Network Security Forum 2003

「プロセス実行履歴に基づく アクセスポリシー自動生成システム」

平成１５年１０月２２日
株式会社ＮＴＴデータ
技術開発本部
原田季栄 haradats@nttdata.co.jp

Linuxのアクセス制御

- ファイルアクセス制御
  - 所有者は制御内容を変更できる。
  - 制御内容が貧弱。
    - 所有者/グループ/その他の分類、グループは1つのみ
      (POSIX ACL導入により改善される予定)
- root (システム管理者) は制限を受けない
  - root権限を奪取されると歯止めが効かない。
    - データやログの削除・改ざん、バックドアの設置・・・

root権限奪取阻止は可能か？

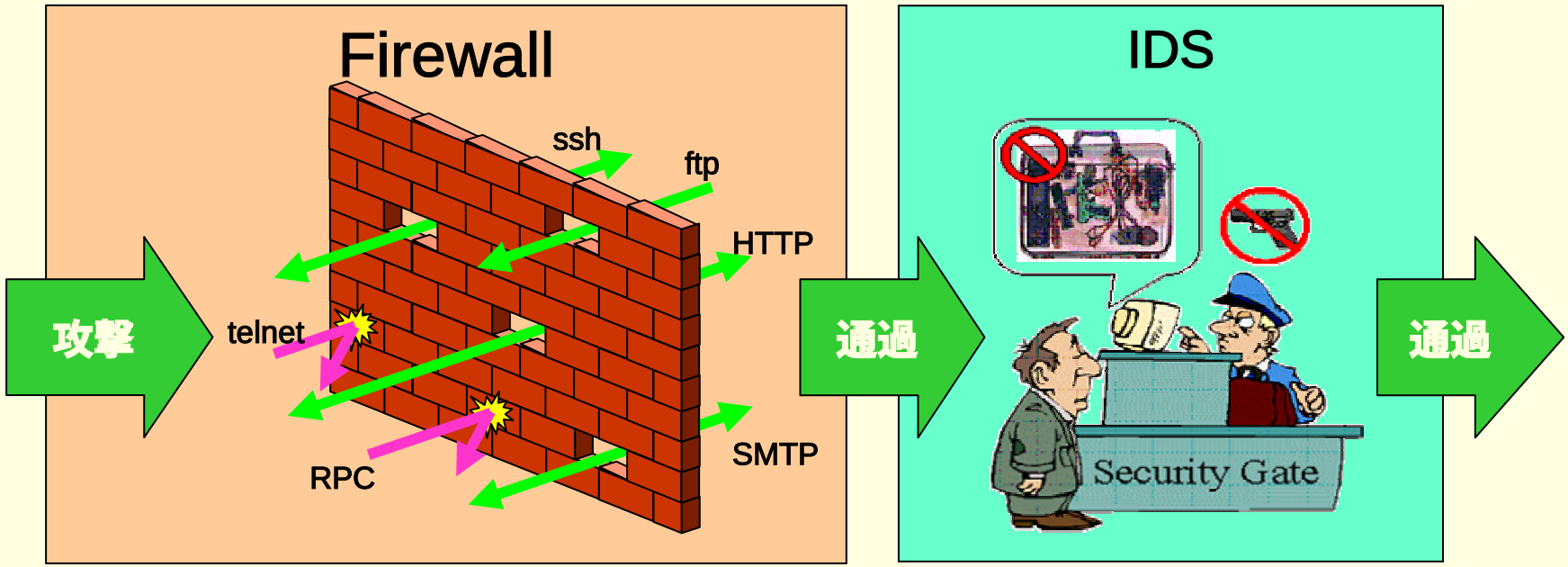
- Linuxではroot権限の奪取を完全に抑止することは不可能
  - システムはroot権限やroot権限でプログラムを動作させる仕組みなしには機能しない。
  - root権限で動作するプログラムの脆弱性はroot権限の奪取に直結する。
  - つまり・・・、

Linuxはセキュアではない

- バッファオーバーフローとroot権限で動作するプログラムが存在する限りは、root権限の奪取が発生しうる。
  - バッファオーバーフロー
    - 改善策はあっても決定的解決策はない
  - root権限で動作するプログラム
    - なくすとOSが動作しない

