必須アクセス制御方式を用いた侵入検出システム保護機能

女部田 武史† 井上 直† 浅香 緑†

不正侵入の増大にともなって侵入検出システムの研究がさかんに行われ，多数の商用システムも登場してきた．しかしながら多くの侵入検出システムはその侵入検出手法にのみ注目し，自身のセキュリティに関してはあまり考慮がなされていない．侵入者が侵入の際にまず行うことはログの削除や監視プロセスの停止であることを考えると，侵入検出システム自身のセキュリティは侵入検出手法同様に非常に重要である．そこで本研究では侵入検出システムを対象に侵入検出システム自身を保護するためのセキュリティ機能について検討を行い，伝統的な必須アクセス制御方式を用いて侵入検出システムを保護するための機構を考案，実装した．我々の考案した保護手法は LOMAC と呼ばれる必須アクセス制御モデルに「限定アクセスモード」を追加し，侵入検出システム用にカスタマイズしたものである．この機能によって侵入検出システムを侵入者から保護することが可能となり，確実に侵入判定を行うことができる．

A Protection Mechanism for an Intrusion Detection System Based on Mandatory Access Control

Takefumi Onabuta,† Tadashi Inoue† and Midori Asaka†

Research regarding intrusion detection systems (IDSs) has become more active with the recent increases in illegal accesses to computer systems. Many researchers focus only to the techniques or mechanisms for detecting intrusions automatically, without considering the security of IDSs themselves. When an intruder attacks and breaks into a system, he or she often deletes system logs and stops auditing processes. Thus, the security of an intrusion detection system is an important aspect of intrusion detection. In this paper, we examine security functions to protect IDSs. We design and implement a mechanism to protect IDSs by use of the traditional mandatory access control method. This mechanism of protecting IDSs is based on Low Water-Mark mandatory access control with the addition of a limited access mode, allowing the protection of the IDS itself from intruders.

1. はじめに

侵入検出システム（以下 IDS）はターゲットシステム上やネットワーク上のアクティビティをモニタリングし，外部・内部の侵入者またはユーザの誤使用により，ターゲットシステム上のセキュリティが侵されているかを監視するシステムである．IDS に関する研究はインターネット上での侵入事件の増大と比例して近年非常に活発化してきており，商用レベルの IDS も多数登場してきている[1),2)]．これまで IDS の研究は主に “侵入検出のための手法の開発” および “その効率化” に焦点が絞られてきた．しかしながら，製品やフリーソフトとして IDS が実環境で利用されるようになってくると,「IDS そのもののセキュリティ」が大きくクローズアップされるようになってきた．一般に侵入者が侵入直後に行うことは，ログの消去や監視プロセスの停止といったことである．侵入者によってログが消去されたり監視プロセスが停止させられた場合，IDS が正常にその侵入を判定するかは不明である．現在 IDS の研究においてはこのような攻撃から IDS を保護する「IDS のセキュリティ機能」を対象としたものは非常に少なく，実際に運用されている IDS においてもそのセキュリティ対策はさほど考慮されていない．

本研究では，一般的な侵入検出システムの動作環境である UNIX 系 OS を対象にした IDS 保護機能を提案する[3)]．我々が提案する IDS 保護の枠組みは UNIX カーネル上に IDS が動作するための安全な領域を確保しようとするものである．つまり通常のプロセスと IDS が動作する領域を区切り，その間のアクセスを強制的に制御するものである．このために，我々はカーネルレベルでの必須アクセス制御方式を導入する．必

† 情報処理振興事業協会
Information-Technology Promotion Agency, Japan

須アクセス制御に関する研究は文献 4) や 5)，6) において以前から研究されているが，これらは非常に高いセキュリティを確保するために複雑で IDS には適用しにくいものとなっている．本研究で導入する必須アクセス制御は非常に単純なモデルであり，IDS の保護に適するように構成したものである．この機能によって IDS を他のプロセスと分離することが可能となり，侵入者から IDS を保護することが可能となる．また IDS は確実に侵入判定を行うことができる．

本論文ではまず 2 章で IDS を保護するためのセキュリティ機能について議論する．3 章では必須アクセス制御方式を用いた IDS 保護機能と実装について述べる．4 章では実装したシステムの評価と考察について述べる．最後にまとめと今後の研究課題について述べる．

2. IDS の保護機能

ホストベースの IDS ではシステムでログを収集し，統計的な手法やパターンマッチングなどによって侵入の判定を行う．IDS において確実に侵入判定を行うためには，収集するログが改竄・削除されていないこと，侵入判定プロセスが正常に動作していることが重要である．侵入者は自身の痕跡を削除するためにログを消去したり，判定プロセスを停止したりする．そのため侵入者からこれら侵入判定において重要となるプロセス，ログなどを保護することが重要となる．

