社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－DPS－134 (7)
2008－CSEC－ 40 (7)
2008／3／6

SELinuxの不要なセキュリティポリシの削減手法の提案

山口拓人[†1] 中村雄一[†2] 田端利宏[†1]

SELinuxを導入する際は，汎用的なセキュリティポリシを適用することが多く，必要以上の権限を許可する可能性がある．また，誤って拒否ログから必要以上の権限を許可してしまった場合，SELinuxはこれを見つけることが困難である．そこで，必要以上に与えてしまったセキュリティポリシを自動的に発見し，システム管理者に削除を提案する手法を提案する．本手法は，カーネルで使用したポリシを取得し，実施中のポリシとの差分を取得することで実現されている．本稿では，SELinuxのポリシの問題点を明らかにし，この問題に対する対策を検討した．また，この対策を実現するシステムの設計を示す．

Proposal of a Method to Cut Redundant Security Policy of SELinux

Takuto Yamaguchi ,[†1] Yuichi Nakamura [†2]
and Toshihiro Tabata[†1]

In installing SELinux, in many cases, general security policy is adapted. This has possibility to authorize excessive rights. In addition, if such rights are permitted from a denied log file, it is difficult for SELinux to detect them. To solve this problem, we propose a method to detect automatically such rights and to cut redundant security policy of SELinux. Our method collects used policies and gains difference between enforcing policies and those of them. In this paper, we clarify problems of security policy of SELinux and take measures against them. We also show the design of this system.

1. はじめに

高度化した様々な攻撃から，コンピュータシステムを守るため，セキュリティを向上させる様々な手法が提案されている．これらの手法の1つとして，Security-Enhanced Linux（SELinux）[1]を代表とするセキュアOSがある．セキュアOSとは，Mandatory Access Control（MAC）と最少特権の2つの機能を実現するオペレーティングシステム（以降，OSと略す）である．MACとは，システムポリシをコンピュータシステム内のユーザやプログラムに対して強制できる機能である．MACのもとでは，ファイルの所有者やプログラムの実行者であっても，自らの思い通りにに制御することはできず，不正なプログラムの実行や無権限者によるアプリケーションの実行を防止することができる[2]．また，最少特権とは，コンピュータシステム内の主体（ユーザやプログラム）の持つ強力な権限を役割や用途に応じて分割し，個々の権限は必要最小限にする仕組みである[2]．

SELinuxを導入する際は，SELinux開発者が作成した汎用的なセキュリティポリシ（以降，ポリシと略す）が配布されているため，このポリシを適用することが多い．しかし，個々のシステムでこのポリシを運用する場合，汎用的なポリシは，個々のシステムでは，本来必要のない権限まで許可していることが多いため，必要以上の権限を許可する可能性がある．

また，SELinuxで拒否された操作について，監査システム[3]が生成したログ（以降，拒否ログと略す）から，audit2allowコマンドを使ってポリシを作成する手法がある．audit2allowコマンドとは，拒否された操作を許可するためのポリシを生成するために，拒否ログからポリシを生成するコマンドである．この手法は，自動的にポリシを作成するため，本来許可してはならない権限までシステム管理者が誤って許可してしまう可能性がある．また，SELinuxを導入する時だけでなく，ソフトウェアのバージョンアップに伴い，ポリシの修正が必要な場合も，audit2allowコマンドを使用すると，同様の理由でシステム管理者が設定ミス

†1 岡山大学 大学院自然科学研究科
Graduate School of Natural Science and Technology, Okayama University
†2 日立ソフトウェアエンジニアリング株式会社
技術開発本部研究部
Hitachi Software Engineering Co., Ltd.
Research & Development Department

を犯す可能性がある．

そこで，必要以上に与えてしまったポリシを自動的に発見し，削除する手法を提案する．本手法は，実際に各システムにおいて使用されたポリシを自動的に記録し，使用されなかったポリシの削除をシステム管理者に提案するという手法である．本手法を導入することにより期待される効果として，より厳密な最少特権の実現が可能となり，安全性の向上が期待できる．

2. SELinux

2.1 SELinux とは

SELinux は，米国家安全保障局（NSA）を中心として開発されるセキュア OS である．SELinux は，セキュア OS として必要な MAC と最少特権を実現している．また，他のアクセス制御機能として，Type Enforcement（TE）がある．TE は，サブジェクト，オブジェクト，操作の 3 つの関係に基づいて，アクセス可否を判定する．この時，実施しようとする操作が，ポリシに記述されているか比較する．