ファイアウォールとIDS

- ファイアウォール・・・不必要な通信の遮断
- IDS・・・既知の攻撃手法との照合による「検知」



ファイアウォールとIDS

- 実は効果は非常に限定されている
  - ファイアウォールは提供していないサービスを遮断することはできても、提供しているサービスを閉じることはできない。
  - IDSやOSへのパッチは既知の脆弱性や不具合に対するものに限定される。
- ファイアウォールやIDSで救済されるのはごく限られた場合だけ。(無駄ではないがそれだけで安心してはいけない)

ではどうするか？

- 「セキュリティ強化OS」の考え方
  - 最後の砦であるOSの強化が不可欠
  - root権限を奪われても完全な支配を許さない
- 「セキュリティ強化OS」の手法
  - アクセス制御の強化
    - 自由裁量ではなくポリシーに基づき、rootも例外扱いされない強制的な制御＝「強制アクセス制御」
    - root権限の細分化 (least privilege)
    - 通信の制限やプロセスおよびデータの隔離

「強制アクセス制御」について

- ポリシーに基づきアクセス可否を判断する。
- 網羅的な制御を「強制」することにより、不適、不要なアクセスを拒否する。
- TCSEC “Labeled Security”
  - ファイル等の資源にラベルをつけて、ラベルに基づいてアクセス制限を規定する。
- Linuxにおいては、カーネル内部でシステムコールをフックする形で実装される。

強制アクセス制御の実装

- 実装の例
  - ＯＳ
    - Flask
    - RSBAC
    - SubDomain
    - LIDS
    - SELinux
  - フレームワーク
    - LSM（Linux Security Modules）
- ポリシーの構文は実装系依存

ポリシーの例

SubDomain

```
foo {
/etc/readme r,
/etc/writeme w,
/usr/bin/var x {
  /usr/lib/otherread r,
  /var/opt/otherwrite w,
  },
}
```

“SubDomain: Parsimonious Server Security”
※parsimonious:倹約した、きりつめた

SELinux

1. ユーザの役割
2. ファイルへのラベル
3. ドメイン（プログラム遷移）

に基づくポリシー（とてもここでは書き切れません・・・）

ポリシー定義は困難である

- 「適切なポリシー」とは？
  - 必要なアクセスのみを認め、不必要なアクセスを拒否するポリシー。
- 「適切なポリシー」を作るための条件
  - 「必要なアクセス」を把握していること。
  - どの粒度で？＝Linuxではシステムコールレベル。

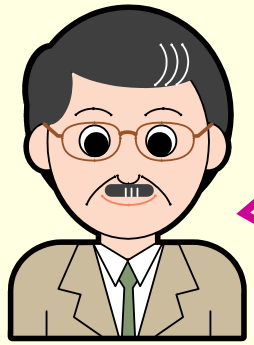
ポリシー定義は困難である

- ポリシー定義を難しくする要因
  - シンボリックリンク、ハードリンク
  - ダイナミックリンクライブラリ
  - プログラム（モジュール）の依存関係
    - 明示的でないプログラムの起動

適切なポリシーを定義する方法

1. ＯＳとサービスの動きを掌握する
   - 現実には困難。
   - 仮にできたとしてもプログラムの設定やシステム構成の変更等により変化する。
   - そこで・・・
2. 考えるのではなく、「記録（追跡）」する
   - どうやって？
   - アクセス許可を「判断」する箇所で「記録」する
     - システムの構成に関わらず最終的に発生したアクセス要求をシステムコール内で捕捉する。

