Empowered by Innovation NEC

タイプ継承を可能にするSELinux ポリシーコンパイラの拡張

平成17年6月2日

NEC Linux推進センター
海外 浩平

NEC & Linux

Empowered by Innovation NEC

Today's Agenda

- SELinuxの概要と考え方
- セキュリティポリシーの記述方法と問題点
- タイプ継承に基づくセキュリティポリシー
- SELinuxポリシーコンパイラの拡張
- タイプ継承を利用したポリシーの記述
- まとめ

はじめに ～SELinuxの概要～

- Linux2.6.x用のセキュリティ強化モジュール
- 万が一、侵入を許しても“何もさせない”アプローチ
  - ➘ FW、IDSのような水際作戦とは異なる。

☺ SELinuxが有効なケース
- ✓ アプリ脆弱性を突いた侵入、不正な権限昇格
- ✓ ウィルス、ワーム、トロイの木馬
- ✓ 目的外の機密情報の参照

- アプリケーションには不必要な権限が多すぎる。
  - ➘ セキュリティホール経由で、これらの権限を悪用される
  - ➘ BOFからシェルを実行されて、ファイルを改ざんされたり
- SELinuxは攻撃者の手足を縛ってしまう。

SELinuxにおける考え方

- アプリケーションを必要最小権限で動作させる
  - アプリケーションには不必要な権限が盛りだくさん
  - 粒度の細かいアクセス制御（約２００種類！）
  - 仮に脆弱性が存在しても「想定の範囲内」の行動しか許可されない

  ⇔　UNIXモデルでは、root権限を取られたら無条件降伏
- セキュリティポリシーによる一括設定
  - アプリケーションが「やっていいこと」を予め列挙しておく
  - 全てのユーザ・プロセスはセキュリティポリシーに従う
  - “誰が”、“誰に”、“何をできる”かを列挙したもの

  ↘ セキュリティポリシーの記述に反する事は一切許さない

  ↘「正しい」セキュリティポリシーが、SELinuxの生命線

セキュリティポリシーのいろは

- 基本は「誰が」「誰に」「何をする」 …… 脚本
  - 誰が？： ほとんどがプロセス
  - 誰に？： ファイル、ソケット、共有メモリ
  - 何をする：「ディレクトリに対するエントリの追加」
    「親プロセスへのシグナルの送信」
    「TCPソケットに対するパケットの送信」
    「共有メモリの書き込み可能マップ」

俳優
タイプ
脚本
アクセス許可

- 「PostgreSQLデータベールファイル」を定義する

```
TYPE postgresql_db_t, file_type, sysadmfile;
```

- 「Apacheが」「Webコンテンツを」「ファイルとして読み込む」

```
ALLOW httpd_t httpd_sys_content_t : file read ;
```

↘ 「俳優」は脚本通りに演技しなければならない。

↘ SELinuxのセキュリティポリシーはこのような記述（脚本）の集合

「誰が」「誰に」「何をする」構文

登場人物の宣言・・・TYPE構文

TYPE <タイプ名> [ALIAS <別名>] [,<属性名> ,...];

[使用例]

TYPE httpd_sys_content_t, file_type;

TYPE httpd_t, domain;

[意　味]]

- ✓ httpd_sys_content_tの名前のタイプを定義し、file_type属性を付与する。
- ✓ httpd_tという名前のタイプを定義し、domain属性を付与する。

脚本の執筆・・・ALLOW構文

ALLOW <タイプ名(Subject)> <タイプ名(Object)>
: <クラス名> <パーミッション> ;

[使用例]

ALLOW httpd_t httpd_sys_content_t : file {getattr read lock ioctl} ;

[　意　味　]

Apache(httpd_t)が、Webコンテンツ(httpd_sys_content_t)の、
ファイルに対してgetattr/read/ioctl/lockの操作を行うことを許可

