SELinuxのポリシ作成時間を短縮する一考察

Linux Conference 2007
9月13日

桝本　圭　　株式会社NTTデータ
masumotok@nttdata.co.jp
村田　裕之　日本データコム株式会社
murata@ndc.ne.jp

目次

- はじめに
- SELinuxの課題
- 作成方法の変更
- プロトタイプについて
- 効果測定
- おわりに

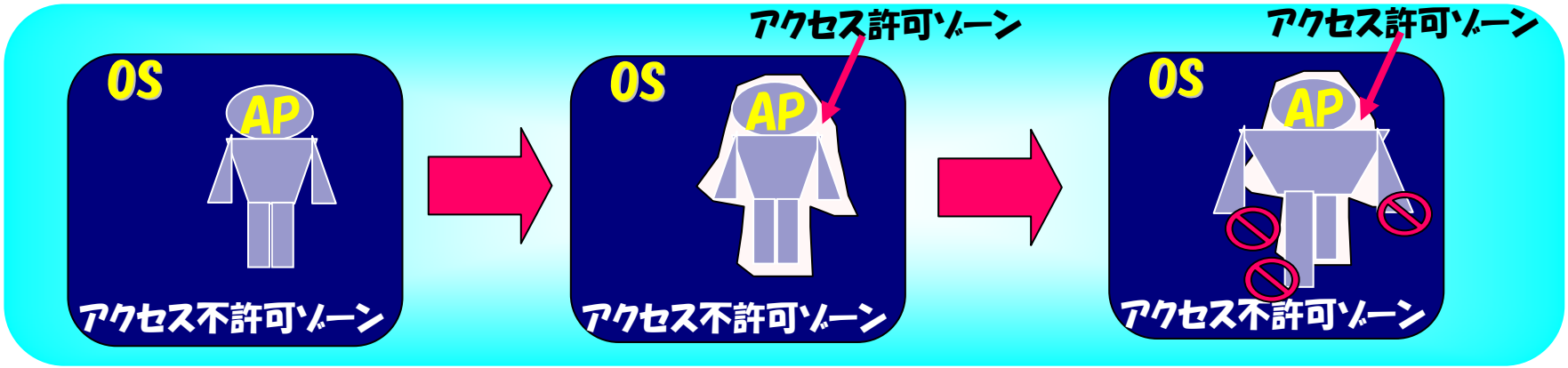
はじめに

- SELinuxについて
  - 強制アクセス制御
  - 最少特権
- 導入のメリット
  - 攻撃されたときの被害の抑止効果
  - ユーザの誤操作防止

SELinuxの課題

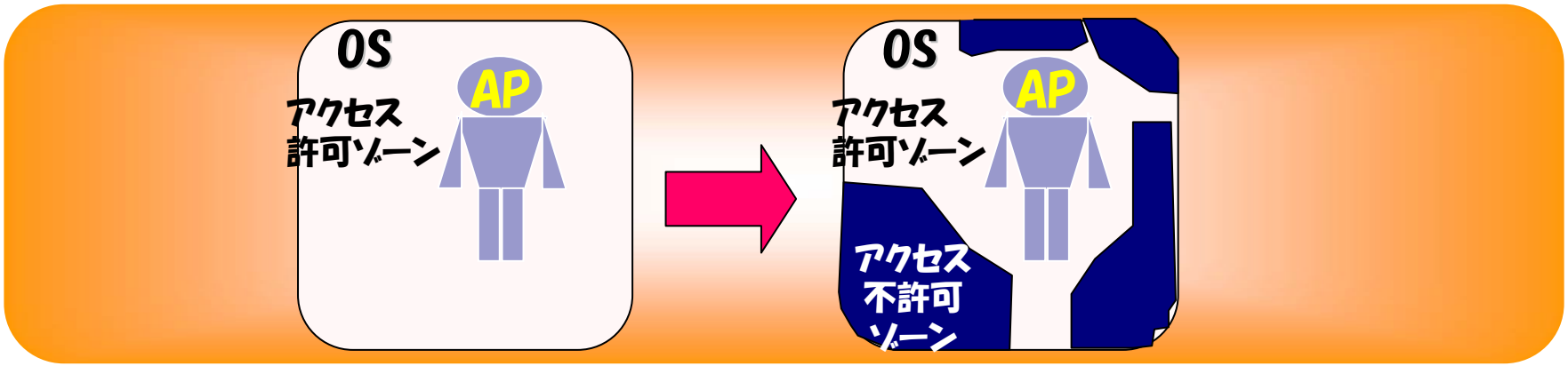
- 動作するまでに時間がかかること
  - 慣れてない技術者にとっては
    - SELinuxの独自言語の習得
    - 設定内容の理解
  - 慣れた(高いレベルを持つ)技術者にとっても
    - …