2.1 保護対象

侵入者から IDS を保護するためには，以下の 2 つの対象の保護が重要となる．

- 重要ファイル保護
  IDS において，侵入検出に必要なログや知識ベースなどのファイルの保護が重要である．これらの改竄を防ぐためには単にファイルを暗号化しただけでは不十分である．たとえ暗号化されていたとしてもファイルそのものを削除される危険があるからである．またシステムの特権モードを奪われた場合には，すべてのファイルが危険にさらされることとなる．さらにログファイルなどは，つねにアップデートされ変更されるものであるため，頻繁に復号化・暗号化を繰り返さなければならず，暗号化による保護によって大きなオーバヘッドが生じる．
- 重要プロセス保護
  IDS は 1 つまたは複数のプロセスによって 1 つのタスクを実行する．どのサブプロセスが機能しなくなってもタスクを完結することはできない．特に侵入を判定する機能が停止させられた場合，その管理下にあるすべてのマシンが監査の対象から外れてしまう．そのため各サブプロセスが侵入者によって停止されたり，動作を妨害されないように保護する必要がある．プロセスもファイル同様，たとえシステムの特権ユーザであってもある条件では停止させることができないといった機能や重要プロセスを隠蔽する機能が必要となる．

2.2 IDS 保護のためのアプローチ

2.1 節の保護対象に対する保護アプローチには以下の 2 つが考えられる．

(1) self protection
IDS の動作上重要となるログファイルの暗号化，プロセス，ログファイルの隠蔽といった機能を IDS の機能の一部として保護機構を実現する．

(2) kernel-level protection
侵入検出システムを保護するための仕組みを kernel に組み込み，侵入検出システムの機能と切り分ける．このアプローチは伝統的なアクセス制御の枠組みと同じである．

(1) のアプローチから IDS の保護機構を提案したものには文献 8)，9) がある．文献 8) は IDS において最も重要な部分であるログを保護するために “逃げログ” というログ保護の機構を提案している．ここではログはシステム中のディレクトリにそのコピーが隠蔽され，たとえログが改竄されても，ディレクトリに隠蔽されたコピーファイルにより，改竄または削除されたファイルの復旧を行うことができる．また文献 9) では IDS の侵入判定部分のみを侵入者から保護するために “ランダムホッピング” というプロセス保護の機構を提案している．ここでは侵入判定のモジュールがランダムなタイミングで動作するマシンを侵入者が推測しがたいマシンに移動するというものである．どちらも IDS のプロセスと同一のプロセスとして機能する．しかし，これらのアプローチはどちらも IDS 自身のプロセスとして保護機能が動作するため，侵入者の攻撃を完全には防御することはできない．文献 8) のアプローチはログを改竄・削除するような攻撃に対しては効果があるものの，IDS のプロセス自身が攻撃を受けて，停止させられる場合には意味がない．また，文献 9) のアプローチはプロセスを移動させて保護するものであるが，これは設計上システムにとどまっていなければならないプロセスを保護することはできない．マシンのログを監視するモジュールなどは移動不可能なので，これを保護しない限り，たとえ解析モジュールのみを保護しても侵入は検出できない．

上記の問題点をふまえ，我々はログファイルおよび

IDS のプロセスを保護するために，(2) の kernel-level protection のアプローチを採用する．このアプローチを採用する理由は以下の 2 つである．

(1) アプリケーションレベルでは，保護機構に制限がある．たとえば特権ユーザの権限がいったん奪われてしまえば，保護機構自身が破壊される恐れがある．

(2) ログの保護やプロセスの保護など保護対象ごとに異なる保護方法を用いるのではなく，カーネルレベルでの統一的な方法で IDS を保護する．

カーネルレベルでの保護方法として必須アクセス制御方式を導入する．IDS に適したアクセス制御をカーネル上に実装することで，IDS を保護するための機構を作成する．

3. 必須アクセス制御を用いた IDS 保護機能

本研究では必須アクセス制御を用いた IDS 保護機能を提案し，Linux 上に必須アクセス制御を用いた IDS 保護機能を試作した．

3.1 IDS のための必須アクセス制御

IDS 保護のための基本的アイデアは，IDS が動作するための安全な領域を作成するというものである．つまり，IDS を通常のプロセスとは違った領域で動作させることによって，通常のプロセスがいっさい IDS に関連するプロセスやファイルにはアクセスできないようにする．これを実現するためには従来の UNIX のアクセス制御方式では不十分である．従来の UNIX では特権ユーザの権限が奪われた場合，すべてのファイル，プロセスへのアクセスが許可されることになる．そのため侵入後に行われる侵入者の行為からプロセスやファイルを保護するための手段はない．そこでカーネルレベルでの必須アクセス制御（MAC: Mandatory Access Control）が必要となる．必須アクセス制御とはユーザによって任意に変更することができないシステムレベルの強制的なアクセス制御である．必須アクセス制御には Bell-LaPadula [4]，RBAC [6]，Chinese Wall [7]，DTE [10]，LOMAC [11] などさまざまなモデルが提唱されており，それぞれデータのインテグリティの保持，データの機密性の保持，プライバシーの保持などを指向している．本研究では IDS 保護にどのような必須アクセス制御モデルが適するのかについて，以下の 4 つの要件にて整理を行った．

