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～

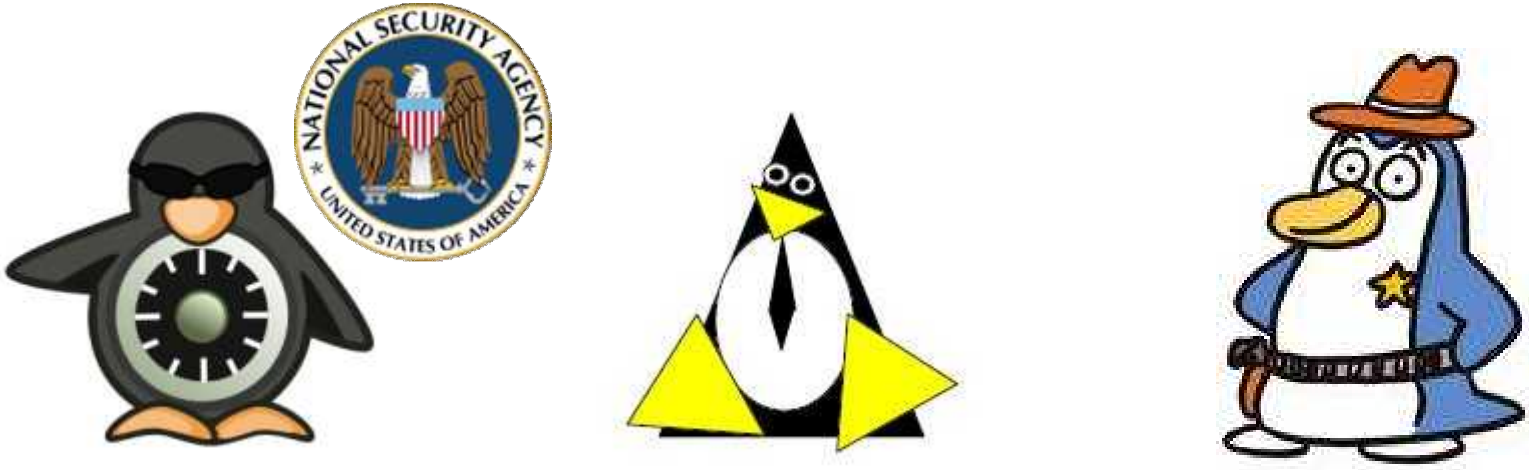
日本セキュアOSユーザ会

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！

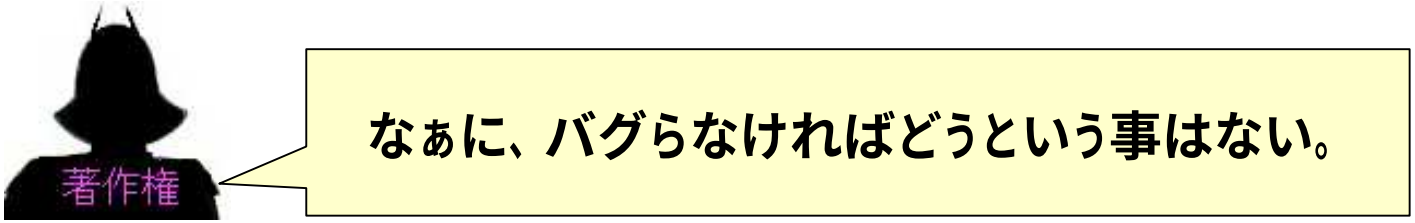
- LSMの上に乗っかる 強制アクセス制御 モジュールとしては、SELinux、Smackに続き3番目の統合
- AppArmorよ何処へ…。

SELinuxとの比較

- ポイント
  - 何が同じなのか？
  - 何が違っているのか？
- 違っているところ
  - SELinuxのイイトコ
  - TOMOYO Linuxのイイトコ
  - 考え方の違い

