情報処理学会論文誌　コンピューティングシステム　Vol.5　No.2　63–73 (Mar. 2012)

SELinuxの不要なセキュリティポリシ削減の自動化手法の提案

矢儀 真也[1]　中村 雄一[2]　山内 利宏[1,a)]

受付日 2011年7月19日, 採録日 2011年10月24日

概要：SELinux のセキュリティポリシは設定が難しいため，汎用的なポリシを利用することが多い．しかし，汎用的なポリシは，個々のシステムに必要のない権限を許可している可能性がある．また，ポリシが占有するメモリ使用量が多く，組み込み機器には適していない．これらの問題への対処として，不要なポリシを自動で検出し，削減する手法を提案する．提案手法は，SELinux が出力するログを利用して不要なポリシを検出する．また，システム管理者にポリシの修正を提案し，システムのセキュリティを向上させ，ポリシのメモリ使用量を削減できる．本論文では，SELinux のポリシの問題点と対処方法を示し，設計と評価について報告する．

キーワード：SELinux，セキュリティポリシ，最小特権，セキュリティ

Proposal of a Method to Automatically Reduce Redundant Security Policy of SELinux

Shinya Yagi[1]　Yuichi Nakamura[2]　Toshihiro Yamauchi[1,a)]

Received: July 19, 2011, Accepted: October 24, 2011

Abstract: In many cases, general security policy is used because of the difficulty of creating security policy. However, general security policy is possible to allow excessive rights in system. In addition, it is difficult to use this security policy in embedded systems because of the memory footprint. To deal with these problems, we propose a method system automatically detects redundant security policies by using log SELinux outputs and deletes them. The proposed system also suggests system administrator and improves security of the system and reduces the memory footprint. This paper shows the problems of security policy and dealing with them. This paper also shows design and evaluation.

Keywords: SELinux, security policy, least privilege, security

1. はじめに

ソフトウェアの脆弱性を利用した様々な攻撃により，個人情報の漏洩などの被害が発生している．攻撃の中でもゼロデイ攻撃は，未知の脆弱性を利用した攻撃のため対処が難しい．また，攻撃者が権限昇格によって管理者権限を取得した場合は，攻撃者がすべての権限を得るため，被害が大きくなる可能性が高い．

これらの問題を解決する手段として，Security-Enhanced Linux（以降，SELinux と略す）[1] に代表されるセキュアOS の利用がある．しかし，SELinux のセキュリティポリシ（以降，ポリシと略す）には，設定の難しさ，最小特権実現の難しさ，およびメモリ使用量の問題がある．ポリシの設定が難しいため，自分でポリシを作成することなく，ポリシ開発者が配布しているポリシを利用する場合が多い．しかし，このポリシは，利用しないデーモンやアプリケーションに関するポリシを含む汎用的なポリシであるため，個々のシステムにはアクセスを許可する必要のないポリシ

[1] 岡山大学大学院自然科学研究科
Graduate School of Natural Science and Technology, Okayama University, Okayama 700–8530, Japan
[2] 株式会社日立ソリューションズ
Hitachi Solutions, Ltd., Shinagawa, Tokyo 140–0002, Japan
[a)] yamauchi@cs.okayama-u.ac.jp

（以降，不要なポリシと呼ぶ）が含まれている可能性が高い．また，配布されているポリシのメモリ使用量は 5 MB を超えるため，組み込み機器のようにメモリのサイズが限られる場合，利用が難しい [2]．

ポリシの設定の難しさの問題を解決するために，SELinux Policy Editor（以降，SEEdit と略す）[3], [4] や SLIDE [5] などのポリシの作成を支援するツールが開発されている．SEEdit は Simplified Policy Description Language と呼ばれる独自の中間言語でポリシを記述するツールである．SEEdit の利用者は，SELinux について詳しい必要はなく，Linux のアプリケーションの動作に詳しい人でも利用できる．また，SLIDE は Reference Policy（以降，refpolicy と略す）[6] を作成するための Eclipse ベースの統合開発環境であり，入力の補完などによりポリシの記述を支援する．

しかし，これらのツールを利用したとしても計算機システムの知識が必要であり，設定工数が多いため，ポリシの作成は簡単ではない．たとえば，SEEdit の利用者は，アプリケーションの動作を詳細に把握する必要があることに加えて，利用するアプリケーションとリソースに，適切なラベルを付与する必要がある．このため，最小特権を実現したポリシを作成するには，多くの時間がかかる．また，SLIDE の利用者が，ポリシの作成や修正をするには，数万行にも及ぶコードに加え，1,000 種類以上のマクロを解読する必要がある．このため，理解が難しく，ポリシの作成に多くの時間がかかる．

そこで，ポリシ開発者が配布しているポリシから，利用する計算機で不要なポリシを自動的に検出し，削除する手法を提案する．提案手法は，SELinux が出力するログを利用して，不要なポリシを検出し，システム管理者にポリシの修正を提案する．システム管理者が修正を許可した場合，自動でポリシを修正する．これにより，最小特権を持つポリシの設定におけるシステム管理者の負担を軽減し，実行時にポリシが占有するメモリ使用量の削減を実現する．

2. SELinux のセキュリティポリシと問題点

2.1 SELinux のアクセス制御機構

SELinux は，National Security Agency を中心するコミュニティで開発されているセキュア OS である．セキュア OS とは，強制アクセス制御（Mandatory Access Control: MAC）と最小特権（Least Privilege）を実現する OS またはカーネルモジュールである．MAC は，OS におけるアクセス権限の管理者が定めたポリシのもとで，すべてのファイルやプログラムのアクセス権限が一元的に制御され，所有者が設定を変更できないアクセス制御である．最小特権は，サブジェクトに必要最小限のアクセス権を与えることができる機能である．SELinux は，Multi Level Security, Role Based Access Control，および Type Enforcement を実現している [7]．

2.2 SELinux のセキュリティポリシの構造と refpolicy

SELinux のポリシは，ポリシルール（policyrule: allow, auditallow，dontaudit，neverallow），ドメイン（subj_t），タイプ（obj_t），オブジェクトクラス（tclass），およびアクセスベクタパーミッション（av）で構成される．以下にポリシの文法を示す．

policyrule subj_t obj_t:tclass{av};