2.2 ポリシの種類

SELinux のポリシは，2 つの種類がある．1 つ目は，Fedora Core 4 以前に採用されていた Example Policy である．もう 1 つは，Fedora Core 5 から採用されている Reference Policy[4] である．これらの大きな違いは，ポリシのモジュール化にある．Reference Policy では，ポリシがモジュール化されたため，ポリシの運用中でも，モジュール単位でポリシの編集及びロードが可能である．本稿では，モジュール単位で扱える方が利便性が高いため，Reference Policy を扱う．

Reference Policy は，targeted ポリシと strict ポリシの 2 つのポリシに分類できる．targeted ポリシとは，よく使用されるプロセスのみにポリシを厳格に設定し，それ以外のプロセスには SELinux のアクセス制御の範囲内で，すべての権限を与えるポリシである．また，strict ポリシとは，すべてのプロセスにポリシを適用し，厳格に SELinux のアクセス制御を適用するポリシである．Fedora 7 では，targeted ポリシを採用した場合，allow ルールは 96,861 行で，モジュールは 50 個ある．また，strict ポリシを採用した場合，allow ルールは 165,113 行で，モジュールは 155 個あり，ポリシの量は少なくない．

2.3 Reference Policy

Reference Policy 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

```
policyrule subj_t obj_t : tclass { av };
```

図 1 リファレンスポリシの文法

(3) te（type enforcement）ファイル

fc ファイルは，プロセスやシステム資源のパス名とラベルを対応付けて，記述するためのファイルである．if ファイルは，te ファイルで使用するマクロを記述するためのファイルである．te ファイルは，SELinux の TE 機能を実現するためのポリシ記述ファイルである．TE は，記述したポリシのみを許可するホワイトリスト方式を採用している．te ファイルに記述される SELinux のポリシは，マクロを使用しない場合，図 1 に示す以下の 5 つの要素で構成される．

(1) ポリシルール（policyrule: allow，auditallow，dontallow，neverallow）
(2) サブジェクト （subj_t）
(3) オブジェクト （obj_t）
(4) オブジェクトクラス （tclass）
(5) アクセスベクタパーミッション （av）

ポリシルールの allow は許可，auditallow は監査システムが出力するログ（以降，監査ログと略す）の出力，dontallow は監査ログの非出力，neverallow は絶対に許可しないことを意味する．サブジェクトは，プロセスのラベルであり，オブジェクトは，操作対象となるファイルやネットワーク等のシステム資源のラベルである．次に，オブジェクトクラスとは，例えばファイルの場合は，ファイル，ディレクトリ，パイプのように，オブジェクトの種類を分類するものである．最後に，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．

3. SELinux のポリシにおける問題点

システム管理者がポリシを設定するにあたり，SELinux のポリシには，2 つの問題点がある．1 つ目は，SELinux のポリシ設定が難しいという問題がある．2 つ目は，最少特権の実現を正確に実施することが難しいという問題である．

3.1 ポリシ設定の難しさ

SELinux におけるポリシ設定の難しさは，非常に細かい粒度でアクセスパーミッションやファイルの種類を設定できることにより，ポリシの記述が複雑になるためである．この問題を解決するため，SELinux のポリシ編集ツールとして，SELinux Policy Editor (SEEdit)[5]，SETools[6]，Virgil[7] を始めとして，様々

なツールが開発されている．しかし，一からポリシを記述することは，設定ツールを使用したとしても簡単ではない．なぜなら，新たなラベルの設定に関しては，システム管理者自身の判断で適切に行わなければならない上，設定工数が多く複雑であるためである．そこで，多くの場合，SELinux と一緒に配布されている汎用的なポリシを使用する．

3.2 最少特権の実現の難しさ

SELinux のポリシが採用するホワイトリスト方式は，ブラックリスト方式に比べ，ポリシに記述されている操作以外はすべて禁止されるため，安全性が高い．しかし，ホワイトリスト方式は万能ではなく，誤って必要以上に与えてしまった権限によって，安全性を低下させる可能性がある．このため，セキュア OS が目的とする最少特権が実現できない．必要以上の権限を与えてしまう要素として，以下の 2 つがある．