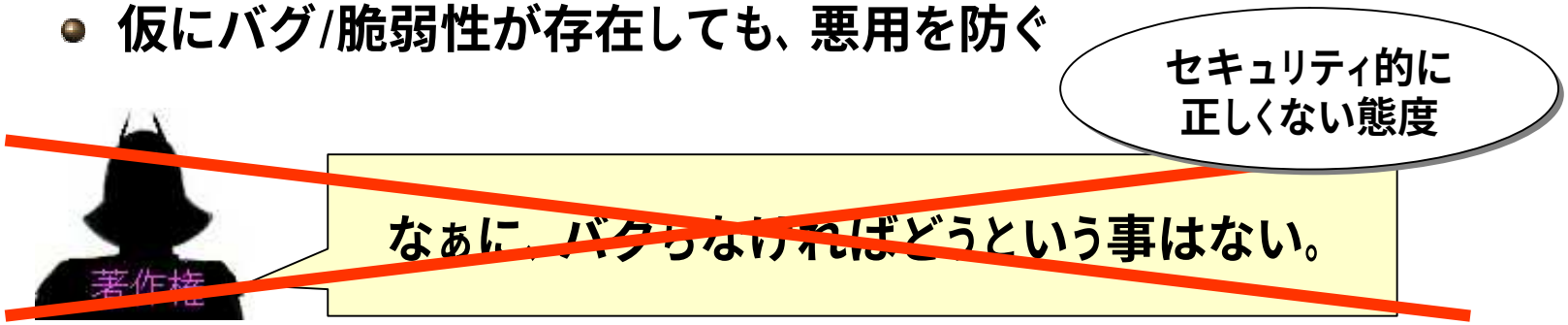
セキュアであるためには？ (1/2)

- 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



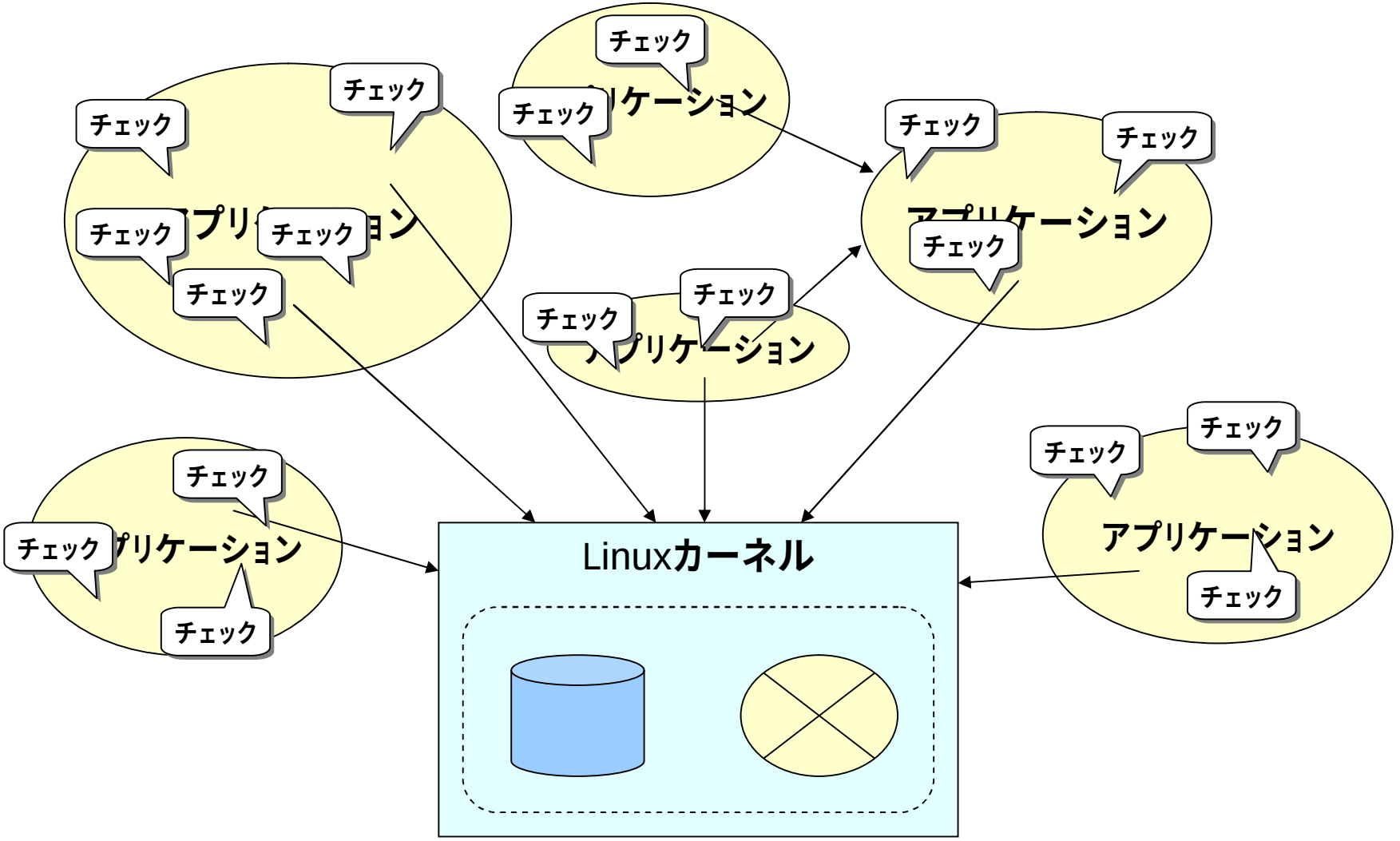
セキュアであるためには？ (1/2)