ポリシルールのうち，allow は許可を意味し，auditallow はアクセスを許可した際に出力するログ（以降，許可ログと略す）の出力を意味する．ドメインはプロセスのラベルであり，タイプは操作対象となるファイルやネットワークなどのシステム資源のラベルである．オブジェクトクラスとは，ファイル，ディレクトリ，ソケットのように，オブジェクトの種類を分類するものである．アクセスベクタパーミッションとは，read や write のようなアクセスパーミッションであり，オブジェクトクラスごとに定義されている．

本研究では，利便性の高さから refpolicy を扱う．refpolicy は，よく利用されるアプリケーションに対して，問題なく動作するように設定されたポリシのサンプルのことであり，Fedora や CentOS では標準で利用されている．また，refpolicy は，ポリシがモジュール化されており，ポリシの運用中でも，モジュール単位でポリシの追加や削除が可能である．さらに，refpolicy には targeted ポリシと strict ポリシがあり，本研究では両方のポリシを対象とする．

refpolicy は以下の 3 つのファイルから構成される．

(1) fc（file context）ファイル
(2) if（interface）ファイル
(3) te（type enforcement）ファイル

fc ファイルには，プロセスやシステム資源のパス名とラベルの対応付けを記述する．if ファイルには，te ファイルで使用するマクロを記述する．te ファイルには，アクセス権限の付与を記述する．

2.3 SELinux のポリシの問題点

2.3.1 ポリシの設定の難しさ

SELinux のポリシの設定の難しさは，以下の 3 つに分類される [8]．

(1) ラベル設定の難しさ
利用する計算機システムについて詳しく知らなければ，どのプログラムや資源にどのラベルを設定すべきか判断が難しい．

(2) パーミッション設定の難しさ
アクセス権限を与えるべきすべてのドメインとタイプの組合せに対して，ルールを記述する必要がある．また，700 を超える種類のパーミッションから，適切なパーミッションを設定する必要がある．

(3) アプリケーションの振舞いの理解が必要
アプリケーションがどのシステムコールを利用するかを知らなければならない．また，システムコールとパーミッションの対応付けも知らなければならない．

2.3.2 最小特権の実現の難しさ

SELinux のアクセス制御は，ポリシに記述したアクセスのみを許可するホワイトリスト方式であるため，誤って必要以上の権限を与えることがある．必要以上の権限を与えてしまう原因の 1 つとして，配布されているポリシを利用することがある．配布されているポリシは，利用していないデーモンやアプリケーションに関するポリシを含んでいるため，個々のシステムに必要のない権限を許可している可能性がある．また，利用しているデーモンやアプリケーションでも，様々な環境で問題なく動作するように，多くの権限が与えられている．このため，動作しているシステムの最小特権とは差が生じる．

2.3.3 メモリ使用量の多さ

現在，SELinux で利用されている汎用的なポリシのメモリ使用量は増加傾向にある．たとえば，Fedora 13 で利用されているポリシ（refpolicy）のメモリ使用量は約 5.6 MB である．このため，組み込みシステムのようにメモリが限られている場合には，必要最小限のポリシを作成することでメモリ使用量を削減する必要がある．

3. SELinux の不要なセキュリティポリシ削減の自動化手法の設計

3.1 考え方

SELinux のポリシの問題点を解決するために，不要なポリシ削減の自動化手法を提案する．提案手法は，refpolicy を対象としたシステムであり，自動で不要なポリシを発見し，削除する．これにより，ポリシを最小特権に近づけ，ポリシのメモリ使用量を減らすことができる．また，誤って必要なポリシを削除してしまった場合のために，ポリシの復元機能を備える．

提案システムは 3 つの利用フェーズ（ログ収集とポリシ削減期間，テスト運用期間，および実運用期間）を想定している．ログ収集とポリシ削減期間は，ポリシの削減に必要となる許可ログを収集し，不要なポリシを削除する期間である．テスト運用期間は，誤って必要なポリシを削除したか否かを判定し，必要があればポリシを復元する期間である．実運用期間は，提案システムを OFF にし，提案システムで不要なポリシを削減したポリシを利用して運用する期間である．

不要なポリシを発見する方法として，SELinux が出力するログを用いる．ログには，SELinux のポリシを作成するために必要不可欠であるドメイン，タイプ，オブジェクトクラス，およびアクセスベクタパーミッションが含まれる．ログを収集した後に，現在運用中のポリシとログから作成したポリシを比較することで不要なポリシを発見し，削除する．

また，誤って削除したポリシを発見する方法も不要なポリシを発見する方法と同様に，SELinux が出力するログを用いて実現する．

3.2 ポリシ削減の自動化手法への要求

(1) ポリシの要否を判断するのに必要なログを取得できること
(2) 誤って削除したポリシを復元できること

不要なポリシを発見するために，アクセスに関するログを取得する必要がある．提案手法では，既存のシステムである，Linux Auditing System を利用して，ログを取得する．

また，ポリシを誤って削除したことを判定するために，アクセスを拒否した際に出力されるログ（以降，拒否ログと略す）を利用する．拒否ログをもとに以前削除したポリシに関するアクセスかどうかを判定し，復元を行う．

3.3 設計内容

3.3.1 設計方針

(1) ポリシが必要か否かは管理者が判断する．
(2) ポリシの修正時に，提案手法は管理者がポリシを削除する際の判断を支援する．
(3) 誤って必要なポリシを削除した場合，できるだけ早くポリシの復元を提案する．
(4) カーネルに修正を加えない．

提案手法では，不要なポリシを検出してもすぐに修正せずに，ポリシの修正をシステム管理者に提案するにとどめ，最終的にはシステム管理者に判断を委ねる．また，誤って必要なポリシを削除してしまった場合に備えて，ポリシのバックアップをとり，復元する機能を備える．これにより，システムの信頼性を高める．さらに，カーネルに修正を加えないことにより，導入時の手間を省くだけでなく，汎用性を高める．