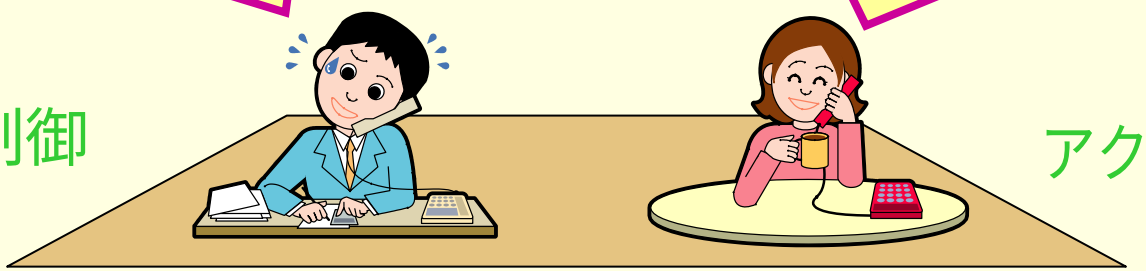
アクセス制御とアクセス記録



passwordですが
／etc／password
をwriteアクセスさせて
ください。

えーと、今ポリシーを確認
しているのでしばらくお待
ちください。

／etc／password
をwriteアクセスですね。
記録しましたのでどうぞ♪



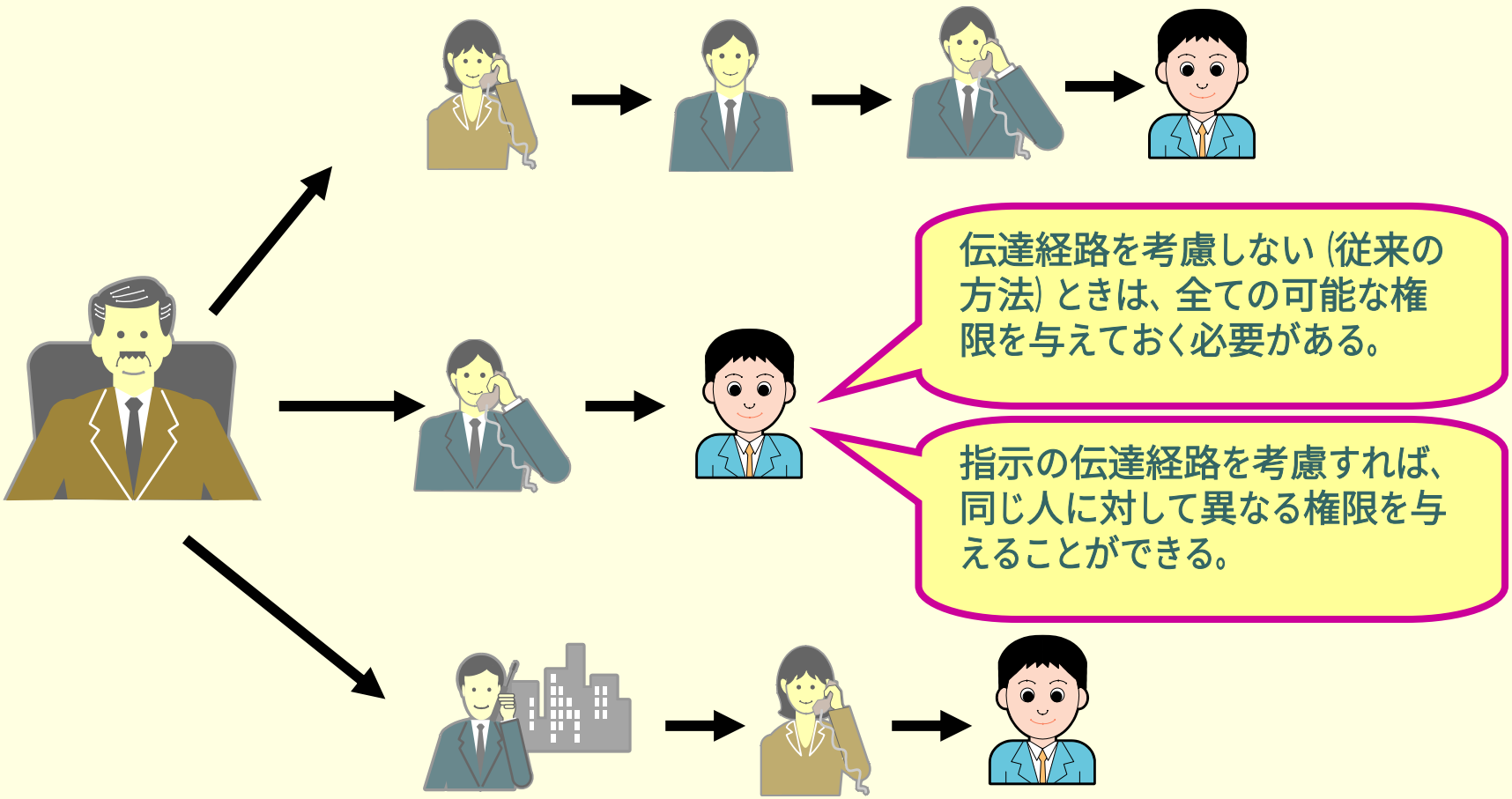
アクセス制御

アクセス記録

