typeattributeset宣言の置き換えによるSELinuxの不要なポリシの細粒度な削減手法の提案

齋藤 凌也[1]　山内 利宏[1]

概要：SELinux のセキュリティポリシは，記述が難しいという問題がある．このため，配布されている汎用的なポリシを利用する場合が多い．汎用的なポリシは，個々のシステムにおいて必要のない権限を許可している可能性がある．この問題を解決するため，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除する手法を提案した．ポリシ削減手法は，SELinux Common Intermeditate Language という中間言語で記述されたファイルと，SELinux が出力する監査ログを利用して不要なポリシを発見し，削除する．一方で，アトリビュートによって不要な権限が付与されているポリシに対しては，ポリシ削減手法によるポリシ削減が不十分である課題がある．そこで，本研究では，タイプをアトリビュートに加える `typeattributeset` 宣言を `allow` 文に置き換え，ポリシ削減手法適用不可なモジュールから `allow` 文を分離することで，権限を細分化し，細粒度で不要なポリシを削除できるように従来手法を拡張した．本稿では，拡張した提案手法について述べ，実際のポリシを用いた攻撃防止実験やポリシの削減の評価により，提案手法の有用性を示す．

Proposal on Fine-grained Reduction Method of Redundant Security Policy by Replacing typeattributeset Statement

Ryoya Saito[1]　Toshihiro Yamauchi[1]

Abstract: The security policy of SELinux has the problem that it is difficult to write. Policies are often used for distributed generic policies due to difficulty of writing. This may allow for redundant privileges in individual systems. In order to solve this problem, we proposed a method to automatically detect redundant policies and delete them according to the target system. The proposed method detects and deletes redundant policies using SELinux Common Intermediate Language and audit logs output by SELinux. However, there is a possibility that the policy reduction by the proposed method is insufficient for the policy to which the redundant privileges is granted by the attribute. Therefore, the proposed method is extended to subdivide the privileges by replacing `typeattributeset` statement that adds types to attributes with `allow` statement. In this paper, we report the proposed method and its evaluation results.

1. はじめに

攻撃者によって，ソフトウェアの脆弱性が悪用され，情報漏えいやデータ改ざん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，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る．このアクセス制御方式により，SELinux は高いセキュリティを実現している．

しかし，SELinux のポリシは記述が難しいため，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ため，個々のシステムに

[1] 岡山大学 大学院自然科学研究科
Graduate School of Natural Science and Technology, Okayama University

はアクセスを許可する必要のない権限（以降，不要なポリシ）を含んでいる可能性がある．

これらの問題を解決するため，文献 [3, 4] において，SELinux CIL を利用した不要なポリシ削減手法（以降，従来手法）を提案した．従来手法は，SELinux が保持している SELinux Common Intermediate Language（以降，SELinux CIL）という中間言語で記述されたファイルと SELinux がアクセスを許可した際に出力するログ（以降，許可ログ）から変換したポリシを比較することで，不要なポリシを発見し，取り除く．

一方で，従来手法では，以下を満たす場合，不要なポリシを削減できない．

(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，ポリシ削減手法適用不可なモジュール内に存在する

これは，タイプ `T` をアトリビュート `A` に加え，ポリシ削減手法適用不可なモジュール内で `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した．これにより，権限を細分化することができる．本稿では，拡張した提案手法とその評価結果について報告する．

本研究の貢献は，以下に示す通りである．

(1) ポリシ削減手法適用不可なモジュールから `allow` 文を分離することで，権限を細分化し，細粒度で不要なポリシを削減

`typeattributeset` 宣言を `allow` 文に置き換え，ポリシ削減を行うことで，従来手法が対処できなかった不要なポリシ削減が可能となる．

(2) 細粒度なポリシ削減による最小特権により近いポリシの実現

提案手法を利用することで，不要なポリシを削除し，最小特権に近づけることができる．たとえば，本稿では，Apache Struts2 の脆弱性を用いた評価を行い，攻撃を防止可能であることを示した．

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として，MAC がある．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，

```
(rule_name domain type (class (av)))
   (1)     (2)   (3)   (4)   (5)
```

