Linux カーネルの動的アクセスポリシー制御

保理江 高志† 原田 季栄 田中 一男

（株）NTT データ 技術開発本部

† E-mail : horietk@nttdata.co.jp

あらまし

筆者らは、オペレーティングシステム（以下、OS）のアクセス制御機能における「セキュリティチェック」と呼ぶ機能拡張の検討を行っている。これはパーミッションチェックと別個にアクセス内容の危険性についてチェックを行い、その判定結果に基づいて動的にアクセスポリシーを制御し、システム全体の防御を図ると言う機構である。本稿は、米NSAの開発したSecurity-Enhanced Linux（以下、SELinux）をベースとした本機能拡張の実装について報告するものである。また、実用上の諸問題の検討、及び対策として実施した「セキュリティレベル制御のフレキシビリティ」に関する2項目の拡張についても併せて報告する。

Dynamic Access Policy Control for Linux Kernel

Takashi HORIE† Toshiharu HARADA Kazuo TANAKA

Research and Development Headquarters, NTT DATA CORPORATION

† E-mail : horietk@nttdata.co.jp

Abstract

The authors had proposed "security check" functionality, which is the extension for the access control of the OS (operating system). It has two unique features. One is the risk judgment for the access context separately from permission check and the other is system protection by dynamic access policy control based on the judgment results. This paper describes the implementations for this extension based on NSA's Security-Enhanced Linux (SELinux). Some issues about it for practical use and effective measures are also discussed.

1. はじめに

NSA による SELinux[1] [2]の公開を受け、近年、セキュア OS が関心を集める。ファイアウォールや IDS（ Intrusion Detection System ）・ IPS（Intrusion Prevention System）等のネットワーク型・パケット監視型のセキュリティ対策と異なり、セキュア OS はサーバーサイドの要塞化手段として位置付けられ、不正侵入を許してしまった場合や内部からの情報漏えいといったような、これまでネットワークレイヤの対策では有効な防止手段がなかった状況に効果を発揮するものである。

オープンソースソフトウェアである SELinux は TCSEC （ Trusted Computer System Evaluation Criteria）[3]の B1 グレード相当とされており、商用に耐え得るセキュリティ強度を持つと言われている[4]。

筆者らは OS カーネルのアクセス制御に基盤をおいた IDS・IPS というコンセプトの元、SELinux に機能拡張を施し、自律的な防御機能「Linux カーネルベース IDS」[5] [6]を実現した。

本稿では、まず Linux カーネルベース IDS の概要について述べ、次に初期プロトタイプにおける課題を整理した上で、その解決策として実施した「セキュリティレベルのフレキシブルな制御」に関する 2 つの拡張について紹介する。

2. SELinux 概要

SELinux は FLASK アーキテクチャ[7]と呼ばれるカーネルのセキュリティ拡張を Linux に実装したものであり、以下のような特長を有する。

- 強制アクセス制御
  セキュリティ管理者が設定したアクセス権が管理者も含め、全てのユーザーにトップダウンに「強制」され、任意の変更が禁止される。Mandatory Access Control（MAC）。
- 管理者権限の分割
  システム管理者権限を役割毎に分割が可能。root への管理者権限の集中を抑える。
- 細粒度のアクセス制御

従来の「uid / gid」及び「read / write / execute」でのアクセス権の管理体系が、「任意ラベル情報」及び「詳細なパーミッション区分」での管理体系に拡張される。

- 監査情報
  パーミッションチェックにおける種々の情報をアクセス制御ログとして記録可能である。

SELinux のこれらの機能は、任意アクセス制御（Discretionary Access Control; DAC）と呼ばれる従来の Linux カーネルのアクセス制御部へのフック挿入による拡張セキュリティモジュールとして実現されている（Linux Security Module; LSM）。カーネル 2.6 以降では、LSM は標準モジュールとして統合された。

3. Linux カーネルベース IDS

本章では、Linux カーネルベース IDS の概要について述べる。

3.1 セキュア OS での不正アクセス防止

通常の Linux に比べ、SELinux は強力なアクセス制御機能を持ち、正しくアクセスポリシーが設定されれば、バッファオーバーフロー（以下、BOF）等の攻撃についても被害を抑えることができる。

図 1 で root 権限にて動作中のプログラムに対し、BOF が成功した場合、攻撃者は任意コードの root としての実行権限を得ることとなり、システムの全権が奪われてしまう。

