情報処理学会論文誌 Vol.61 No.9 1519–1530 (Sep. 2020)

SELinux CILを利用した不要なセキュリティポリシ削減手法

齋藤 凌也[1] 山内 利宏[1,a)]

受付日 2019年12月9日, 採録日 2020年6月1日

概要：SELinuxを利用する際，個々のシステムには必要のない権限が許可されている汎用的なセキュリティポリシを利用する場合が多い．そこで，我々は，汎用的なポリシから不要なポリシを削減する手法を提案した．しかし，従来手法は，ポリシのソースファイルがない場合は適用不可であり，アトリビュートを含むポリシに対応していない．また，ログ収集とポリシ削減期間において，同一ポリシにより許可されたアクセスのログが出力され続け，オーバヘッドが大きい．さらに，baseモジュール内の不要なポリシを削減できない．本論文では，これらの問題に対処するため，従来手法を拡張した手法を提案する．提案手法では，中間言語であるSELinux CILで記述されたファイルに着目し，ポリシを削減する．また，アトリビュートを考慮し，細粒度でポリシを削減する．さらに，1度変換されたポリシに関する`auditallow`文をポリシから削減することで，オーバヘッドを抑える．最後に，`typeattributeset`宣言を置き換えることで，baseモジュールに変更を加えずに不要なポリシを削減する．本論文では，ポリシ削減の評価やApache Struts2の脆弱性を用いた攻撃防止実験により，提案手法の有効性を示す．

キーワード：SELinux，セキュリティポリシ，SELinux CIL，強制アクセス制御，最小特権

Method to Reduce Redundant Security Policy Using SELinux CIL

Ryoya Saito[1] Toshihiro Yamauchi[1,a)]

Received: December 9, 2019, Accepted: June 1, 2020

Abstract: Application of SELinux involves incorporating a general security policy that permits redundant privileges for individual systems. Hence, we previously proposed a method that eliminates redundant policies from the general policy. However, the said method cannot be applied when there is no policy source file or policies include an attribute that is not supported. During eliminating policies period, the log of access permitted by a particular policy is continually produced as an output, and the associated overhead is large. Furthermore, redundant policies in the base module cannot be eliminated. To address these issues, we propose a new method that extends the previously proposed method. The new method involves the processing of files written in SELinux CIL (an intermediate language) for eliminating redundant policies. Additionally, the new method considers attributes and eliminates policies with fine granularity. The overhead is reduced by eliminating the `auditallow` statement associated with the policy once converted to the policy format from the policy. Furthermore, by replacing the `typeattributeset` statement, redundant policies can be eliminated without modifying the base module. In this study, the effectiveness of our method is demonstrated through evaluation of policy elimination and through an attack prevention experiment by incorporating the vulnerabilities in Apache Struts2.

Keywords: SELinux, security policy, SELinux CIL, mandatory access control, least privilege

1. はじめに

サイバー攻撃により，システムの改変や機密情報漏洩などの被害が発生している [1]．これらの被害を抑制する手

[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
[a)] yamauchi@cs.okayama-u.ac.jp

段として，Security-Enhanced Linux [2]（以降，SELinux）の利用がある．SELinux が持つ代表的なアクセス制御機能として，強制アクセス制御（MAC: Mandatory Access Control）がある．MAC とは，アクセス権限の管理者が定めたセキュリティポリシ（以降，ポリシ）のもとで，すべてのファイルやプログラムなどのアクセス権限が一元的に管理されるアクセス制御方式である．このアクセス制御機能により，SELinux は高いセキュリティを実現している．

しかし，ポリシは記述の難しさから，配布されている汎用的なポリシをそのまま利用する場合が多い．ここで，汎用的なポリシには，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（以降，不要なポリシ）が含まれている可能性が高い．そこで，我々は，実行対象のシステムに合わせて，汎用的なポリシから不要なポリシを自動で削減する手法 [3]（以降，従来手法）を提案した．従来手法は，監査システムが出力するログ（以降，監査ログ）から変換したポリシと運用中のポリシを比較することで，不要なポリシを削減する．しかし，従来手法には，4 つの問題点が存在する．まず，従来手法は，ポリシのソースファイルがない場合，ポリシ削減手法を適用できない．また，従来手法はアトリビュートを含むポリシに対応できない．ここで，アトリビュートとは，複数のドメインやタイプを束ね，グループ化したものである．さらに，同一ポリシにより許可されたアクセスのログが出力され続け，ログ収集とポリシ削減期間のオーバヘッドが大きい．最後に，base モジュール内に不要なポリシがある場合，削減できない．

これらの問題点に対処するために，我々は従来手法を拡張し，SELinux CIL で記述されたファイル（以降，cil ファイル）を利用した不要なセキュリティポリシ削減手法（以降，提案手法）を提案する．これにより，ポリシのソースファイルが含まれていないパッケージを利用する場合でもポリシを削減できる．また，提案手法は，従来手法が対応できないアトリビュートを含むポリシ削減に対応し，従来手法に比べてより細粒度で不要なポリシを削減できる．さらに，1 度変換されたポリシに関する `auditallow` 文をポリシから削除することで，ログの出力を抑える．これにより，提案手法適用によって発生するオーバヘッドが徐々に抑えられていく．最後に，提案手法は，`typeattributeset` 宣言を `allow` 文に置き換えることで，base モジュールに対して変更を加えることなく，不要なポリシを削減できる．

本論文では，従来手法の問題点とそれに対する対処について述べ，ポリシ削減量の評価や Apache Struts2 脆弱性を用いた評価により，提案手法の有効性を示す．

2. SELinux と不要なポリシ削減手法

2.1 SELinux のアクセス制御方式

SELinux は，National Security Agency を中心とするコミュニティで開発されている Linux カーネルのセキュリティ拡張機能である．SELinux の代表的なアクセス制御機能の MAC は，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などのアクセス権限が一元的に管理されるアクセス制御方式である．アクセス制御の設定変更は，アクセス権限の管理者のみが行うことができ，リソースの所有者は，アクセス制御の設定を自由に変更できない．SELinux は，Type Enforcement，Role Based Access Control，および Multi-Level Security などのアクセス制御機能により，高いセキュリティを実現している．

2.2 SELinux のセキュリティポリシ

2.2.1 SELinux CIL

SELinux CIL とは，高水準言語とバイナリのポリシとの中間言語となるように設計された言語 [4] であり，SELinux Userspace（SELinux のライブラリやツール）2.4 から取り込まれている [5]．SELinux CIL におけるポリシの文法を以下に示す．

(`rule_name domain type` (`class` (`av`)))

SELinux のポリシは，以下の 5 つの要素で構成される．

(1) アクセスルール（`rule_name:allow`, `auditallow` など）
(2) ドメイン（`domain`）
(3) タイプ（`type`）
(4) オブジェクトクラス（`class`）
(5) アクセスベクタパーミッション（`av`）

アクセスルールの `allow` はアクセスを許可することを意味する．また，`auditallow` は監査ログのうち，アクセスを許可した際のログ（以降，許可ログ）を出力させることを意味する．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リソースのラベルである．ドメインやタイプは，複数束ねてグループ化でき，束ねたものをアトリビュートという．アトリビュート宣言の文法を以下に示す．`attribute` 宣言でアトリビュートを定義し，`typeattributeset` 宣言で，複数のラベルをグループ化する．

(`typeattribute` attribute_id)

(`typeattributeset` attribute_id (label_id ...))