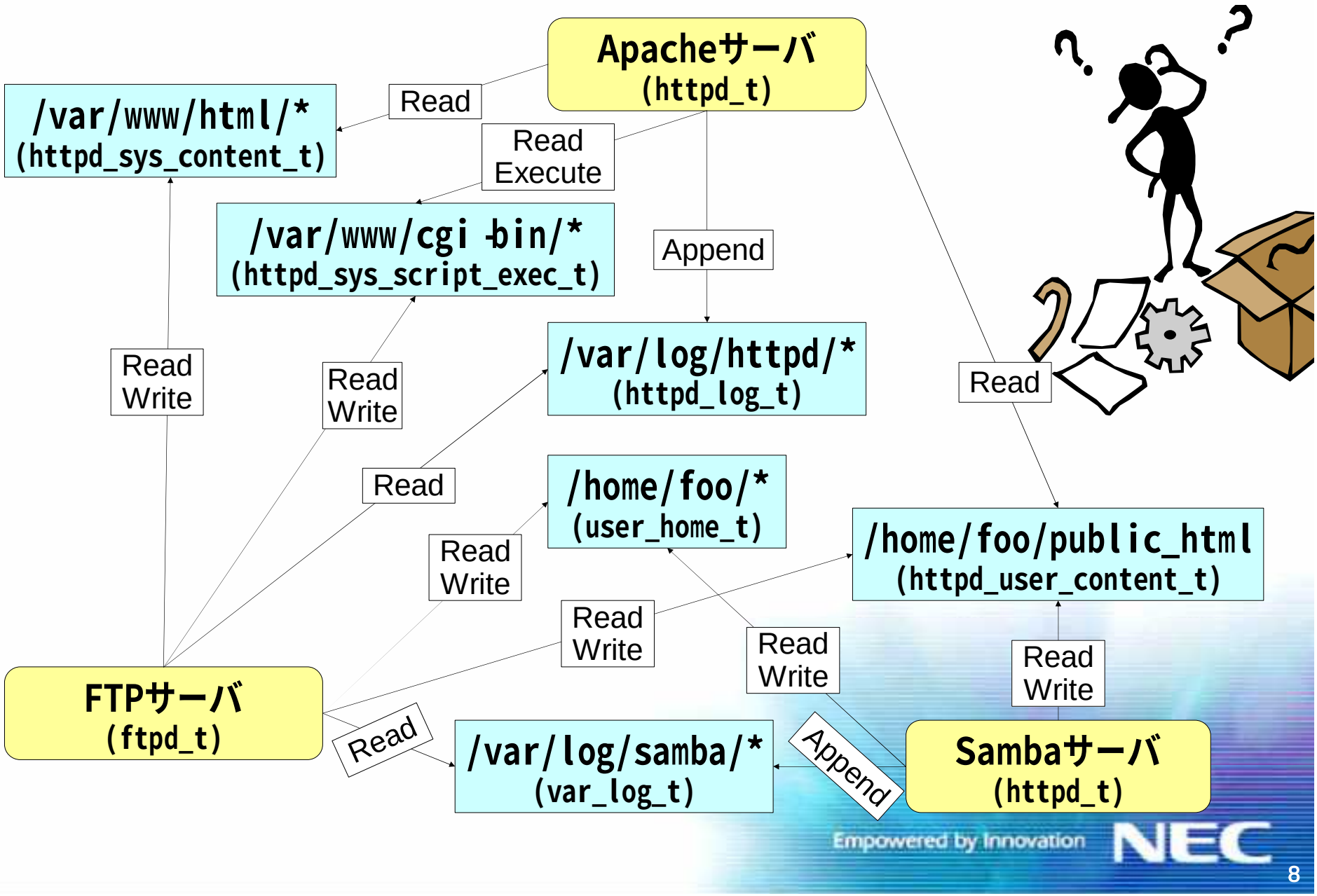
セキュリティポリシー記述の課題

- 複数のアプリケーションに関連する設定が複雑
  - FTP/Samba/Apacheから共通にアクセスされるファイルに対するアクセス権の設定をどのように行うか？
    - ↘ アプリケーションの組み合わせが増えるほど煩雑･冗長になる
  - PHP/CGIスクリプトがPostgreSQLへ接続するための設定は、どこの設定ファイルに記述すればよいのか？
    - ↘ ポリシーの相互依存関係が強くなりすぎる
    - ↘ インターフェース“的”なものを明確に定義することが必要
- 「細かすぎる」パーミッションの粒度
  - ↘ 直感的に把握可能なものに抽象化することが必要

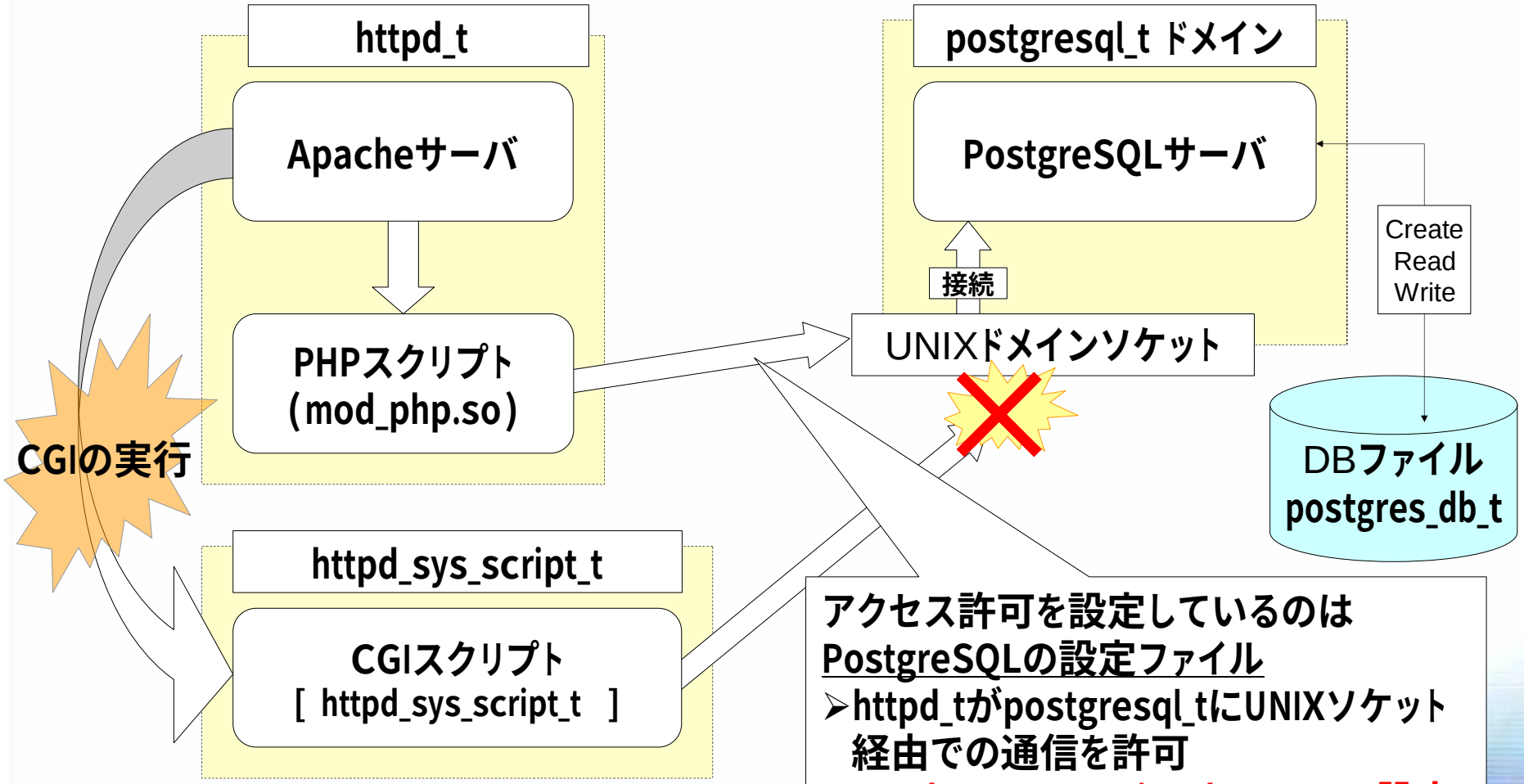
⇨ セキュリティポリシーの記述に高いスキルを要求される

- ✓ 大量のポリシーを一貫性を持ったまま記述しなければならない。
- ✓ アプリケーション間の依存関係を理解している必要がある。

FTP/Samba/Apacheの例



PHP/CGIとPostgreSQLの例(1)