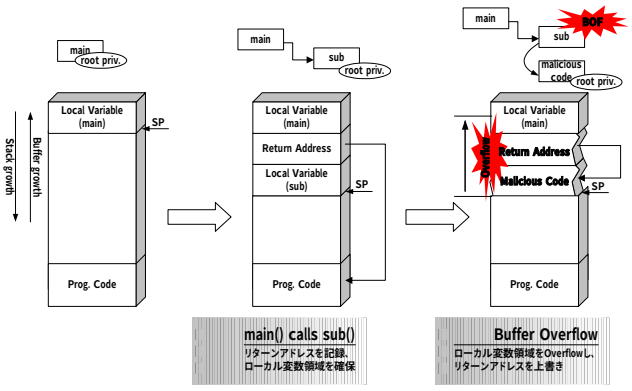

図 1： Buffer Overflow

SELinux においては、元のプロセスの「ラベル情報」が、派生して起こる全てのアクセスに継承され、BOF により発生する不正アクセスにも例外なく適用されるため、アクセスポリシーが適切に設定されている限りは、実際の被害を招くことはない。例えば、ftpd からの BOF を介した不正な shell 起動を防ぐためには、プロセス（ftpd）とアクセス対象（/bin/bash）に適当なラベル情報を付与した上で、その組み合わせに対しては実行許可（'execute'）を与えない限りは、その被害を防ぐことが可能である。

外部からの攻撃だけでなく、内部利用者、特に悪意ある管理者等によるアクセスポリシー違反に対しても同様にチェック可能であることから、改ざんや情報漏えい等への対策ともなり得る。

SELinux のアクセスポリシーの記法は以下のように、アクセス許可とする対象を allow ステートメントにより定義する。

【SELinux のポリシー記法】
例）**allow httpd_t sys_content_t:file**
{ read search getattr } ;
… httpd によるコンテンツファイル参照許可

allow <subject> <object>:<class> <perms> ;
subject:アクセス主体(httpd)のラベル
object:アクセス客体（web コンテンツ）のラベル
class:オブジェクトクラス（file クラス）
perms:対象となるパーミッション（read、他）

3.2 カーネルベースの不正アクセス検知

「ネットワーク型 IDS」とは異なる観点で、サーバーサイドでの不正行為の検知を行うための種々の機構が存在する。一例として、ファイルシステムのインテグリティチェックのためのツール Tripwire[8]が挙げられる。これらはしばしば「ホスト型 IDS」に分類されるものである。

本稿の「カーネルベース IDS」という用語であるが、ホスト型 IDS の１つとして、「OS カーネルのアクセス制御機構を基盤とした不正アクセスの検知・防止機構」を指すものとして用いる。

カーネルベース IDS はプロセスの通常の振る舞いから大きく逸脱する行為の検出を試みるものである。例えば先に挙げた「権限のない ftpd からの shell の実行」等は不正アクセス発生の決定的な証拠と言える。

SELinux はこのような非常に重大な局面に瀕したとしても、そのアクセス制御機能に基づく「抑止力」としては機能するものの、その情報を有効に利用する術を持たない。アクセス制御ログとして syslog への出力は可能であるが、事象の重要度による差別化は特に行われないため、他のログ情報に埋もれてしまうのみであり、不正行為を取り逃がしてしまうこととなる。

このような重大なアクセスポリシー違反は、他のアクセス拒否のケースとは区別して扱われるべきである。そこで「アクセス許可」の対象のみを

規定する、SELinux のアクセスポリシーに対し、特に注意を要する「重大なアクセスポリシー違反」に関する定義情報を追加し、その定義情報との比較判定（セキュリティチェック）をパーミッションチェックとは別個に行う、という機構を筆者らは提案し、SELinux への拡張として実装した。セキュリティチェックの定義対象としては以下のようなものが考えられる。

【侵入・権限取得】
- 権限の無いプロセスからの shell 等の起動
- 権限奪取・権限拡大の試み

【改ざん・破壊行為】
- 実行ファイル・設定ファイル・ログファイルの改ざん・消去

【情報漏えい行為】
- 機密ファイルへの不正な参照行為
- リムーバブルメディアの mount/umount

アクセスそのものの成功・不成功に関わらず、これらのアクセスが試みられた時点で「アクセスポリシー違反」として検知される。検知対象を定義するための構文として、以下の記法、strict ステートメントを追加した。

strict ftpd_t shell_exec_t:file { execute } ;

