の reference policy で約 10 万の allow 、約 7 万の type_trans が定義されている
  - 普通の目的でゼロから書き起こすのは事実上無理
- 「 targeted ポリシーをもとに必要最小限のカスタマイズだけを行って利用する」のが SELinux ライフの現実的な所

targeted ってどんなポリシー？

- モジュール式で制限をかけるサービスだけを定義したポリシー
- 定義されていないものはほとんど制限 (保護) されない
  - 未定義のサービス (unconfined_service_t) は、 他サービスとの通信などをすこしだけ制限される
  - ログインしたユーザが起動するプロセス (unconfined_t) はほぼ何も制限されない
- 「 semanage module --disable MODULE 」のようにしてモジュール毎の有効・無効を設定
  - デフォルトでは全モジュール有効
  - カスタマイズするときはユーザ作成モジュールを追加

概要

- SELinux って何？
- SELinux の概要
  - パーミッションと比較、 type 、ポリシー
- SELinux の設定
  - permissive ではじめる
  - モジュール、 boolean 、 TCP/UDP ポート設定 , file context
  - ひっかかったらどうするか？
  - enforcing
- Appendix

permissive で使いはじめる

- setroubleshoot-server パッケージを導入 (後述)
- systemctl enable restorecond
- permissive mode に設定変更
  - /etc/selinux/config で SELINUX=permissive に
  - permissive はログだけ出力するモード。何も保護されない。
- ファイルのラベル付けをやりなおす指定をする
  - 再起動時にファイルに SELinux で使うラベルの再設定も行う
    「 touch /.autorelabel 」
- 再起動 !
  - 再起動時に /.autorelabel を発見してラベル付けを行う
  - 全ファイルに拡張属性を付与するのでそれなりに時間がかかる
  - 数分待ってログインプロンプトが出たら OK

モジュールの利用方法確認

- targeted ポリシーは各モジュールに man page があります
  - < モジュール名 >_selinux という名前
  - 一覧したいときは man -k selinux
- モジュールで定義している type の情報や、このあと触れる boolean, port, file context などの設定項目がまとまっています

boolean

- 「ポリシーの中で典型的によく変更するところ」を、 true/false の boolean として設定可能にしている
- モジュールの man ページで BOOLEAN を確認する
- 例 : samba で kerberos 認証を有効にする
  - 「 man samba_selinux 」で関連しそうな boolean を確認 →“ kerberos_enabled” を発見
  - 「 semanage boolean --list|grep kerberos_enabled 」として現状を確認
  - 「 semanage boolean -1 kerberos_enabled 」として有効化
  - 「 semanage boolean --list -C 」でカスタマイズ状況を確認

TCP/UDP ポート設定

- デフォルトやよく使われるポート番号は設定されている
- ポートを変更する場合には利用するポートの type を変更する
- モジュールの man ページで PORTS を確認する
- 例 : TCP 8888 番ポートを http に使いたい
  - 「 man httpd_selinux 」で PORTS を確認
  - 「 semanage port --list 」 で現状を確認
  - 「 semanage port -a -t http_port_t -p tcp 8888 」のようにして type をポートに付与
  - 「 semanage port --list -C 」 でカスタマイズ状況を確認

file context

- ファイルのパスと type の対応を変更する
  - 読み書きするファイルの位置が標準のポリシーと異なる場合
  - パスは同じでもシンボリックリンク経由になっている場合
- モジュールの man ページで FILE CONTEXTS を確認する
- 例 : tomcat でデータを置くところを /data 以下にしたい
  - man tomcat_selinux を参照→ tomcat_var_lib_t が相当する
  - 「 semanage fcontext -a -t tomcat_var_lib_t “/data(/.*)?” 」としてパターンを追加
  - 「 restorecon -R -v /data 」として /data 以下をラベルつけ直し
  - 「 semanage fcontext --list -C 」としてカスタマイズ内容を確認

引っかかった時にどうする？

普通はなにかしらひっかかります。 SELinux には詳細なログ出力があり、その対応も半自動化されています。

- setroubleshootd: ポリシーと SELinux のログから、boolean の修正やポリシーの拡張を提案する
- audit2allow: 拒否された操作を許可するポリシーを自動生成
- 対応作業のイメージ
  - 「テスト→ setroubleshootd のログに従って許可を追加→適用」を繰り返す
  - アプリケーションのテストがあれば、それを走らせると網羅的にチェックできるので良い

※ この手順では緩いポリシーになりがちで、最善ではありません。しかし SELinux がないよりはずっとマシな環境を簡単に実現できます。まずはここから始めましょう。

緩い設定でもないよりマシなの？

Q: 「アプリケーションのポリシーを緩くしても disabled よりマシ」なのはなぜ？

A: **攻撃を実現するために利用できる経路 (attack surface) を制限できるから**

- その他のプロセスが SELinux で保護されていれば「複数の脆弱性を利用した合わせ技」で攻撃することが難しくなる
- イメージとしては「 web サーバを守りたい」時に「 80 と 443 を素通しするだけの firewall をとりあえずいれておく」場合と近い