3.3.2 基本構成

本手法では，最初に，配布されたポリシで動作をさせ，SELinux が出力するログを収集する．一定期間収集後，収集したログを利用して不要なポリシを発見し，取り除く（以降，ポリシ削減機能と呼ぶ）．しかし，ポリシ削減の問題点として，今までは許可されていた必要な操作にもかかわらず，誤ってポリシを削減した場合，アクセスの拒否が起こる可能性がある．そこで，ポリシの削減終了後に，修正したポリシでシステムを運用し，誤って必要なポリシを削除したことを発見した場合，削除したポリシを復元する（以降，ポリシ復元機能と呼ぶ）．本手法の全体像を**図 1** に示す．ただし，図 1 の番号は，3.3.3 項のポリシ削減機能の説明に対応している．また，図 1 の管理用ポリシとは，

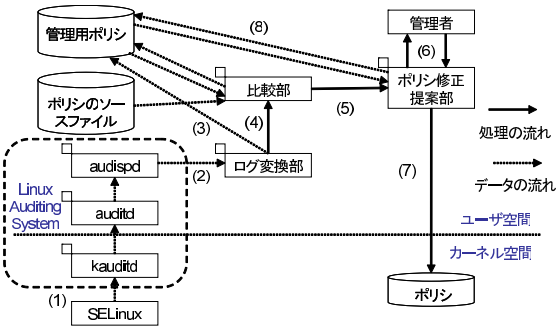

図 1　システムの全体像
Fig. 1　System overview.

利用されたポリシや削除したポリシなどの情報を保存しているファイルのことである．ログ変換部，比較部，およびポリシ修正提案部については後述する．

3.3.3　ポリシ削減機能

ポリシ削減機能は，ログの収集と不要なポリシの削減の 2 つのフェーズからなる．以下に，ポリシ削減機能の流れを示す．

(1) SELinux がログを出力
(2) ログ変換部が，Linux Auditing System の一部である audit dispatcher daemon（以降，audispd と略す）からログを受信
(3) ログ変換部がログをポリシに変換し，管理用ポリシに保存
(4) (2)，(3) を一定時間繰り返し，ログを収集した後，比較部を起動
(5) 比較部が，ログから作成したポリシとポリシのソースファイルを比較し，差分のポリシを作成した後，ポリシ修正提案部を起動
(6) ポリシ修正提案部が管理者に差分のポリシの内容を通知し，ポリシの修正を提案
(7) 管理者がポリシの修正を許可した場合，ポリシ修正提案部がポリシを修正し，システムに反映
(8) ポリシ修正提案部が修正したポリシを管理用ポリシに保存

3.3.4　ポリシ復元機能

ポリシ復元機能は，アクセスの拒否が発生した際に，誤って削除したポリシを復元する機能である．以下に，ポリシ復元機能の流れを示す．

(1) SELinux が拒否ログを出力
(2) ログ変換部が audispd から拒否ログを受信し，ポリシに変換後，比較部を起動
(3) 比較部が，拒否ログから作成したポリシと管理用ポリシを比較し，以前に削除したポリシと一致するかを調査
(4) 一致した場合，比較部がポリシ修正提案部を起動し，ポリシ修正提案部が管理者にポリシの復元を提案
(5) 管理者がポリシの修正を許可した場合，ポリシ修正提案部がポリシを修正し，システムに反映
(6) ポリシ修正提案部が修正したポリシを管理用ポリシに保存

3.3.5　ログ変換部

ログ変換部は audispd からログを受信し，ログをポリシに変換するデーモンである．本構成部の処理の流れを述べる．

(1) audispd からログを受信
(2) ログをポリシに変換し，管理用ポリシに保存
(3) ログが拒否ログだった場合，比較部を起動

本構成部を実現するための課題と対処を以下に示す．

(課題 1)　ログを出力させる方法

不要なポリシを発見するために，許可ログ，または拒否ログを出力させる必要がある．そこでログを出力させる方法として auditallow 文を利用した．

(課題 2)　ログを出力する量を制限する方法

すべての許可ログを出力させた場合，ログの出力によるオーバヘッドが大きく，audit の設定によっては，計算機が正常に動作しなくなる問題がある．また，カーネル空間からユーザ空間にログを転送する際，一定時間に出力するログの量が，ログを一時的に保管する領域である backlog を超えるとユーザ空間にログを転送する前に，ログが消失してしまうという問題がある．そこで，すべてのモジュールに auditallow 文を適用せずに，調査したいモジュールのみに適用し，ログの出力を制限する．

(課題 3)　ポリシを復元する契機

誤って削除したポリシの復元は，システムが知りえた早い段階で行ったほうがよい．そこで，拒否ログが出力されたとき，以前削除したポリシか否か判定を行い，復元を行う．このタイミングであれば必要な可能性のあるアクセスを 1 回拒否するだけで，影響は小さいといえる．また，一定時間経過したときにまとめてポリシを復元することもできるようにする．

3.3.6　比較部

比較部は，ログから作成したポリシと，ポリシのソースファイルを比較し，差分のポリシを作成する．比較部は 2 種類の比較を行う．1 つ目は，ポリシ削減機能のための比較であり，2 つ目はポリシ復元機能のための比較である．本構成部の処理の流れを述べる．

(1) ポリシ削減機能の比較
  (a) ログ変換部が一定時間ログを収集した後，比較部を起動
  (b) 管理用ポリシに保存されている許可ログから作成されたポリシとポリシのソースファイルを比較し，差分を作成
  (c) 不要なポリシが存在した場合，管理者に提示する情報を管理用ポリシに保存した後，ポリシ修正提

表 1　コマンドの仕様
Table 1　Specifications of command.

modify_policy のオプション	内容
-a ⟨モジュール名⟩	モジュールを調査対象に追加
-d ⟨モジュール名⟩	モジュールを調査前の状態に復元
-e ⟨モジュール名⟩	モジュールの調査を終了し，修正
-f	調査済みのモジュールのリストを表示
-l	調査中のモジュールのリストを表示
-m	修正済みのモジュールのリストを表示

案部を起動

(2) ポリシ復元機能の比較
- (a) 拒否ログが出力されたとき，ログ変換部が比較部を起動
- (b) 拒否ログから変換したポリシと管理用ポリシを比較し，以前削除したポリシと一致するものがあるか探索
- (c) 以前削除したポリシと一致した場合，管理者に提示する情報を管理用ポリシに保存した後，ポリシ修正提案部を起動