【解説：ftpd への BOF 検知】
ftpd による不正な shell 起動を検知対象とする。

3.3 IPS 機能 ～ 動的アクセスポリシー制御

本節では先の strict ステートメント定義で得られる検知情報がどのように用いられるかを述べる。

不正アクセスを検知し、自動的に対処を行う機構がいくつか提案されている。例として、IDS とファイアウォールを連携させ、IDS での検知をトリガとして、ファイアウォールのルールを動的に変更し遮断する、といった手法が挙げられる [9]。

Linux カーネルベース IDS においては、アクセスポリシー違反の検知後、以下の対応を行う。

- 動的アクセスポリシー制御
  違反検知に連動して、アクセス権を縮退する。
- 動的ロギング制御
  ログ監視の対象範囲・取得粒度を動的に制御する。
- 通知機能
  カーネルのセキュリティレベル参照のためのインタフェースを提供。

前述の通り、MAC の機能のみでポリシー違反行為の防止は可能だが、「動的アクセスポリシー制御」を利用した遮断機能により、攻撃を受けた際のリスクをさらに低減可能である、という点が本拡張のメリットとして挙げられる。

一例として、権限奪取のための手段である BOF を検知した段階で、攻撃者が奪おうとしている権限そのものを縮退させてしまうことにより、その後の破壊・改ざん・漏えい等の行為に対するリスク低減が期待できる。

アクセスポリシーの動的制御情報の記述においては、「状態ラベル」を allow ステートメント末尾に付与し、状態毎にポリシーのルールセットを定義する。また、検知対象定義（strict ステートメント）に関しても状態毎に定義可能とした。

状態毎の allow / strict ステートメント定義
状態ラベル　通常…1 / 縮退…2

```
allow sysadm_t httpd_conf_t:file
     { create ioctl read getattr write unlink..} 1;
  … 通常時の sysadm の httpd 設定ファイルに
     対するパーミッション定義

allow sysadm_t httpd_conf_t:file
     { read getattr ioctl }  2;
  … 権限の縮退（create, write,unlink を禁止）
```

検知から権限縮退までの流れを図 2 に示す。

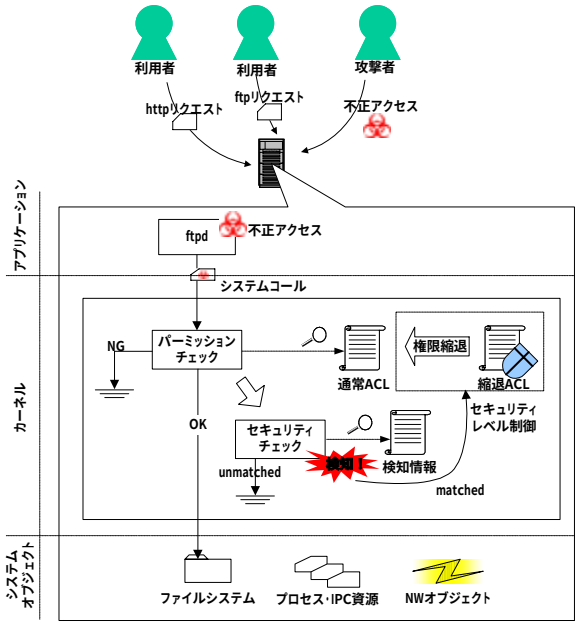

図 2：Linux カーネルベース IDS

カーネルに検知･防御の基盤を置くことにより、システム資源への最終防衛線として機能し、以下

に示すようなこれまでのホスト型 IDS・IPS では得られなかった優位性を備えることが可能となる。

- リアルタイム性
  検知と防御を OS レイヤで極めて密に連携させるため、リアルタイム性が高い。
- 一般性・汎用性
  亜種攻撃であっても、「逸脱の形態」が類似であれば、検知・防御が可能である。

3.4 カーネルによる自律制御

SELinux のアクセスポリシーはランタイムでの動的変更に対応する [10]。権限縮退のための動的アクセスポリシー制御の実装にあたっては、このような既存の機能も利用可能であるが、外部インタフェースを介したポリシー変更の許可は、「ポリシーの改ざん」という新たなリスクに結びつくものである。

