

# 強敵よ！ ~ SELinuxとの比較 ~

日本セキュアOSユーザ会

海外 浩平 <kaigai@kaigai.gr.jp>



日本セキュア OS ユーザ会  
Japan Secure Operating System Users Group since 2007

TOMOYO

# ～ SELinuxとの比較～

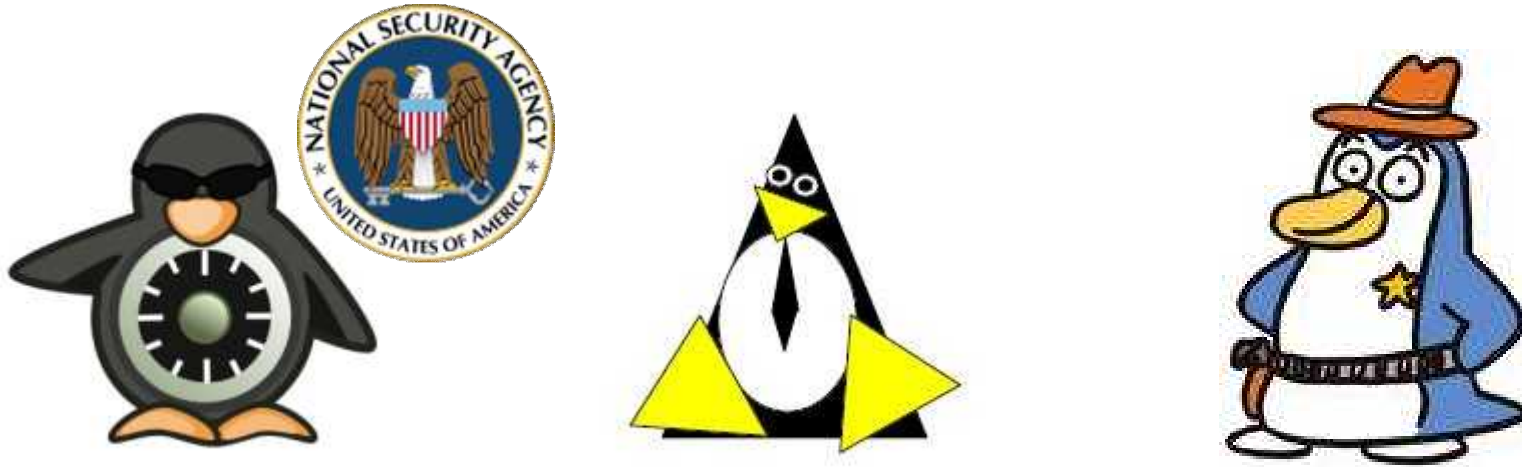
日本セキュアOSユーザ会

海外 浩平 <kaigai@kaigai.gr.jp>



日本セキュア OS ユーザ会  
Japan Secure Operating System Users Group since 2007

# はじめに



- 2.6.30カーネルでの TOMOYO Linux の統合  
**おめでとうございます！**
- LSMの上に乗っかる 強制アクセス制御 モジュールとしては、SELinux、Smackに続き3番目の統合
- AppArmorよ何処へ....。

# SELinuxとの比較

- ポイント
  - 何が同じなのか？
  - 何が違うのか？
- 違うところ
  - SELinuxのイイトコ
  - TOMOYO Linuxのイイトコ
  - 考え方の違い

# セキュアであるためには？ (1/2)

- セキュアであるとは？
  - 機密性 ... 'read'権限なしに情報を読み出せない
  - 完全性 ... 'write'権限なしに情報を更新できない
  - 可用性 ... 必要な時に必要な情報にアクセスできる
- 全てのアプリケーションが正しく動作すればよい
  - 本当にアクセス権チェックの漏れはないのか？
  - バグ/脆弱性が存在しない事は検証できない
- 望ましいデザインとは？
  - チェック漏れのリスクを最小化
  - 仮にバグ/脆弱性が存在しても、悪用を防ぐ