本構成部を実現するための課題と対処を以下に示す．

(課題 4)　モジュール単位でポリシを修正する方法

モジュールを構成するソースファイル（.te，.if，.fc）は m4 マクロ [9] を利用している．3 つのソースファイルのうち，.te ファイルと.if ファイルはこのマクロをすべて展開しなければポリシの比較に必要な情報を得ることができない．

そこで，次の手法で対処した．モジュールのソースファイルからロード可能なモジュールに変換する際，⟨モジュール名⟩.tmp というファイルを経由する．このファイルは，.te ファイルと.if ファイルのすべての m4 マクロを展開し，1 つのファイルにしたものである．このファイルを利用して，モジュール単位でポリシの比較と修正を行う．

3.3.7　ポリシ修正提案部

ポリシ修正提案部は，ポリシの修正の提案や，ポリシの修正を行う．また，管理者がコマンドを入力することで，ポリシの状態の確認，ポリシの修正ができる．本構成部の処理の流れを述べる．

(1) 管理者にポリシの修正を提案
(2) 管理者が修正を許可した場合，ポリシのソースファイルを修正し，システムに反映
(3) ポリシを修正した内容を管理用ポリシに記述

表 1 に，管理者が利用可能なコマンドである modify_policy の仕様を示す．また，本構成部を実現するための課題と対処を以下に示す．

(課題 5)　ポリシの削除，復元を提案する際に提示する情報

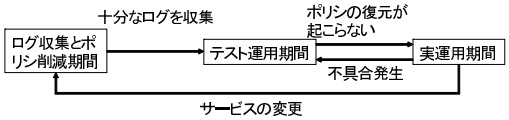

図 2　システム遷移図
Fig. 2　System transition.

本手法では，ポリシの削除，復元を管理者に提案する際に，管理者がポリシの変更をするか否かを判断するために，有益な情報を提示する必要がある．ポリシの削除，およびポリシの復元を提案する際に提示する情報を以下に示す．

(1) ポリシの削除を提案する際に提示する情報
module=⟨モジュール名⟩ subj=“関連するサブジェクト” obj=“関連するオブジェクト” allow subj_t obj_t:tclass{av};

(2) ポリシの復元を提案する際に提示する情報
subj=“サブジェクトのフルパス” obj=“オブジェクトのフルパス” syscall=システムコール名 allow subj_t obj_t:tclass{av};

3.4　利用形態

提案システムの利用形態は，ログ収集とポリシ削減期間，テスト運用期間，および実運用期間の 3 つに分類される．システムの利用形態を**図 2** に示す．実運用期間にサービスを変更する場合，提案システムを ON にし，もう 1 度ログを収集する必要がある．

具体的には，追加したサービスが，提案システムで削除したモジュールを必要とする場合，ポリシ復元機能によりモジュールを復元する．その後，ポリシ削減機能により，モジュールに含まれる不要なポリシを削減したポリシを作成する．ただし，最初から refpolicy に含まれていないサービスを追加した場合，自分でポリシを作成する必要がある．これは，提案システムの適用の有無にかかわらず起こる問題である．

また，既存のサービスを削除した場合，対応するモジュールを調査対象に加えることで，システムが自動的に削除を提案する．提案システムがモジュールを削除することで，不要なモジュールを削除したポリシを作成する．

サービスの処理内容を変更した場合，そのサービスが必要とするモジュールについて，ログの収集からやり直す必要がある．

組み込みシステムの場合，利用するサービスは製品の出荷時に決定され，その後変更することはないと考えられる．このため，上記の問題は発生しない．

3.5　組み込みシステムで想定する利用方法

組み込みシステムでは，製品を出荷後にソフトウェアの構成を変更することはないと考えられるため，製造者があ

らかじめポリシを組み込んでおくことを想定している．提案システムを組み込みシステムで利用する場合，組み込みシステムとは別の計算機（以降，ポリシ作成用計算機と呼ぶ）を用意し，提案システムを適用する．提案システムではサイズの大きい refpolicy のソースファイル，refpolicy を作成するコマンド，および GUI による操作が必要となるため，多くの組み込みシステムでは対応できないためである．

組み込みシステムでの提案システムの利用手順を以下に示す．

(1) ポリシ作成用計算機でログの収集設定
(2) 組み込みシステムにポリシを複写し，ログを収集
(3) ログの収集後，ログをポリシ作成用計算機に移し，組み込みシステム用のポリシを作成
(4) 作成したポリシを組み込みシステムに複写
(5) テスト運用
(6) 実運用

テスト運用で正しく動作しなかった場合，拒否ログをポリシ作成用計算機に移すことにより，拒否された権限を付与したポリシを作成する．この方法を正しく動作するまで繰り返すことで，組み込みシステム用のポリシを作成する．

3.6　期待される効果

本手法を適用することで，期待される効果を述べる．

(1) 最小特権に近づけること

不要なポリシを削除することにより，必要以上のアクセス権限を与えずにすむ．これにより，最小特権に近づけることができる．

(2) 管理者の負担の軽減

SELinux を利用しているシステムの管理者は，利用するサービスが必要とするモジュールを判断し，そのモジュール内のどのポリシが必要かを判断しなければならない．しかし，不要なポリシを発見するためには，利用しているサービスの動作を把握する必要がある．また，SELinux のラベルの仕組みやポリシの文法についての知識が必要である．さらに，refpolicy を利用する場合，マクロの知識が必要であることに加え，必要なモジュールを判断しなければならない．以上のことから，従来手法ではシステム管理者への負担が大きい．

提案システムでは，不要なモジュールの削除や必要なモジュールに含まれる不要なポリシの削減を自動的に提案し，ポリシを修正する．また，システム管理者は SELinux のラベルや refpolicy で利用しているマクロに関する知識は不要である．