- ApacheのポリシーとPostgreSQLのポリシーが強く依存している。
- 「PostgreSQLに接続可能なタイプ」を抽象化する仕組みが必要。

PHP/CGIとPostgreSQLの例(2)

```
test.cgi – PostgreSQLに接続するCGI
#!/usr/bin/perl
use DBI;
$conn = DBI ->connect("dbi:Pg:
          dbname=test; user=test;");
$SQL = "select * from testtbl";

$res = $conn ->prepare($SQL);
$code = $res ->execute();
                :
```

mod_perl経由で実行
(httpd_t)

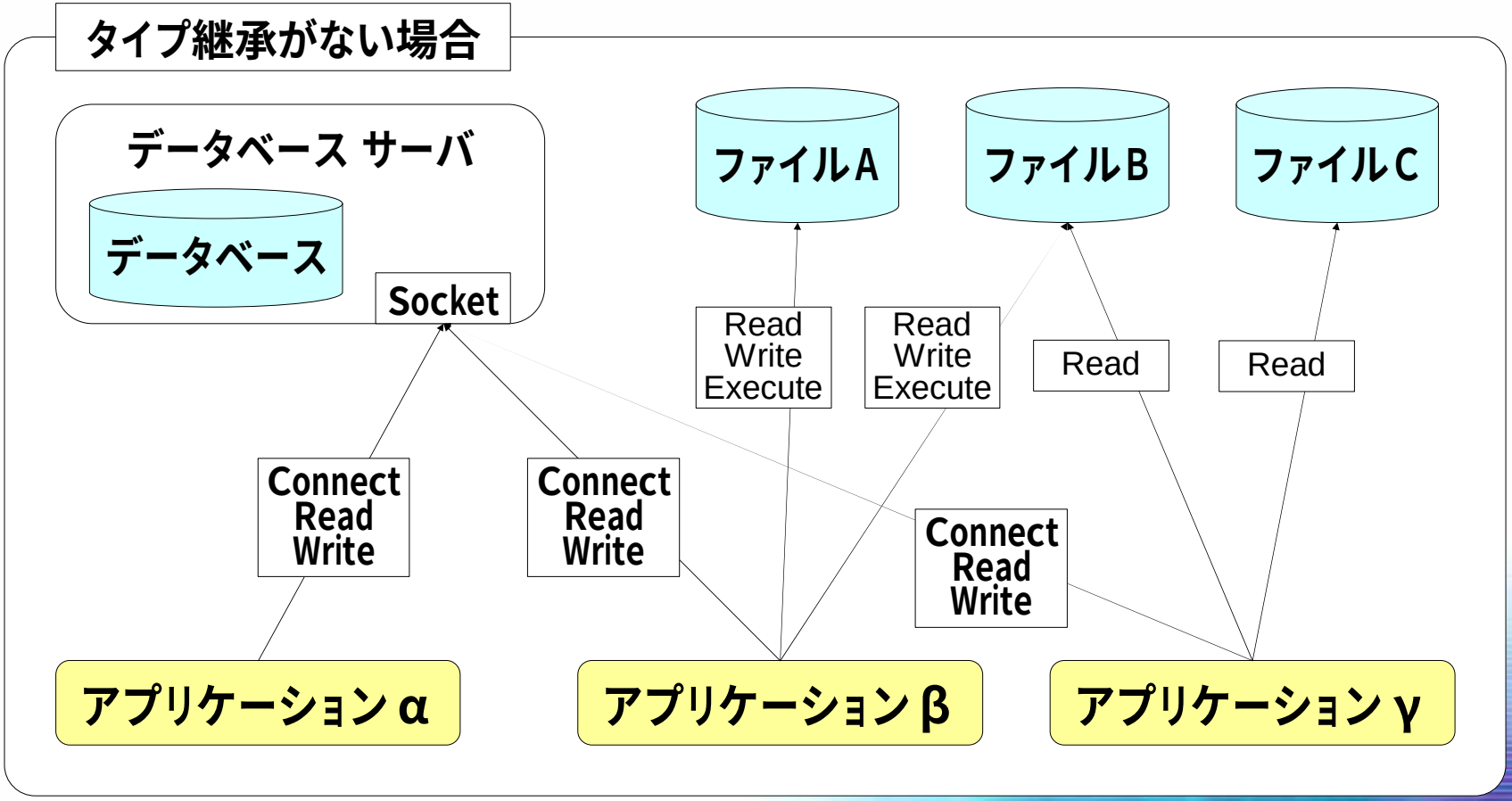
独立のCGIとして実行
(httpd_sys_script_t)

【 アクセス拒否ログ 】

```
audit(1115774411.448:0): avc:  denied
{ write } for pid=20243 exe=/usr/bin/perl
name=.s.PGSQL.5432 dev=sda2 ino=2205106
scontext=root:system_r:httpd_sys_script_t
tcontext=root:object_r:postgresql_tmp_t
tclass=sock_file
```