図 1 SELinux CIL におけるアクセスルールの記述

```
(typeattribute httpd_script_exec_type)
(typeattributeset httpd_script_exec_type (httpd_sys_script_exec_t httpd_user_script_exec_t))
        :
(allow httpd_t httpd_script_exec_type (file (ioctl read getattr lock open)))
        :
```

図 2 アトリビュートの宣言例

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 Reference Policy

SELinux では，セキュリティポリシと呼ばれる設定ファイルで定義された権限をプロセスに許可することでアクセス制御を行う．Fedora や CentOS では，Reference Policy [5]（以降，refpolicy）というポリシが利用されている．refpolicy は，多くの環境で問題なく動作するように権限が与えられた汎用的なポリシである．また，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．

2.2.2 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 [6] であり，Fedora では，Fedora 23（2015 年 10 月リリース）から取り込まれている [7]．アクセスルールとして記述されるポリシは，マクロを使用しない場合，以下の 5 つの要素で構成される．SELinux CIL におけるアクセスルールの記述を**図 1** に示す．

(1) ポリシルール（`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`）

ポリシルールの `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．タイプは，複数束ねてグループ化することが可能であり，束ねたものをアトリビュートという．アトリビュートの例を**図 2** に示す．図 2 の例では，`attribute` 宣言で `httpd_script_exec_type` というアトリビュートを宣言し，`typeattributeset` 宣言で，`httpd_script_exec_t` と `httpd_user_script_exec_t` とい

う 2 つのタイプをグループ化している．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．

2.3 ポリシの問題点

2.3.1 ポリシ記述の難しさ

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アクセスルールの記述がある [8]．

アクセスルールの総数

SELinux のセキュリティ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．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．実際に，Fedora 9 において，デフォルトで利用されているポリシ内のアクセスルールの数は 150,000 を超える．

アクセスルールの記述

(1) パーミッションの設定

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．このため，Linux カーネルの知識が必要になる．さらに，700 を超える種類のパーミッションがパーミッションの設定をより難しくする．

(2) ラベルの設定

システム内すべてのファイルとポートに対してラベルを付与する必要がある．ここで，標準の Linux システムには 10,000 を超えるファイルが存在する．このため，ラベルの設定は非常に難しい．

ポリシ記述の難しさを解決するため，SELinux のポリシ設定ツールとして，SEEdit [9] などの様々なツールが開発されている．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，一からポリシを記述することは簡単ではない．

2.3.2 アトリビュートへの権限付与

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る．アトリビュートに対して，権限を付与した場合，複数のタイプに対して権限を付与したことになる．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 [10]．

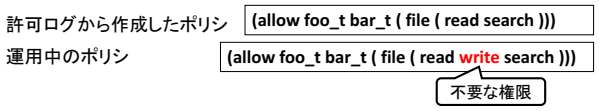

図 3 不要なポリシの発見

2.3.3 最小特権実現の難しさ

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，必要以上のアクセス権限を与えないことである．SELinux のポリシはホワイトリスト方式を採用しているため，ポリシに記述されている操作以外はすべて禁止される．このため，ブラックリスト方式に比べ，安全性が高い．一方で，誤って必要以上の権限を与えることで，安全性を低下させる可能性がある．必要以上の権限を与えてしまう原因の一つとして，汎用的なポリシ（refpolicy）の利用がある．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．

2.3.4 ポリシのメモリ使用量の多さ

IoT 機器に利用されている OS の約 86%が Linux であり [11]，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で利用されている refpolicy のメモリ使用量は約 3.7MB 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）

2.3 節で述べた問題を解決するため，我々は，実行対象のシステムに合わせて，不要なポリシを自動で削除する手法を文献 [3, 4] で提案した．従来手法では，SELinux が保持している SELinux CIL という中間言語で記述されたファイルと SELinux が出力する監査ログを利用して不要なポリシを発見し，削除する．従来手法では，SELinux が出力する許可ログを一定期間収集し，収集した許可ログをポリシに変換する．その後，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し，運用中のポリシから不要なポリシを削除する．不要なポリシを発見する例を**図 3** に示す．この例では，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した結果，`write` を不要な権限であるとして，運用中のポリシから削除する．従来手法では，SELinux に許可ログを出力させるために，運用中のポリシに `auditallow` 文を追加している．