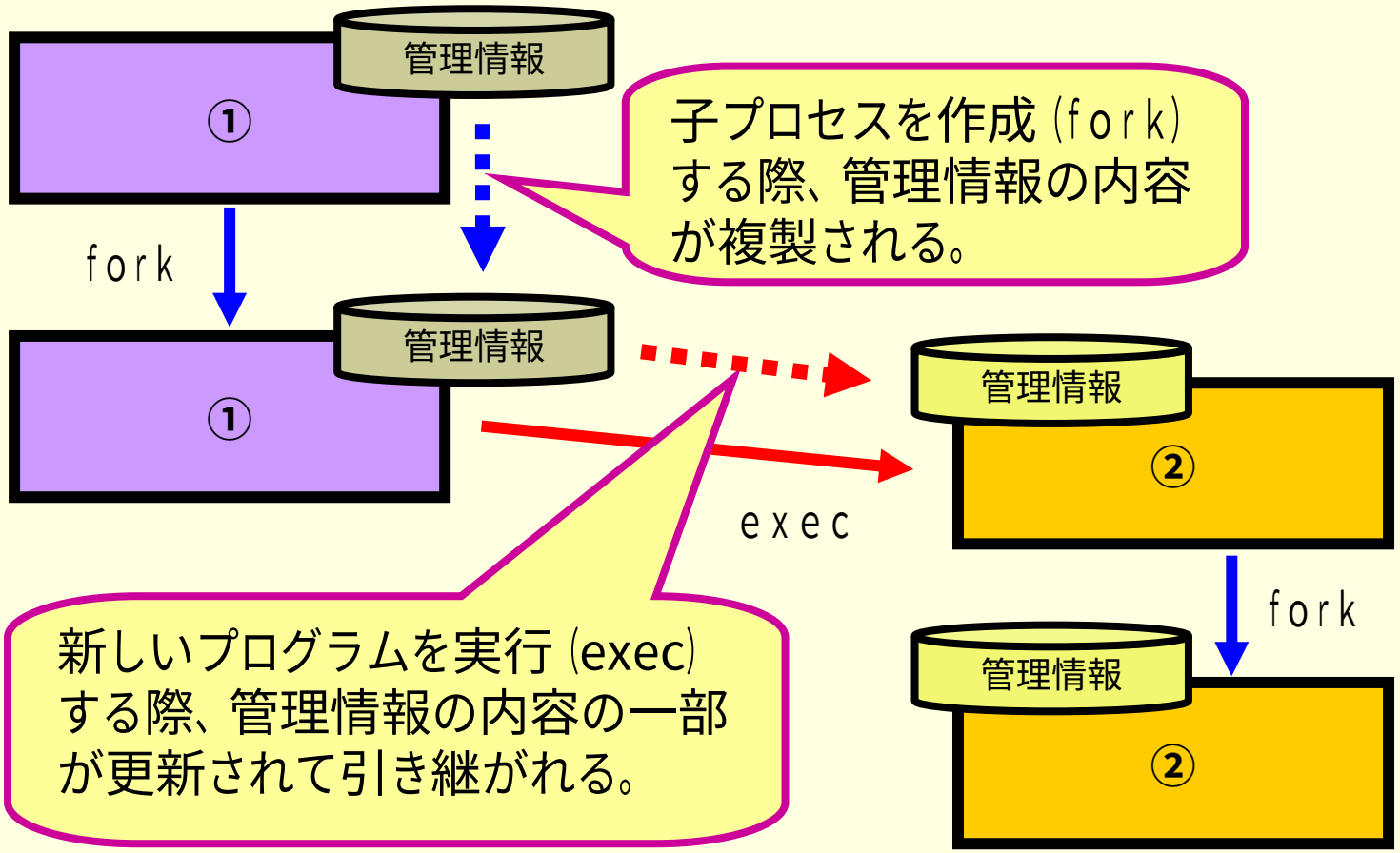
「プロセス履歴」の考え方

- 現在実行中のプログラムの情報だけでアクセス権限を規定すると・・・
  - 状況に応じてアクセス対象が異なる場合、アクセス対象の和集合に対して権限を付与＝本来アクセスが必要でない状況でもアクセスが可能となる。
- 現在実行中のプログラムの情報だけでなく、そのプログラムが起動されるまでの情報も加味し、与えるアクセス権限を規定すると・・・
  - そのときに必要な最小限の権限を付与できる。