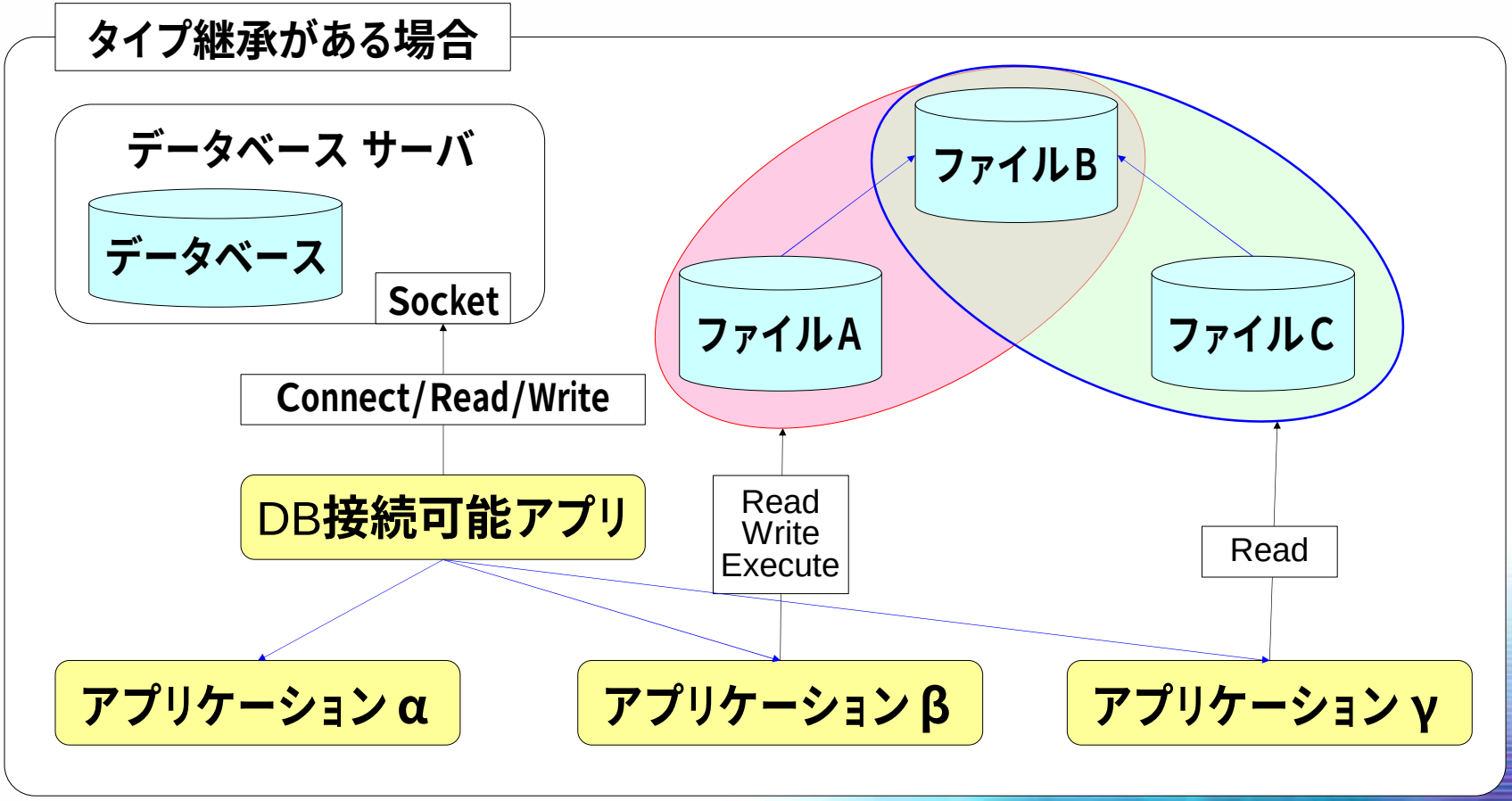
タイプ継承に基づくセキュリティポリシー

- タイプ間に親子関係を定義、権限を継承させることができる。
- 多重継承を許し、タイプは複数の親を持つことができる。

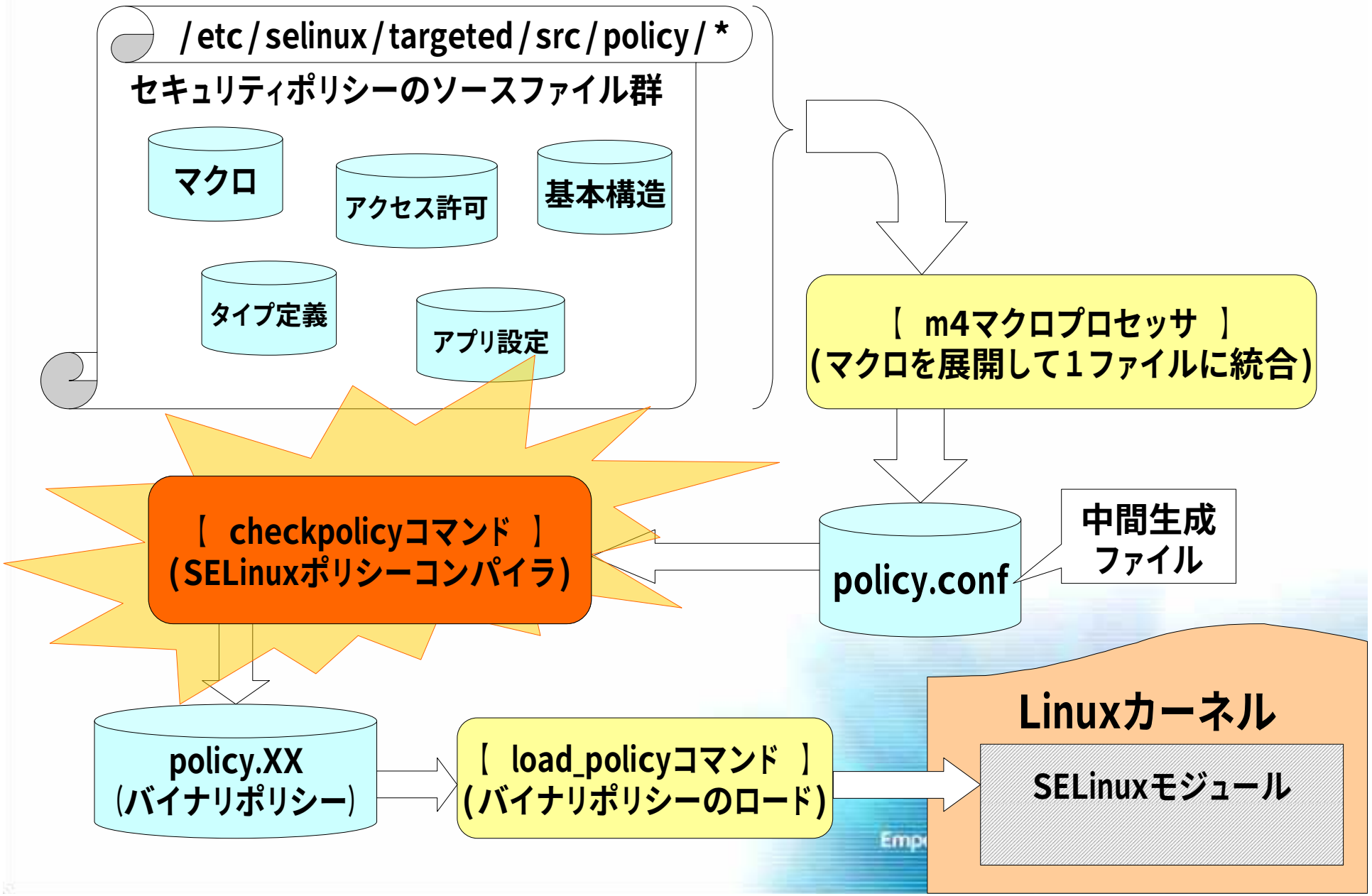


タイプ継承に基づくセキュリティポリシー

- タイプ間に親子関係を定義、権限を継承させることができる。