なあに、バグらなければどうという事はない。

# セキュアであるためには？ (1/2)

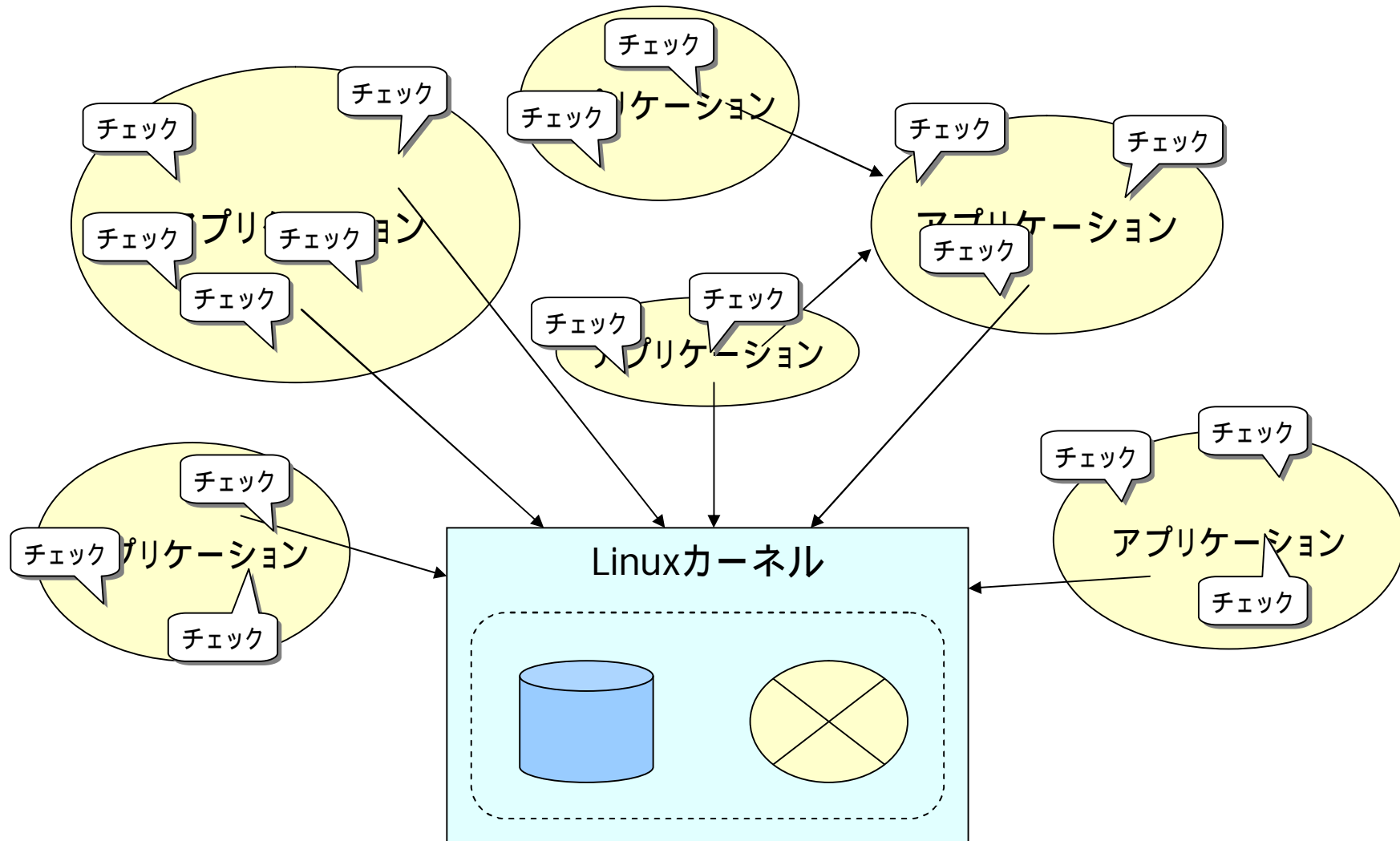
- セキュアであるとは？
  - 機密性 ... 'read'権限なしに情報を読み出せない
  - 完全性 ... 'write'権限なしに情報を更新できない
  - 可用性 ... 必要な時に必要な情報にアクセスできる
- 全てのアプリケーションが正しく動作すればよい
  - 本当にアクセス権チェックの漏れはないのか？
  - バグ/脆弱性が存在しない事は検証できない
- 望ましいデザインとは？
  - チェック漏れのリスクを最小化
  - 仮にバグ/脆弱性が存在しても、悪用を防ぐ