また、検知手法との連携においてはログ監視ツール等からの通知に基づく場合、リアルタイム性も損なわれると考えられる。

筆者らのアプローチとしては、アクセスポリシーの動的制御内容・制御条件も含めて、ブート時にカーネルメモリにロードし、以後のポリシー制御はカーネルに閉じた形で、自律的に動作させるというものである。これにより、ポリシー情報への不正な干渉を防ぎつつ、安全に動的アクセスポリシー制御を実現することが可能である。

3.5 実装概要

ここまでの指針をベースに、表 1 に挙げた拡張を SELinux（Kernel 2.4 系）に行った。（「初期プロトタイプ」と呼ぶ）

表 1：改造箇所（初期プロトタイプ）

対象	機能拡張
パーミッション構造体（Access Vector）	・strict / watch メンバの追加 ・動的制御情報の定義（パーミッション情報の配列化）
アクセス制御関数	・セキュリティチェック追加
状態遷移関数	・検知に基づく状態遷移制御ロジックの追加
ツール	・ポリシーパーザの拡張、入出力関数の変更
制御インタフェース	・avc_securitylevel 等を追加

4. 検知・防御の高度化

本章では、Linux カーネルベース IDS の初期プロトタイプにおける課題を整理し、実施したいくつかの機能改善について解説する。

4.1 プロセス・ドメイン単位の権限縮退

初期プロトタイプにおいては全プロセス一括で権限縮退を行う。このため、権限縮退の影響はすべてのプロセスに及び、アクセスポリシー違反者だけでなく、正規利用者のサービスレベルも低下させてしまう可能性がある。

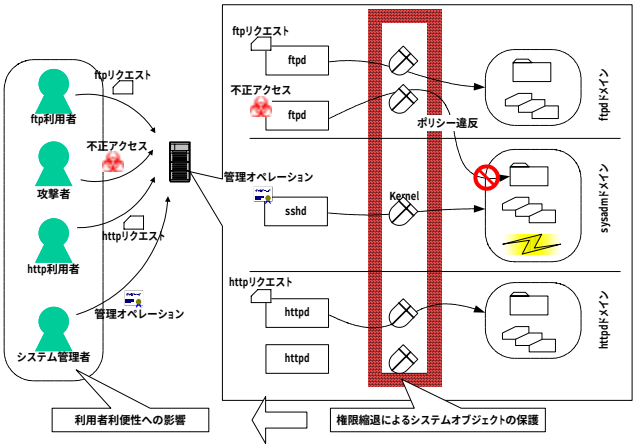

図 3：初期プロトタイプでの制御範囲

これを逆手にとり、サービスレベル低下を目的とした、意図的な違反行為が行われてしまうケースも考えられるため、違反者と正規利用者を区分するための対策が必要となる。

このような場合、違反行為に関連するプロセス・アプリケーションドメインのみを対象とした限定的なアクセス権の縮退を状況に応じて選択可能であるのが望ましい。

そこで検知対象の定義記法 strict ステートメントにおいて、セキュリティレベルの制御範囲を規定するための制御オプション（scope）についての検討を行った。制御範囲として表 2 に示した 8 項目を指定可能とした。各々の用法について述べる。

表 2：ポリシー制御範囲

表記	範囲
self	検知対象となったプロセス自身 軽微なポリシー違反で、権限縮退がそのプロセスの存在期間のみでよい場合
parent	上記 self を生成した親プロセス 軽微なポリシー違反で、検知対象を生成したプロセスの権限縮退を行う場合。後述の「セキュリティレベルの継承」参照
pparent	上記 parent を生成した親プロセス
selfdom	上記 self と同一ドメインのプロセス 中危険度のポリシー違反で、攻撃対象となった AP ドメイン全体や攻撃者のセッション全体に権限縮退を行う場合

pdom	上記 self の遷移元ドメイン
ppdom	上記 pdom の遷移元ドメイン
procs	全てのプロセス ⇒ 初期プロトタイプ仕様 高危険度のポリシー違反で、管理者権限も含め、縮退を行う場合
kernel	暗黙的セキュリティレベルの制御（後述）

strict user_t etc_t:file {write unlink} {parent} ;
… 一般ユーザーによる/etc への不正操作を検知した場合、ユーザーが使用中のシェル権限を縮退。
※ self は検知対象（違反したコマンド）、parent はそのプロセスを起動したシェルに相当