- 多重継承を許し、タイプは複数の親を持つことができる。



セキュリティポリシーの生成フロー



SELinuxポリシーコンパイラの拡張

- 継承関係を定義するための構文

  TYPE <タイプ名> EXTENDS <親タイプ> [, ...] ;

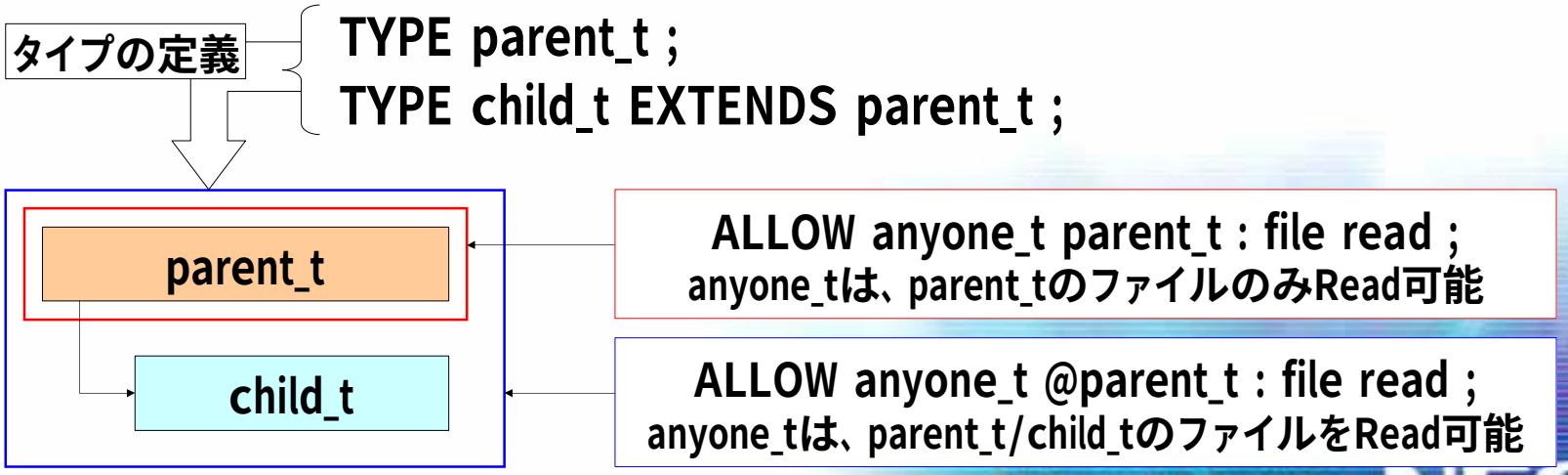
  例） TYPE child_t EXTENDS parent_t ;