(1) 汎用的なポリシは，個々のシステムでは，本来必要のない権限まで許可していることが多い．

(2) auditd が生成した拒否ログから，audit2allow コマンド等を使ってポリシを作成する際，本来許可してはならない権限までシステム管理者が誤って許可してしまう可能性がある．必要以上の権限を許可した場合，ホワイトリスト方式では，これを見つけることは困難である．

また，SELinux のポリシは，非常に複雑であるため，より扱い易いように，高級言語にあたるポリシが開発されている．この代表例として，SEEdit の Simplified Policy Description Language (SPDL)[8] がある．しかし，SPDL は SELinux のポリシを統合し，容易に扱えるようにしているため，一部のポリシを必要以上に与えてしまう問題[9] がある．この問題を解決することは，(2) と同様に困難である．

4. 提案方式

4.1 対処方法の検討

3.1 項の問題は，設定ツールの開発によって，一部解決されている．しかし，3.2 項の問題は解決されていないため，本研究では，この問題への対処を検討する．3.2 項で示した問題が発生するのは，実施しているポリシの中で，必要以上のポリシが記述されている時である．これは，SELinux のポリシ導入時にポリシの設定が不十分な際や，ソフトウェアのバージョンアップや新たなソフトウェアの導入に伴い，ポリシを修正する際に起こる可能性がある．

そこで，この問題を解決するには，システム管理者は，最少特権を実現するために十分なポリシと，不要なポリシを分離できなければならない．最少特権を実現するために十分なポリシは，実際にシステムの動作に必要とされるポリシである．また，不要なポリシとは，ポリシとして記述されているにもかかわらず，ポリシが実際のシステムの動作に必要とされないポリシである．実際のシステムの動作に必要とされるかどうかは，カーネルレベルで SELinux のポリシの確認において許可された動作に関して，サブジェクト，オブジェクト，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を蓄積することで知ることできる．

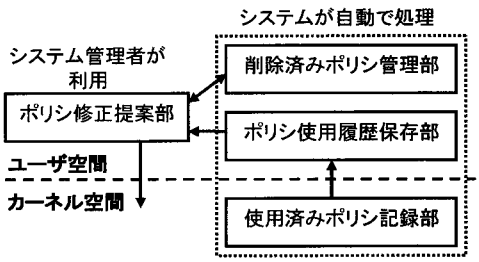

図 2 システムの全体像

我々は，ポリシとして記述されているにもかかわらず，長い間使用されていないポリシほど，不要なポリシである可能性が高いと考え，この対処を実現する手法として，次のシステムを提案する．まず，システム稼働中に使用されたポリシを記録し，記録されたポリシと実施中のポリシとの差分のポリシ（以降，差分ポリシと呼ぶ）を出力する．次に，この差分ポリシの中に，不要なポリシが存在する可能性があると考え，システム管理者に差分ポリシの一部の削除を提案する．この提案に基づき，システム管理者は必要がないと判断したポリシを削除し，ポリシモジュールをコンパイルし，カーネルにロードする．また，システム管理者が誤って必要なポリシを削除した場合に備え，削除したポリシは保存しておき，必要に応じて復元できるようにしておく．ポリシを復元する場合は，システム管理者が改めてポリシの内容を確認した上で，ポリシモジュールをコンパイルし，カーネルにロードする．

この対処内容から，本システムを機能ごとに分割した結果，使用済みポリシ記録部，ポリシ使用履歴保存部，ポリシ修正提案部，削除済みポリシ管理部の 4 つの機能で構成する．図 2 に，このシステムの全体像を示す．また，以下で，図 2 における各機能部の処理の流れを示す．

(1) 使用済みポリシ記録部において，許可された動作に関する情報を蓄積し，ポリシ使用履歴保存部に情

報を送る．

(2)　ポリシ使用履歴保存部は，使用済みポリシ記録部から受け取った情報を保存する．また，保存しているポリシの情報と，実施中のポリシとの差分を取り，ポリシ修正提案部に送る．