特に時間のかかるポイント



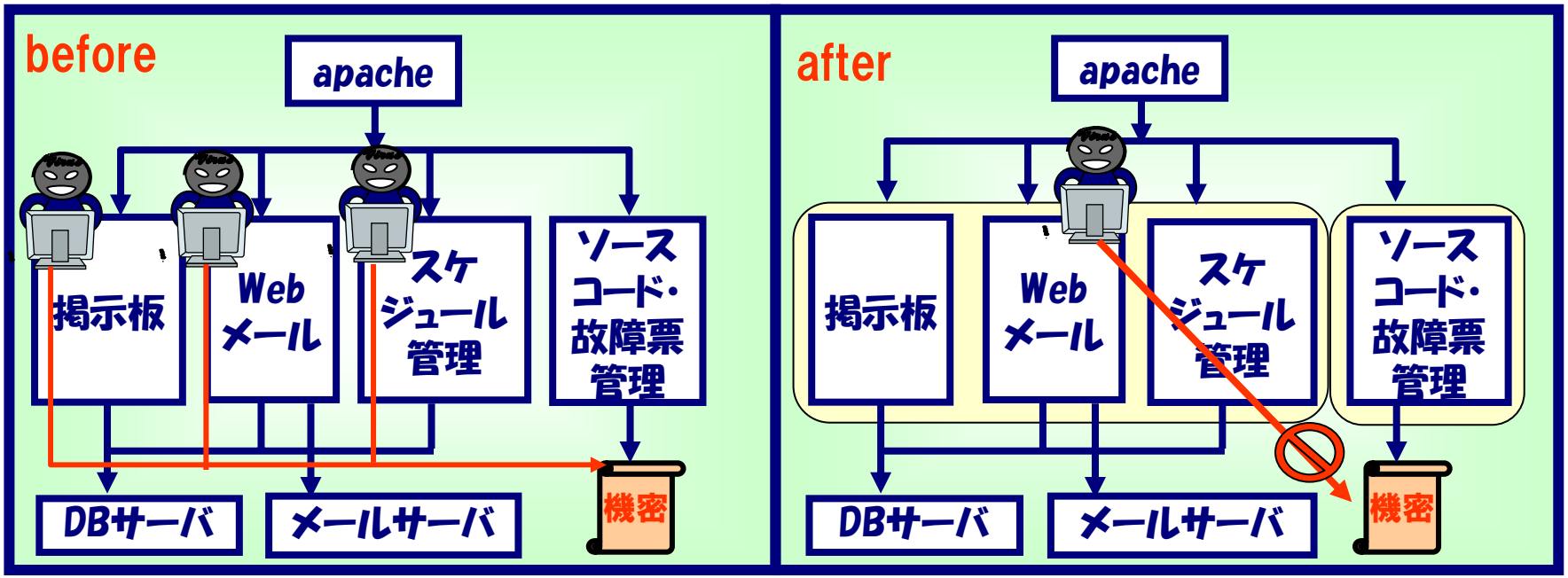
- ポリシ作成の基本はホワイトリスト
  - アプリケーションをトレースし、それを元にポリシを作成
  - 「トレースしたデータは十分か？」が課題
    - 不十分＝動作妨害 → 不安 → さらに時間をかけてデータ収集
  - この作業は導入時だけではない
    - 構成変更、拡張モジュール追加、バージョンアップ時も同じ

作成方法の変更



- ブラックリストも状況に応じて使用していいのでは？
  - アプリケーションの動きをトレースする必要がなくなる
    - 「守りたいこと」ベースで制限をかけることができるため
  - 不必要に厳しい制限をかけなくてもよくなる
    - アプリケーションの動作が妨害される可能性を下げるため