- 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



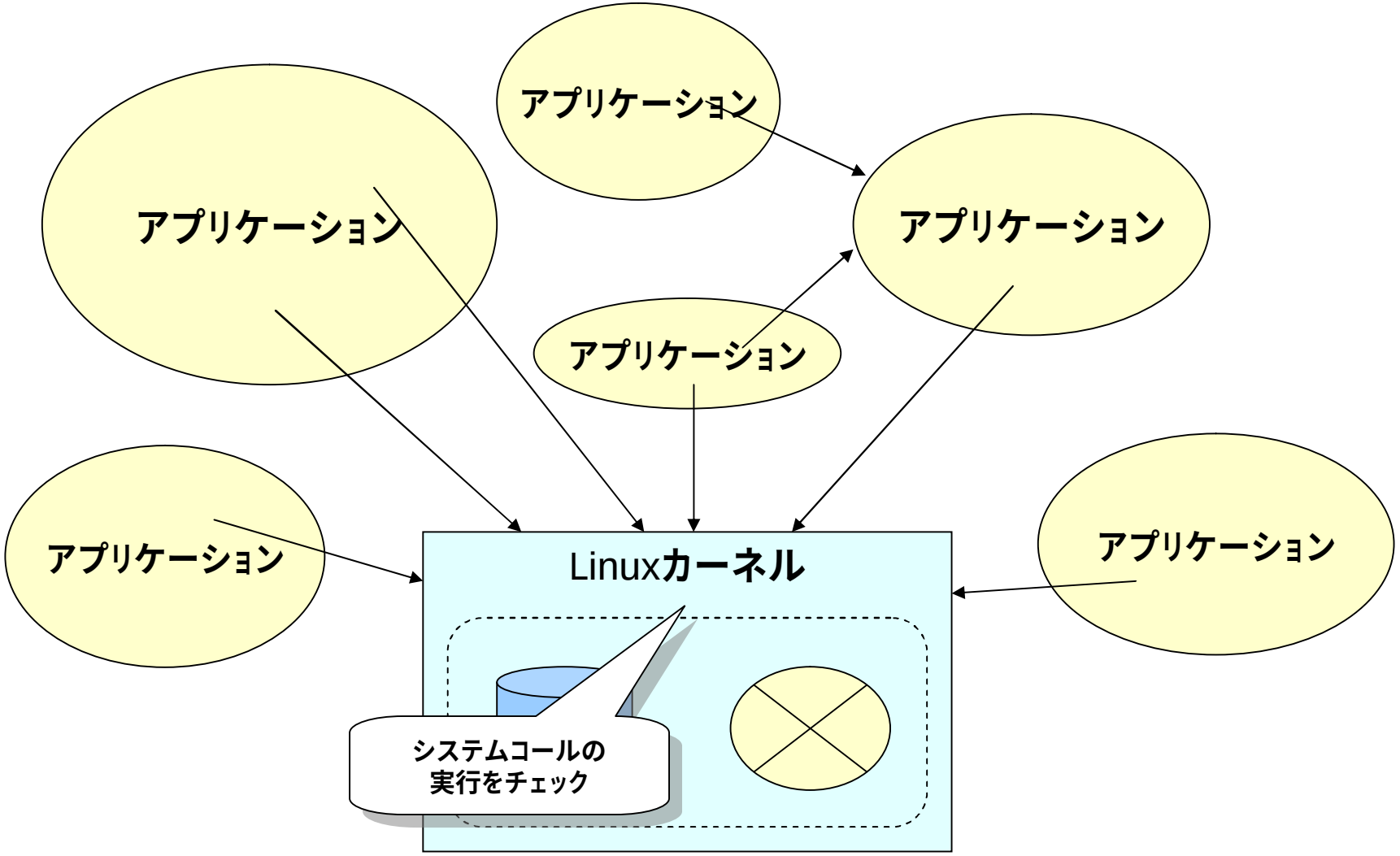
セキュアであるためには？ (2/2)