サービスレベル低下が懸念される状況では、プロセスやアプリケーションドメインを指定し、正規利用者への影響を防ぐ。一方、危険度が高いポリシー違反を検知した場合には初期プロトタイプと同様、全プロセスを対象とする。これが scope の基本的な用法となる。scope は必要に応じて、複数を同時指定することも可能とした。

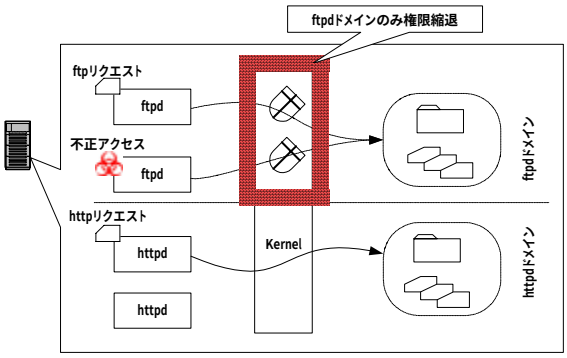

図 4：プロセス・ドメイン単位での制御

4.2 セキュリティレベル継承について

各プロセスのセキュリティレベルはタスク構造体 task_struct -> task_security_struct 内への追加メンバ seclevel / audlevel により管理を行い、fork() / execve() / clone()等により、子プロセス等が生成される場合には生成元の親プロセスのセキュリティレベルが継承されることとした。

xinetd が起動する ftpd 等のサービスが攻撃を受けた場合の権限縮退において、「parent」、「pdom」等、プロセス生成元を対象とした指定が効果的である。xinetd が違反行為の検知以降に生成する全てのサービスデーモンについては、セキュリティレベル継承の機能に基づき、継続的に権限縮退を行う一方で、検知以前からの既存ユーザーのセッションについては権限縮退を行わない、というような制御が可能となる。（図 5 参照）

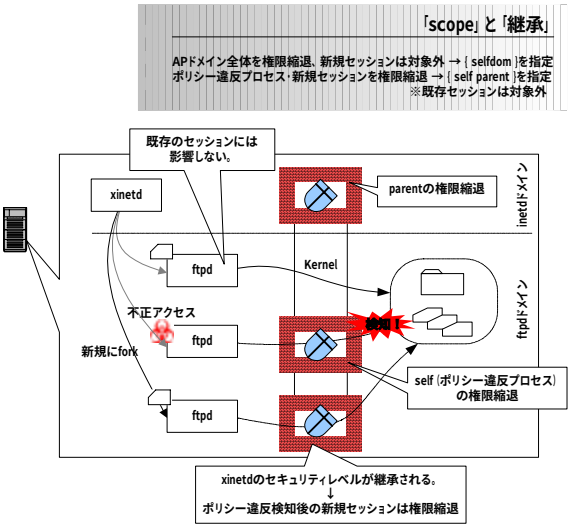

図 5：セキュリティレベルの継承

4.3 暗黙的セキュリティレベル

Linux カーネルベース IDS において、セキュリティレベルの概念は、ポリシー設定構文上、サブジェクト（アクセス主体）となり得るシステムオブジェクトに適用される。アクセス制御で扱われるサブジェクトのほとんどはプロセスだが、一部、プロセス以外のシステムオブジェクトもサブジェクトとして扱われる。

管理をタスク構造体で行っているため、その管理下にないこれらのサブジェクトに関してはセキュリティレベルの適用対象外であるが、アクセス制御のロジック上、何らかの値が定義される必要がある。

そこで、意味的な妥当性、コーディング上の利便性などから、それらのベースとなる「暗黙的セキュリティレベル」を定義し、適用することとした。以下に例示する。

① 実質的なサブジェクトがプロセスではない。
例）IPC 資源クラス等
allow sysadm_t self:sem associate ;
allow sysadm_t self:msgq enqueue ;

② 影響範囲が大きいため、個別プロセスのレベル値による判定が好ましくない。
例）ファイルシステム、ネットワーククラス等
allow iso9660_t iso9660_t:filesystem associate ;
allow ypbind_t netif_t:netif tcp_send ;

4.4 状態遷移の文脈性

Linux カーネルベース IDS における検知 → 権限縮退の一連の流れは、図 6 のような状態遷移図として表すことができる。セキュリティレベルは各状態に対応し、状態毎にパーミッションが定義さ