- strength：IDS の領域への情報の流れを厳密に制御（アクセス制御）できること
- flexibility：多数の IDS 保護の要件を表現可能な柔軟性を持つこと
- compatibility：現在動作しているアプリケーションに影響を与えないこと
- performance：高パフォーマンスであること
- availability：制御モデルが実際のシステムにマッピングしやすいこと

上記の要件に従って主要な必須アクセス制御モデルについて検討した．

(1) Bell-LaPadula に代表されるマルチラベリングシステムはデータの読み書きを機密レベルとコンパートメントからなるセキュリティラベルによって厳密かつ柔軟にアクセスを制御することができるが，セキュリティラベルを正しく設定し，現在動作しているアプリケーションを正しく動作させるためには多くの管理コストを必要とする．また他のアプリケーションを正常に動作させるためには機密レベルの設定を全アプリケーションに対して行わなければならない．このモデルは機密情報を扱う軍事目的のシステムに多く採用されているモデルであるが，本研究の要件には複雑すぎて適さない．

(2) RBAC はユーザの役割（ロール）をベースにアクセス制御を行うもので，頻繁にユーザの役割が変更されるような，医療や金融関係のために開発されたモデルである．役割による制御のため，どのようにデータを操作するかといった操作レベルでの制御を行うことはできないため，本研究の要件には適さない．

(3) Chinese Wall Model はユーザの関心と，データ間の参照利害に基づいてアクセス制御を行うモデルである．データ間ではデータどうしの参照利害関係が定義され，ユーザはいったんあるデータを参照すると，参照したデータと相対する利害関係にあるデータにはアクセスできないように制限される．この方式は IDS の保護のためには適さない．IDS のデータとそれ以外のデータを参照利害関係があるものとして設定しても，もし仮に侵入者がはじめに IDS のデータにアクセスしようとした場合，そのアクセスは許可されてしまい，IDS が停止させられる危険性があるからである．

(4) LOMAC は Low Water-Mark Model を拡張したモデルで，すべての対象にレベルが割り当てられ，このレベルに従ってアクセス制御が行われるものである．データの完全性を保証するためのモデルで，IDS の保護に適している．保護対象（本研究においては IDS）を高いレベルに

設定し，その他のものを低いレベルに設定することで，この間の情報フローを厳しく制限することが可能となる．またこのモデルは実際のシステムにマッピングするための方法についても考慮されたモデルである．

このような比較検討により，本研究ではカーネル内にIDSのための論理的な領域を確保するために，文献11)によって提案されたLow Water-Mark Modelを拡張したLOMACと呼ばれる必須アクセス制御モデルを利用することとした．LOMACはデータのインテグリティを保持するための必須アクセス制御で，セキュリティレベルをLOW_LEVELとHIGH_LEVELの2段階に設置した単純なモデルである．LOMACでは動作するプロセスをサブジェクトと呼び，i-nodeを持ち，サブジェクトによって扱われるものをオブジェクトと呼ぶ．LOMACでは低いレベルのものが高いレベルのものにアクセスすることを許可しない．LOW_LEVELに属するサブジェクトは，HIGH_LEVELに属するオブジェクトにアクセスすることはできない．高いレベルに指定されたものへのアクセスはたとえ特権ユーザであっても制限される．また高いレベルのサブジェクトが低いレベルのオブジェクトにアクセスした場合には降格（demote）が生じる．これは低いレベルのオブジェクトにアクセスしたために，高いレベルのサブジェクトの信頼性が低下するため，以降高いレベルでの動作を許可しないようにするものである．

LOMACをそのままIDS保護に適用した場合，いくつかの問題が生じる．それはIDSをその他のプロセスと完全に切り離すことができないことに起因している．つまりIDSで利用するオブジェクトを他のプロセスで利用する必要がある場合，LOMACではアクセス制御を記述することができない．IDSと他のプロセスの双方が利用するオブジェクトとしては以下のものがある．

- ログファイル
- ライブラリ
- カーネルメモリ
- IPC

これらをHIGH_LEVELに設定すると，IDS以外の他のプロセスは利用できなくなる．LOW_LEVELに設定すれば，どのユーザもアクセス可能となりIDSの動作に影響を与えることになる．つまりIDSの動作を他のプロセスと切り離せなくなる．そこで我々はLOMACに「限定アクセスモード」という概念を導入し，限定したデータアクセスの制御を可能とした．「限定アクセスモード」とは，サブジェクトとオブジェクトのレベルが違ってもアクセス可能なモードのことである．限定アクセスモードはREADONLYからEXECUTEまでの9つのアクセスモードを持つ（**表1**）．

表1 限定アクセスモード
Table 1 Limited Access Mode (LAM).