以上のことから，提案システムは，システム管理者の負担を軽減できる．

(3) メモリ使用量の削減

不要なポリシを削除することで，カーネルにロードするポリシが減るため，メモリ使用量を削減できる．

4.　評価

4.1　目的と評価環境

本評価の目的と評価の観点を以下に示す．

(1) ポリシの削減量

提案システムにより，ポリシのサイズ，ラベルの数，モジュールの数，および allow 文の数を，どの程度削減できるのかを示す．

(2) ログの収集期間

1 週間のログを収集し，ログの内容を解析することで，どのくらいの期間のログを収集する必要があるかを考察する．

(3) ポリシの解析

削減したポリシや必要なポリシには，どのような傾向があるのかを考察する．

(4) オーバヘッド

提案システムのオーバヘッドの要因とその影響を示す．また，提案システムが，現実的な時間でログを収集し，不要なポリシを削減できるか否かを考察する．

(5) システム管理者への負担

従来手法と提案手法を比較することで，システム管理者への負担を比較する．

(6) 組み込みシステムでの利用

組み込み機器を用いた評価により，提案システムが組み込みシステムでも利用できることを示す．

評価環境は，(1) から (5) の評価については，カーネルは Linux 2.6.34.6-54.fc13.i686.PAE（Fedora 13），CPU は Pentium 4 2.80 GHz，メモリは 512 MB，ポリシのバージョンは selinux-policy-targeted-3.7.19-62.fc13 である．また，(6) の評価については，組み込みシステムの評価環境は，カーネルは Linux 2.6.24.3，CPU は SH7760 (SH4) 200 MHz，メモリは SDRAM 64 MB であり，ポリシ作成用計算機の評価環境は，カーネルは，Linux 2.6.21-1.3194.fc7，CPU は Pentium 4 3 GHz，メモリは 1 GB である．ポリシは，refpolicy-20060307 を利用した．

4.2　ポリシの削減量

ポリシのサイズ，ラベルの数，モジュールの数，および allow 文の数を評価した．ポリシはカーネルにロードして利用されるため，ポリシのサイズの削減は，メモリ使用量の削減と対応している．また，ラベルの数，モジュールの数，および allow 文の数は最小特権に近づけることに対応している．

refpolicy のモジュールには，base モジュールとその他のモジュールがある．base モジュールは，システムに必須のモジュールであり，容易に修正を行うものではないため調査対象から除外した．本評価は，その他のモジュールすべてを調査対象に追加した後，計算機を再起動し，2 日間

ログを収集した後，評価した．計算機は，HTTP，FTP，ファイル共有，メールサーバ，およびDNSが動作しているサーバである．

表 2 に評価結果を示す．allow文の数から，本環境ではポリシの約8割は実際には利用されていないことが分かる．また，モジュールの数から，モジュールの9割以上が利用されていないことが分かる．これは，利用されていないデーモンやアプリケーションのモジュールが多く含まれていたからだと考えられる．

表 3 にポリシの削減量と内訳を示す．表3から，以下のことが分かる．ポリシのサイズはモジュールの削除や，利用しているモジュール内の不要なポリシの削除により削減された．ラベルの数は，ラベルを定義しているモジュールの削除により，削減された．allow文の数は，モジュールの削除や，利用しているモジュール内の不要なポリシの削除により削減された．

以上のことから，提案手法は利用しているモジュールに含まれるポリシと，利用していないモジュールに含まれるポリシをどちらも削減でき，ポリシを最小特権に近づけることができたといえる．また，ポリシのメモリ使用量（サイズ）も約8割削減できた．

4.3 ログの収集期間

ログを1週間収集し，1日ごとにログのバックアップを行った．これにより，バックアップしたログの内容を比較することで，ログの収集期間に関する考察を行う．ログを収集する対象は，サーバ用のアプリケーションは，HTTP，FTP，ファイル共有，NTP，およびプロキシサーバであり，クライアント用のアプリケーションは，Google Chromeである．サーバの利用人数は10人程度である．

収集したログは，同じアクセスに関するログを多く含むため，重複したログが多い．ログのサイズと重複を削除したログのサイズを**表 4** に示す．

表4の重複を削除したサイズを参照すると，最初の1日で必要なログをほとんど収集したことが分かる．この中で3日目と4日目の差分に着目した．2日目と3日目は重複を削除したログのサイズが同じであることに対し，3日目と4日目は重複を削除したログのサイズが違うためである．**図 3** にログの差分を示す．差分はログのローテートに関連したものが多い．たとえばhttpd_tのラベルが付いているapacheに関連したログは，ログのローテートの際にapacheが自動で再起動したために発生したログである．また，squid_tのラベルが付いているsquidに関連したログも，ログのローテートに関するログである．ログのローテートは，時刻やログのサイズなどの条件で動作するため，偶然4日目に発生したと考えられる．

以上のことから，提案システムのログの収集期間はログのローテートのように，定期的に行われる処理を考慮して決める必要があることが分かる．

4.4 ポリシの解析

4.3節の重複を削除したログを利用してポリシの解析を行った．ログから作成したポリシと削減したポリシの数を**表 5** に示す．表5より，どのモジュールに含まれるポリシも約60%削減できたことが分かる．

ログから作成したポリシのパーミッションの詳細を**表 6** に示し，削減したポリシのパーミッションの詳細を**表 7** に

表 2　ポリシの削減量
Table 2　Amount of reduced policy.

	デフォルト	提案手法適用後	削減量（%）
ポリシのサイズ（B）	5,852,591	1,074,081	81.8
ラベルの数	3,083	1,417	54.0
モジュールの数	223	19	91.5
allow 文の数	271,296	49,289	81.8

表 4　ログのサイズ
Table 4　Log size.

日数	サイズ（KB（MB））	重複を削除したサイズ（KB）
1	101,057（98.7）	34.1
2	185,273（180.9）	34.4
3	239,942（234.3）	34.4
4	279,244（272.7）	35.0
5	339,769（331.8）	35.0
6	445,730（435.3）	35.0
7	517,954（505.8）	35.0

```
allow httpd_t httpd_t:capability { kill };
allow httpd_t httpd_t:process { signal };
allow httpd_t httpd_t:sem { destroy };
allow httpd_t httpd_var_run_t:file { write };
allow smbd_t devlog_t:sock_file { write };
allow smbd_t smbd_t:unix_dgram_socket { connect create };
allow squid_t logrotate_t:fd { use };
allow squid_t squid_log_t:dir { add_name remove_name search };
allow squid_t squid_log_t:file { rename setattr };
allow squid_t squid_t:process { signal };
allow squid_t system_cronjob_t:fd { use };
```

図 3　ログの差分
Fig. 3　Difference of the log.