オブジェクトクラスは，ファイルやディレクトリ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は，読み取り権限や書き込み権限のようなアクセスパーミッションであり，オブジェクトごとに定義されている．

2.2.2 Reference Policy

Fedora や CentOS では，Reference Policy [6]（以降，refpolicy）という汎用的なポリシが利用されている．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．

refpolicy を利用する方法として，ソースファイルをダウンロードする方法と，提供されているパッケージを利用する方法がある．ソースファイルをダウンロードする方法では，ダウンロードした後，コンパイルし，システムに適用する必要がある．

ソースファイルは，中間ファイルを経て，ロード可能なモジュールファイル（ポリシパッケージファイル）にコンパイルされる．このポリシパッケージファイルは高水準言語（hll）ファイルと呼ばれ，cil ファイルに変換される．最後に，すべての cil ファイルをまとめて，バイナリのポリシにコンパイルする．

提供されているパッケージを利用する方法では，バイナリのポリシ，hll ファイル，cil ファイル，およびその他の設定ファイルがパッケージとして提供されており，ポリシを切り替えるだけで利用できる．

2.3 ポリシの問題点

SELinux のポリシ記述を難しくする要因として，アクセスルールの総数と記述がある [7]．SELinux のポリシはホワイトリスト方式を採用しているため，アプリケーションを正常に動作させるためには多くのアクセスルールが必要となる．また，SELinux のパーミッションは，システムコールの観点から設計されているため，Linux カーネルの知識が必要になる．ここで，パーミッションの種類は 700 を超えるため，アクセスルールの数が増大する．このため，ポリシは記述が難しい．

ポリシ記述の難しさから，汎用的なポリシを利用する場合が多い．しかし，汎用的なポリシは，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，個々のシステムには必要のない権限を許可している可能性がある．また，利用しているアプリケーションでも，多くの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，セキュリティが低下する可能性が高い [8]．ここで，最小特権とは，各プロセスに用途に合った必要最小限のアクセス権限のみを与えることである．

IoT 機器に利用されている OS の約 86%が Linux であり [9]，提供する機能が限られていることから，SELinux を適用し，最小限のポリシでセキュリティを強固にできる可能性がある．一方で，Fedora 27 において，デフォルトのポリシのメモリ使用量は約 3.7 MB である．IoT 機器のようにメモリサイズが限られている場合には，必要最小限のポリシを作成し，メモリ使用量を削減する必要がある．

2.4 不要なポリシ削減手法（従来手法）

ポリシの問題点を解決するため，我々は，実行対象のシステムに合わせて，汎用的なポリシから不要なポリシを自動で削減する手法を提案した [3]．従来手法では，収集した許可ログをポリシの形式に変換し，運用中のポリシと許可ログから変換したポリシを比較することで，運用中のポリシにのみ存在するポリシを不要なポリシとして削減する（以降，ポリシ削減機能）．監査ログには，ポリシの形式に変換するために必要な情報であるドメイン，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の情報が含まれている．また，誤って必要なポリシを削除した場合，誤って削除したポリシを発見し，復元する（以降，ポリシ復元機能）．ポリシ復元機能も，SELinux がアクセスを拒否した際に出力する監査ログ（以降，拒否ログ）を利用して実現される．

従来手法の利用形態は，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．ログ収集とポリシ削減期間では，ポリシ削減機能を利用し，この期間中に利用されなかったポリシを不要なポリシとして削減する．テスト運用期間では，ポリシ復元機能を利用し，誤って削除してしまったポリシを復元する．

3. SELinux CIL を利用した不要なポリシ削減手法

3.1 従来手法の問題点

従来手法には，以下の問題点が存在する．

（**問題点 1**） ポリシのソースファイルがない場合，ポリシ削減手法を適用できないこと

Fedora や CentOS において，デフォルトで利用されているポリシはコンパイル済みのバイナリ形式であり，ポリシのソースファイルが含まれていない．このため，従来手法で必要となる ⟨ モジュール名 ⟩.tmp ファイルを生成できない．また，提供されているパッケージを利用する場合においても，同様にポリシのソースファイルが含まれていない．これらの環境では，ソースファイルを入手した後，コンパイルし，適用しなければ従来手法を適用できない．

（**問題点 2**） アトリビュートを含むポリシに対応していないこと

従来手法は，アトリビュートを含むポリシに対応できておらず，削除対象外としている．このため，従来手法のポリシ削減は粒度が粗い．

（**問題点 3**） 同一ポリシにより許可されたアクセスのログが出力され続けること

従来手法では，同一ポリシにより許可されたアクセスのログが出力され続けるため，従来手法適用時，ログ収集とポリシ削減期間のオーバヘッドが大きくなる．たとえば，ログ収集とポリシ削減期間において，あるファイルに 10 回アクセスした場合，そのファイルに対してアクセスを行ったという許可ログが 10 回出力される．これらの許可ログから変換されるポリシは 1 種類であるため，2 回目以降の許可ログの出力は不要である．

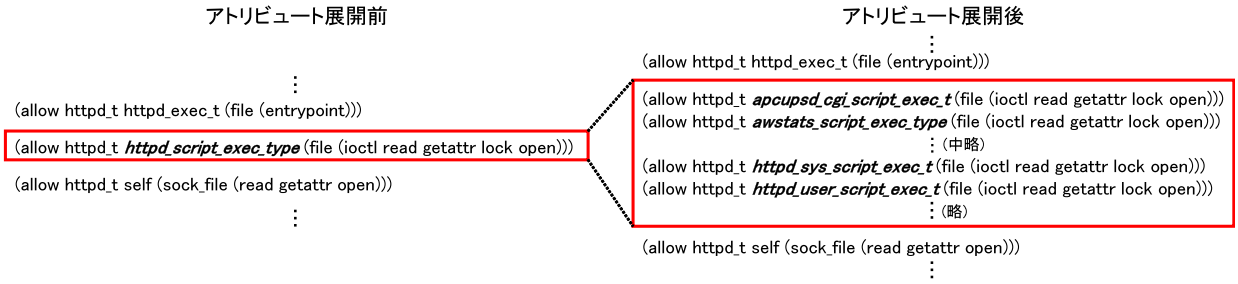

図 2 アトリビュート展開
Fig. 2 Attribute expansion.