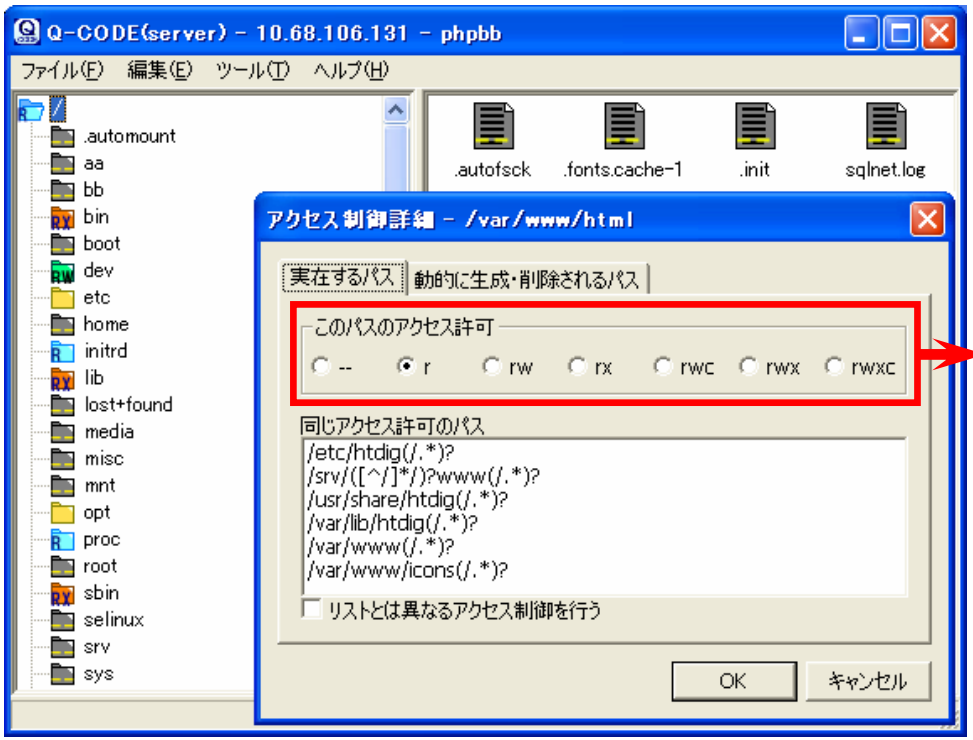
事例



- 要求条件:『不正アクセスがあっても、機密情報が漏洩しない対策を！』
  - システムの全動作をトレースして、ソースコード管理機能以外はアクセスできないようにした
- ホワイトリストによる作成は、時間もかかり、トラブルも発生
  - 要件を満たす設定をするだけなら、時間もかからずトラブルも少く安全なシステムを提供できた