セキュリティ的に  
正しくない態度

~~なあに、バグらなければどうという事はない。~~

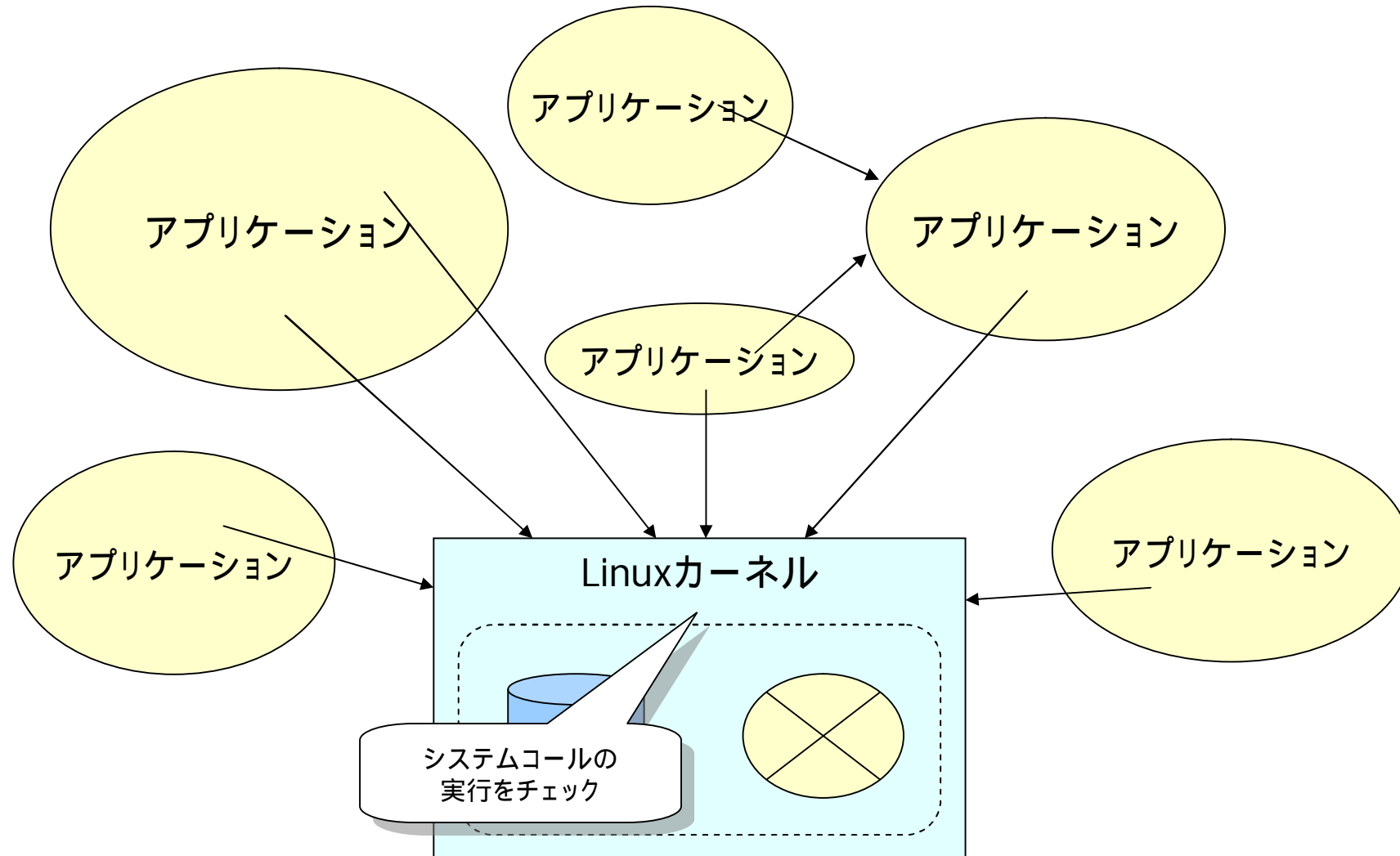
# セキュアであるためには？ (2/2)

- 鉄則： チェック箇所は少なければ少ないほど“良い”