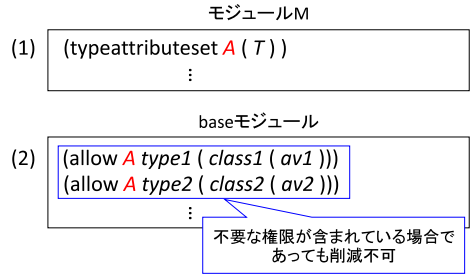

図 1 削減不可なポリシの例
Fig. 1 Example of policy that cannot be deleted.

図 3 auditallow 文の削除
Fig. 3 Delete auditallow statement.

(問題点 4) base モジュール内に存在する不要なポリシを削減できないこと

従来手法は，汎用的なポリシに含まれるモジュールのうち，base モジュールを削減対象外としている．これは，base モジュールは，システムに必須のモジュールであり，大きな変更を加えるべきではないためである．ここで，以下を満たす場合，不要なポリシを削減できない．削減不可な例を**図 1** に示す．

(1) タイプ T がアトリビュート A に追加されている．

(2) A に関する allow 文が，base モジュール内に存在する．

アトリビュート A に対して権限を付与することは，タイプ T を含む複数のタイプに権限を与えることと同じである．アトリビュート A に関する allow 文が base モジュール内に存在し，その allow 文によってタイプ T に対して不要な権限が付与されている場合，従来手法では，ポリシを削減できない．

3.2 対処

3.1 節で述べた問題点を解決するため，我々は，従来手法を拡張したポリシ削減手法を提案する．それぞれの問題点に対する対処を以下に示す．なお，それぞれの対処は，3.1 節の各問題に対応している．

(対処 1) cil ファイルを利用する．

提案手法では，デフォルトのポリシやパッケージに含まれる cil ファイルを利用する．ログを出力させる際や運用中のポリシと許可ログから変換されたポリシを比較する際は，cil ファイルを利用することで，不要なポリシを削減できる．

(対処 2) アトリビュートを元のタイプに置き換える．

アトリビュートを含むポリシを削減する際に，アトリビュートを元のタイプに置き換え，権限を細分化する（以降，アトリビュート展開）．アトリビュート展開の例を**図 2** に示す．

不要なポリシを削減する際の比較には，アトリビュート展開後のポリシを比較することにより，システムに必要な権限のみを残すことができる．

(対処 3) 許可ログから変換されたポリシのサイズが閾値を超えた際，1 度変換されたポリシに関する auditallow 文をポリシから削除する．

運用中のポリシに含まれる auditallow 文と許可ログから変換されたポリシを比較し，1 度変換されたポリシに関する auditallow 文をポリシから削除する．これにより，削除後は出力されなくなるため，ログの出力を抑えることができる．削除する例を**図 3** に示す．

図 3 に示す例では，ドメイン foo_t，タイプ bar_t，オブジェクトクラス file，アクセスベクタパーミッション read と write が 1 度ポリシの形式に変換されているため，運用中のポリシからそれらに関する auditallow 文を削除し，ログの出力を抑えている．

(対処 4) typeattributeset 宣言を allow 文に置き換える．

（問題点 4）は，タイプ T をアトリビュート A に加え，base モジュール内で A に対して権限を付与することで発生する．この問題に対処するために，タイプ T をアトリビュート A に加えずに，別に allow 文を定義するように従来手法を拡張する．具体的には，typeattributeset 宣

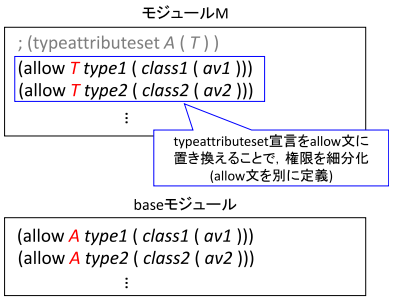

図 4 typeattributeset 宣言の置き換えの例
Fig. 4 Example of replacing typeattributeset statement.

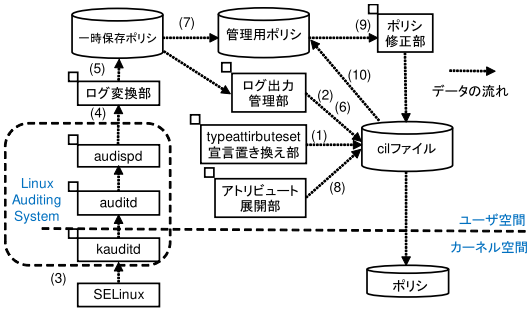

図 5 提案手法の全体像
Fig. 5 Overview of the proposed method.

言を allow 文に置き換える（以降，typeattributeset 宣言の置き換え）．例を図 4 に示す．

図 4 の例では，タイプ T をアトリビュート A に加えず，タイプ T に関する allow 文を別に定義している．これにより，base モジュールに変更を加えることなく，不要なポリシを削減できる．

3.3 提案手法の基本構成

3.3.1 概要

提案手法では，収集したログをポリシの形式に変換し，ログから変換したポリシと cil ファイルを比較することにより，不要なポリシを削減する．また，従来手法と同様に，ポリシ復元機能も備えている．本手法の全体像を図 5 に示す．図 5 の番号は，3.3.5 項のポリシ削減機能の説明に対応している．提案手法を実現するためのシステムは，以下の (1)〜(5) の構成となっている．

(1) ログ変換部
(2) ポリシ修正部
(3) typeattributeset 宣言置き換え部
(4) ログ出力管理部
(5) アトリビュート展開部

ログ変換部は，audit dispatcher daemon（以降，audispd）からログを受信し，ポリシの形式に変換した後，一時保存ポリシに保存する．ポリシ修正部では，管理用ポリシと運用中の汎用的なポリシを比較し，運用中の汎用的なポリシにのみ存在するポリシを不要なポリシとして削減する．(3)〜(5) について以降の項で説明する．なお，一時保存ポリシとは，ログから変換されたポリシを保存しているファイルであり，ログの出力を抑制するために，ログ出力管理部が利用する．管理用ポリシとは，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である．

3.3.2 typeattributeset 宣言置き換え部

typeattributeset 宣言置き換え部では，cil ファイル内に記述されている対象の typeattributeset 宣言を allow 文に置き換える．本構成部を実現するための課題として，置き換える対象を選択する必要がある．すべての typeattributeset 宣言を allow 文に置き換えた場合，ポリシ内の allow 文の数が増大し，ポリシのサイズが膨大になる可能性がある．タイプをほぼ制限を受けないアトリビュート（unconfined）に誤って加えたバグについて報告されている [10] ことから，本研究では，ほとんど制限を受けないアトリビュート（unconfined）を置き換えの対象とした．