また，アトリビュートを含むポリシを削減する際に，ア

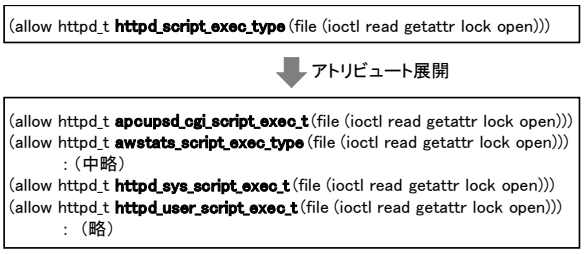

図 4　アトリビュート展開の例

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 4 に示す．許可ログから作成したポリシと運用中のポリシを比較する際に，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．これにより，粒度の細かいポリシ削減を実現している．

3. 提案手法

3.1 従来手法が対処できないポリシ

従来手法は，不要なポリシを自動で発見し，削除することができる．一方で，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 5 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，ポリシ削減手法適用不可なモジュール内に存在する

アトリビュートは，複数のタイプを束ねたものであるため，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 文がポリシ削減手法適用不可な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，ポリシ削減を行うことができない．

ここで，ポリシ削減手法適用不可なモジュールとは，以下のモジュールである．

(1) base モジュール
(2) 監査システムに関するアクセスルールを含むモジュール
(3) audit dispatcher daemon（以降，audispd）が起動する前に発生する操作に関するアクセスルールを含むモジュール

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．また，監査システムに関するアクセスルールを含むモジュールに auditallow 文を追加した場合，ログが出力された際，ログの出力に伴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．さらに，ポリシ削減手法は，audispd から受け取っ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．このため，audispd が起動する前に発生する操作に関するアクセスルールを不要な権限として削減してしまう恐れがある．

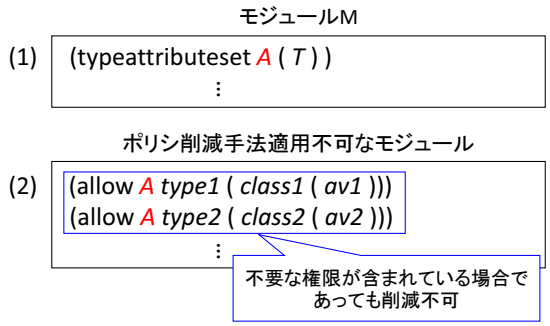

図 5　従来手法が対処できないポリシの例

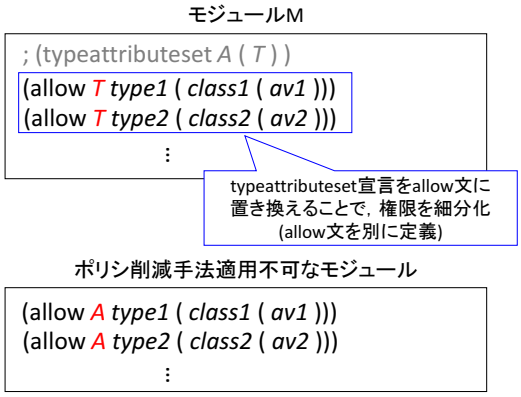

図 6　typeattributeset 宣言の置き換え

3.2 考え方

3.1 節で述べた問題点は，タイプ T をアトリビュート A に加え，ポリシ削減手法適用不可なモジュール内で 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．ポリシ削減手法適用不可なモジュールから allow 文を分離することで，権限を細分化することができる．具体的には，typeattributeset 宣言を allow 文に置き換える（以降，typeattributeset 宣言の置き換え）．例を図 6 に示す．図 6 の例は，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，ポリシ削減手法適用不可なモジュールからタイプ T に関する allow 文を切り分けることができ，ポリシ削減手法適用不可なモジュールに対してポリシ削減手法を適用せずに不要な権限を削減できるようになる．