- 鉄則： チェック箇所は少なければ少ないほど”良い”



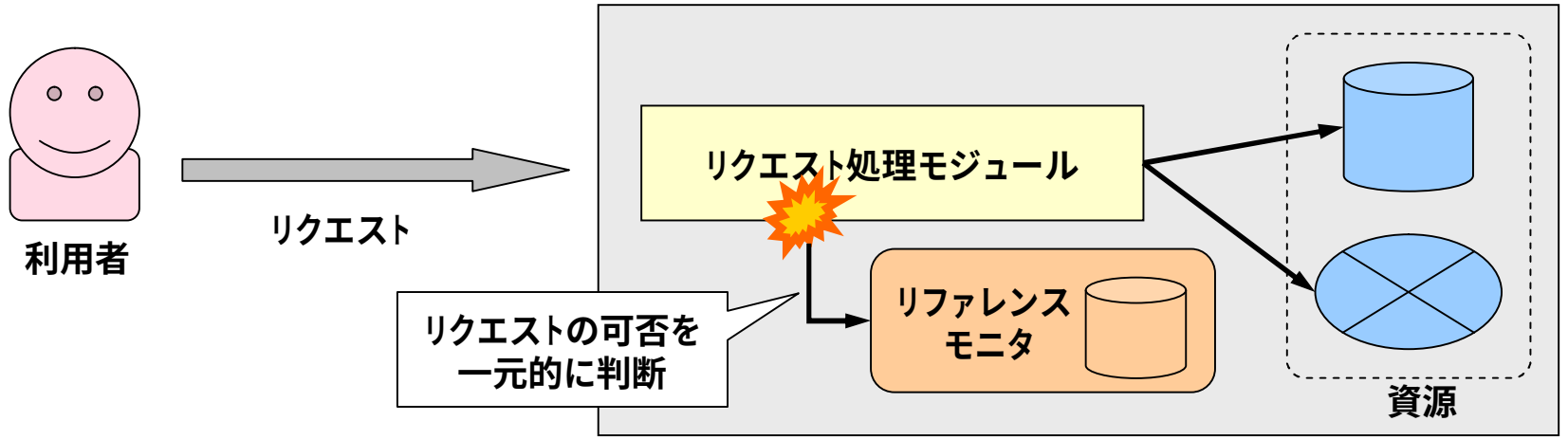
セキュアであるためには？ (2/2)

- 鉄則： チェック箇所は少なければ少ないほど”良い”