3.3.3 ログ出力管理部

ログ出力管理部では，cil ファイルに対し，auditallow 文の追加または削除することで，ログの出力を管理する．auditallow 文の追加は，許可ログを収集するために行う．cil ファイルに対し，auditallow 文を追加することで，SELinux がアクセスを許可した際，許可ログを出力するようになる．auditallow 文の削除は，ログの出力を抑制するために行う．ログ収集とポリシ削減期間において，同一ポリシにより，許可されたアクセスの許可ログが出力され続けることにより，オーバヘッドが大きくなる．このため，ログ変換部が変換したポリシ（一時保存ポリシ）のサイズが閾値を超えた際，1 度変換されたポリシに関する auditallow 文をポリシから削除する．

3.3.4 アトリビュート展開部

アトリビュート展開部では，細粒度でポリシ削減を行うために，アトリビュートを含むポリシをもとのタイプに置き換える．ポリシ修正部は，アトリビュート展開後のポリシと管理用ポリシに保存されたポリシを比較することで，細粒度でポリシを削減できる．

3.3.5 提案手法の処理流れ

以下にポリシ削減機能の処理流れを示す．

(1) 運用中の汎用的なポリシ（cil ファイル）に対し，typeattributeset 宣言の置き換えを実行．
(2) ログ出力管理部が運用中の汎用的なポリシに対し，auditallow 文を追加．
(3) SELinux が許可ログを出力．
(4) ログ変換部が Linux Auditing System の一部である audispd からログを受信．
(5) ログ変換部がログをポリシの形式に変換し，一時保存ポリシに追加．
(6) 一時保存ポリシのサイズが閾値を超えた場合，ログ

出力管理部が一時保存ポリシの内容と運用中の汎用的なポリシを比較し，1度変換されたポリシに関するauditallow文をポリシから削減．

(7) 一時保存ポリシの内容を管理用ポリシに保存．

(8) (3)〜(7)を一定期間繰り返した後，運用中の汎用的なポリシに対し，アトリビュート展開を実行．

(9) ポリシ修正部が管理用ポリシと運用中の汎用的なポリシを比較し，不要なポリシを削減．

(10) 修正したポリシを管理用ポリシに保存．

また，ポリシ復元機能の処理流れを以下に示す．

(1) SELinuxが拒否ログを出力．

(2) ログ変換部がLinux Auditing Systemの一部であるaudispdからログを受信．

(3) ログ変換部がログをポリシの形式に変換．

(4) ポリシ修正部が拒否ログから変換したポリシと管理用ポリシを比較し，以前に削除したポリシか否かを調査．

(5) 以前に削除したポリシであった場合，ポリシの復元を提案．

(6) ポリシを復元後，復元したポリシを管理用ポリシに保存．

3.4 提案手法が削減可能な不要なポリシ

不要な権限が付与される原因として，以下がある．

（原因1） 汎用的なポリシやツールの利用による不要な権限の付与

汎用的なポリシは，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，個々のシステムには必要のない権限を許可している可能性がある．また，利用しているアプリケーションでも，多くの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．

拒否ログからポリシを生成するツールとして，audit2allowコマンドがある．このコマンドは，拒否ログを解析し，アクセス拒否された操作について，アクセス許可を追加するためのポリシを自動生成する．ここで，本来許可してはならない権限まで，ポリシに追加されてしまう可能性がある．

（原因2） アトリビュートに対する不要な権限の付与

アトリビュートは，複数のタイプをまとめてグループ化したものであり，複数のタイプに対して権限を付与できる．一方で，複数のタイプに権限を付与することから，意図せずに不要な権限を一緒に付与する可能性がある．このため，アトリビュートに対して権限を付与する際は，より慎重になる必要がある[11]．

（原因3） 誤ったタイプの割当てによる不要な権限の付与

あるリソースに対して，誤ったタイプを割り当てた場合，不要な権限が付与される可能性がある．たとえば，「ドメインDがタイプTのファイルを読み取り可能」というポリシがあるとする．ここで，ドメインDが読み取る必要のあるファイルF1と読み取る必要のないファイルF2があり，F1とF2に対して，同じタイプTを割り当てた場合，F1とF2の両方をドメインDが読み取ることができる．ドメインDはF2を読み取る必要がないため，これは不要な権限である．

提案手法では，ポリシ削減機能により，allow文に含まれる不要な権限を削減し，（原因1）に対応できる．また，アトリビュート展開，およびtypeattributeset宣言の置き換えにより，アトリビュートを含むポリシにおける不要な権限を削減し，（原因2）に対応できる．一方で，提案手法は，タイプの割当ての正当性を確認していない．たとえば（原因3）の例では，ログ収集とポリシ削減期間中にタイプTが割り当てられたファイルF1をドメインDが読み取った場合，提案手法によって「ドメインDがタイプTのファイルを読み取り可能」というポリシは必要であると判断される．このため，提案手法適用後もドメインDは読み取る必要のないファイルF2を読み取ることもできる．このため，（原因3）には対応できない．

4. 評価

4.1 目的と評価環境

提案手法の有効性を明らかにするために，以下の評価を行った．

（評価1） アトリビュート展開による権限の細分化

アトリビュート展開後のポリシにおいて，ログから変換したポリシとcilファイルを比較することにより，ログから変換したポリシ以外のポリシが削減されているか否かを示す．

（評価2） ポリシの削減量

提案手法を適用することにより，ポリシのサイズ，ラベルの数，モジュールの数，allow文の数，およびパーミッションの数と種類をどの程度削減できるかを示す．

（評価3） Apache Struts2の脆弱性[12]を用いた評価

提案手法が従来手法の（問題点4）に対処できたか否かを示すために，baseモジュール内に不要なポリシが存在するというバグのあるポリシに対して，提案手法を適用することによって，脆弱性による被害を抑制可能であるか否かを評価した．また，ポリシにバグがない場合でも，提案手法を適用することにより，脆弱性の被害を抑制可能であるか否かを明らかにするため，ポリシにバグがない場合もあわせて評価した．

（評価4） 提案手法適用によるアプリケーションの性能への影響

提案手法が従来手法の（問題点3）に対処できたか否かを示すために，提案手法の適用によって発生するオーバヘッドを計測し，アプリケーションへの影響を評価した．

（評価 5） システム管理者への負担

従来手法と提案手法それぞれの導入の手順を比較することで，システム管理者への負担を比較した．

（評価 6） ポリシ復元機能の評価

（評価 2）の後，提案手法によってポリシの復元が行われるか否かを評価した．

（評価 1），（評価 2），（評価 3），および（評価 6）における評価環境は，カーネルは，Linux 3.10.0-514.el7.x86_64（CentOS 7），ポリシのバージョンは selinux-policy-targeted-3.13.1-102.el7 である．また，評価対象の計算機では，HTTP サーバ, SFTP サーバ, およびサーブレットコンテナ（Tomcat）が動作している．1 度変換されたポリシに関する auditallow 文は，一時保存ポリシのサイズが 1 MB を超えていた場合に削除するものとする．さらに，selinux-policy-targeted-3.13.1-102.el7 で宣言されているアトリビュートのうち，置き換えの対象となったアトリビュートは 13 個である．削減対象のモジュールすべてに auditallow 文を追加した後，計算機を再起動し，評価した．なお，ログ収集とポリシ削減期間を 2 日間とした．