(3)　ポリシ提案部において，システム管理者は自身の判断の下，提案されたポリシの一部を削除し，ポリシをコンパイルし，カーネルにロードする．

(4)　ポリシ提案部において，システム管理者によって削除されたポリシは，削除済みポリシ管理部に送られ，削除済みポリシ管理部に保存される．

4.2 実現課題

この対処を実現するためには，既存のシステムには用意されていない機能を追加する必要がある．新たな機能を実現するために，以下の3つの課題を検討する．

(課題1)　使用したポリシを記録する方法の検討

(課題2)　不要なポリシを提案する方法の検討

(課題3)　誤って削除したポリシ復元方法の検討

(課題1) 使用したポリシを記録するために，以下の2つの案を検討した．

(案1)　監査システムを利用する方法

(案2)　使用済みフラグを設置する方法

(案1) 監査システムは，既存のシステムをそのまま利用することで実現可能である．具体的には，許可するすべてのポリシに対して，auditallow ルールを適応することで，許可されたログを出力させることができる．しかし，OS 起動時や常時動作するプロセスに関するログが，大量に生成されてしまうという問題がある．この時，監査システムはバッファに入りきらなかった場合，ログを消失する仕様になっている上に，監査システムの設定によっては，パニックを引き起こしてしまう．また，大きなデータを読み込む時，ブロックの読み込みに関するログが重複して記録されるため，システムにかかるオーバヘッドが大きいという問題もある．

(案2) この方法は，SELinux のポリシの確認において許可されたポリシに対して，使用したことを示すフラグを立てる方法である．この機能は，既存のシステムに用意されていないため，新たに実装する必要がある．しかし，**(案1)** と比較してシステムにかかるオーバヘッドが少ない．そこで，本システムでは，**(案2)** を採用する．

(課題2) 差分ポリシは，不要なポリシである可能性があるため，システム管理者が検証する．この時，差分ポリシだけを提示するだけでは，本当に削除してもよいポリシなのか検討が難しい可能性がある．例えば，エラー処理のように，本来必要であるけれども，使用頻度が低いために，提示された可能性がある．この問題に対応するために，本システムでは，ラベルに対応するプロセス名やファイル名を同時に示すことで，システム管理者を支援する．また，誤って削除してしまった場合に備えて，削除したポリシを保存し，元に戻せるようにする．

(課題3) 誤って削除したポリシを復元するためには，削除したポリシを保存しておく必要がある．そこで，システム管理者がポリシを編集するのはユーザ空間であるため，ユーザ空間に削除したポリシを保存する機能を作成する．保存したポリシが誤って削除したポリシかどうかを確認するためには，ポリシで許可されていない操作が実施された記録を確認することが有効である．この記録は，監査システムが常時拒否ログとして生成しているため，拒否ログの利用が有効である．削除したポリシと拒否ログの内容が一致する場合，本来このポリシがシステムの動作に必要であったにもかかわらず，システム管理者が誤って削除したことがわかる．そこで，該当する削除したポリシを復元するように，システム管理者に提案し，システム管理者の判断の下でポリシを修正する．

4.3 期待される効果

本システムを導入し，利用することにより，以下の3つの効果が期待される．

(1)　最少特権の実現がより正確に行えるようになり，安全性の向上が期待できる．

(2)　システム管理者は，本システムを利用することにより，自動的に不要なポリシの可能性があるポリシを提案してもらえる．これにより，ポリシの導入や修正にあたり，正確にポリシの追加ができたかを確認できるため，システム管理者のポリシ編集作業を支援できる．このため，システム管理者の負担を減らせる．

(3)　ポリシ設定ツールは，ポリシ設定を簡易化してシステム管理者の負担を軽減させる反面，本当に正確なポリシが生成されたかどうかはシステム管理者にとって不安であった．しかし，本システムを利用することにより，必要以上に与えてしまったポリシを発見でき，削除できる．このため，ポリシ設定ツールの利用を促進できる．