# セキュアであるためには？ (2/2)

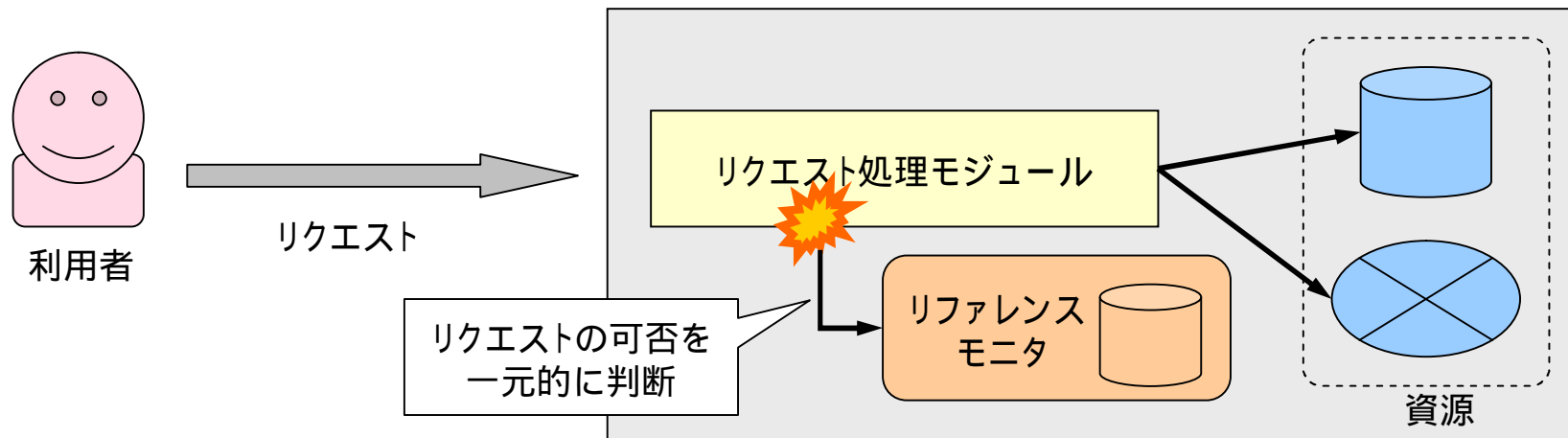
- 鉄則： チェック箇所は少なければ少ないほど“良い”





# リファレンスモニタ

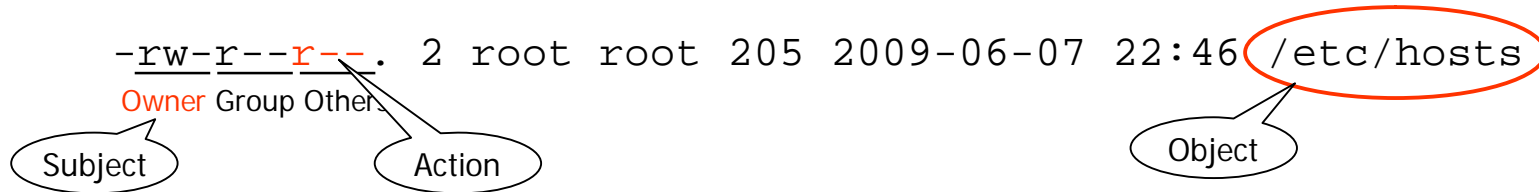
- リファレンスモニタとは？
  - 一元的にアクセス制御の意思決定を行うモジュール
  - 利用者の資源に対するアクセスを全て捕捉する (Always-Invoked)
  - 改ざんから保護 (Tamperproof)され、自身にバグがない (Small-Enough)
- 実装例
  - SELinux, Smack, TOMOYO Linux
  - LSM = リファレンスモニタを実装する枠組み
    - ✓ 利用者のシステムコール呼出しを確実に捕捉できる



# Subject, Object, Action

- アクセス制御とは？

- 「誰が (Subject)」「何に (Object)」「何をできるか (Action)」を決める事
- UNIXパーミッションで例えると。。。。



- では、どうやって、Subject、Objectを識別するのか？

アクセス制御マトリックス

Subject \ Object	/home/*	/etc/hosts	/var/www/html/*	/var/lib/pgsql/*
ユーザシェル(一般)	(read, write)	(read)	(-)	(-)
ユーザシェル(管理者)	(read,write)	(read,write)	(read, write)	(-)
Apache/webサーバ	(-)	(read)	(read)	(-)
PostgreSQLサーバ	(-)	(read)	(-)	(read,write)