表 3　ポリシの削減量と内訳
Table 3　Detail of reduced policy.

	削減量	モジュールの削除による削減	モジュール内のポリシの削除による削減
ポリシのサイズ（B）	4,778,510	4,455,414	323,096
ラベルの数	1,666	1,666	0
allow 文の数	222,007	195,167	26,840

表 5　削減したポリシの割合
Table 5　Rate of reduced policy.

モジュール名	ログから作成したポリシ	削減したポリシ	削減したポリシの割合（%）
apache	123	242	66.3
chrome	84	125	59.8
ftp	123	178	59.1
ntp	95	155	62.0
samba	216	349	61.8
squid	136	194	58.8
合計	777	1,243	61.5

表 6　必要なパーミッションの詳細
Table 6　Detail of needed access vector permission.

モジュール名	パーミッションの総数	パーミッションの種類	出現回数が 5 位までのパーミッション（出現回数）
apache	123	35	read(21), search(21), getattr(15), open(13), write(8)
chrome	84	25	read(21), search(12), getattr(11), open(8), write(6)
ftp	123	37	read(20), getattr(16), search(13), open(11), write(10)
ntp	95	29	getattr(13), read(13), search(11), open(10), write(9)
samba	216	39	search(33), read(29), getattr(26), write(21), open(20)
squid	136	36	read(23), getattr(18), search(17), open(13), write(8)
合計	777	58	read(127), search(107), getattr(99), open(75), write(62)

表 7　不要なパーミッションの詳細
Table 7　Detail of unneeded access vector permission.

モジュール名	パーミッションの総数	パーミッションの種類	出現回数が 5 位までのパーミッション（出現回数）
apache	242	44	ioctl(38), lock(38), getattr(32), open(23), read(23)
chrome	125	29	lock(22), ioctl(21), getattr(19), open(15), read(6)
ftp	178	42	ioctl(26), lock(26), getattr(23), open(17), read(13)
ntp	155	38	lock(25), ioctl(23), getattr(16), open(13), read(13)
samba	349	44	ioctl(49), lock(47), getattr(44), open(37), read(27)
squid	194	37	ioctl(30), lock(30), getattr(22), open(16), read(15)
合計	1,243	66	lock(188), ioctl(177), getattr(156), open(121), read(97)

示す．

表 6 より，必要なアクセスベクタパーミッションは，read，search，getattr，open，および write の数が多いことが分かる．これらは，ディレクトリを探索し，ファイルへ読み書きをする際に必要なパーミッションであり，アプリケーションが業務を行う際に最低限必要なパーミッションであると考えられる．

表 7 より，不要なアクセスベクタパーミッションは，lock，ioctl，getattr，open，および read の数が多いことが分かる．これらは，ファイルを読み込むために必要なパーミッションであり，r_file_perms と定義されているマクロを利用することで，付与される．refpolicy は様々な環境で動作するように設定されているため，評価環境では必要としていないファイルを読み込むアクセスベクタパーミッションが付与されていたと考えられる．提案システムがこれらのポリシを削除することで，不要なポリシを削減できたといえる．

以上のことから，提案システムは様々な環境で動作するように設定されている refpolicy から不要なポリシを削減することで，ポリシを最小特権に近づけることができたといえる．

4.5　オーバヘッド

4.5.1　abrtd の再起動時間

短時間に多くのログを出力させる場合のオーバヘッドを測定するために，デーモンの 1 つである abrtd の再起動処理で評価した．下記の 4 つの条件で abrtd の再起動を 10 回行い，再起動の平均時間を測定した．4 つの条件の測定結果を比較することで，提案手法のオーバヘッドはどのような場合にどの程度発生するかを考察する．

(条件 1)　提案手法適用時，abrt に許可ログを出力させる．
(条件 2)　提案手法適用時，abrt に許可ログを出力させ

ない．

(条件 3)　提案手法非適用時，abrt に許可ログを出力させる．

(条件 4)　提案手法非適用時，abrt に許可ログを出力させない．

(条件 1) は，abrtd に関する許可ログ収集時の abrtd の再起動時間を示す．(条件 2) は，提案システムは動作しているが，許可ログの収集を行っていない場合の abrtd の再起動時間を示す．(条件 3) は，許可ログを出力する設定を行った後，提案システムを停止させた場合の abrtd の再起動時間を示す．(条件 4) は，デフォルトの状態での abrtd の再起動時間を示す．

表 8 に測定結果を示す．(条件 1) と (条件 4) の差である 0.250 秒（110%）が，提案手法のオーバヘッドである．これは，許可ログの出力によるシステムコール回数の増加，ログ変換部までのログの送受信時間，およびコンテキストスイッチの発生が影響していると考えられる．

(条件 1) と (条件 3) の差である 0.133 秒（39%）が，ログ変換部が audispd からログを受信する際に発生するオーバヘッドである．ログの送受信によるオーバヘッドや，コンテキストスイッチの発生が影響しているものと考えられる．

(条件 2) と (条件 4) の差である 0.002 秒（1%）が，許可ログの出力設定をしていない際のオーバヘッドである．ほとんど差がない理由は，ログを受信していないとき，ログ変換部はログの受信待ちの状態であるため，性能に影響を与えないためであると考えられる．

以上のことから，オーバヘッドは許可ログを収集しているときだけに発生する一時的なものであり，実用に耐えうる程度であることが分かる．

また，許可ログはシステムコールの発行を契機に出力されるため，提案システムのオーバヘッドはシステムコールの発行を契機に発生する．本評価では，abrtd の再起動 1 回につき，システムコールは平均 795 回発行された．表 8 より，オーバヘッドは約 250 ミリ秒であり，システムコール 1 回につき，約 0.3 ミリ秒のオーバヘッドがあることが分かる．以上のことから，一定時間内にシステムコールの発行が多い処理ほどオーバヘッドが大きくなると推察できる．

表 8　提案手法のオーバヘッド
Table 8　Overhead of the proposal method.