  ➘ child_t は parent_t を親タイプに持つ

- 継承を意識したアクセス許可を与えるための拡張

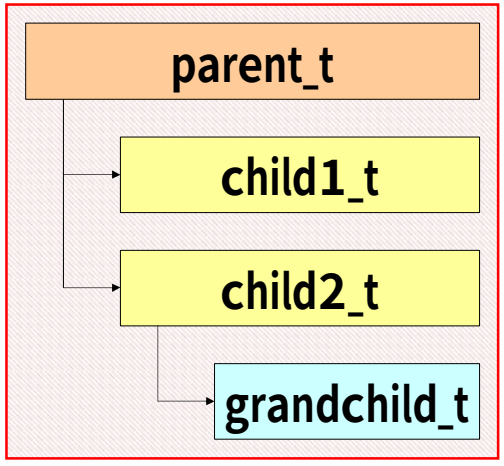
  タイプ名に対する '@'プレフィクスの追加

  ➘ 親とその子孫がタイプ指定の範囲となる。



従来のポリシーコンパイラとの互換性

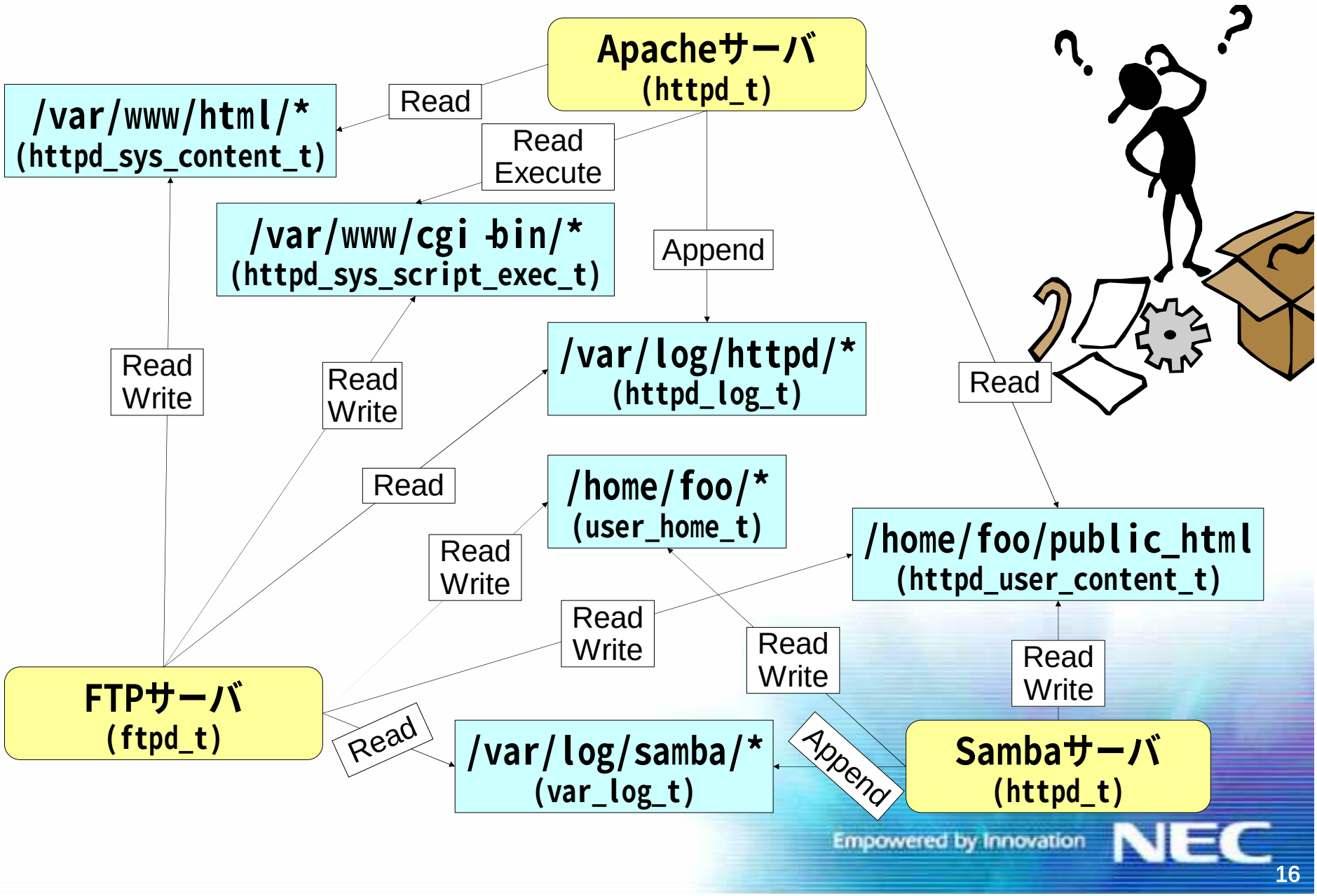
- セキュリティポリシー・ソースレベル互換性
  - ✓ “EXTENDS”のないタイプ宣言は継承関係を定義しない。
  - ✓ “@”を付与しない権限の付与は、従来通りに認識される。
  - ➘ EXTENDSや@を使用しない従来の構文の解釈は全く変わらない。
- セキュリティポリシー・バイナリレベル互換性
  - ✓ タイプ間の親子関係を内部的に展開してコンパイルする。
  - ✓ バイナリ形式は従来のものを一切変更していない。
  - ➘ 既存のポリシー分析ツールなどをそのまま使用することができる。



@parent_t ⇔{parent_t child1_t child2_t grandchild_t}
@child2_t ⇔{child2_t grandchild_t }
@parent_t - @child2_t ⇔{ parent_t child1_t }

ALLOW anyone_t @parent_t : file {read write};
⇔ ALLOW anyone_t {parent_t child1_t child2_t grandchild_t}: file {read write};