プロトタイプ　～特徴１～

- 独自言語の習得期間を削減する
  - 独自言語でなく、エクスプローラ形式で設定
  - 設定漏れや設定内容を容易に確認

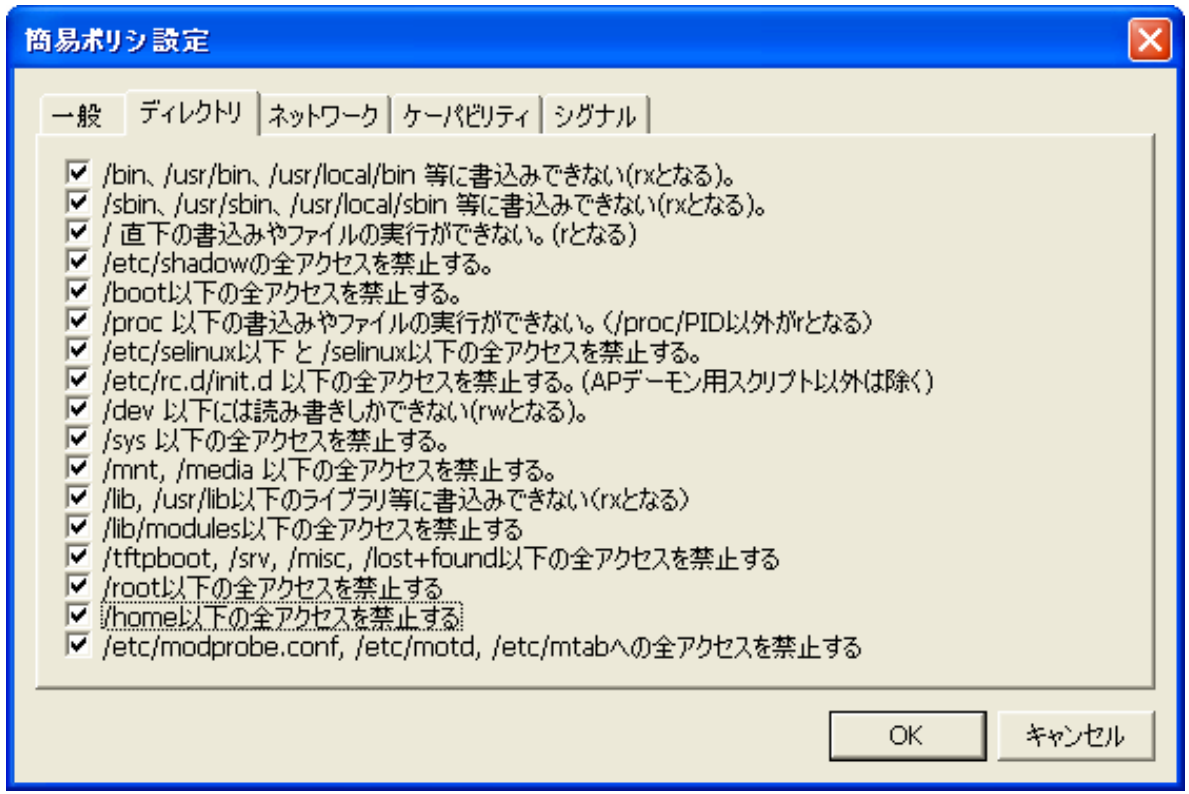


7種類のアクセス制御

- 全て不許可
- 読込のみ
- 読み書きのみ
- 読み・実行のみ
- 読み・実行・作成削除
- 読み・書き・実行
- 全て許可

プロトタイプ　～特徴２～

- より短期間で作成できるように
  - アプリケーション共通の設定項目をテンプレートとして用意



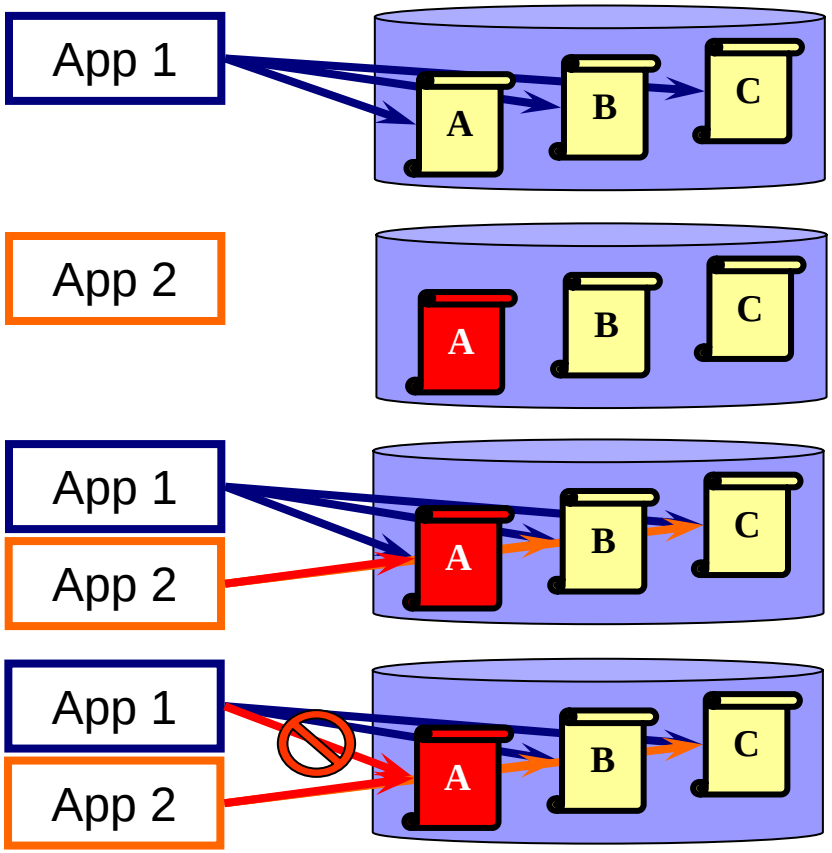
プロトタイプ　～特徴３～

- GUIだけでなくポリシも
  - SELinuxのオブジェクトクラスは複雑で使い分けることが困難
  - まとめられるオブジェクトクラスをグループ化

No.	項目	オブジェクトクラス代表例	制限の種類
1	FS上に存在するリソース	file, dir, fifo_file, blk_file	r,w,x,c 4種類の組み合わせ
2	ケーパビリティ	capability	指定したケーパビリティの設定ができないようにする
3	ネットワークポート	tcp_socket, udp_socket	あるポート番号にbindできる/できない
4	Inetドメインソケットを用いたNW通信	tcp_socket, udp_socket	ソケットを用いた通信ができる/できない
5	シグナル	signal	SIGCHLD, SIGKILL, SIGSTOP, SIGNULL、それ以外の4種類のシグナルが送れる/送れない
6	その他クラス	security, raw_socket, passwd, isem, msg, msgq, shm, ipc	制御しない