全部で 403 個のモジュールのうち，削減対象としたモジュールは 376 個である．削減対象としていないモジュールは，base モジュール，監査システムに関するモジュール，audispd が起動する前に発生する操作に関するポリシを含むモジュールである．

監査システムに関するモジュールに auditallow 文を追加した場合，ログが出力された際，ログの出力にともなってさらにログが生成され，無限にログが生成される．これにより，システムが動作を停止する可能性がある．また，ポリシ削減手法は，audispd から受信したログを用いるため，audispd 起動前に発生する操作を認識できない．audispd が起動する前に発生する操作に関するポリシを不要なポリシとして削減してしまう恐れがあるため，削減対象から外した．

4.2 アトリビュート展開による権限の細分化

4.2.1 評価内容

apache モジュールで定義されているアトリビュートを含む allow 文の例を以下に示す．

```
(allow httpd_t file_type (dir (getattr
search open)))
```

file_type は，794 のタイプを束ねているアトリビュートである．本評価では，この allow 文に着目し，提案手法適用後，ログから変換したポリシと cil ファイルを比較することにより，ログから変換したポリシ以外のポリシが削減されているか否かを評価した．

4.2.2 評価結果と考察

提案手法適用後のアトリビュート file_type を含む allow 文を**図 6** に示す．提案手法適用後，上記の allow 文は 23 のタイプに対してアクセスを許可する allow 文に置き換えられた．

```
(allow httpd_t httpd_config_t (dir (getattr open search)))
(allow httpd_t httpd_log_t (dir (getattr search)))
(allow httpd_t httpd_sys_content_t (dir (getattr search)))
(allow httpd_t lib_t (dir (getattr search)))
(allow httpd_t var_t (dir (getattr search)))
(allow httpd_t httpd_modules_t (dir (search)))
(allow httpd_t httpd_var_run_t (dir (search)))
(allow httpd_t bin_t (dir (search)))
⋮
```

図 6 置き換えられた allow 文（一部）
Fig. 6 Replaced allow statement.

httpd_config_t には 3 種類の権限を与えている．また，httpd_log_t，httpd_sys_contents_t，lib_t，および var_t には 2 種類の権限，その他タイプには 1 種類の権限をそれぞれ残し，不要な権限を削除していることが分かる．このことから，アトリビュートを含む allow 文を削減する際，各タイプに必要な権限を残し，不要な権限を削除していることが分かる．なお，上記以外のアトリビュートを含む allow 文の削減においても，同様の結果が得られた．また，図 6 の allow 文とログから変換したポリシの内容を比較した結果，ログから変換したポリシ以外が削除できていたことを確認した．

以上より，提案手法は，アトリビュートを含む allow 文を削減する際，アトリビュートを含むポリシから不要な権限のみを削減でき，システムに必要な権限のみを残し，最小特権に近づけることができるといえる．これにより，提案手法は，従来手法の（問題点 2）に対処したといえる．

4.3 ポリシの削減量

4.3.1 評価内容

allow 文の数，ポリシのサイズ，ラベルの数，およびモジュールの数を評価した．ポリシのサイズは，メモリ使用量の削減についての評価である．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，allow 文の数，およびパーミッションの数と種類は，最小特権に近づけているかを示すための評価である．

4.3.2 評価結果

まず，評価対象の計算機で運用しているプログラムのポリシに着目する．HTTP サーバ，SFTP サーバ，サーブレットコンテナに対応する 3 つのモジュールの allow 文の削減数を**表 1**，パーミッションの個数と種類の削減数を**表 2** に示す．なお，合計は，重複を除いた値となっている．表 1 の合計の削減数から，本環境では，3 つのモジュールで定義されている 9 割以上の allow 文は実際には利用されていないことが分かる．また，表 2 の合計の削減数から，

表 1 3 つのモジュールにおける allow 文の削減数
Table 1 Number of allow statements reduced in the three modules.

モジュール	デフォルト	アトリビュート展開後 (N1)	削減後 (N2)	削減数 (N1-N2)
apache	1,201	151,761	54	151,707 (99.96%)
ssh	709	14,336	87	14,249 (99.39%)
tomcat	145	71,377	74	71,303 (99.90%)
合計	2,051	237,462	213	237,249 (99.91%)

表 2 3 つのモジュールにおけるパーミッションの個数と種類
Table 2 Number and kinds of permissions in the three modules.

モジュール	デフォルト		アトリビュート展開後 (A1)		削減後 (A2)		削減数 (A1-A2)	
	個数	種類	個数	種類	個数	種類	個数	種類
apache	6,342	149	1,435,432	240	117	34	1,435,315 (99.99%)	206 (85.83%)
ssh	3,411	122	28,132	122	191	49	27,941 (99.32%)	73 (59.84%)
tomcat	1,067	121	1,128,255	240	206	31	1,128,049 (99.98%)	209 (87.08%)
合計	10,811	166	2,591,801	240	509	58	2,591,292 (99.98%)	182 (75.83%)

表 3 ポリシの削減量
Table 3 Amount of reduced policy.

	デフォルト	削減後	削減量 (%)
allow 文の数	101,056	15,385	84.8
ポリシのサイズ (B)	3,704,933	1,053,217	71.6
ラベルの数	4,729	2,044	56.8
モジュールの数	403	107	73.4

本環境では，3 つのモジュールで定義されている約 76%の種類のパーミッションは利用されていないことが分かる．

次に，ポリシ全体の allow 文，サイズ，ラベルの数，およびモジュールの数に着目する．**表 3** に評価結果を示す．なお，デフォルトはアトリビュート展開前の値，削減後はアトリビュート展開して不要なポリシを削減した後の値である．

ここで，allow 文は，不要な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削減により削減された．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削減により，約 72%削減された．ラベルは，ラベルを定義しているモジュールの削除によって削減された．allow 文の数から，本環境では，各モジュールで定義されている allow 文の約 85%は，実際には利用されておらず，不要な権限であることが分かる．ここで，本環境で利用されていた 107 のモジュールにのみ着目した場合，デフォルトの allow 文の数は 36,471 であった．このことから，本環境で利用されていた 107 のモジュールにのみ着目した場合，allow 文の削減量は，約 57.82%となった．

また，本環境では，ラベルは約 57%，モジュールは約 73%が実際には利用されていないことが分かる．たとえば，本環境では，wireshark モジュール（パケットキャプチャツールのモジュール）や wine モジュール（Linux 上で Windows のアプリケーションを動作させるためのプログラムのモジュール）が不要なモジュールとして削除された．