FTP/Samba/Apacheの例

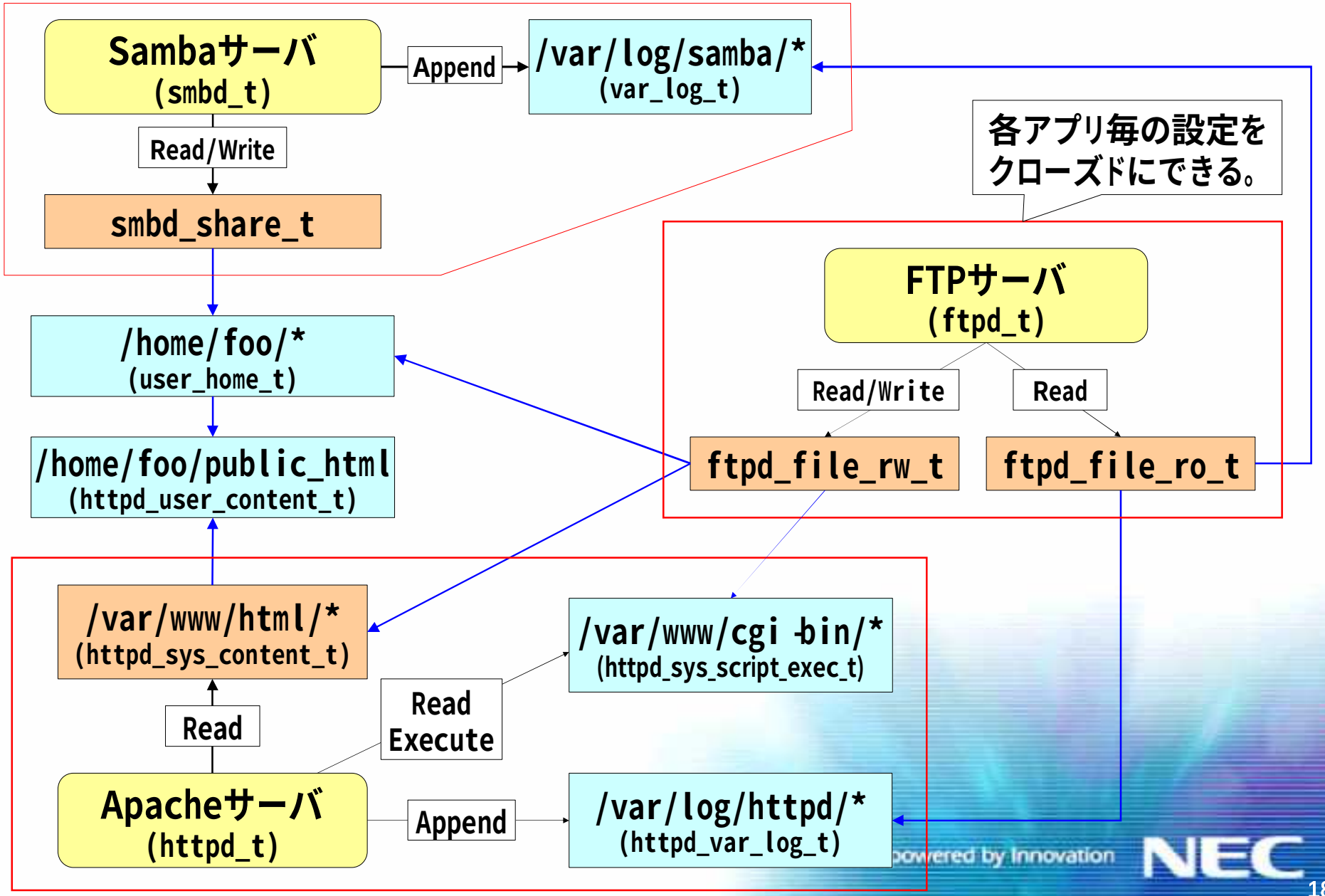


FTP/Samba/Apacheの例

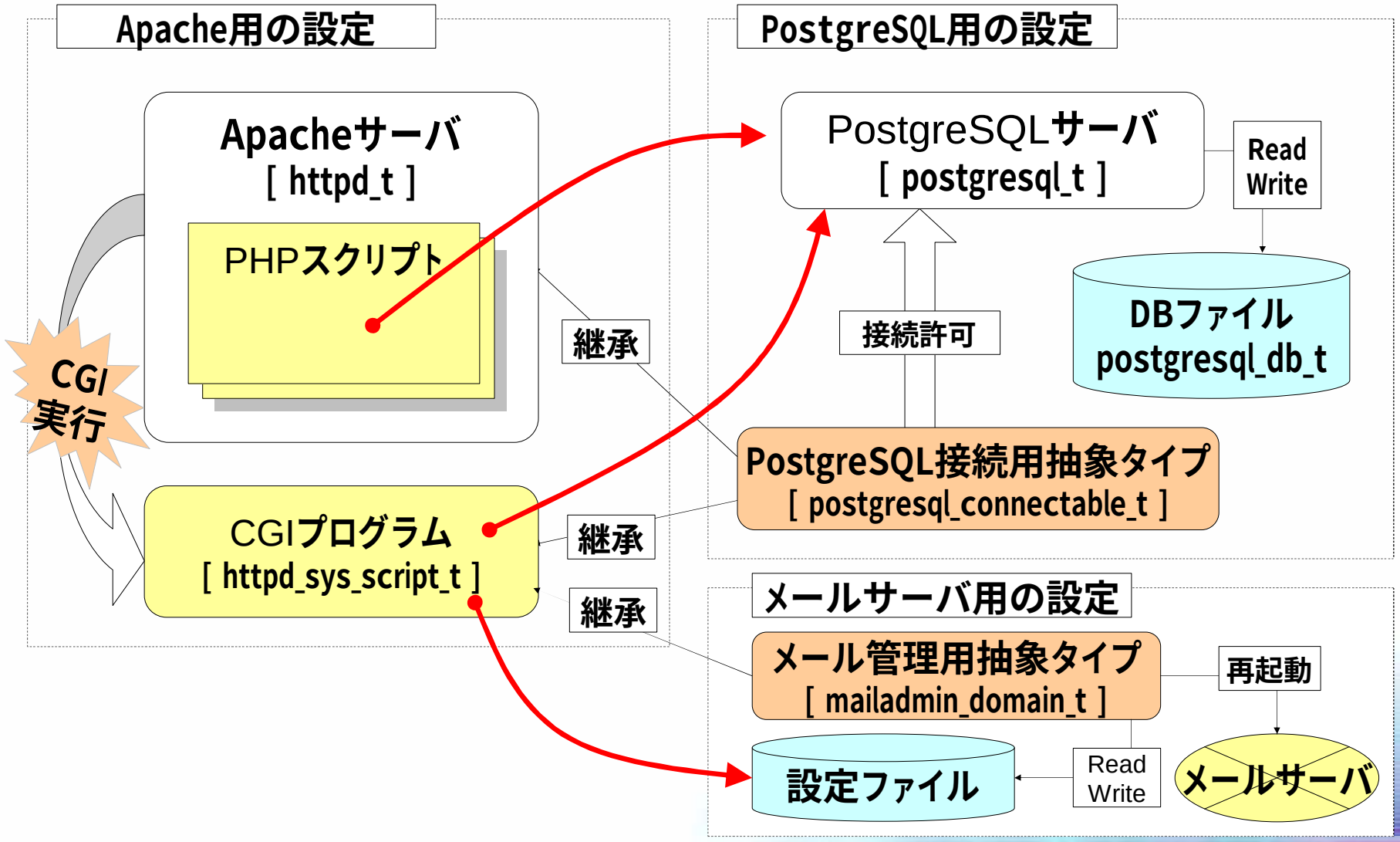
- FTP/Apache/Sambaが共通のファイルにアクセス

	Apache	FTP	Samba
/var/www/html/	Read	Read/Write	×
/var/www/cgi-bin/	Read Execute	Read/Write	×
/var/log/httpd/	Append	Read	×
/var/log/samba/	×	Read	×
/home/foo/	×	Read/Write	Read/Write
/home/foo/public_html	Read	Read/Write	Read/Write

継承を使ったFTP/Samba/Apacheの例



継承を使ったPHP/CGIとPostgreSQLの例



- 細かすぎる権限を集約して付与することが可能になる。
- 各アプリケーションのポリシー記述の独立性が向上。

継承を利用したポリシーの記述(1)

- 継承"される"側に必要な設定
  - インターフェースとなるタイプの定義
  - アプリケーション～インターフェース間のアクセス権の定義
- FTP/Sambaの合成タイプの例
  - FTPとSambaから共通にアクセスしたいファイルタイプの定義
  - FTPのみ/Sambaのみのファイルタイプの定義を利用する

```
type samba_share_t;      # Samba共用ファイル用タイプ
type ftpd_file_ro_t;     # FTP読み取り専用タイプ
type ftpd_file_rw_t;     # FTP読み書き可能タイプ

allow smbd_t @samba_share_t : file create_file_perms;
allow smbd_t @samba_share_t : dir  create_file_perms;
allow ftpd_t @ftpd_file_rw_t : file create_file_perms;
allow ftpd_t @ftpd_file_rw_t : dir  create_dir_perms;
allow ftpd_t @ftpd_file_ro_t : file r_file_perms;
allow ftpd_t @ftpd_file_ro_t : dir  r_dir_perms;
```

継承を利用したポリシーの記述(2)

- 継承"する側"に必要な設定
  - インターフェースとして提供されたタイプを継承する
- Samba/FTPの合成タイプの例

【継承を使用した場合】

```
type samba_ftpd_file_ro_t extends
  samba_share_t, ftpd_file_ro_t ;
type samba_ftpd_file_rw_t extends
  samba_share_t, ftpd_file_rw_t ;
```

↘ Sambaが読み書き可能で、FTPはRead-Only又はRead/Write可能

✓ 同様の記述を繰り返すことに。
✓ FTPの設定なのか、Sambaの設定なのか不明確