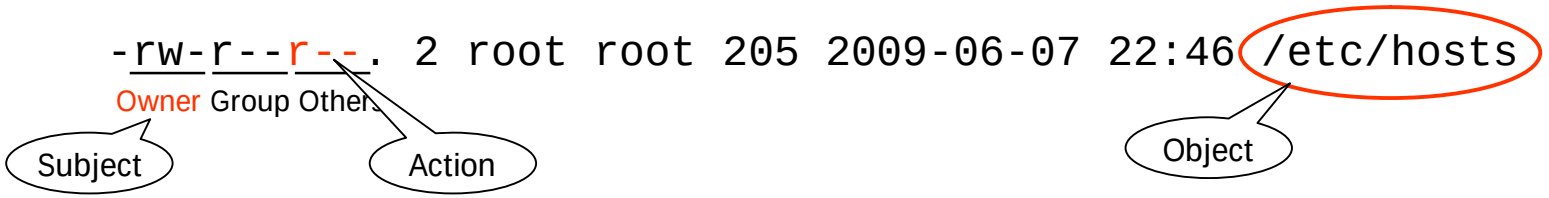
リファレンスモニタ

- 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。。。**



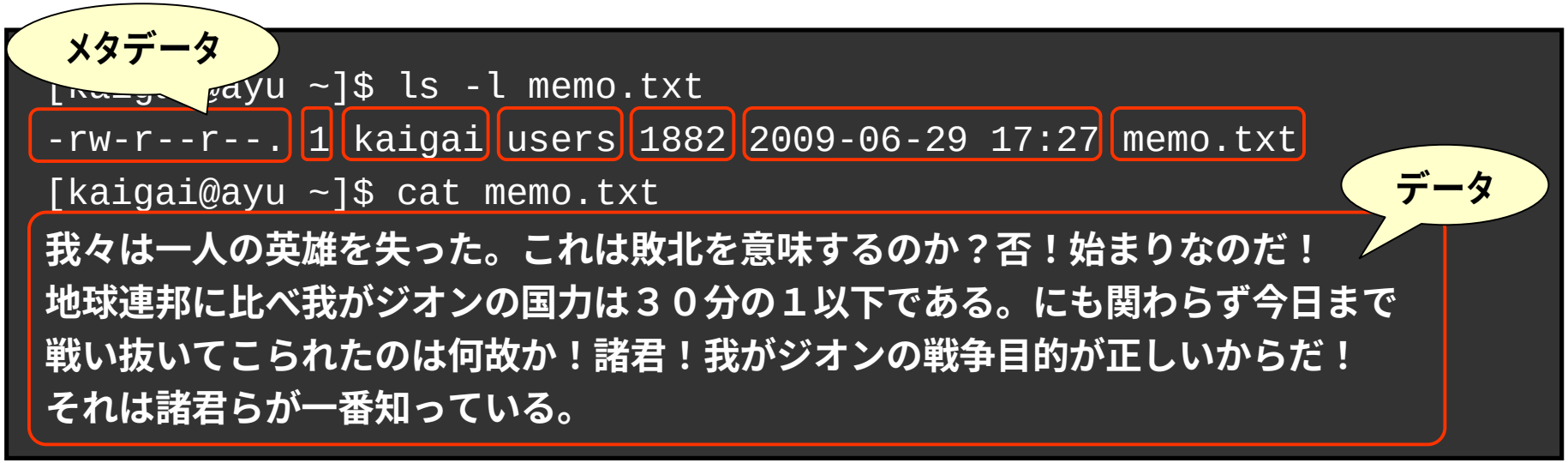
- **では、どうやって、**Subject、Object**を識別するのか？**

アクセス制御マトリックス

Object / Subject	/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、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。