5. 設　　計

5.1 設計方針

本章では，4.1 節で説明した対処を実現するために，システムの設計について説明する．本システムの設計方針は，以下の通りである．

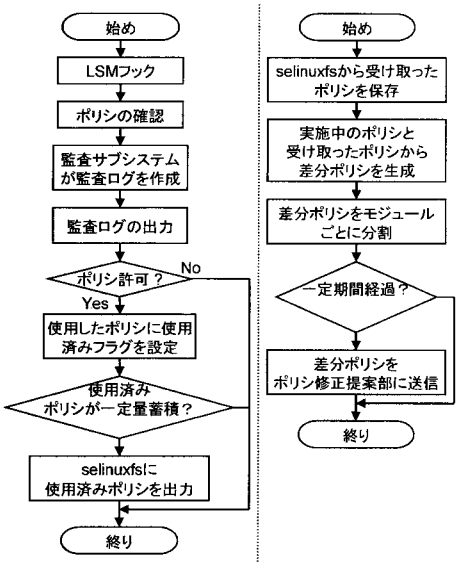

図 3 使用済みポリシ記録部とポリシ使用履歴保存部の処理の流れ

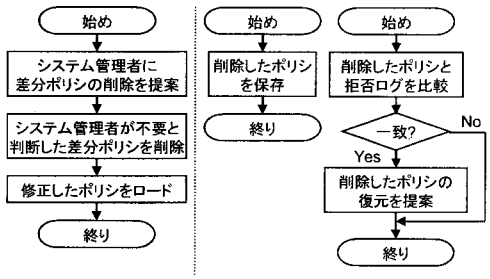

図 4 ポリシ修正提案部と削除済みポリシ管理部の処理の流れ

(1) ポリシが必要か不要かの判断は，システム管理者に任せる．

(2) ポリシの修正時に，システム管理者のポリシ削除における判断を支援する．

(3) 誤って必要なポリシを削除してしまった場合，システムが知り得た段階で，できるだけ早く修正する．

(4) パフォーマンスの低下を抑える．

図 3 に，使用済みポリシ記録部とポリシ使用履歴保存部における処理の流れを示す．また，図 4 に，ポリシ修正提案部と削除済みポリシ管理部における処理の流れを示す．本設計では，Linux Security Modules (LSM)[10] からフックされた関数において，ポリシの確認を実施する箇所で，使用されたポリシの情報を取得するため，すべての使用されたポリシの情報を取得できる．また，取得した情報をシステム管理者に提案するに留め，実際にポリシを削除するかどうかは，システム管理者の判断に任せる．このため，二重に確認されることになり，信頼性を高められる．最後に，システム管理者が誤って判断した場合に備えて，削除したポリシのバックアップを取ることにより，信頼性を高めている．

各部の機能の詳細については，以下で説明する．

5.2 使用済みポリシ記録部

本構成部の目的は，使用されたポリシの記録を取ることである．この目的を達成するために，本構成部では，以下の 1 つの機能を用意する．

(機能 1) 使用済みフラグ設定機能

以下で，本構成部の処理の流れを説明する．

(1) SELinux では，システムコールや割り込みに伴い，SELinux のアクセス制御を実施するために，LSM を通して，フック関数が呼ばれる．このフック関数において，SELinux の処理が実施され，ポリシに適合した操作かどうか検証が実施される．

(2) 削除済みポリシ管理部において，拒否ログの確認する必要があるため，監査システムに監査ログの生成を実施させる．

(3) ポリシの確認において許可された場合，サブジェクト，オブジェクト，オブジェクトクラス，アクセスベクタパーミッションを取得する．

(4) Access Vector Cache（AVC）において，該当するポリシ（allow ポリシ）に使用済みフラグを設定する（**機能 1**）．使用済みフラグには，許可されたアクセスベクタパーミッションに対応するビットを立てる．

(5) カーネル空間で確保できるメモリ領域は限られているため，定期的に使用済みフラグが設定されたポリシをユーザ空間に送信し，フラグをクリアする．この時，カーネル空間とユーザ空間とのデータのやり取りに，SELinux 用に用意された疑似ファイルシステム selinuxfs を利用する．また，送信データの内容は，文字列としてデータを送信するとオーバヘッドが大きいため，数値として送信する．

5.3 ポリシ使用履歴保存部

本構成部の目的は，使用済みポリシの情報をカーネル空間から受信し，定期的に保存することである．この目的を達成するために，本構成部では，以下の 3 つの機能を用意する．

(機能 2) 使用済みポリシ保存機能

(機能 3) 実施中のポリシとの差分取得機能

(機能 4) 差分ポリシのモジュール分割機能

以下で，本構成部の処理の流れを説明する．

(1) 使用済みポリシ記録部から，ユーザ空間のプロ

セスは，定期的にデータを受信する．
(2)　受信したデータは，数値であるため，サブジェクトやオブジェクトのラベル名等の文字列に変換し，保存する（**機能 2**）．
(3)　保存したポリシと実施中のポリシの差分を取り，未使用のポリシを生成する（**機能 3**）．
(4)　Reference Policy の特性を活かすために，ポリシはモジュールごとに分割する（**機能 4**）．これにより，ポリシを運用しながら，ポリシの編集と運用が可能になる．