モード	説明
READONLY	読み出し専用
WRITE	書き込み専用
APPEND	追加
CREATE	新規作成
DELETE	削除
LINK	リンク作成・削除
MODIFY	ファイル情報の修正
STATUS	ファイル情報の参照
EXECUTE	実行

限定アクセスモードを導入したLOMACを定式化すると以下のようになる．

サブジェクトSとオブジェクトOはそれぞれセキュリティレベルと限定アクセスモードからなるセキュリティラベルを持つ．サブジェクトSのセキュリティラベル R_s を $R_s = (l_s, m_s)$ とおく．ここで l_s はSのセキュリティレベル，m_s はSの限定アクセスモードとする．セキュリティレベルにはLOW_LEVELとHIGH_LEVELの2つのレベルがある．オブジェクトOについても同様に $R_o = (l_o, m_o)$ と定義する．このときアクセス制御を行うためのセキュリティラベル評価関数 $F(R_s, R_o)$ を以下のように定義する．

$F(R_s, R_o) :=$
if $(l_s == l_o)$ permit;
else if $(l_s > l_o)\ l_s = l_o$; permit;
else if $(m_s == m_o)$ permit;
else prohibit;

このセキュリティラベル評価関数Fによってアクセス制御が行われる．

3.2 アクセス制御例

3.1節に従った具体的なアクセス制御の例を示す．ここではsubject S_1 がobject O_1 にアクセスする例とsubject S_2 がobject O_2 にアクセスする例を示す．

［例1］
$Rs_1 = (LOW\_LEVEL, EXECUTE)$
$Ro_1 = (HIGH\_LEVEL, EXECUTE)$
（ただし，$HIGH\_LEVEL > LOW\_LEVEL$）

S_1 のセキュリティレベルが O_1 のセキュリティレベルより低いのでアクセスが拒否されるが，O_1 がEXECUTEだけ低いレベルのサブジェクトに許可しているため，S_1 は O_1 をEXECUTEすることができる．ただしWRITEやLINKなどを S_1 が行うことはで

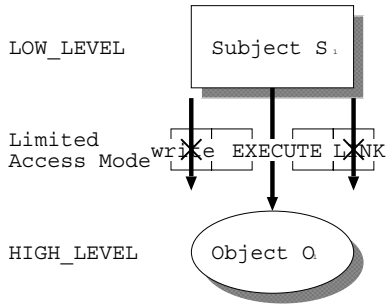

図 1 アクセス制御例（1）
Fig. 1 Example of access control (1).

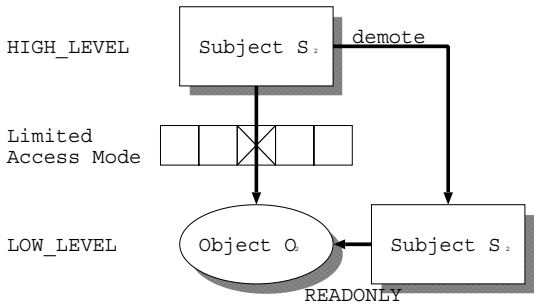

図 2 アクセス制御例（2）
Fig. 2 Example of access control (2).

きない（**図 1**）.

［例 2］

$Rs_2 = (HIGH\_LEVEL, READONLY)$

$Ro_2 = (LOW\_LEVEL, READONLY)$

S_2 は O_2 よりも高いレベルにあるため，O_2 へのアクセスは許可される．しかし自分より低いレベルのオブジェクトにアクセスした場合は S_2 のレベルを以後保証することはできない．たとえば O_2 がシェアードライブラリの場合を考える．LOW_LEVEL のユーザによってシェアードライブラリが不正なものと置き換えられてしまっていた場合，S_2 は不正な動作を行うことになる．もし S_2 が高いレベルで動作するならば保護対象に大きな被害を与えることになる．そのため，レベルが低いオブジェクトにアクセスする際は S_2 が降格され $Rs_2 = (LOW\_LEVEL, READONLY)$ となる（**図 2**）.

3.3 システム構成

限定アクセスモードによって拡張した LOMAC を E-LOMAC と呼ぶ．我々は E-LOMAC を Linux（Redhat5.2）上に実装した．実装した機構は**図 3** に示すように，2 つのテーブルと 3 つのモジュールから成る．詳細は以下のとおりである．

（1） OLM（Object Level Map）
オブジェクトに対応したセキュリティラベルを保持したテーブル．オブジェクトを識別するために i-node とデバイス番号を利用する．セキュリティラベルとしてはセキュリティレベルと限定アクセスモードを持つ．

（2） ULM（User Level Map）
サブジェクトのセキュリティラベルを保持したテーブル．UID とセキュリティラベルを持つ．

（3） ACM（Access Control Module）
必須アクセス制御のコントロールモジュール．LOMAC に限定アクセスモードを導入したルールを実装したモジュール．サブジェクト，オブジェクト間のアクセス制御を行う．