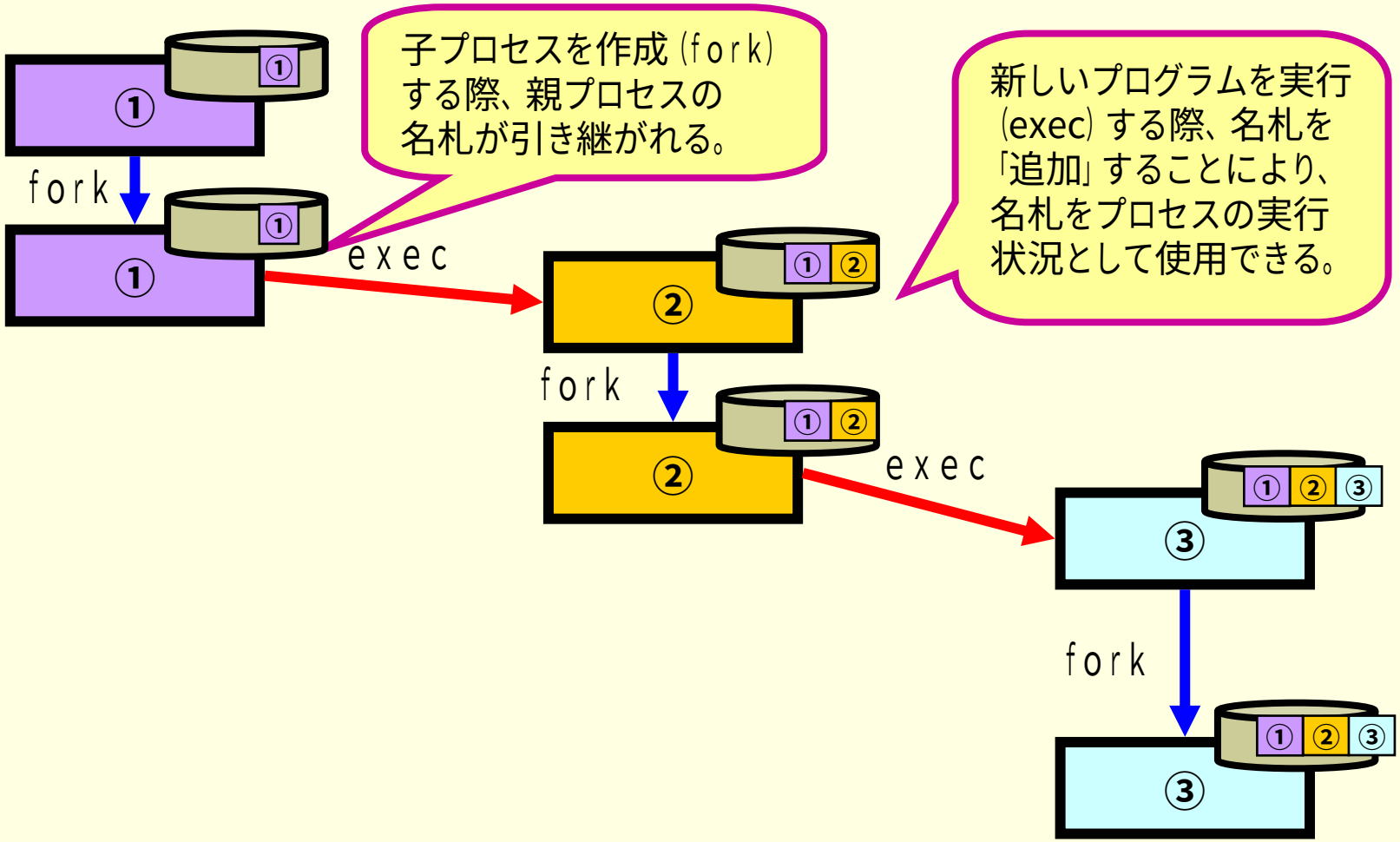
「プロセス履歴」のイメージ



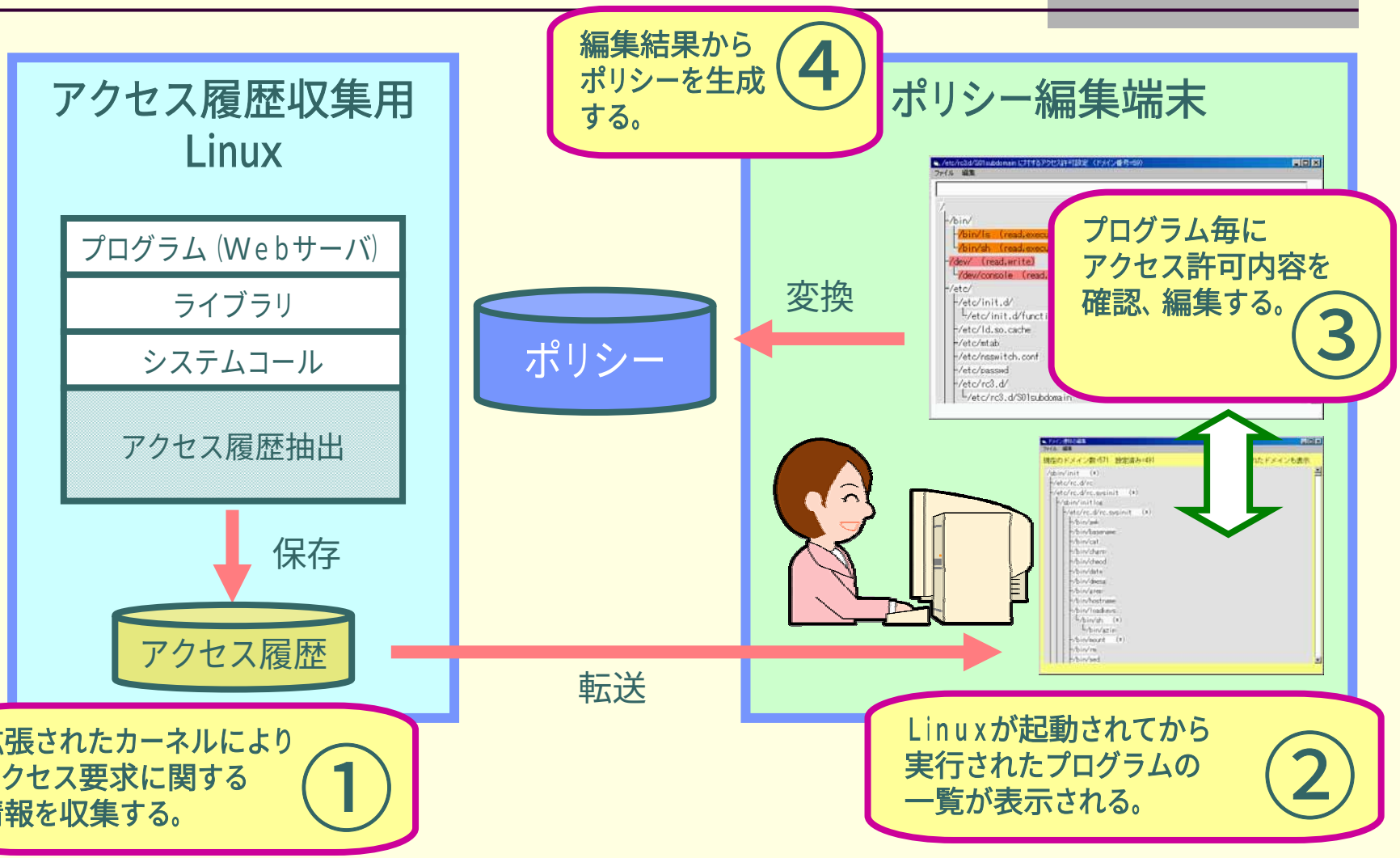
Linuxにおけるプロセス起動



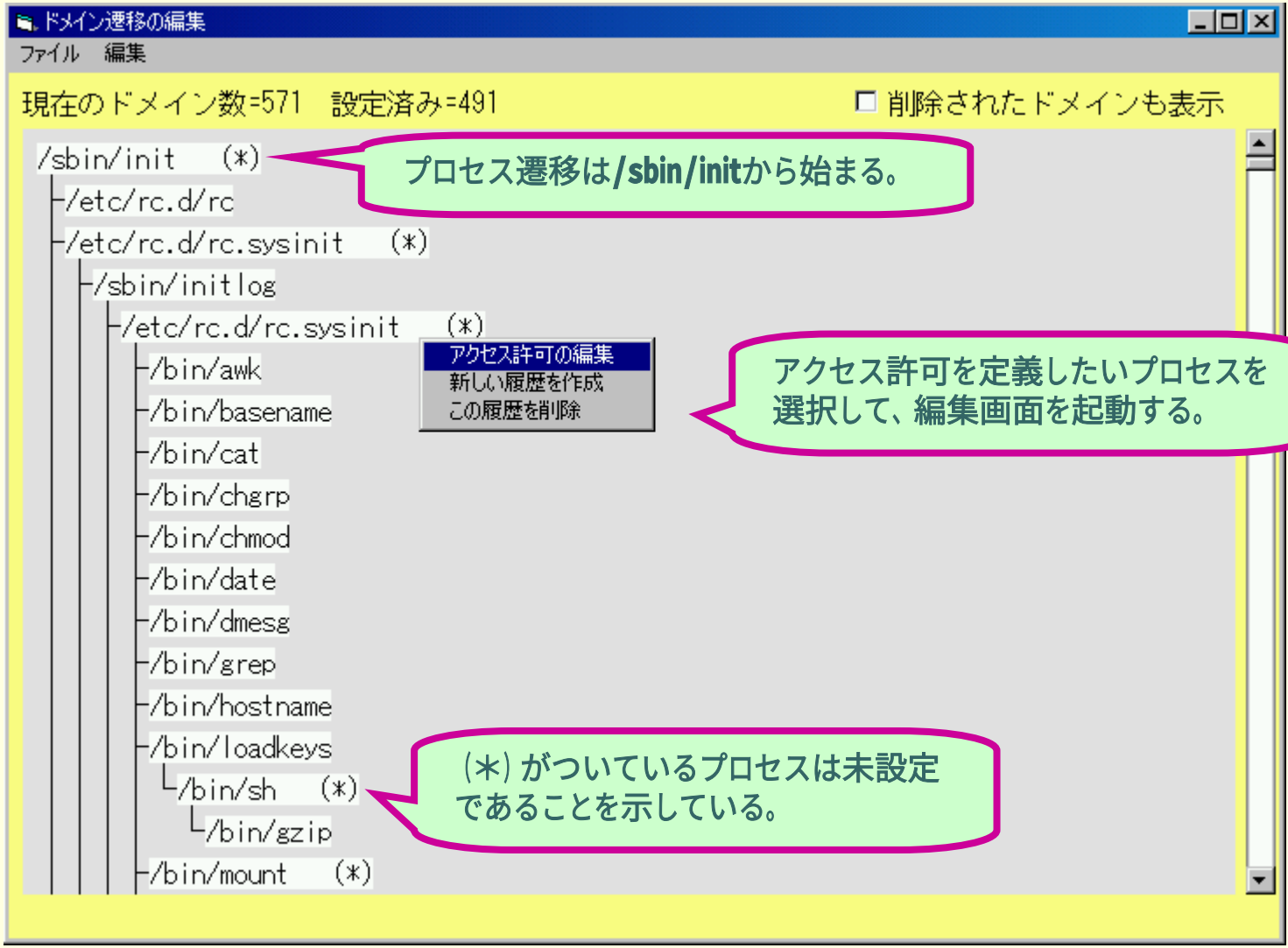
プロセス遷移の追跡方法



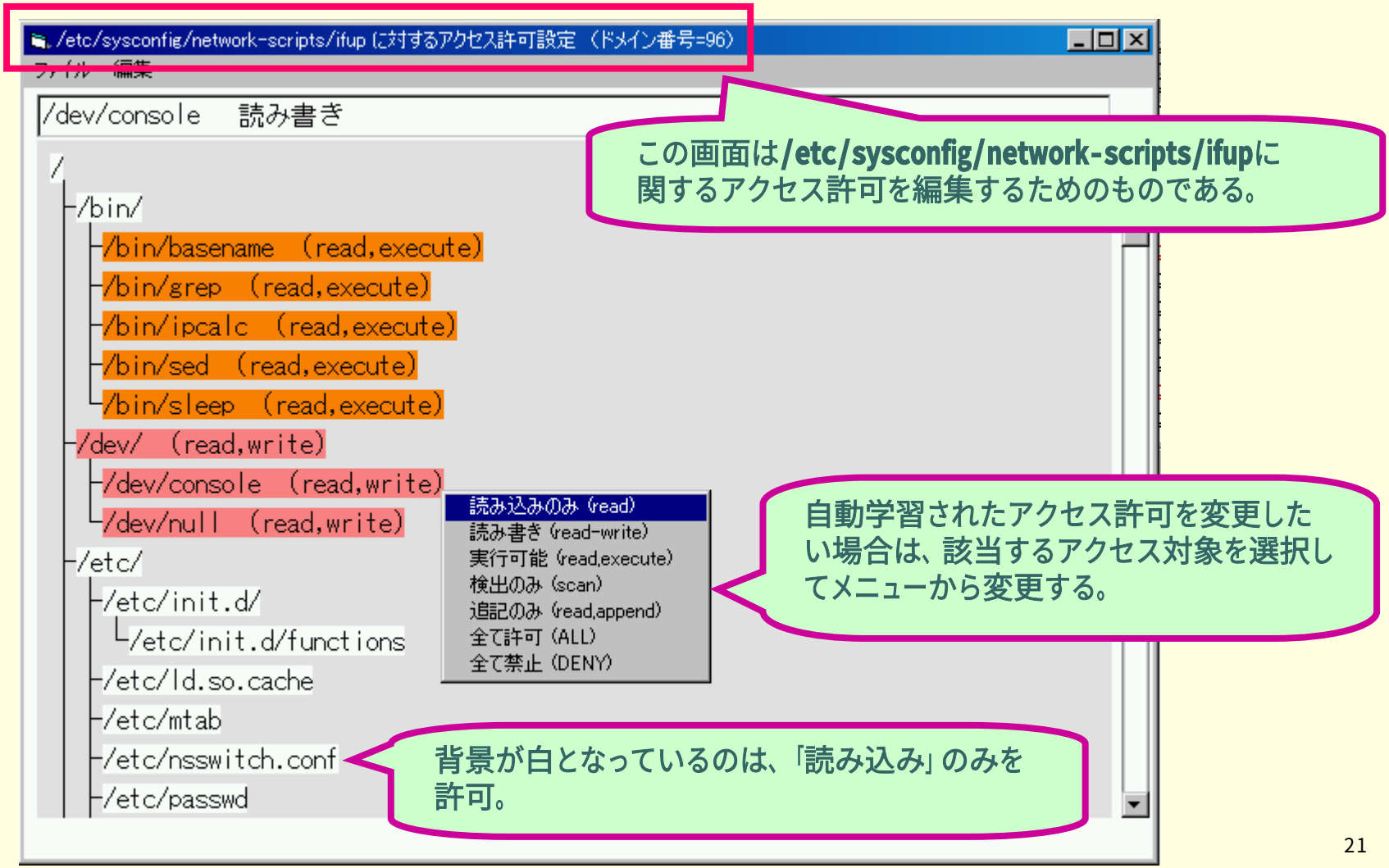
システム構成



プロセス遷移の編集画面



プロセス毎のアクセス許可設定



生成したポリシーの例

```
# /sbin/init
rx    /sbin/init
{
    r     / ,
    r     /dev/ ,
    rw    /dev/console ,
    rw    /dev/initctl ,
    rw    /dev/tty0 ,
    r     /etc/ ,
    r     /etc/inittab ,
    r     /etc/ioctl.save ,
    r     /etc/ld.so.cache ,
    r     /etc/rc.d/ ,
    # /sbin/init→/etc/rc.d/rc
    rx    /etc/rc.d/rc
    {
        r     / ,
        r     /bin/ ,
        # /sbin/init→/etc/rc.d/rc→/bin/egrep
        rx    /bin/egrep
        {
            r     / ,
```

まとめ

- 強制アクセス制御の適用はセキュリティ強化策として有効だが、「適切なポリシー」の定義が課題となる。
- Linuxにおけるプロセス生成の仕組みを利用し、(1) プロセスの実行状況に応じたアクセス要求履歴を抽出し、(2) それをもとに不要なアクセス許可を含まないファイルアクセスポリシーの定義方法を実現した。
- 本手法によれば、もっとも細かな (厳密な) ポリシーを定義することが可能である。