5.4 ポリシ修正提案部

本構成部の目的は，差分ポリシをシステム管理者に提示し，不要なポリシの削除を促すことである．また，誤ってポリシを削除した時，該当するポリシの復元をシステム管理者に促す．この目的を達成するために，本構成部では，以下の 2 つの機能を用意する
（機能 5）　ポリシ修正提案機能
（機能 6）　削除ポリシ復元機能
以下で，本構成部におけるポリシ修正処理の流れを説明する．
(1)　ポリシ使用履歴保存部から差分ポリシを受け取り，モジュールごとにポリシの削除をシステム管理者に提案する（**機能 5**）．この時，サブジェクトやオブジェクトのラベル名をパス名に変換して，削除提案をする．
(2)　システム管理者は，削除するポリシを選択し，ポリシを修正する．
(3)　ポリシをコンパイルし，カーネルにポリシをロードする．
(4)　削除したポリシは，誤って削除した場合に備えて，削除済みポリシ管理部に送る．
以下で，本構成部におけるポリシ復元処理の流れを説明する．
(1)　削除済みポリシ管理部から，システム管理者は，復元するポリシを提案される（**機能 6**）．
(2)　復元するポリシを確認した後，ポリシを修正し，ポリシを反映する．

5.5 削除済みポリシ管理部

本構成部の目的は，不要と判断された削除したポリシを保存することである．この目的を達成するために，本構成部では，以下の 2 つの機能を用意する．
（機能 7）　削除ポリシの保存機能
（機能 8）　拒否ログと削除ポリシ比較機能
以下で，本構成部の処理の流れを説明する．
(1)　ポリシ修正提案部で削除されたポリシを受け取り，モジュール名とともに保存する（**機能 7**）．
(2)　システム管理者が，誤って必要なポリシを削除した場合に備えて，拒否ログの中に削除ポリシに該当するものがないか比較する．もし該当するポリシが見つかれば，すぐにポリシ修正提案部に情報を送り，システム管理者にポリシの復元を促す（**機能 8**）．

6. おわりに

本稿では，システム管理者がポリシの設定時における SELinux のポリシの問題点として，ポリシ設定の難しさと最少特権の難しさの 2 つがあることを明らかにした．これらの問題の中で，最少特権の実現の難しさに対する対策として，SELinux の不要なセキュリティポリシの削減手法を提案した．この対策を実現するシステム設計として，カーネルで自動的に使用したポリシを記録し，一定期間内に使用されなかったポリシをシステム管理者に削除提案する方式を説明した．
残された課題として，本手法の実装と評価がある．

謝辞　本研究の一部は，C&C 振興財団 若手研究員助成の支援を受けて行った．

参考文献

1) P. Loscocco, and S. Smalley, "Integrating Flexible upport for Security Policies into the Linux Operating ystem," In Proceeding 10th USENIX Conference (FREENIX Track) 2001.
2) みずほ情報総研株式会社，"電子政府で利用する情報システムへのセキュリティ機能を強化した OS の適用可能性等に関する調査研究報告書，" 内閣官房情報セキュリティセンター（NISC）調査研究，p.18，2006.
3) T.R. Chavez, "A Look at Linux Audit, " Linux-World Conference & Expo, 2006.
4) C. PeBenito, F. Mayer, K. MacMillan, "Reference Policy for Security Enhanced Linux," SELinux Symposium 2006，2006.
5) 中村雄一，鮫島吉喜，"Security-Enhanced Linux のアクセス制御ポリシ設定の簡易化，" 暗号と情報セキュリティシンポジウム（SCIS2003）予稿集，Vol.II，pp.831–836，2003.
6) Tresys Technology,"SETools," http://www.tresys.com/
7) Virgil, http://sourceforge.net/projects/sepolicy-virgil/.
8) Y. Nakamura, "Progress of SELinux Policy Editor, " SELinux Symposium, http://seedit.sourceforge.net/presentations/2006selinuxsymposium.pdf，2006.
9) 山口拓人，中村雄一，田端利宏，"ファイルのアクセスベクタパーミッションを統合したアクセスパーミッションの安全性評価，" 情報処理学会コンピュータセキュリティシンポジウム 2007 論文集，pp.625–630，2007.
10) C. Wright, C. Cowan, S. Smalley, J. Morris, and G. Kroah-Hartman, "Linux Security Modules: General Security Support for the Linux Kernel, " In Proceedings USENIX Security Symposium，2002.