# データとメタデータ

- データ
  - ファイルの中身、コンテンツ
- メタデータ
  - データの持つ属性を示す情報
  - ファイル名、拡張子、オーナー、更新時刻、セキュリティコンテキスト、etc...
  - ➡ SELinuxもTOMOYOも、メタデータを使ってObjectを識別する  
**どのメタデータを利用するか**という点が異なる。

メタデータ

```

[kaigai@ayu ~]$ ls -l memo.txt
-rw-r--r--. 1 kaigai users 1882 2009-06-29 17:27 memo.txt
[kaigai@ayu ~]$ cat memo.txt
  
```

データ

我々は一人の英雄を失った。これは敗北を意味するのか？否！始まりなのだ！地球連邦に比べ我がジオンの国力は30分の1以下である。にも関わらず今日まで戦い抜いてこられたのは何故か！諸君！我がジオンの戦争目的が正しいからだ！それは諸君らが一番知っている。

# TOMOYO Linuxの場合

プロセスのメタデータ

- プロセス起動履歴とパス名に基づくアクセス制御

ファイルのメタデータ

```

<<< Domain Policy Editor >>> 28 entries '?' for help
<kernel> /etc/init.d/gdm /sbin/start-stop-daemon /usr/sbin/gdm /etc/gdm/Xsession
0: allow_read /usr/lib/locale/ja_JP.utf8/LC_ADDRESS
1: allow_read /usr/lib/locale/ja_JP.utf8/LC_COLLATE
2: allow_read /usr/lib/locale/ja_JP.utf8/LC_CTYPE
3: allow_read /usr/lib/locale/ja_JP.utf8/LC_IDENTIFICATION
4: allow_read /usr/lib/locale/ja_JP.utf8/LC_MEASUREMENT
5: allow_read /usr/lib/locale/ja_JP.utf8/LC_MESSAGES
6: allow_read /usr/lib/locale/ja_JP.utf8/LC_MONETARY
7: allow_read /usr/lib/locale/ja_JP.utf8/LC_NAME
8: allow_read /usr/lib/locale/ja_JP.utf8/LC_NUMERIC
9: allow_read /usr/lib/locale/ja_JP.utf8/LC_PAPER
10: allow_read /usr/lib/locale/ja_JP.utf8/LC_TELEPHONE
11: allow_read /usr/lib/locale/ja_JP.utf8/LC_TIME
12: allow_read /var/log/tomoyo/reject_log
13: allow_env COLORTERM
14: allow_env DISPLAY
15: allow_env HOME
16: allow_env LANG
17: allow_env LOGNAME
18: allow_env PATH
19: allow_env SHELL
20: allow_env SUDO_COMMAND
  
```

プロセス起動履歴

```

端末
<<< Domain Transition Editor >>> 602 domains '?' for help
<kernel>
0: 1 <kernel>
1: 1 * /etc/init.d/acpi-support
2: 1 /bin/sed
3: 1 /etc/acpi/power.sh
4: 1 /bin/grep
5: 1 /sbin/on_ac_power
6: 1 /bin/grep
7: 1 /sbin/acpi_available
8: 1 /usr/sbin/invoke-rc.d
9: 1 /bin/ls
10: 1 /bin/sed
    /etc/init.d/anacron ( -> 32 )
11: 1 /sbin/runlevel
12: 1 /usr/bin/xargs
13: 1 /bin/echo
14: 1 /sbin/usplash_write
15: 1 /usr/bin/expr
16: 1 /usr/bin/tput
17: 1 /usr/sbin/dmidecode
18: 1 * /etc/init.d/acpid
19: 1 /bin/cat
  
```

アクセス可能なパス名

# SELinuxの場合

- セキュリティコンテキストに基づくアクセス制御

プロセスとファイルのメタデータ

```

ayuu.myhome.cx - Tera Term VT
ファイル(E) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
#####
#
# postgresql Local policy
#
allow postgresql_t self:capability [ kill dac_override dac_read_search c
hown fowner fsetid setuid setgid sys_nice sys_tty_config sys_admin ];
dontaudit postgresql_t self:capability [ sys_tty_config sys_admin ];
allow postgresql_t self:process signal_perms;
allow postgresql_t self:fifo_file rw_fifo_file_perms;
allow postgresql_t self:sem create_sem_perms;
allow postgresql_t self:shm create_shm_perms;
create_stream_socket_perms;
create_stream_socket_perms;
create_stream_socket_perms;
create_stream_socket_perms;
create_stream_socket_perms;
create_stream_socket_perms;
type:db_database *;
process:db_database sepgsql_db_t;
process:db_database install_module;
process:db_database load_module;
  