プロトタイプ　～特徴４～

- アプリケーション間のアクセス許可の調整を実現



① APP1は、A,B,Cに対してアクセスを許可する

② Aは、App2 固有のファイルのため、B,Cとは異なるグループに割り当てる

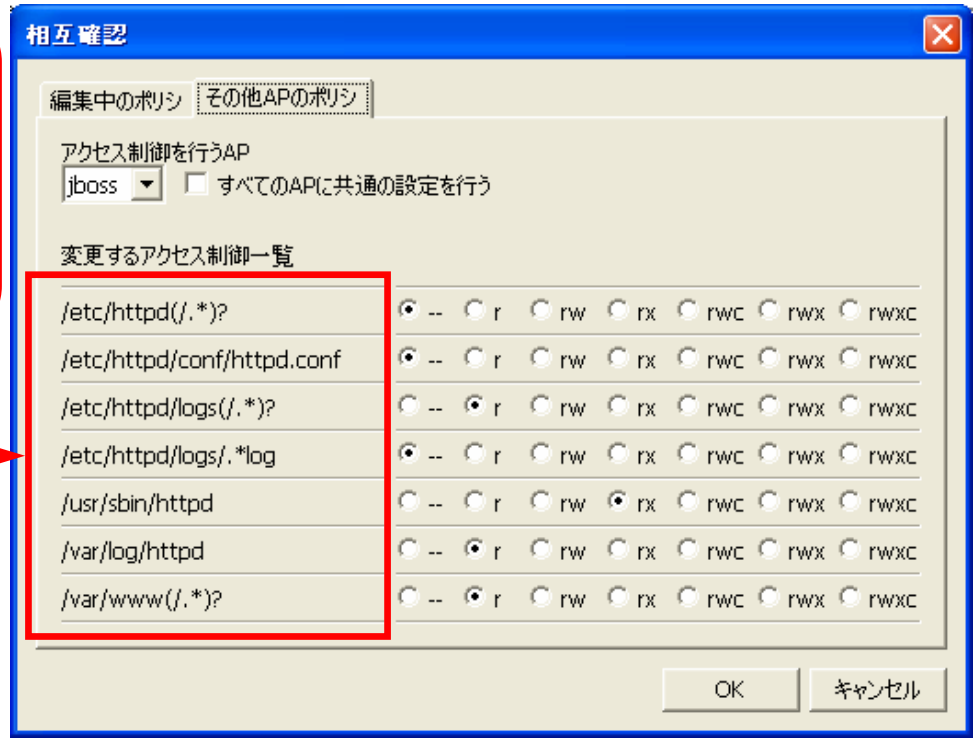
③ 割当てただけでは、App1は A にアクセスできてしまう

④ App1からAにアクセスできないようにする必要がある

プロトタイプ　～特徴４～

- アプリケーション間のアクセス許可の調整を実現

新規に定義したグループのパスを覚えておき、他のアプリーケーションに対するアクセス許可を簡単に設定できる

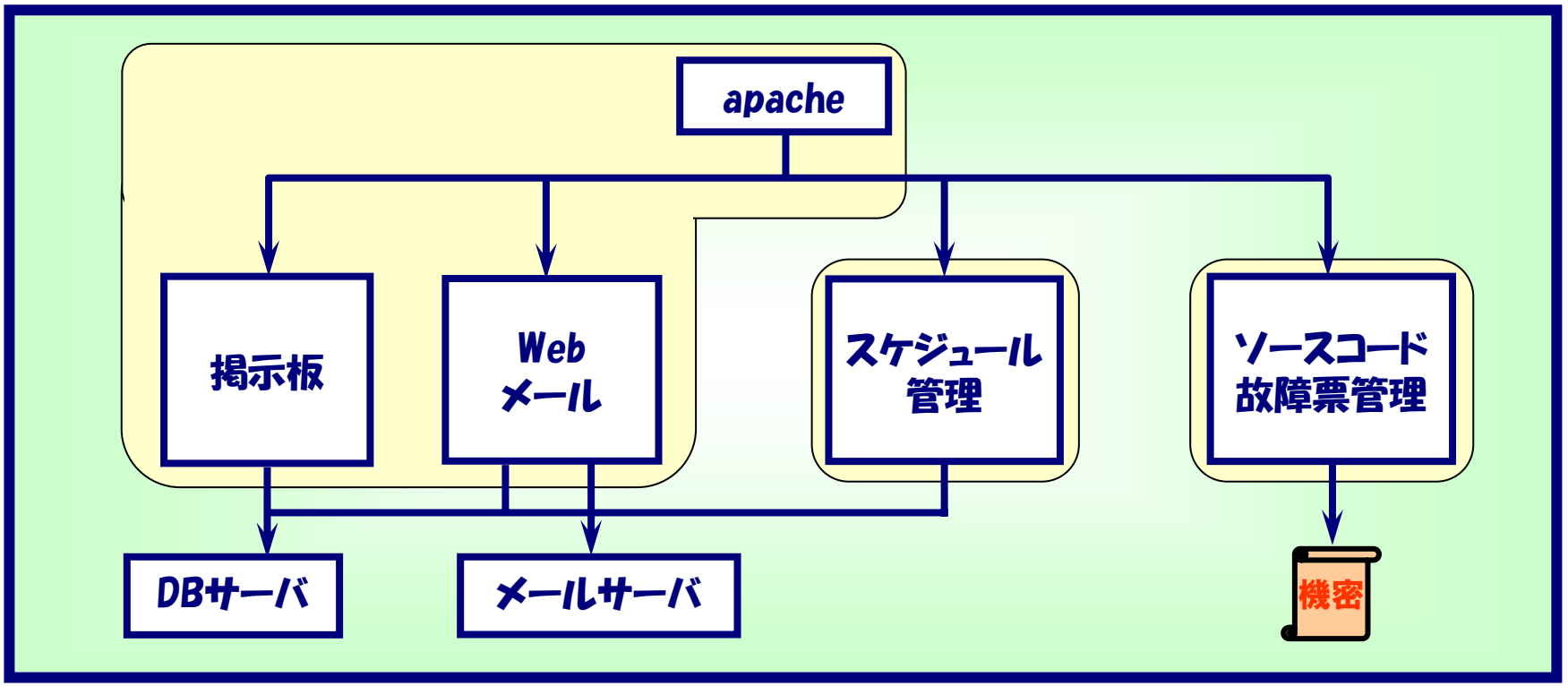


効果測定

- 測定方法
  - 以前にSELinuxを導入した実システムを対象システムとする
  - 共通の要件を元に、テスターを使って従来方式とプロトタイプでポリシを作成する
  - それぞれの時間を比較・検討する