れ、違反検知を契機として状態遷移が行われる、というモデルで表現される。

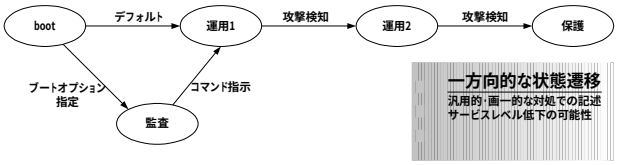

図 6：カーネルの状態遷移

初期プロトタイプのもう一つの問題点として、セキュリティレベルがインクリメンタルに制御される仕様であったため、ポリシー違反内容、検知時の状況等を制御要因としては用いず、単純に次の状態に遷移するのみであった。このため、権限縮退の記述としては、複数事象に対応可能な形の、汎用的内容が求められ、必ずしも妥当な対処とならないような場合も起こり得る。

このため、状態遷移パスに自由度を与え、より的確に対処が可能となるように拡張を行う。具体例について、図 7 に示す。

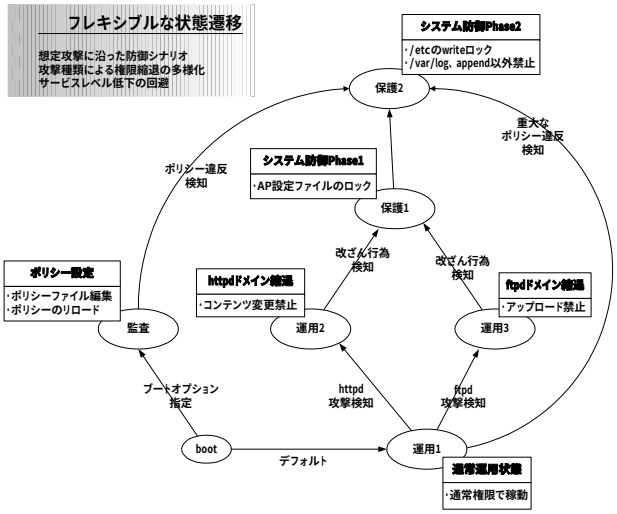

図 7：フレキシブルな状態遷移

本機能は検知・権限縮退の制御に「文脈性」という制御要因を持たせるためのものであり、攻撃者により行われる一連の手順 ～情報収集（ポートスキャン等）、侵入（BOF 等）、活動（破壊・改ざん等）、後処理（バックドアの設置、侵入痕跡の消去等）～ に対し、その時点の状況、違反行為の内容に応じ、最適な防御を実施させることを可能とする。

表記上は strict ステートメントに goto フレーズを追加することにより、任意の遷移先を指定可能とした。

strict user_t etc_t:file {write unlink} goto 3 {self} 1;

セキュリティレベル 1 で、一般ユーザーによる /etc への不正操作を検知した場合、そのプロセスのセキュリティレベルを 3 に変更。

4.5 MAC 権限に関して

セキュリティレベルの取得・変更等のオペレーション権限は security クラスに新規パーミッションを追加し、以下のように定義する。

allow secadm_t self:security { getlevel setlevel } ;
セキュリティレベルの取得・変更
allow secadm_t security_t:security { getlevel } ;
暗黙的セキュリティレベルの取得

例示した通り、これらの権限はセキュリティ管理者（secadm）に許可する。また、getlevel についてはレベル値参照の必要な管理者・アプリケーションについても許可する。

5. 実装

SELinux（Kernel 2.6 系）をベースとして、初期プロトタイプにこれまで述べた機能改善を実施した。実装に当たっては、表 1 に示した内容に加えて、表 3 に挙げた内容に関する変更を行った。

表 3：変更箇所（現行プロトタイプ）

対象	機能拡張
パーミッション構造体（Access Vector）	・scope / goto メンバの追加
task_security_struct 構造体	・seclevel / audlevel メンバの追加
アクセス制御関数	・セキュリティレベルのプロセス個別管理対応
状態遷移関数	・goto 状態遷移制御ロジックの追加
制御インタフェース	・/selinux、/proc/<pid>/attr の拡張

制御インタフェースに関連して、Kernel 2.6 系 SELinux では、すべて selinuxfs なるファイルシステムを介して、SELinux 関連のオペレーションが行われる。また、各プロセスのセキュリティコンテキスト等の情報取得手段として proc ファイルシステムが使用されるため、今回拡張についてもそれらに倣い、表 4 に示す制御用インタフェースを追加した。