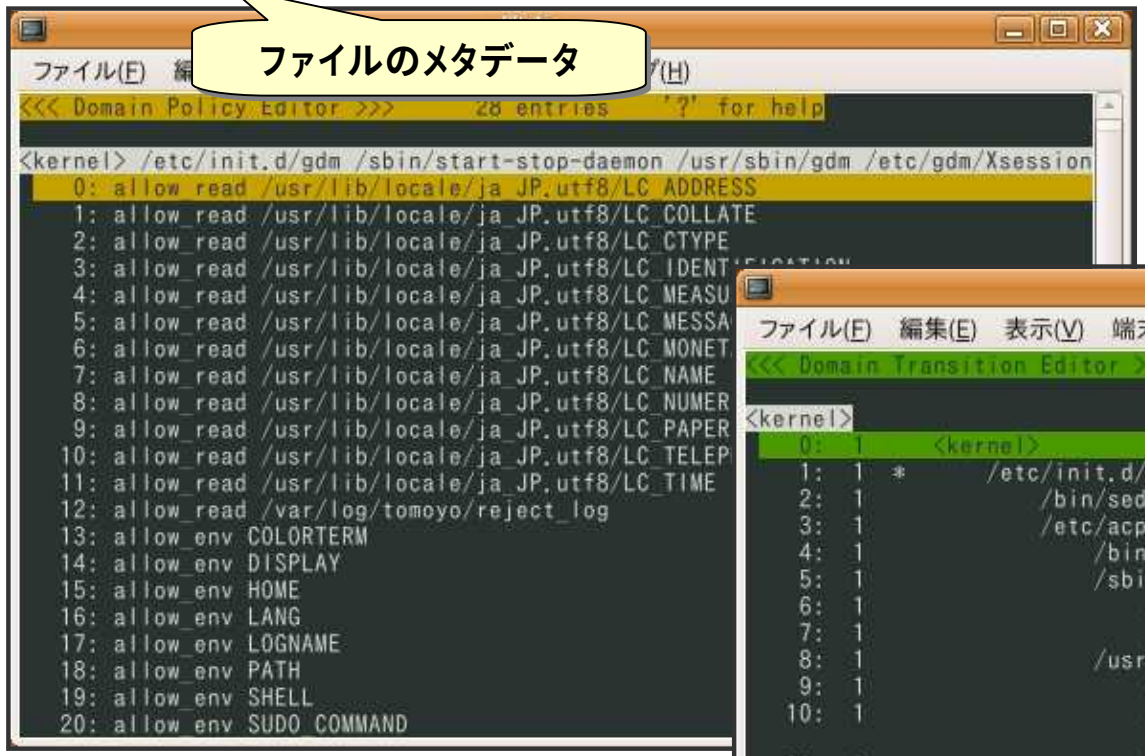


TOMOYO Linuxの場合

- **プロセス起動履歴**と**パス名**に基づくアクセス制御

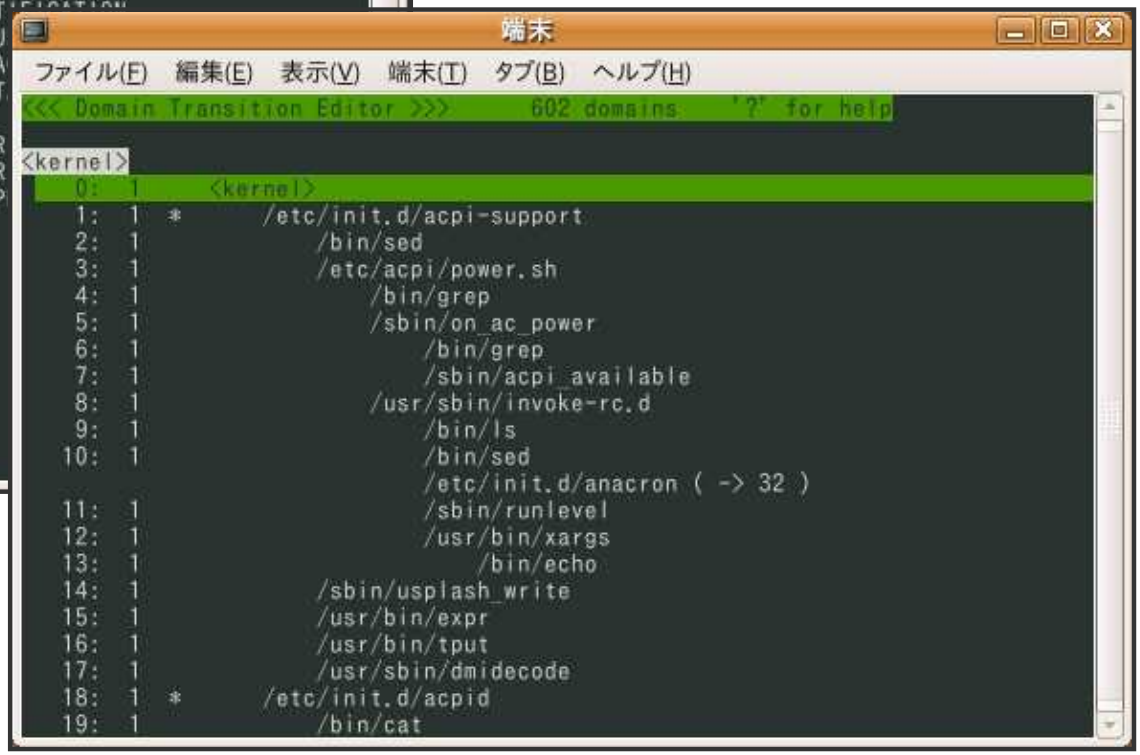
プロセスのメタデータ

ファイルのメタデータ



アクセス可能なパス名