（4） LM（Labeling Module）
ULM に従ってすべてのサブジェクトにセキュリティラベルを設定する．ラベルの設定は以下のように行う．
（a） ログイン時のユーザ ID を監査 ID として保持しておく．
（b） 保持しておいた監査 ID と ULM によってセキュリティラベルを設定する．
（c） fork を行った場合などは親のセキュリティラベルを引き継ぐ．
（d） su コマンドを用いて実行ユーザ ID を変更しても，セキュリティラベルは変更されない．

（5） E-LOMAC インタフェース
OLM や ULM のマッピングを変更するためのカーネルインタフェースを提供する．

このシステムではすべてのシステムコールはアクセス制御を実施するために疑似システムコール（pseudo system call）で置き換えられる．プロセスがシステムコールを実行した場合はこの疑似システムコールが呼び出され，次にアクセス制御モジュール（ACM）が呼び出される．ACM は OLM を参照し，アクセス制御を行う．もしアクセスが許可される場合には本来のシステムコール（real system call）が呼ばれ，許可されない場合はアクセス権限エラーが返される．プロセスへのラベルづけはプロセスが生成された時点で行われる．プロセスは execve コマンドを通じて生成されるので execve が実行された場合には，ACM が LM を呼び出し，ULM を参照してサブジェクトへのラベルづけを実行する．これらすべてはカーネル内部のモジュールとして動作し，ユーザレベルからは E-LOMAC インタフェースを用いて内部情報を変更することができる．

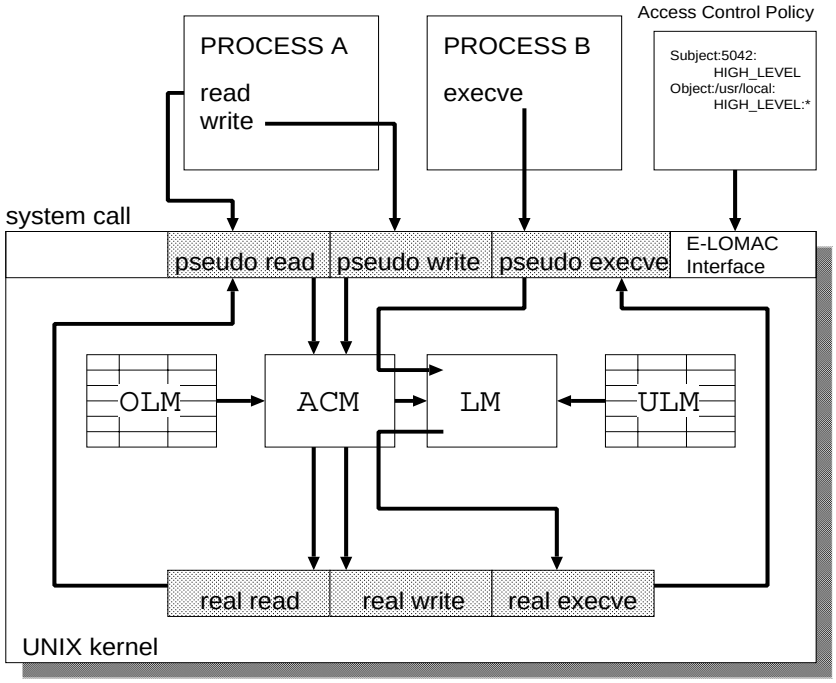

図 3 システム構成
Fig. 3 Structure of system.

3.4 アクセス制御ポリシー

サブジェクト，オブジェクトのセキュリティラベルはアクセス制御ポリシーという形でファイルに記述される．このポリシーはシステムの起動時に読み込まれ，E-LOMAC インタフェースを通じて OLM，ULM にセットされアクセス制御対象が決定される．このファイルは HIGH_LEVEL に属するものとして登録され保護される．アクセス制御ポリシーの記述例を**図 4** に示す．

```
Subject:5046:HIGH_LEVEL
Subject:5010:LOW_LEVEL
Object:/usr/local/bin/ida:HIGH_LEVEL:*
Object:/usr/lib:HIGH_LEVEL:READONLY
Object:/usr/local/ida/log:HIGH_LEVEL:APPEND
```

図 4 アクセス制御ポリシー
Fig. 4 Access control policy.

ここでは Subject はサブジェクトのセキュリティレベルを定義するためのラベル，Object ではオブジェトのセキュリティレベルを定義するためのラベルである．Subject に続く数値はユーザ ID である．ここで指定されたユーザ ID のユーザによって起動されたプロセスが指定したレベルで動作することとなる．また Subject の限定アクセスモードはユーザが明示的に指定できない．Subject の限定アクセスモードは実際にオブジェクトを操作しようとする際にカーネルによって自動的に割り振られる．たとえば，stat システムコールを実行した場合には，“STATUS” という限定アクセスモードが自動的にサブジェクトに設定される．5 行目の項目は：/usr/local/ida/log ディレクトリのセキュリティラベルを (HIGH_LEVEL, APPEND) に設定している．ここで明示的に設定されないオブジェクトおよびサブジェクトはすべて LOW_LEVEL に設定される．