3.3 実現における課題と対処

（課題 1）typeattributeset 宣言を置き換えるために必要な情報の取得

typeattributeset 宣言の置き換えにあたり，置き

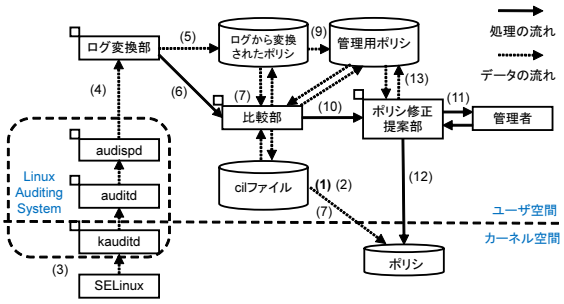

図 7　ポリシ削減機能

換えるために必要な情報をどのように取得するかという問題がある．allow 文に置き換えるためには，オブジェクトクラス（class）やアクセスベクタパーミッション（av）の情報が必要となる．この情報を取得するために，cil ファイルを利用した．具体的には，置き換え対象のアトリビュートに関する allow 文を cil ファイルからすべて取得し，取得した allow 文から class や av の情報を取得する．

（課題 2）置き換える対象の選択

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．このため，置き換える対象の typeattributeset 宣言を絞り込む必要がある．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 [12] 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．

3.4　提案手法の処理流れ

提案手法は，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て，不要なポリシを削除する．その後，テスト運用期間でポリシ復元機能を利用して，誤って削除してしまったポリシを復元する．提案手法におけるポリシ削減機能の処理流れを図 7 に示し，以下で説明する．

(1) 運用中のポリシ（cil ファイル）に対して，typeattributeset 宣言の置き換えを実行
(2) 運用中のポリシに auditallow 文を追加し，システムに適用
(3) SELinux がログを出力
(4) ログ変換部が 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
(6) ログ変換部によって変換されたポリシのサイズが閾値を超えていた場合，比較部を起動
(7) 比較部がログ変換部によって変換されたポリシと運用中のポリシを比較し，一度ポリシの形式に変換されたアクセスルールに関する auditallow 文をポリシから削除
(8) 一度ポリシの形式に変換されたアクセスルールに関する auditallow 文を削除したポリシをシステムに反映
(9) ログ変換部が変換したポリシを管理用ポリシに保存
(10) （2）～（8）を一定期間繰り返したあと，管理用ポリシと cil ファイルを比較部が比較し，差分のポリシを作成した後，ポリシ修正提案部を起動
(11) ポリシ修正提案部が差分のポリシの内容を通知し，ポリシの修正を提案
(12) 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 cil ファイルを修正し，システムに反映
(13) 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

なお，(1) が文献 [4] の手法からの拡張部分である．

また，提案手法におけるポリシ復元機能の処理流れは，文献 [3] の手法の処理流れと同様である．

4.　評価

4.1　評価内容と評価環境

提案手法の有用性を明らかにするために，以下の評価を行った．

（評価 1）Apache Struts2 の脆弱性 [13] を用いた評価

提案手法を適用したシステムにおいて，Apache Struts2 の脆弱性を悪用した攻撃を行い，提案手法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する．

（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，および allow 文の数をどの程度削減できるかを示す．

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．

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．各モジュールのログの収集期間は 2 日間とする．

また，本評価では，selinux-policy-targeted-3.13.1-102.el7 で宣言されているアトリビュートの内，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．置き換えの対象となったアトリビュートは 13 個 (例：files_unconfined_type，unconfined_domain_type など) である．

4.2　Apache Struts2 の脆弱性 [13] を用いた実験

4.2.1　実験内容

Apache Struts とは，Apache Software Foundation によっ

表 1　CVE-2017-5638 を用いた評価結果

条件	ポリシのバグ	手法の適用	任意コードの実行による攻撃防止の可否
（条件 1）	あり	無	攻撃防止失敗（tomcat の権限で任意のファイルにアクセス可能）
（条件 2）		従来手法	攻撃防止失敗（tomcat の権限で任意のファイルにアクセス可能）
（条件 3）		提案手法	攻撃防止成功
（条件 4）	なし	無	権限内に制限（ポリシの範囲内で被害が発生）
（条件 5）		従来手法	権限内に制限（ポリシの範囲内で被害が発生）
（条件 6）		提案手法	攻撃防止成功