効果測定

- 測定対象システム

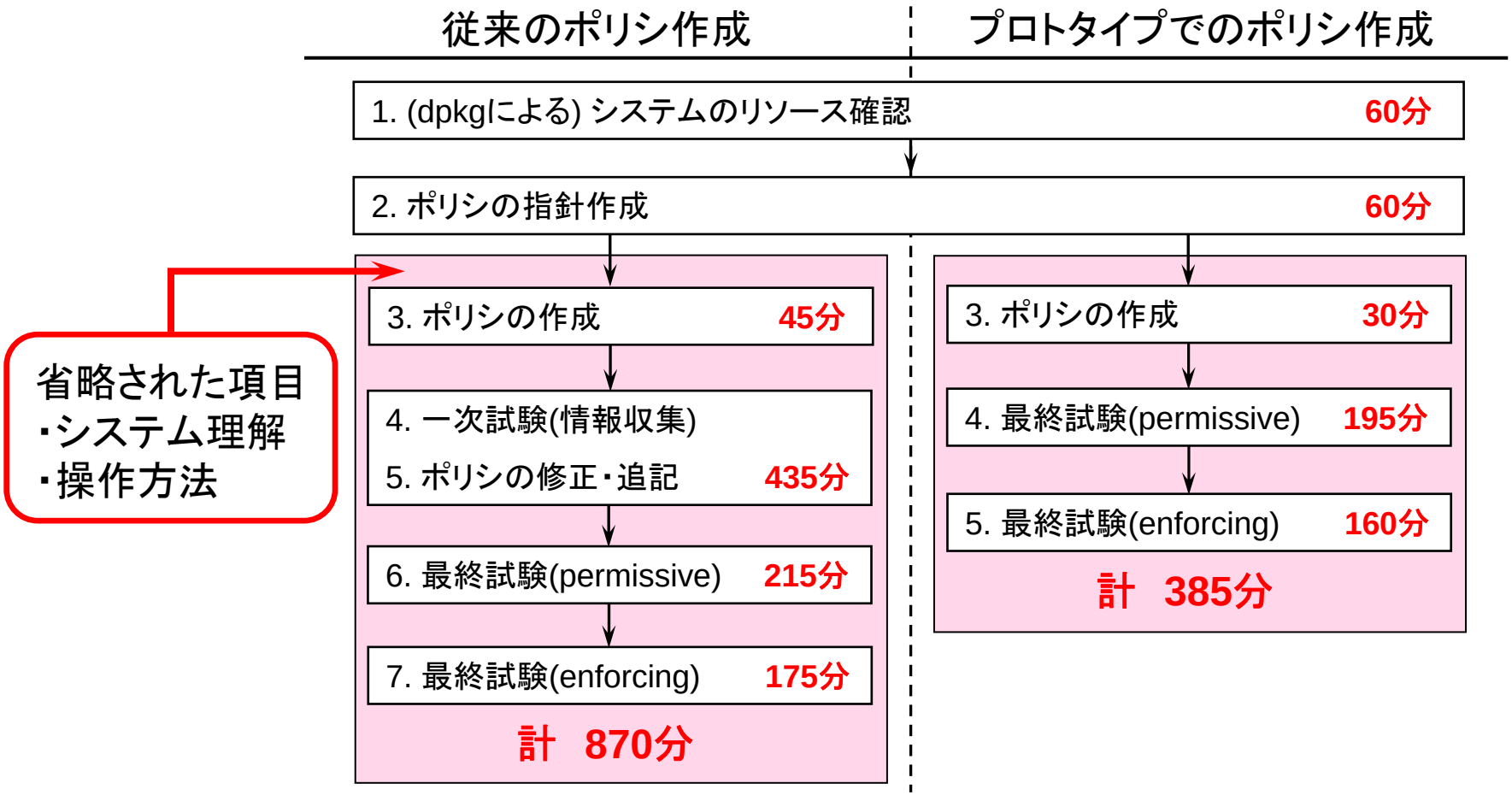


効果測定

- ポリシ作成を行う上での要件定義

No.	要件
1	機密情報には、機密管理機能以外からアクセスできない
2	各CGI（*.php等）やカーネル、/bin, /usr/bin, /sbin, /usr/sbin にあるバイナリの改ざんができない
3	スケジュール管理機能、機密管理機能からはWebメールが保持するデータにアクセスできない
4	ユーザホームディレクトリ（/home）以下のユーザデータにはアクセスできない
5	/var/log, /tmp等の共有ディレクトリについて、本システム以外のプログラムが作成したリソースにアクセスできない
6	他のプログラムに対してシグナルを送れない
7	カーネルモジュールを ロード / アンロード できない
8	SELinuxのアクセスポリシの変更ができない
9	/dev 以下のデバイスファイルの改ざんができない
10	デーモンのブートスクリプト(/etc/init.d 以下) にはアクセスできない

測定結果


従来のポリシ作成
プロトタイプでのポリシ作成
1. (dpkgによる) システムのリソース確認 60分
2. ポリシの指針作成 60分
3. ポリシの作成 45分
4. 一次試験(情報収集)
5. ポリシの修正・追記 435分
6. 最終試験(permissive) 215分
7. 最終試験(enforcing) 175分
計 870分
3. ポリシの作成 30分
4. 最終試験(permissive) 195分
5. 最終試験(enforcing) 160分
計 385分
省略された項目
・システム理解
・操作方法

考察

- ポリシ作成に関する利点
  - 前提知識の不要と作業量の減少
  - ポリシ作成時間の短縮
- 効果測定に関して
  - SELinuxに精通した測定者での実施
  - 不慣れな測定者の場合、時間差はより増大

おわりに

- プロトタイプを用いたポリシ作成の事例を増やし、要件にあわせた効果を検討