この設定ファイルをカーネルに反映させることができるのは root ではない．root が操作可能とした場合には，E-LOMAC そのものが侵入者によって攻撃される可能性があるためである．E-LOMAC ではポリシーを操作することのできる特権ユーザを root とは別に用意している．このユーザへはコンソールからログインした root で，かつ 128 文字以下のパスワードを入力することでスイッチすることが可能である．

3.5 必須アクセス制御の影響範囲

本アクセス制御を導入した場合，その設定によっては他のシステム（アプリケーション）の動作に影響を与える．たとえば，/tmp 以下のファイルを HIGH_LEVEL に設定した場合，LOW_LEVEL のユーザはログインすることができなくなってしまう．ログイン時には login プロセスが/tmp の下のログファイルに情報を書

きこもうとするが，/tmp 以下は HIGHT_LEVEL なので，LOW_LEVEL のユーザは情報を書き出すことはできない．こういった事例は必須アクセス制御の設定の困難さを示すものである．仮に今回導入した限定アクセスモードを導入しなかった場合，この問題を解決するためには，/tmp 以下は必ず LOW_LEVEL にするか，ログインするユーザをすべて HIGH_LEVEL に設定する必要がある．両方のアプローチとも十分なアクセス制御を実施することはできない．しかしながら限定アクセスモードの導入により，/tmp 以下を HIGH_LEVEL に設定したとしても，以下のファイルに対して，追加書き込みのみの限定したアクセスを許可することで，/tmp 以下のファイルを安全に保ちながら，ログインなどの通常の動作に影響を与えないように設定することが可能となる．

4. 評価と考察

本章では実装した機能の評価および考察を行う．

4.1 IDS 保護機能の評価

E-LOMAC を導入した場合の有効性を評価するために，数種類の疑似攻撃を数種類の IDS に対して行い，本機能の利用によって実際に侵入判定にどのような影響があるか実験した．実験では以下の 3 種類の IDS を用いた．

(1) IDA
モバイルエージェントを用いて情報の収集を行い，侵入判定をマネージャと呼ばれる 1 つの判定エンジンで解析する IDS である[12]．侵入行為の検出においては「痕跡」と呼ばれる侵入者によって共通的に残されるログをトリガとして，侵入判定のコストおよび誤認識を最低限に抑えるように設計されている．

(2) ASAX
ルールベースの言語を用いてホストのログと侵入行為のパターンマッチングを行い侵入の判定を行うためのエキスパートシステムである[13]．RUSEL と呼ばれる独自のスクリプト言語を持ち，これによってルールを記述する．基本的に 1 つのマシン上で動作する．

(3) AAFID
エージェント技術を用いて複数台のマシン上で侵入の検出を行うための IDS である[14]．各ターゲットマシン上には，侵入パターンごとに侵入の判定を行うエージェントと呼ばれるプログラムが常駐する．このエージェントは侵入を検出するとネットワークにおかれたモニタと呼ばれるデーモンに侵入の情報を受け渡す．モニタは複数のターゲットマシンを監視しており，状況に応じて情報を管理者に報告する．

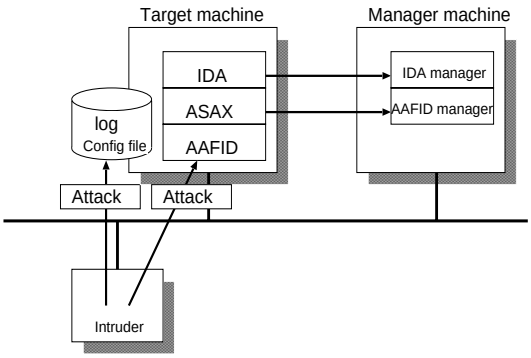

図 5 実験環境
Fig. 5 Test environment.

これらの IDS を**図 5** に示す状態に配置した．これらの IDS に以下の侵入手法で攻撃を加え，侵入判定時の状態を調査した．

攻撃 1 IDS プロセスの停止：
setuid されたプログラムをバッファオーバフローさせ特権モードを取得し，直後に IDS のプロセスを kill コマンドを用いて停止させる．

攻撃 2 ログファイル（ログ出力プロセス）の削除：
setuid されたプログラムをバッファオーバフローさせて特権モードを取得し，直後に IDS が利用しているログファイルを削除またはログ出力プロセスを停止する．

攻撃 3 ログファイルの改竄：
setuid されたプログラムをバッファオーバフローさせて特権モードを取得し，直後に IDS が利用しているログファイルに不正なデータを書き込む．

攻撃 4 IDS システムファイルの削除：
setuid されたプログラムのバグをついて IDS のシステムファイルのモードを変更し，削除してしまう．

E-LOMAC を導入しなかった場合のテスト結果を**表 2** に，導入した場合の結果を**表 3** に示す．IDS 保護機能を利用しなかった場合，IDA と AAFID は攻撃 1 によって，侵入を検出する前にプロセスとして停止してしまった．ASAX は侵入を検知したものの，プロセスは検出後に停止した．攻撃 2 によって全 IDS のプロセスは入力待ち状態になってしまった．IDA だけはその後プロセスが停止した．すべての IDS はログファイルが削除されたことを検知することはできなかった．攻撃 3 によってログを改竄した場合，IDA 以

表 2 テスト結果 1（IDS 保護機能なし）
Table 2 Result of test 1 (without E-LOMAC).