```

```

ayuu.myhome.cx - Tera Term VT
ファイル(E) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
[kaigai@ayu ~]$ ls -Z /usr/
drwxr-xr-x. root root system_u:object_r:bin_t:SystemLow bin
drwxr-xr-x. root root system_u:object_r:etc_t:SystemLow etc
drwxr-xr-x. root root system_u:object_r:usr_t:SystemLow games
drwxr-xr-x. root root system_u:object_r:usr_t:SystemLow include
drwxr-xr-x. root root system_u:object_r:usr_t:SystemLow kerberos
drwxr-xr-x. root root system_u:object_r:lib_t:SystemLow lib
drwxr-xr-x. root root system_u:object_r:bin_t:SystemLow libexec
drwxr-xr-x. root root system_u:object_r:usr_t:SystemLow local
drwxr-xr-x. root root system_u:object_r:bin_t:SystemLow sbin
drwxr-xr-x. root root system_u:object_r:usr_t:SystemLow share
drwxr-xr-x. root root system_u:object_r:src_t:SystemLow src
lrwxrwxrwx. root root system_u:object_r:usr_t:SystemLow tmp -> ../var/tmp
[kaigai@ayu ~]$
[kaigai@ayu ~]$ ps -Z
  PID TTY          TIME CMD
unconfined_u:unconfined_r:unconfined_t:SystemLow-SystemHigh 2147 pts/5 00:00:00 ps
unconfined_u:unconfined_r:unconfined_t:SystemLow-SystemHigh 5590 pts/5 00:00:01 bash
[kaigai@ayu ~]$
  
```



# 良いトコ悪いトコ (1/2)

- Objectをユニークに識別できるか？
  - パス名の場合
    - ハードリンク、bind-mount、namespace
    - ➡ 同じ Object が複数のパス名を持ち得る
  - セキュリティコンテキスト
    - 常に inode と 1:1 対応するため、矛盾する事はない
- パス名は"使い方"を表現する
  - open(2)の引数はパス名、inode番号ではない
  - アクセス"対象"の制御か？ アクセス"手段"の制御か？
- パス名は情報フローを記憶しない
  - 新規ファイルのセキュリティコンテキストは、「誰が」作成したかによって変化する

# 良いトコ悪いトコ (2/2)

- セキュリティポリシー

- SELinux

- 標準セキュリティポリシーがかなり良くできている
- セキュリティの専門家が作成したベストプラクティスの導入
- 一方、スクラッチからのポリシー体系の作成は困難

- TOMOYO Linux

- ポリシーの自動学習機能
- アプリケーションの正常動作時の振る舞いをヒントにポリシー生成
  - ✓ それを検証するのは人間の役割

# TOMOYO Linuxの今後に期待

- 標準Auditdとの統合
  - イベントディスパッチャとか、Audit-Logをリモートに飛ばして集中管理する機能がある
- アプリケーション向けの機能
  - SE-PostgreSQLとかXACE/SELinuxを作るためのインターフェース
  - Named objectなら、既存構文の拡張で無理？  
`allow_select database.schema.table.column`
- “プロセス起動履歴”の相対指定
  - sshdからログインしても、telnetdからログインしても、「ある起点」からの起動履歴として扱えないか？



# まとめ

- SELinux も TOMOYO Linux も、  
**本質的にはリファレンスモニタの一種**
  - アクセス制御に使うメタデータの種類が違う程度
    - ✓ どのメタデータを利用するかが、アクセス制御システムの性格を決定付ける
- 思想の違い
  - ポリシーは誰が作るべきか？
    - セキュリティの専門家が作るべきだ、ベストプラクティスはそこにある。
    - システム管理者が作るべきだ、アプリケーションは "あるべき振る舞い" を教えてくれる
- エール
  - 何事も、好敵手があってこそその発展です。  
SELinux、TOMOYO Linux 共に頑張っていきましょう！