そして enforcing へ

- permissive モードである程度テストできたと感じたら、enforcing にします
  - enforcing モードはポリシーで許可されていない動作は失敗させるモード。実際に保護したい環境ではこれを利用します。
- /etc/selinux/config で以下のように設定します
  - SELINUX=enforcing
- 再起動！
  - enforcing で起動に失敗する場合、 9 割以上ファイルのラベルつけ忘れです。 kernel オプション selinux=0 で selinux を disable して一旦起動してから、 /.autorelabel を作成して再起動しましょう

まとめ

- SELinux はそれなりに使われています ;)
- RHEL は SELinux を enforcing している前提で開発されています
- 現在の SELinux ポリシーはモジュール分割され、モジュール毎の有効 / 無効の切り替えや、各種カスタマイズをサポートしています
  - boolean, TCP/UDP ポート設定 , file context
- 必要な所だけゆるめのポリシーに変更して利用することも十分意味があります

Appendix

SELinux のモード

- disabled
  - 無効。何もしない。
- permissive
  - チェックしてログを書き出す (同種の拒否は 1 回)
  - ポリシーで許可されていなくても処理は成功する
  - ポリシー開発やテスト時に利用する
- enforcing
  - チェックしてログを書き出す
  - ポリシーで許可されていないと処理は失敗する

setroubleshoot ログ出力例

/var/log/messages 内 :

```
Jul 24 18:42:22 snake.usersys.redhat.com setroubleshoot[18520]: SELinux is
preventing gsd-xsettings from setattr access on the directory
/usr/lib/fontconfig/cache. For complete SELinux messages run: sealert -l
8e9e87e7-b1ba-4312-9771-cb5765fcffdb
```

どのプロセスが何に何をしようとして失敗した

```
Jul 24 18:42:22 snake.usersys.redhat.com python3[18520]: SELinux is
preventing gsd-xsettings from setattr access on the directory
/usr/lib/fontconfig/cache.

                *****  Plugin catchall (100. confidence) suggests
**************************

                If you believe that gsd-xsettings should be allowed setattr
access on the cache directory by default.
                Then you should report this as a bug.
                You can generate a local policy module to allow this access.
                Do
                allow this access for now by executing:
                # ausearch -c 'gsd-xsettings' --raw | audit2allow -M my-
gsdxsettings
                # semodule -X 300 -i my-gsdxsettings.pp
```

この問題についての提案

実稼働後のトラブル

- setroubleshootd を動作させておくと、 /var/log/ messages に検出された問題と対応策がログ出力されます
  - `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`
- 粛々と対応する
  - やることは permissive でのテスト時と同じ
  - 多くは自分の設定漏れ、 boolean の提案がログに出ることも多い
  - 最初は「 audit2allow でとにかく許可」でよい
    - それでも disabled よりずっとマシ
  - RHEL で targeted policy 自体がおかしそうな場合はサポート窓口へ
  - upstream のメーリングリストや IRC もかなりサポートしてくれます

SELinux で最初にハマる穴はこれ

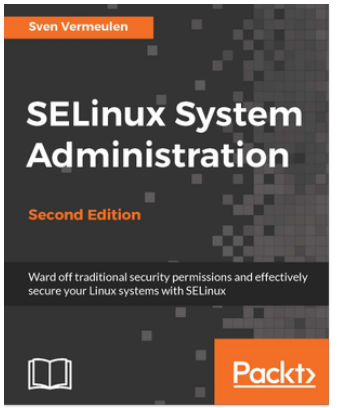
- /tmp で適当なファイルを作ってから mv したら type が tmp_t になっていてアクセス失敗
  - ファイルの作成時に type が決まり、 mv では元の type を維持するため
  - 移動したあとに restorecon foobar.conf のようにしてコンテキストを修正
- permissive mode で運用したらログが膨大で /var/log があふれる or すごく遅い
  - 運用環境は disable か enforcing で
  - テストはちょっとずつやりましょう
  - アクセス拒否が発生すると自動的に起動する setroubleshoot-server が重い原因になりがち

ちょっと便利な機能

- enforcing だけど一部だけ permissive にしたい
  - semanage permissive で type ごとに設定可
- 失敗しても問題ない呼び出しがログに出力されるのを抑制したい
  - ポリシーで dontaudit ルールを付与して成功失敗にかかわらずログ出力しない
    - semanage dontaudit で一時的にログ出力させることも可能
  - 典型的には特定ディレクトリのファイルをスキャンして open できたものを読むような動作

読むといいもの

- 利用時に必要なツールや用語の知識
  - SELinux ユーザーおよび管理者のガイド
    http://red.ht/2A07Pgc
- 本がほしい場合は……
  - SELinux System Administration - Second Edition
    http://bit.ly/2yF7bIb
  - 利用ツールが最新バージョンのため
    RHEL7 にマッチしない記述もあります
  - おそらく今入手できる本の中ではベスト
- hbstudy#28 SELinux HandsOn 公開版
  - http://bit.ly/2AmdEnG



読むといいもの（発展的）

- Security Enhanced Linux for Mere Mortals
  - Red Hat Summit 2017 でのセッション資料
  - http://bit.ly/2AsGNP3
- SELinux for Red Hat developers
  - 新規に作成したプログラム用に SELinux モジュールを作成するホワイトペーパー
  - http://red.ht/2vTXDWx
- SELinux Notebook
  - SELinux project のリファレンスドキュメント
  - https://selinuxproject.org/page/Category:Notebook
  - 大量に書いてあるので辞典的に利用する