	攻撃 1	攻撃 2	攻撃 3	攻撃 4
IDA	N ⇒ S	N ⇒ S	N	D
ASAX	D ⇒ S	N	D	D
AAFID	N ⇒ S	N	D	D

* 停止：S 検出：D 検出不能 N

表 3 テスト結果 2（IDS 保護機能利用）
Table 3 Result of test 2 (using E-ROMAC).

	攻撃 1	攻撃 2	攻撃 3	攻撃 4
IDA	D	D	N	D
ASAX	D	D	D	D
AAFID	D	D	D	D

外は侵入を検出することができた．攻撃 4 は侵入判定プロセスに特に支障をきたすことはなかった．

IDS 保護機能を利用した場合，IDA に対する攻撃 3 以外はすべての攻撃から保護され，侵入の判定を正常に行った．

以上の実験を通じて明らかになったことをまとめると以下のようになる．

- 作成した IDS 保護機能によって IDS のプロセス自身が保護され，侵入によって特権ユーザ権限が奪われても安全に動作した．
- IDS 保護機能によってログなどの重要なファイルが保護された．
- IDA 以外の IDS では保護機能によって攻撃 3 を避けることができた．ただしログ自身は削除されないため，侵入の判定は可能であるが，ログの内容のチェックを正確に行っていない場合は，システムがパニックを起こし，侵入の判定を正常に行うことができない．

このように IDS 自身を攻撃された場合に生じる侵入検出が行えない状況を，提案した IDS 保護機能によってある程度回避できることが示された．

4.2 パフォーマンス評価

カーネルレベルでの必須アクセス制御を用いた IDS 保護機構は，システムコールレベルでアクセス制御を行うため，システム全体のパフォーマンスに影響を与える．ここでは 2 つのベンチマークの結果を**表 4** に示す．ベンチマーク 1 は execve を 1,000 回繰り返し，サブジェクトへのレベル設定のパフォーマンスを見るものである．ベンチマーク 2 はファイルをオープンし，1,000 バイトのデータを書き込み，クローズするものを 100,000 回繰り返す．制御のパフォーマンスを見るためのものである．実験環境は Pentium 333 MHz，メモリ 128 MB，OS: RedHat5.2 である．

表 4 ベンチマーク結果
Table 4 Result of benchmark tests.

test	without E-LOMAC	E-LOMAC
Benchmark 1	31.379 sec	31.913 sec
Benchmark 2	4.326 sec	5.077 sec

ベンチマークの結果より，アクセス制御にかかる時間は実用上問題のあるものではないことがうかがえる．

4.3 考　　察

E-LOMAC を導入する利点は以下である．

(1) カーネルレベルでの保護
UNIX のデフォルトのアクセス制御ではいったん特権ユーザ権限が奪取されてしまうと，ファイルやプロセスを保護する手段はない．E-LOMAC ではアクセス制御ポリシーに基づきカーネルレベルでアクセス制御を行うため，たとえ特権を奪取されても保護対象を安全に保つことができる．つまり E-LOMAC でのアクセス制御は，特権ユーザのアクティビティも厳密に制御することを可能にする．

(2) 統一的な保護アプローチ
従来のアプローチではアプリケーションごと，または保護対象ごとに保護のための仕組みを用意する必要があったが，E-LOMAC ではアクセス制御という統一したメカニズムで保護が可能となる．また IDS ごとに保護機能を実装する必要もなく，現在存在するホストベース IDS の多くに対してコードの変更なく適用することが可能となる．

(3) 保護メカニズムのオーバヘッド
従来のアプローチではアプリケーションレベルで複雑な保護機能を実装するものが多く，保護のために多くのオーバヘッドが生じていたが，E-LOMAC ではアプリケーションレベルでの保護に比べてカーネルレベルで単純な制御機能が働くため，処理のオーバヘッドが少なく，高速である．

一方，以下のような問題点もある．

(1) 設定の複雑さ
E-LOMAC ではアクセス制御ポリシーによって保護対象の保護方法を規定する．このポリシーを記述するためには，対象となるアプリケーションがどのようなファイルやプロセスに，どのような形でアクセスするのかを十分に把握している必要性がある．これを完全に把握することは管理者にとって容易ではなく，ポリシーの記述には熟練度が必要となる．さらにポリシーを正