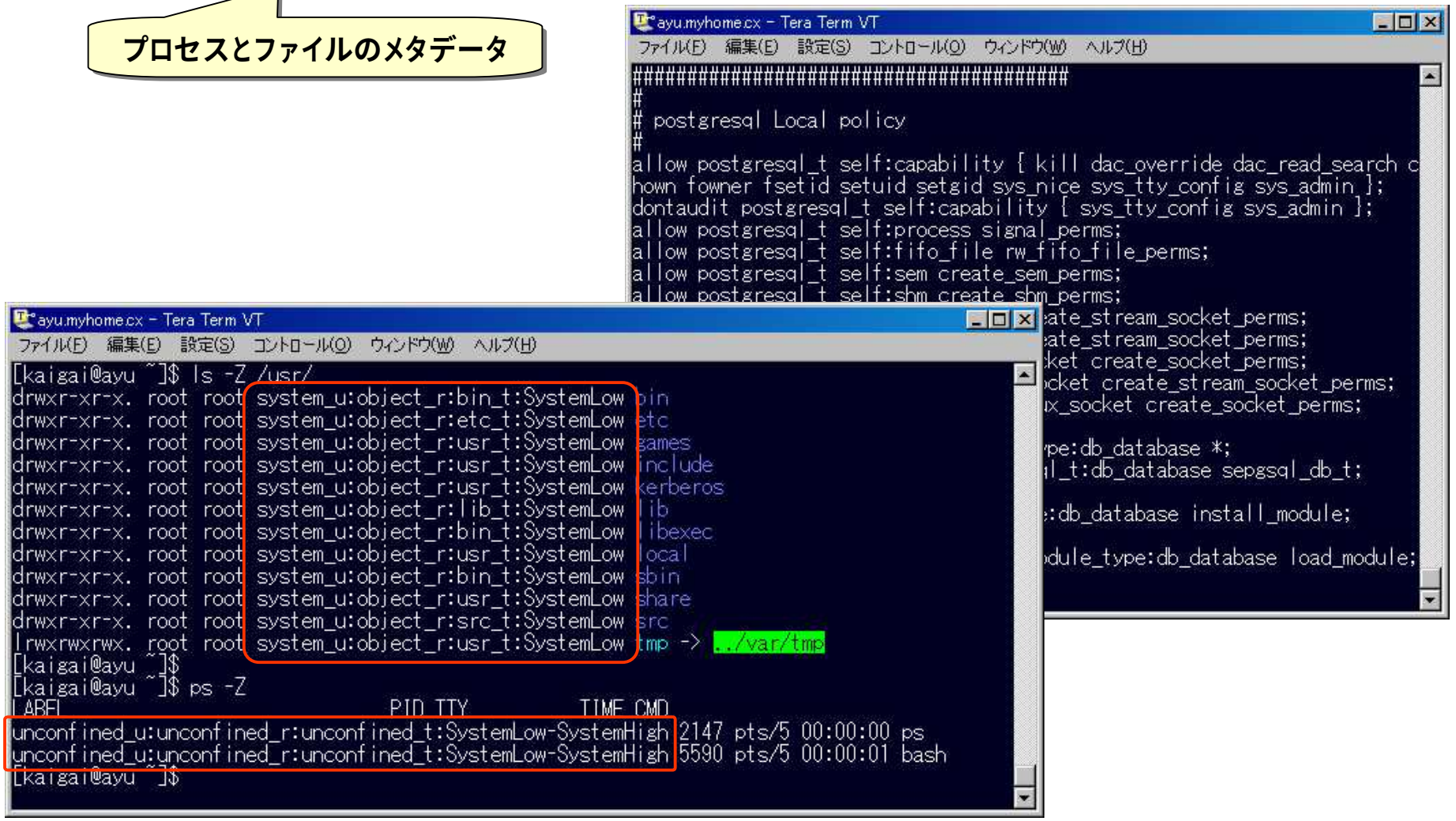
プロセス起動履歴



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



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の発展です。
    SELinux、TOMOYO Linux 共に頑張っていきましょう！