4.3.3 考察

本評価では，多くの allow 文，ラベル，およびモジュールが削減された．これは，refpolicy は汎用的なポリシであり，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を多く含んでいるため，多くのポリシが削減されたと推察できる．また，本環境で利用されていたモジュールにのみに着目した allow 文の削減量の結果から，refpolicy は，多くの環境で問題なく動作するように必要以上の権限が与えられているため，利用されているモジュールであっても，多くのポリシが削減されたと推察できる．

組み込み機器や IoT 機器のようにメモリサイズが限られているデバイスでは，メモリサイズは多くの場合，64 MBytes よりも小さく，スワップが有効でないため，デフォルトのポリシのサイズは大きく，許容できない [13]．メモリを割り当てることができない場合，アプリケーションが正常に動作しなくなる．本環境では，提案手法適用後，ポリシのサイズは約 1 MBytes となった．これは，64 MBytes のメモリを持つデバイスの場合，メモリの約 1.57%程度の消費であり，デフォルトのポリシの約 5.52%の消費と比べて小さい．これにより，提案手法を適用し，不要なポリシを削減することで，メモリサイズが限られているデバイスであっても，SELinux を適用できるようになる可能性がある．

以上のことから，提案手法により，各モジュールに含まれる不要なポリシを削減することで，ポリシのメモリ使用量を削減し，最小特権に近づけることができるといえる．

4.4 CVE-2017-5638 [12] を用いた評価

4.4.1 評価内容

Apache Struts とは，Apache Software Foundation によって開発されている Java の Web アプリケーションを作成

表 4 CVE-2017-5638 を用いた評価結果
Table 4 Evaluation using CVE-2017-5638.

条件	ポリシのバグ	提案手法	攻撃防止の可否
（条件 1）	あり	適用前	攻撃防止失敗（tomcat の権限で任意のコマンドを実行可能）
（条件 2）		適用後	攻撃防止成功（任意のコマンドの実行に失敗）
（条件 3）	なし	適用前	権限内に制限（ポリシの範囲内で被害が発生）
（条件 4）		適用後	攻撃防止成功（任意のコマンドの実行に失敗）

するためのソフトウェアフレームワークである．Apache Struts2 には，「Jakarta Multipart parser」のファイルアップロード処理に起因するリモートで任意のコードが実行される脆弱性（CVE-2017-5638）が存在する [12]．また，ドメイン `tomcat_t` をほとんど制限を受けないアトリビュート（`unconfined`）に誤ってタイプを割り当てるバグが存在しており [10]，このバグを含む汎用的なポリシ（selinux-policy-targeted-3.13.1-102.el7）とバグが修正された汎用的なポリシ（selinux-policy-targeted-3.13.1-166.el7）の 2 つを評価に用いた．本評価では，エクスプロイトコード [14] を用いて，上記のポリシを利用しているシステムに対して，任意のコマンド（`whoami` コマンドによるユーザ名の表示，`cat` コマンドによるファイルの閲覧）を実行可能か否か実験した．

4.4.2 評価結果

評価結果を**表 4** に示す．提案手法適用前の（条件 1）と（条件 3）では，`whoami` コマンドを実行できた．また，`cat` コマンドによるファイルの閲覧に関して，（条件 1）は，ポリシのバグにより，tomcat の権限で任意のファイルを閲覧できた．（条件 3）では，ポリシのバグが修正されているため，任意のファイルにアクセスできないものの，ポリシの範囲内でファイルを閲覧できた．

一方で，提案手法適用後の（条件 2）と（条件 4）では，任意のコマンドの実行を防止し，攻撃を防止できた．攻撃を防止した際にドメイン `tomcat_t` がタイプ `shell_exec_t` のファイル（`bash`）の実行（`execute`）に失敗したことを示す拒否ログが出力された．

4.4.3 考察

ポリシにバグがある（条件 2）の結果から，base モジュール内に不要な権限がある場合でも，提案手法を適用することで脆弱性による被害を抑制できる可能性があるといえる．また，ポリシにバグがない（条件 4）の結果から，バグのないポリシの場合であっても，提案手法を適用することで，脆弱性による被害を抑制できる可能性があるといえる．

提案手法が `typeattributeset` 宣言を置き換え，ポリシ削減を行った結果を**図 7** に示す．図 7 から，

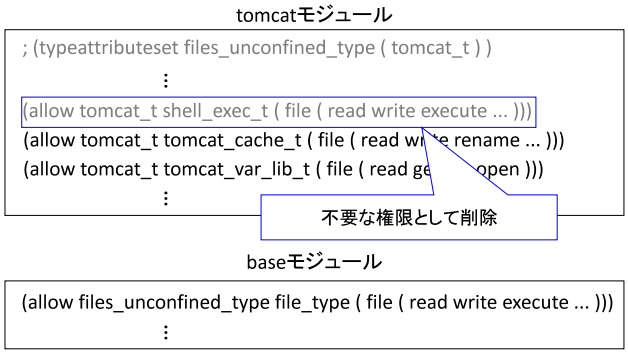

図 7 typeattributeset 宣言の置き換えの結果
Fig. 7 Result of replacing `typeattributeset` Statement.

`typeattributeset` 宣言の置き換えにより，不要な権限を削減したことで，エクスプロイトコード実行中に SELinux によるアクセス拒否が発生し，任意コードの実行が失敗したと推察できる．

以上のことから，提案手法を適用することで，base モジュール内に不要なポリシが存在する場合であっても，`typeattributeset` 宣言の置き換えにより，不要なポリシを削減し，最小特権に近づけることができるといえる．このことから，提案手法は，従来手法の（問題点 4）に対処したといえる．

4.5 提案手法適用によるアプリケーションの性能への影響

4.5.1 評価内容

提案手法が従来手法の（問題点 3）に対処できたか否かを示すために，提案手法適用によるアプリケーションの性能への影響を評価した．アプリケーションとして，許可ログを出力していない期間と許可ログを出力している期間における Web サーバの性能を測定した．ここで，許可ログを出力している期間は，以下の 2 つの条件でそれぞれ測定した．

（条件 1） `auditallow` 文を追加した直後
（条件 2） 1 度変換されたポリシに関する `auditallow` 文をポリシから削除した直後（1 回目の `auditallow` 文の削減処理後）

性能を測定するために Web サーバに対するすべてのリクエスト完了に要した処理時間を比較することで，アプリケーションの性能への影響を考察する．本評価では，Web サーバは Apache 2.4.29，ベンチマークツールは ApacheBench 2.3 を用いた．10 KB のファイルに対し，合計リクエスト 100,000 で，同時接続数が 10 の場合を 5 回測定し，すべてのリクエスト完了に要した処理時間の平均を算出した．また，サーバ側の評価環境は，カーネルは，Linux 4.13.16-302.fc27.x86_64（Fedora 27），CPU は Intel(R) Core(TM) i5-6500 3.20 GHz，メモリは 4 GB，ポリシのバージョンは selinux-policy-targeted-3.13.1-283.17.fc27 である．クライアント側の環境は，CPU は Intel(R) Core(TM) i7-6700

表 5 すべてのリクエスト完了に要した処理時間（単位：s）
Table 5 Processing time required to complete all requests (s).