て開発されている Java の Web アプリケーションを作成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 [13]) が存在する [14]．本実験では，Web 上から入手できるエクスプロイトコード [15] を用いて，任意コードを実行できるか否か実験した．このエクスプロイトコードは，Apache Struts2 へ細工したリクエストを送信することで，任意のコマンドを実行する．

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，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．それぞれに対して，ポリシ削減手法適用無し，従来手法適用後，提案手法適用後の条件下で実験を行った．

ここで，ポリシのバグとして，ドメイン tomcat_t が誤ってアトリビュート files_unconfined_type に加えられていると報告されている [12]．また，アトリビュート files_unconfined_type は任意のファイルにアクセス可能であるように base モジュール内にアクセスルールが記述されている．

4.2.2　実験結果

実験結果を**表 1** に示す．ポリシ削減手法を適用していない（条件 1）では，ポリシのバグにより，tomcat の権限で任意のファイルにアクセスが可能であった．また，従来手法を適用した（条件 2）では，従来手法により，不要な権限が削減されたものの，tomcat の権限で任意のファイルにアクセスが可能であった．これは，ポリシ削減手法適用不可なモジュールである base モジュール内に files_unconfined_type に関するアクセスルールが記述されており，files_unconfined_type に関する不要な権限を削減できなかったためであると推察できる．さらに，従来手法を適用していない（条件 4）と従来手法を適用した（条件 5）では，ドメイン tomcat_t がアトリビュート files_unconfined_type に含まれてはいないため，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にアクセス可能であった．

一方で，提案手法を適用した（条件 3）と（条件 6）では，任意コードの実行を防止し，攻撃を防止できた．攻撃を防止した際に SELinux が出力した拒否ログ（一部）を以下に示す．

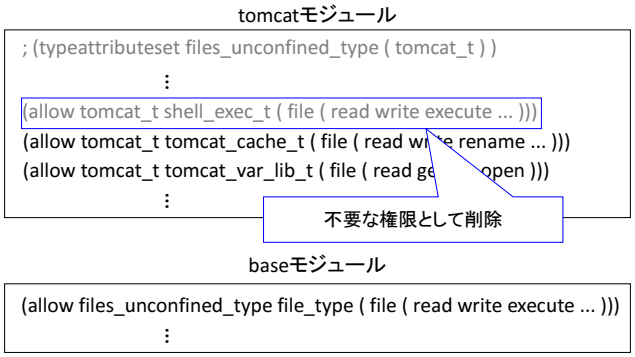

図 8　typeattributeset 宣言の置き換えの結果

```
avc:  denied { execute } for
pid=23254 comm="java" name="bash"
dev="dm-0" ino=50545940
scontext=system_u:system_r:tomcat_t:s0
tcontext=system_u:object_r:shell_exec_t:s0
tclass=file
```

これは，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である．また，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 8** に示す．図 8 から，files_unconfined_type を allow 文に置き換え，不要な権限 (shell_exec_t) 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．

以上から，提案手法は，従来手法に比べ，細粒度で不要なポリシを削減できる．

4.3　ポリシ削減量

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，および allow 文の数は，最小特権に近づけているかを示すための評価である．

表 2　3 つのモジュールにおける allow 文の削減数

モジュール	ポリシ削減手法	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	従来手法	1,201	84,155	21	84,134 (99.98%)
	提案手法		151,761	54	151,707 (99.96%)
ssh	従来手法	709	14,334	76	14,258 (99.47%)
	提案手法		14,336	87	14,249 (99.39%)
tomcat	従来手法	145	7,104	19	7,085 (99.73%)
	提案手法		71,377	74	71,303 (99.90%)
合計	従来手法	2,055	105,593	116	105,477 (99.89%)
	提案手法		237,474	215	237,259 (99.91%)

表 3　ポリシの削減量

	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

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．本評価の手順を以下に示す．

(1) 各モジュールに auditallow 文を適用し，許可ログの収集を開始
(2) 計算機を再起動
(3) 2 日間許可ログを収集
(4) 不要なポリシを削除