表 4：制御インタフェース

参照インタフェース	説明
/proc/<pid>/attr/seclevel	プロセス<pid>のセキュリティレベル
/proc/<pid>/attr/audlevel	プロセス<pid>の監査レベル
/selinux/seclevel	暗黙的セキュリティレベル
/selinux/audlevel	暗黙的監査レベル

4.5 章で述べた、setlevel 権限を与えられた管理者はこれらのインタフェースを介して、各値の変更を行う。

6. 性能試験

6.1 LMBENCH による性能測定

本拡張によるパフォーマンスへの影響を検証するため、LMBENCH[11]により以下の 3 種類のカーネルについて性能比較を行った。

- Kernel 2.6.3（non-SELinux）
- Kernel 2.6.3（SELinux）
- Kernel 2.6.3（SELinux-IDS：今回拡張）

同一のマシン環境でこれらのカーネルをビルドした。” Security options”以外のカーネルビルドオプションはいずれも同一である。

LMBENCH のうち、パーミッションチェックを多く伴い、本拡張による影響が特に大きいと思われる以下の項目についてデータ取得を行った。また、本 IDS 拡張の代表的用途は Web サーバーへの適用であると考えられ、アプリケーション性能への影響の検証として LMBENCH に含まれる lat_http を用いた Web ベンチマークを行った。

- システムコールの発生（lat_syscall）
  … read() / write() / open() / close()等の性能
- プロセスの生成（lat_proc）
  … fork() / execve()等の性能
- Web ベンチマーク（lat_http）
  …http 処理性能

測定マシンのスペックは以下の通りである。

表 5：実験条件

CPU	Pentium Ⅲ 933MHz
Memory	2GB, (512MBx4 SDRAM 133MHz)
Disk	18GB SCSI Disk, (18 GByte, Ultra160)

6.2 LMBENCH 測定結果

各 3 回ずつ測定を行った。結果を平均値にて、表 6 に示す。

表 6：LMBENCH 結果（Latency）

		non-selinux	selinux	selinux-IDS
lat_syscall [μsec]	read	0.35	0.64	0.71
	write	0.27	0.55	0.61
	stat	2.27	3.85	4.07
	fstat	0.65	0.90	0.95
	open/close	2.90	4.58	4.89
lat_proc [μsec]	fork+exit	148	161	168
	fork+execve	755	847	847
	fork+/bin/sh -c	2658	2863	2872
lat_http[msec]		1.67	1.96	1.98

各カーネルのレイテンシ、及び non-SELinux カーネルのレイテンシを 1 とした場合の伸び率について以下のグラフで示す。

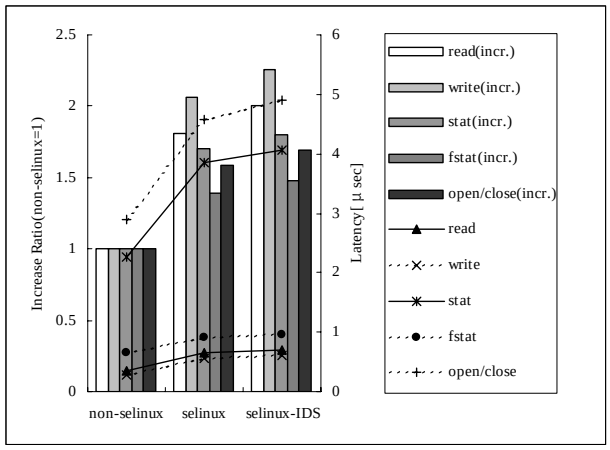

図 8：lat_syscall 結果

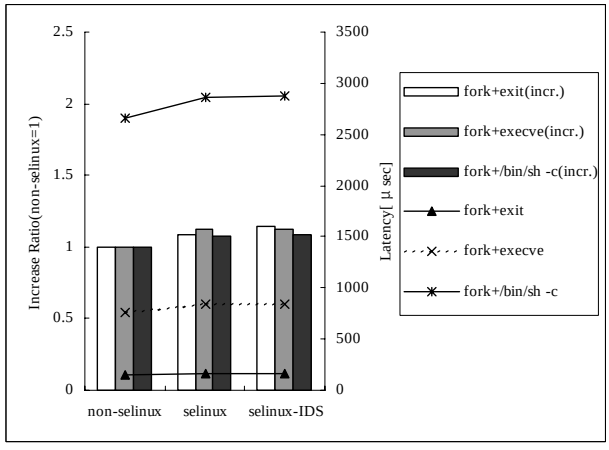

図 9：lat_proc 結果

レイテンシの伸び率で見た場合、non-SELinux に対する SELinux のオーバーヘッドは最大約 110%、SELinux に対する SELinux-IDS のオーバーヘッドは最大約 10%であることを確認した（いずれも lat_syscall,write）。