【継承を使用しない場合】

```
allow smbd_t samba_ftpd_file_ro_t : file create_file_perms;
allow ftpd_t samba_ftpd_file_ro_t : file r_file_perms;
allow smbd_t samba_ftpd_file_rw_t : file create_file_perms;
allow ftpd_t samba_ftpd_file_rw_t : file create_file_perms;
                    :
```

継承を利用したポリシーの記述(3)

- サーバアプリケーションに接続可能なタイプの定義
  - ↘ PostgreSQLへ接続可能なインターフェース･タイプを定義
- PostgreSQL側の設定

```
type postgresql_connectable_t;
allow @postgresql_connectable_t postgresql_tmp_t
   : sock_file rw_file_perms;
allow @postgresql_connectable_t postgresql_t
   : unix_stream_socket connectto;
```

- Apache側の設定

```
typeextends httpd_t extends
   postgresql_connectable_t;
```

中間にインターフェースとなるタイプを挟むことで、PostgreSQLの設定と、Apacheの設定を分離できる。

継承を利用したポリシーの記述(3)

- サーバアプリケーションに接続可能なタイプの定義
  - ↘ PostgreSQLへ接続可能なインターフェース･タイプを定義
- PostgreSQL側の設定

```
ifdef(`apache.te', `
allow httpd_t postgresql_tmp_t
  : sock_file rw_file_perms;
allow httpd_t postgresql_t
  : unix_stream_socket connectto;
')
```

- Apache側の設定

~~typeextends httpd_t extends~~
~~postgresql_connectable_t;~~

中間にインターフェースとなるタイプを挟むことで、PostgreSQLの設定と、Apacheの設定を分離できる。

まとめ

- タイプ継承によって可能になったこと
  - 複数アプリケーションが共有するリソースへの権限付与
  - インターフェース･タイプを継承することで、個別アプリケーション毎の設定の独立性を高くできる

  ↘ 既存の構文より少ない記述で一貫性のある設定が可能に
- これからの課題
  - “継承される”ことを意識したセキュリティポリシーの記述

    ☹　今までの蓄積もあるので大変。
- SELinuxはまだまだ始まったばかりの技術
  - プログラミング言語で言えば、アセンブリ言語時代

  ↘　高級言語“的”なものが求められている。
  - タイプ継承も、これらのアプローチの一つ

ご静聴ありがとうございました。

ＮＥＣ Linux推進センター

海外 浩平

(kaigai@ak.jp.nec.com)