	提案手法適用時	提案手法非適用時
許可ログの出力あり	(条件 1)　0.477 s	(条件 3)　0.344 s
許可ログの出力なし	(条件 2)　0.229 s	(条件 4)　0.227 s

4.5.2　ApacheBench

よく利用されるサーバプログラムとして Apache を利用して提案システムのオーバヘッドを評価した．Apache は，ファイルアクセスやネットワーク処理をともない，多くのシステムコール処理をともなうサービスの例として選択した．ApacheBench を用いて，Apache 利用時の提案システムのオーバヘッドを測定することで，オーバヘッドによるシステムへの影響を考察する．Apache に対応するモジュールである apache に提案手法を適用して，許可ログを出力させた．許可ログを出力しているときと許可ログを出力していないときの ApacheBench の実行時間を比較することでオーバヘッドを測定した．なお，測定回数はそれぞれ 5 回であり，平均時間を算出した．

表 9 に測定結果を示す．なお，表 9 の n は合計リクエスト数であり，c は同時接続数である．合計リクエスト数が 10,000 で同時接続数が 10 の場合，許可ログの出力によるオーバヘッドは 3.64 秒（70.73%）であり，合計リクエスト数が 10,000 で同時接続数が 100 の場合もほぼ同様のオーバヘッドとなった．

これらのことから，ApacheBench を利用した場合，許可ログの出力は約 70%程度のオーバヘッドであることが分かる．また，表 8 のオーバヘッドを考慮すると，他のケースでも許可ログ出力によるオーバヘッドは 100%前後であると推測できる．

しかし，このオーバヘッドは，ログの出力が特に多いログ収集とポリシ削減期間に生じるもので，この期間は計算機システムの性能よりも，ポリシを削減するために計算機システムが行う処理の許可ログを多数出力させ，収集することが目的である．このため，計算機システムの性能は落ちるものの，サービスが不可能となるほど処理時間が増加しないことから，許容できるオーバヘッドであるといえる．なお，テスト運用期間は，拒否ログのみを出力させるため，ログ収集とポリシ削減期間ほどオーバヘッドは大きくならない．また，実運用期間では，提案システムを停止させ，不要なポリシを削減したポリシを適用して運用するため，オーバヘッドは生じない．

4.6　システム管理者への負担

従来手法と提案手法の不要なポリシを発見し，削除する

表 9　ApacheBench による測定結果
Table 9　Result of ApacheBench.

条件（n = 10,000）	許可ログ非出力時（s）	許可ログ出力時（s）	オーバヘッド（%）
c = 10	5.17	8.81	70.73
c = 100	5.38	9.21	71.33

手順を以下に示し，システム管理者の負担を比較する．

＜従来手法の手順＞

(1) システム管理者がモジュールのソースファイルを参照する．

(2) m4 マクロを解読し，不要なポリシを発見する．

(3) ポリシを削除し，正しく動作するかテストを行う．

(4) 正しく動作した場合，次のモジュールを調査し，正しく動作しなかった場合，削除したポリシを復元し，再設定を行う．

従来手法では，システム管理者がポリシのソースファイルを参照し，アプリケーションの動作を把握したうえで，m4 マクロを解読する必要がある．m4 マクロによるポリシの定義は 1,000 種類以上あり，理解が難しい．このため，ポリシの開発者と同程度の知識が必要となる．また，ポリシのソースファイルはモジュール単位で記述されている．このため，モジュールの数に比例して参照するポリシのソースファイルが増えるため，すべてのモジュールの調査に多くの時間を費やす必要がある．

＜提案手法の手順＞

(1) modify_policy コマンドを使い，1 つ以上のモジュールを調査対象に追加する．

(2) システムが自動で不要なポリシを発見し，ウィンドウを表示する．

(3) システム管理者が，3.3.7 項の情報をもとに，必要なポリシかどうかを判断し，Yes，または No のボタンをクリックする．

(4) Yes をクリックした場合は，システムが自動でポリシを修正し，適用する．No をクリックした場合は何も行わない．

提案手法では，システム管理者がモジュールを調査対象に追加した後，システムが自動で調査対象のモジュールの不要なポリシを発見して，システム管理者に提案する．システム管理者は，提案された情報をもとに判断をするだけであるため，負担が少ない．以上のことから，提案手法はシステム管理者のポリシ設定の負担を軽減できるといえる．

4.7　組み込みシステムでの利用

組み込みシステムで提案システムを利用できることを示すために，シリコンリナックス株式会社の CAT760 を用いて実験を行った．実験は 3.5 節の手順で行い，syslogd に関するポリシの削減を行った．

組み込みシステムで syslogd に関する許可ログを 1 日収集し，ポリシ作成用計算機にログを移した後，不要なポリシを削減した．この結果，syslogd に関するポリシの数は 464 から 128 に減少し，ポリシを削減した割合は，約 72%であった．これは，表 5 の結果とほぼ同等であり，syslogd に関する不要なポリシを削減できたといえる．また，削減したポリシを組み込みシステムに複写することで，動作を確認した．

以上のことから，提案システムは組み込みシステムでも利用可能であるといえる．また，本評価では，syslogd のみを対象としたが，他のサービスを対象として，ログを収集した場合でも，ポリシの削減が可能であると推察できる．

5.　関連研究

アプリケーションに関するログを収集し，ポリシを作成する関連研究として，文献 [10] と文献 [11] がある．文献 [10] はシステムコールが呼ばれたときに，呼び出し元のプログラムやアクセスの対象となるファイルなどを記録するようにカーネルを修正することで，履歴を収集し，ポリシを作成する．問題点として，アクセス許可の内容が統合されているため，粒度が粗いことや，プロセスやファイルごとにラベルを付与しているため，ポリシが膨大になる点がある．文献 [11] は strace を利用して，アプリケーションの振舞いを調査し，既存のポリシから不要なポリシを削減する方式である．しかし，この論文では有用性の評価については述べられていない．

ポリシを記述する工数を削減する研究として，文献 [12] と文献 [13] がある．文献 [12] はシステムを初期化部分とプロトコル処理部分に分け，初期化部分はすべてのアクセスを許可し，プロトコル処理部分は必要なポリシを記述することにより，初期化部分のポリシの記述の手間を削減する．しかし，プロトコル処理部分のポリシはシステム管理者が記述する必要があるため，SELinux のポリシの知識が必要となる．文献 [13] はすべてのアクセスを許可した後，ユーザが指定したアクセスを制限する方式であり，GUI でアクセスの制限を行う．この方法では，システム管理者がアクセス制限を指定する必要があるので，システム管理者がシステムについて詳しい必要がある．