確に記述することが保護の正確さを高めるうえで非常に重要である．システムが自動的にポリシーを生成するための機能が今後重要となる．

(2) 制限

実験結果で示したように，今回提案した機構では，ログファイルへの不正なデータの書き込みを避けることができないという制限がある．ログファイルはすべてのアプリケーションから書き込みが可能でなければならないという特性を持っている．実験ではログファイルをHIGH_LEVELに設定し，限定アクセスモードとしてAPPENDを設定したが，この設定では追加書き込みが可能なため，悪意を持つユーザがなんらかの不正なデータ（一貫性のないログ）を書き込むことを防ぐことができない．そのため，ログの一貫性を前提したIDSでは，正常なログの判断ができない．本機能を導入したとしても，ログの一貫性に関してはIDS自身で対応しなければならない．

5. まとめ

本論文では必須アクセス制御方式を用いたIDSのための保護機構を提案し，試作システムを構築した．本手法によって，IDS自身に対する攻撃からある程度IDSを守ることが可能であることが示せた．またアクセス制御ポリシーという統一的な方法を用いてファイル，プロセスといった区別なく，保護方法を規定できる．また提案手法は，さまざまなIDSに導入可能であることも示した．

今後は試作したシステムを完全に実装し，実環境で運用評価および改良を行う．また今回提案した手法と他の必須アクセス制御モデルの比較検討を行う．

参考文献

1) http://www-rnks.informatik.tu-cottbus.de/sobirey/ids.html.
2) Mukherjee, B., Herberlein, L.T. and Levitt, K.N.: Network intrusion detection, *IEEE Network*, Vol.8, No.3, pp.26–41 (1994).
3) 女部田武史，井上 直，浅香 緑：侵入検出システムにおけるセキュリティ機能の検討，電子情報通信学会情報セキュリティ研究会，研究報告ISEC2000-235-57, pp.181–188 (2000).
4) Bell, D.E. and Padula, L.L.: Secure Computer System: Unified Exposition and Multics Interpretation, Technical Report No.ESD-TR-75-306, Electronics Systems Division, AFSC, Hanscom AF Base, Bedford MA (1976).
5) Biba, K.J.: Integrity Considerations for Secure Computer Systems: Technical Report ESD-TR-76-372, Hanscom AF Base, Bedford MA (1977).
6) Ferraiolo, D. and Kuhn, R.: Role-Based Access Controls, *Proc. 15th National Computer Security Conference*, pp.554–563 (1987).
7) Brewer, D.F.C. and Nash, M.J.: The Chinese Wall Security Policy, *Proc. 1989 IEEE Symposium on Security and Privacy*, pp.206–214 (1989).
8) 高田哲司，小池英樹：逃げログ—削除まで考慮にいれたログ情報保護手法，情報処理学会コンピュータセキュリティ研究会，研究報告99-CSEC-5, pp.13–18 (1999).
9) Bhattacharya, S. and Ye, N.: Design of robust, survivable intrusion detection agent, *Proc. 1st Asia-Pacific Intelligent Agent Technology Conference* (*IAT'99*), pp.274–278 (1999).
10) Badger, L., Sterne, D.F., Sherman, D.L., Walker, K.M. and Haghighat, S.A.: A Domain and Type Enforcement UNIX Prototype, *Proc. 5th USENIX UNIX Security Symposium Salt Lake City* (1995).
11) Fraser, T.: LOMAC: Low Water-Mark Integrity Protection for COTS Environments, *Proc. 2000 IEEE Symposium on Security and Privacy* (2000).
12) Asaka, M., Tsuchiya, M., Onabuta, T., Okazawa, S. and Goto, S.: Local Attack Detection and Intrusion Route Tracing, *IEICE Trans. Comm.*, Vol.E82-B, No.11, pp.1826–1833 (1999).
13) Mounji, A. and Charlier, B.L.: Continuous Assessment of a Unix Configuration Integrating Intrusion Detection and Configuration Analysis, *Proc. ISOC'97 Symposium on Network and Distributed System Security*, San Diego (1997).
14) Balasubramaniyan, J.S., Garcia-Fernandez, J.O., lsacoff, D., Spafford, E. and Zamboni, D.: Architecture for Intrusion Detection using Autonomous Agents, COAST Technical Report, COAST Laboratory, Purdue University (1998).

(平成12年11月30日受付)
(平成13年 6月19日採録)

女部田武史（正会員）

平成8年北陸先端科学技術大学院大学博士前期課程修了.（株）日本総合研究所勤務．平成10年より情報処理振興事業協会に出向．侵入検出システムの研究に従事.

井上　　直（正会員）

平成8年北陸先端科学技術大学院大学博士前期課程修了.（株）SRA勤務．平成11年より情報処理振興事業協会に出向．侵入検出システムの研究に従事.

浅香　　緑（正会員）

昭和59年上智大学大学院博士前期課程修了.（株）日本総合研究所勤務．平成5年より情報処理振興事業協会に出向．電子情報通信学会，IEEE各会員.