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（apache モジュール，ssh モジュール，および tomcat モジュール）の allow 文の削減数を**表 2** に示す．

表 2 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 割以上の不要な権限を削減できたことがわかる．また，表 2 のアトリビュート展開後の allow 文の数から，typeattributeset 宣言の置き換えにより，各モジュール内の allow 文の数が増加している．

以上のことから，提案手法は従来手法に比べ，より多くの allow 文に対して，不要な権限が含まれていないか確認し，細粒度で不要なポリシを削減可能であると推察できる．

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．ここで，allow 文の数は，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．ラベルの数は，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わかる．また，ラベルの数から，本環境では，ラベルの約 57%は実際には利用されていないことがわかる．さらに，モジュールの数から，モジュールの約 73%が実際には利用されていないことがわかる．これは，refpolicy が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に関するモジュールを多く含んでいるからであると考えられる．

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．

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等により，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発見し，取り除く．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，大規模な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．これに対して提案手法は，許可ログを利用して，既存のポリシから不要なポリシを発見し，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] は，SEAndroid ポリシの知識エンジンである SPOKE を提案し，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．文献 [19] は，ポリシに関する詳しい知識が必要となり，ポリシの開発，洗練に利用される．一方で，提案手法は，ポリシに関する詳しい知識は必要なく，不要なポリシを自動で発見し，削除する．

6. おわりに

従来手法では，以下を満たす場合，不要なポリシを削減できない．

(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，ポリシ削減手法適用不可なモジュール内に存在する

この問題を解決するために，従来手法を拡張し，その方式と評価結果について述べた．提案手法では，タイプ `T` をアトリビュート `A` に加えずに，別に `allow` 文を定義することで，ポリシ削減手法適用不可なモジュールから allow 文を分離し，権限を細分化することができる．

Apache Struts2 の脆弱性を用いた評価により，提案手法は，従来手法に比べ，細粒度で不要なポリシを削減できることを示した．また，ポリシの削減量の評価として，HTTP サーバ，SFTP サーバ，およびサーブレットコンテナが動作している計算機に対して提案手法を適用し，376 個のモジュールを対象として，ポリシの削減を行った．評価結果から，提案手法適用により，ラベルの数を約 57%，モジュールの数を約 73%，allow 文の数を約 85%削減し，最小特権に近づけることが可能であることを示した．また，不要なポリシを削減することで，ポリシのサイズを約 72% 削減し，メモリ使用量の削減が可能であることを示した．

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, 入手先 ⟨https://www.nsa.gov/what-we-do/research/selinux/⟩（参照 2017-12-26）．

[3] 齋藤凌也，山内利宏：SELinux CIL を利用した不要なポリシ削減手法の提案，情報処理学会研究報告，Vol.2018-CSEC-82，No.30，pp.1–8（2018）．

[4] 齋藤凌也，山内利宏：ログ出力の抑制による SELinux の不要なポリシ削減手法，情報処理学会シンポジウムシリーズ コンピュータセキュリティシンポジウム 2018 (CSS2018) 論文集， vol.2018，no.2，pp.956-963（2018）．

[5] TresysTechnology: SELinux Reference Policy, GitHub, available from ⟨https://github.com/TresysTechnology/refpolicy⟩ (accessed 2017-01-12).

[6] 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).

[7] 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).

[8] 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).

[9] Nakamura, Y., Sameshima Y. and Tabata, T.: SEEdit: SELinux Security Policy Configuration System with Higher Level Language: Proc. 23rd Large Installation System Administration Conference (LISA'09), pp.107–117 (2009).

[10] 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).

[11] 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).

[12] Kazuki Omo: 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).

[13] 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).

[14] 独立行政法人情報処理推進機構：更新：Apache Struts2 の脆弱性対策について (CVE-2017-5638)(S2-045)(S2-046)，入手先 ⟨https://www.ipa.go.jp/security/ciadr/vul/20170308-struts.html⟩（参照 2018-02-19）．

[15] 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).

[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).