SELinux のポリシのサイズを削減する研究として，文献 [14] と文献 [15] がある．文献 [14] は，携帯電話向けのポリシを作成するために，システムに付与するラベルを trusted_t，untrusted_t，および kernel_t の 3 つにすることで（実際には依存性の関係で 700 程度必要）ラベルの数を削減し，ポリシのサイズを約 90%削減している．文献 [15] は，ポリシを記述する独自の言語である PLEASE を利用して，refpolicy に含まれる重複したラベルを削除することで，ポリシのサイズを削減している．

上記より，関連研究はシステム管理者に多くの知識を要求するものが多い．また，有用性の評価について詳しく述べられていないものが多い．

提案手法は，SELinux が出力するログを利用して，既存のポリシから不要な権限を自動で発見し，ポリシの削除の提案を行う．システム管理者は，与えられた情報をもとに削除の判断をするだけであるため，負担が小さい．また，文献 [11] は提案システムと似ているものの，評価について

述べられておらず，ポリシの作成に必要な情報のすべてを取得したログから得られない点が，提案手法と異なる．

6. おわりに

SELinux のアクセス制御で利用しているポリシには，ポリシの設定の難しさ，最小特権の実現の難しさ，およびメモリ使用量の多さの問題があることを述べた．これらの問題を解決するために，SELinux が出力するログに着目して，利用されていないポリシを自動で削減する方法を提案し，設計と評価結果について述べた．

評価では，配布されているポリシの約 8 割は，評価環境では利用されていないことを示し，検出したポリシを削減することで，最小特権に近づけることと，メモリ使用量の削減を実現した．次に，ログの収集期間は，定期的に行われる処理を考慮して決める必要があることを示した．また，不要なポリシは，ファイルの読み込みに関するものが多いこと，および提案手法のオーバヘッドは約 100%であり，許可ログを収集しているときだけに発生する一時的なものであることを示した．さらに，不要なポリシの検出，提案，および削除を自動化することで，システム管理者の負担を軽減できることを示した．最後に，提案システムは組み込みシステムでも利用できることを示した．

参考文献

[1] NSA: Security-Enhanced Linux, available from ⟨http://www.nsa.gov/research/selinux/⟩.

[2] Nakamura, Y. and Sameshima, Y.: SELinux for Consumer Electronics Devices, *Proc. Linux Symposium* (2008), available from ⟨http://elinux.org/images/8/88/Nakamura-reprint.pdf⟩.

[3] SELinux Policy Editor Project: SELinux Policy Editor, available from ⟨http://seedit.sourceforge.net/⟩.

[4] Nakamura, Y., Sameshima, Y. and Yamauchi, T.: SELinux Security Policy Configuration System with Higher Level Language, *Journal of Information Processing*, Vol.18, pp.201–212 (2010).

[5] Tresys Technology: SELinux Policy IDE (SLIDE), available from ⟨http://oss.tresys.com/projects/slide⟩.

[6] Tresys Technology: SELinux Reference Policy, available from ⟨http://oss.tresys.com/projects/refpolicy⟩.

[7] 海外浩平：Linux のセキュリティ機能：2．SELinux のアーキテクチャとアクセス制御モデル，情報処理学会会誌，Vol.51, No.10, pp.1257–1267 (2010).

[8] 中村雄一，山内利宏：Linux のセキュリティ機能：3．セキュリティポリシー設定簡易化手法，情報処理学会会誌，Vol.51, No.10, pp.1268–1275 (2010).

[9] GNU Project: GNU m4, available from ⟨http://www.gnu.org/software/m4/m4.html⟩.

[10] 原田季栄，保理江高志，田中一男：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，*Network Security Forum 2003* (2003), 入手先 ⟨http://sourceforge.jp/projects/tomoyo/document/nsf2003.pdf⟩.

[11] Marouf, S., Phuong, D.M. and Shehab, M.: A Learning-Based Approach for SELinux Policy Optimization with Type Mining, *Proc. 6th Annual Workshop on Cyber Security and Information Intelligence Research* (*CSIIRW '10*) (2010).

[12] Yokoyama, T., Hanaoka, M., Shimamura, M., Kono, K. and Shinagawa, T.: Reducing Security Policy Size for Internet Servers in Secure Operating Systems, *IEICE Trans. Information and Systems*, Vol.E92-D, No.11, pp.2196–2206 (2009).

[13] 桝本　圭，村田裕之：SELinux のポリシ作成時間を短縮する一考察，Japan Linux Conference 抄録集，Vol.1 (2007).

[14] Muthukumaran, D., Sawani, A., Schiffman, J., Jung, B.M. and Jaeger, T.: Measuring integrity on mobile phone systems, *Proc. 13th ACM SACMAT*, pp.155–164 (2008).

[15] Quigley, D.P., Johnson, K.K. and Zadok, E.: PLEASE: Policy Language for Easy Administration of SELinux, *Computer Security Conference 2008* (2008).

矢儀 真也　（学生会員）

2011 年岡山大学工学部情報工学科卒業．同年同大学大学院自然科学研究科博士前期課程入学．現在，在学中．コンピュータセキュリティに興味を持つ．

中村 雄一　（正会員）

1999 年東京大学理学部物理学科卒業，2001 年同大学大学院理学系研究科物理学専攻修士課程修了．2001 年より日立ソフトウェアエンジニアリング（現日立ソリューションズ）に勤務．The George Washington University に社費留学，2006 年 Master of Science in Computer Science 取得．組み込みソフト・M2M の研究開発に従事．2010 年度情報処理学会論文賞．

山内 利宏　（正会員）

1998 年九州大学工学部情報工学科卒業．2000 年同大学大学院システム情報科学研究科修士課程修了．2002 年同大学院システム情報科学府博士後期課程修了．2001 年日本学術振興会特別研究員（DC2）．2002 年九州大学大学院システム情報科学研究院助手．2005 年岡山大学大学院自然科学研究科助教授．現在，同准教授．博士（工学）．オペレーティングシステム，コンピュータセキュリティに興味を持つ．2010 年度情報処理学会論文賞受賞．電子情報通信学会，ACM，USENIX 各会員．