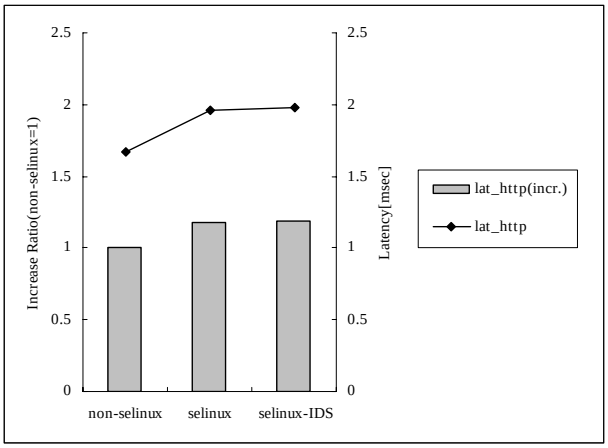

図 10：lat_http 結果

lat_http の結果より、アプリケーション性能への影響も微少という結果を得た。

以上の結果より SELinux と比較した場合の本拡張による性能への影響は僅かであると見られる。

7. まとめ

Linux カーネルベース IDS の概要及び正規利用者のサービスレベルへの影響の局所化という観点で行った二つの機能拡張の取り組み「範囲限定的な権限縮退」と「フレキシブルな状態遷移」について報告を行った。

今後は運用性の改善や検知・対応の高度化を中心とした、以下の技術課題について検討を行う予定である。

- セキュリティツールとの連携
  他の IDS による検知情報を、動的アクセスポリシー制御の際のトリガとして用いる、もしくは本カーネルの検知機構を他の IPS のセンサとして用いる等、他のセキュリティツールとの連携に関する検討。
- ログからのポリシー自動定義
  アクセス制御ログに記録された攻撃者のふるまいに基づいて、検知対象及び防御シナリオを自動的に定義する手法の検討。

なお、本稿の成果物に関しては、GPL 等のライセンスに基づく公開準備を現在、進めている。

謝辞

すべての Linux 開発者の皆様に、敬意と感謝の意をここに表します。

本研究開発の検討・設計から実装まで、全般に渡りご協力いただいた、ソフトウエア興業株式会社 碓井 啓弘氏に、ここに感謝の意を表します。

Linux カーネルのソースコード解析、コーディングについて助言いただいた Vigo 大学 フェルナンド・バスケス氏に、ここに感謝の意を表します。

参考文献

[1] NSA Security-Enhanced Linux ホームページ
http://www.nsa.gov/selinux/

[2] C. Wright et al., “Linux Security Modules: General Security Support for the Linux Kernel”, USENIX Security Symposium ’02

[3] DoD Trusted Computer System Evaluation Criteria (Orange Book), 1985

[4] IPA 調査報告「オペレーティングシステムのセキュリティ機能拡張の調査」
http://www.ipa.go.jp/security/fy13/report/secure_os/secure_os.html

[5] 保理江、原田、田中 "アクセスベクタ拡張による Linux カーネルの自律防御"電子情報通信学会ソサイエティ大会、2003、A-7-1

[6] 保理江、原田、田中 他 "OS カーネルにおける動的アクセス制御" 情報処理学会 研究報告「コンピュータセキュリティ」、2003、CSEC-23-005

[7] The Flux Research Group ホームページ
http://www.cs.utah.edu/flux/fluke/html/flask.html

[8] Tripwire ホームページ
http://www.tripwire.org/

[9] JNSA WG 活動報告「ダイナミックディフェンスの概要と適用について」
http://www.jnsa.org/active00_8.html

[10] Tresys 社 SELinux ホームページ
http://www.tresys.com/selinux/checkpolicy_prototype.html

[11] LMBENCH ホームページ
http://www.bitmover.com/lmbench/