許可ログ非出力時 ($t1$)	許可ログ出力時（条件 1）($t2$)	許可ログ出力時（条件 2）($t3$)	オーバヘッド ($t2-t1$)	オーバヘッド ($t3-t1$)
11.579	57.111	11.519	44.532 (393.23%)	−0.06 (−0.52%)

3.40 GHz，メモリは 8 GB，カーネルは Linux 4.4.0 である．Apache に対応するモジュールである apache モジュールに提案手法を適用し，許可ログを出力させた．

4.5.2 評価結果と考察

測定結果を**表 5** に示す．許可ログを出力させることによるオーバヘッドは 44.532 秒（393.23%）となった．また，許可ログ出力時（条件 2）の測定結果からは，提案手法非適用時と同等の処理時間となっていることが分かる．これは，1 度変換されたポリシに関する `auditallow` 文をポリシから削除し，同一ポリシにより許可されたアクセスのログが出力されなくなったためであると推察できる．

従来手法では，ログ収集とポリシ削減期間中，オーバヘッドがつねに発生する．一方，提案手法では，提案手法適用直後は，計算機の性能が落ちるものの，`auditallow` 文を削除する処理により，ログの出力を抑制し，オーバヘッドが減少していくと推察できる．このことから，従来手法の（問題点 3）に対処したといえる．

4.6 システム管理者への負担

従来手法と提案手法のそれぞれを導入するまでの手順を以下に示す．この評価では，文献 [15] で説明されている Fedora の例で説明する．

＜従来手法の導入までの手順＞
(1) ポリシのソースファイルをダウンロード
(2) ソースファイルに Fedora 用のパッチを適用
(3) 設定ファイルなどを適切なディレクトリに配置
(4) ポリシの設定ファイルを記述
(5) ポリシをコンパイル
(6) 利用するポリシを変更

(3) で，適切なディレクトリに配置できていない場合，SELinux によるアクセス拒否が発生し，システムが動作を停止する可能性がある．(4) では，ポリシ名や，ポリシをモジュール化するかどうかなどの設定を記述する．このため，これらの操作を行うためには，SELinux のポリシに関する知識が必要となる．

＜提案手法の導入までの手順＞
(1) Fedora 用のポリシパッケージをダウンロード
(2) 利用するポリシを変更

提案手法では，ポリシのソースファイルがない場合においても，cil ファイルを利用することで，ポリシを削減できる．このため，従来手法のように，ポリシをソースファイルからコンパイルする必要がない．また，SELinux のポリシに関する知識は少なくて済む．

表 6 ポリシの復元量
Table 6 Amount of restored policy.

モジュール	復元した `allow` 文の数	パーミッションの個数	パーミッションの種類
apache	4	7	5
crond	29	71	26
合計	33	78	26

以上から，提案手法は，ポリシのソースファイルがない場合においても，cil ファイルを利用してポリシを削減でき，従来手法に比べ，導入が容易であり，システム管理者の負担を軽減できるといえる．これにより，提案手法は，従来手法の（問題点 1）に対処したといえる．

4.7 ポリシ復元機能の評価

4.7.1 評価内容

（評価 2）の後，提案手法によってポリシが復元されるか否か評価した．不要な権限を削減し，ログ収集とポリシ削減期間からテスト運用期間に移行した直後の計算機を利用した．この際，拒否ログが出力された場合，ポリシ復元機能により，ポリシを復元する．なお，テスト運用期間を 2 日間とした．

4.7.2 評価結果と考察

復元されたポリシを**表 6** に示す．ポリシ復元機能により，apache モジュールと cron モジュール内のポリシが復元された．cron とは，利用者の設定したスケジュールに従って指定されたプログラムを定期的に起動するデーモンである．ログ収集とポリシ削減期間中に cron によるジョブが実行されず，テスト運用期間中に cron によるジョブが実行されたため，cron モジュール内のポリシが多く復元されたと推察できる．このため，文献 [3] で述べているようにログ収集とポリシ削減期間は，定期的に行われる処理を考慮して決める必要がある．

また，復元された cron モジュールのポリシ（一部）を**図 8** を示す．なお，赤色で強調してあるポリシが復元されたポリシである．図 8 のように，提案手法のポリシ復元機能により，アトリビュート `exec_type` をアトリビュート展開した後のポリシが復元された．

以上より，アトリビュート展開後のポリシを復元できる

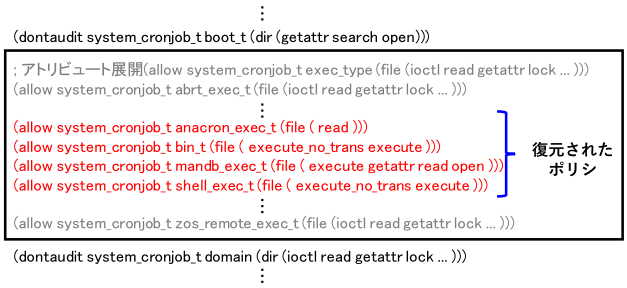

図 8 復元された cron モジュールのポリシ（一部）
Fig. 8 Result of restored `allow` Statement in cron module.

ことから，提案手法は従来手法に比べ，より細粒度でポリシを復元できるといえる．

5. 関連研究

ポリシ作成の工程数を減らす研究として，文献 [16] がある．文献 [16] は，すべてのアクセスを許可するルールをポリシに記述した後，ユーザによって指定された箇所のみのアクセス制限を GUI で行う．これにより，ポリシ作成に必要な前提知識の量と作業量が少なく済むことが期待される．一方で，ユーザの設定し忘れなどにより，アクセス制限が行われていないリソースが存在した場合，ポリシの安全性が低下するという欠点がある．また，ユーザがアクセス制限を指定する必要があるため，設定を行うユーザがシステムに詳しい必要がある．一方で，提案手法では，既存のポリシから自動で不要なポリシを削減する．このため，システムの管理者に必要な知識が少なくて済む．

収集したログからポリシを作成する研究として，文献 [17], [18] がある．文献 [17] は，一定期間システムを稼働させ，プログラムの実行履歴とアクセス要求の情報を収集することによりポリシを作成する．履歴を収集している間にアクセス拒否が発生した場合は無視され，アクセス拒否が発生しないように必要なアクセスルールをポリシに追加する．一方，提案手法は，不要なポリシを削減することを目的としており，許可ログを収集している期間にポリシに記述されていない操作が行われた場合はアクセスを拒否する．このため，元々のポリシに記述されていないアクセスルールが追加されることはない．

文献 [18] では，監査ログとポリシを解析し，SEAndroid のポリシを洗練化するプラットフォームを提案している．このプラットフォームでは，半教師あり学習を利用して，監査ログからポリシを作成し，ポリシの開発に利用される．一方，提案手法は，許可ログを利用して，既存のポリシから不要なポリシを削除することで，ポリシを最適化する．

ポリシの攻撃面分析を行う研究として，文献 [19] がある．文献 [19] では，Android の機能テストからドメイン知識を体系的に抽出する．抽出したドメイン知識から，正当化できないアクセスルールを明らかにし，グラフとして視覚化する．ポリシエンジニアは，グラフを活用し，ポリシを修正することで，ポリシの開発，洗練に利用する．このため，ポリシに関する詳しい知識が必要となる．一方で，提案手法は，ポリシに関する知識は少なくて済み，不要なポリシを自動で削減する．

6. おわりに

従来手法の 4 つの問題点に対処するため，SELinux CIL を利用した不要なセキュリティポリシ削減手法を提案した．提案手法では，SELinux CIL ファイルを対象とし，アトリビュートを含むポリシの削減に対応できる．さらに，1 度変換されたポリシに関する `auditallow` 文をポリシから削減することで，オーバヘッドを抑える．最後に，`typeattributeset` 宣言を `allow` 文に置き換えることで，base モジュールに対して変更を加えることなく，不要なポリシを削減できる．

評価環境では，汎用的なポリシの約 85%の `allow` 文を削減できることを示した．また，アトリビュート展開により，アトリビュートを含むポリシを削減する際，システムに不要な権限のみを削除できることを示した．Apache Struts2 の脆弱性を用いた評価より，提案手法は，細粒度でポリシ削減を行い，脆弱性を悪用した攻撃を防止できることを示した．性能評価では，提案手法適用直後は，計算機の性能が落ちるものの，`auditallow` 文を削減する処理により，ログの出力を抑制し，オーバヘッドが減少していくことを示した．さらに，提案手法は従来手法に比べ，導入手順が少なく，導入時に必要となる SELinux のポリシに関する知識が少ないことを示した．最後に，提案手法は従来手法に比べ，より細粒度でポリシを復元できることを示した．

謝辞 本研究の一部は，JSPS 科研費 JP19H04111, JP19H05579 の助成を受けたものです．

参考文献

[1] 独立行政法人情報処理推進機構：脆弱性対策情報データベース JVN iPedia の登録状況 [2019 年第 2 四半期（4 月～6 月）]，入手先 ⟨https://www.ipa.go.jp/security/vuln/report/JVNiPedia2019q2.html⟩（参照 2019-08-06）．
[2] Security-Enhanced Linux, available from ⟨https://www.nsa.gov/what-we-do/research/selinux/⟩ (accessed 2017-12-26).
[3] 矢儀真也，中村雄一，山内利宏：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，情報処理学会論文誌コンピューティングシステム（ACS），Vol.5, No.2, pp.63–73 (2012).
[4] MacMillan, K., Case, C., Brindle, J. and Sellers, C.: SELinux Common Intermediate Language Motivation and Design, GitHub, available from ⟨https://github.com/SELinuxProject/cil/wiki⟩ (accessed 2017-06-07).
[5] Moore, P.: The State of SELinux, Linux Security Summit 2015, available from ⟨http://kernsec.org/files/lss2015/lss-state_of_selinux-pmoore-082015-r1.pdf⟩ (accessed 2017-12-13).
[6] TresysTechnology: SELinux Reference Policy, GitHub,

available from ⟨https://github.com/TresysTechnology/refpolicy⟩ (accessed 2017-01-12).
[7] 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).
[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，Vol.51, No.10, pp.1268–1275 (2010).
[9] Costin, A., Zaddach, J., Francillon, A. and Balzarotti, D.: A Large-Scale Analysis of the Security of Embedded Firmwares, *Proc. 23rd USENIX Security Symposium*, pp.95–110 (2014).
[10] Kazuki, O.: Bug 1432083 – tomcat_t domain is in unconfined_domain, Red Hat Bugzilla, available from ⟨https://bugzilla.redhat.com/show_bug.cgi?id=1432083⟩ (accessed 2018-02-19).
[11] Chen, H., Li, N., Enck, W., et al.: Analysis of SEAndroid Policies: Combining MAC and DAC in Android, *Proc. 33rd Annual Computer Security Applications Conference* (*ACSAC' 17*), pp.553–565 (2017).
[12] Common Vulnerabilities and Exposures: CVE-2017-5638, available from ⟨https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5638⟩ (accessed 2018-02-19).
[13] Nakamura, Y., Sameshima, Y. and Yamauchi, T.: Reducing Resource Consumption of SELinux for Embedded Systems with Contributions to Open-Source Ecosystems, *Journal of Information Processing*, Vol.23, No.5, pp.664–672 (online), DOI: 10.2197/ipsjjip.23.664 (2015).
[14] Vex Woo: Apache Struts 2.3.5 < 2.3.31 / 2.5 < 2.5.10 - Remote Code Execution, EXPLOIT DATABASE, available from ⟨https://www.exploit-db.com/exploits/41570/⟩ (accessed 2018-02-19).
[15] Richard Haines: The SELinux Notebook (4th Edition), available from ⟨http://freecomputerbooks.com/books/The_SELinux_Notebook-4th_Edition.pdf⟩ (accessed 2017-04-14).
[16] 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).
[17] 原田季栄，半田哲夫，橋本正樹，田中英彦：アプリケーションの実行状況に基づく強制アクセス制御方式，情報処理学会論文誌，Vol.53, No.9, pp.2130–2147 (2012).
[18] Wang, R., Enck, W., Reeves, D., et al.: EASEAndroid: Automatic Policy Analysis and Refinement for Security Enhanced Android via Large-Scale Semi-Supervised Learning, *Proc. 24th USENIX Security Symposium*, pp.351–366 (2015).
[19] Wang, R., Azab, A.M., Enck, W., et al.: SPOKE: Scalable Knowledge Collection and Attack Surface Analysis of Access Control Policy for Security Enhanced Android, *Proc. 2017 ACM on Asia Conference on Computer and Communications Security* (*ASIACCS'17*), pp.612–624 (2017).

齋藤 凌也 （正会員）

2018 年岡山大学工学部情報系学科卒業．2020 年同大学大学院自然科学研究科博士前期課程修了．コンピュータセキュリティに興味を持つ．

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 JIP Outstanding Paper Award, 2012 年度情報処理学会論文賞等受賞．電子情報通信学会，ACM，USENIX，IEEE 各会員．本会シニア会員．