コンピュータセキュリティ　10－26
（2000．　7．25）

侵入検出システムにおけるセキュリティ機能の検討

女部田　武史、　井上　直、　浅香　緑
情報処理振興事業協会　技術センター
〒113-6591 東京都文京区本駒込 2-28-8 文京グリーンコート　センターオフィス 16 階
Tel: 03-5978-7507　Fax: 03-5978-7517
E-Mail:　{t-onabu, t-inoue, asaka}@ipa.go.jp

あらまし　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盛んに行われ、商用レベルのものも登場してきた。しかしながら多くの侵入検出システムで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 IDS を保護するための機構を考案、実装したので報告する。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、IDS 用にカスタマイズしたものである。この機能によって IDS を他のプロセスから保護することが可能となり、確実に IDS は侵入判定を行うことができる。

キーワード　侵入検出システム、必須アクセス制御、LOMAC、限定アクセスモード

The Study of Security Mechanisms for Intrusion Detection System.

Takefumi Onabuta,　Tadashi Inoue,　Midori Asaka
Information-Technology Promotion Agency, Japan, Software Technology center
Bunkyo Green Court Center Office 2-28-8 Honkomagome, Bunkyo-ku Tokyo, 113-6591 Japan
Tel: +81-3-5978-7507　Fax: +81-3-5978-7517
E-Mail:　{t-onabu, t-inoue, asaka}@ipa.go.jp

Abstract　Increasing illegal access on the Internet, there are many research of intrusion detection and some commercial intrusion detection systems (IDSs). But researchers have no attention of IDS's security, because they are too active to study of technique or mechanism for detecting intrusions automatically. When an intruder breaks into a computer system, he or she first detects system logs and kills the auditing processes provided for security tools such as IDS. So IDS's security is as important as a method of intrusion detection. Our IDS called IDA (Intrusion Detection Agent system) has not been considered security so far. We discuss about some security functions to protect IDA and other IDSs.　We designed and implemented a protecting mechanism with LOMAC which is one of traditional mandatory access control models.　We provide a new access model based on extended LOMAC which is added limited access mode to original one, as a result we can enhance security of IDSs.

key words　*Intrusion Detection System, Mandatory Access Control, LOMAC, Limited Access Mode*

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など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 も登場してきた[1][2]。これまで IDS の研究は"侵入発見の正確さ"や"効率"に絞られていた。しかし、製品やフリーソフトとして IDS が実際の環境で利用させるようになってくると、「IDS 自身のセキュリティ」が大きな問題として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といわれている。侵入者によって IDS 自身が狙われた場合、システムが正常に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うした「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ウェイトを占める位置づけがされていない。

そこで本研究では、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。我々が提案する IDS 保護機能は、LOMAC と呼ばれる必須アクセス制御方式を IDS に適した形に拡張したもので、アクセス制御を通じて IDS を保護するものである。この機能によって IDS を他のプロセスから保護することが可能となり、確実に IDS は侵入判定を行うことができる。

本稿ではまず 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 のセキュリティ機能

2.1. セキュリティ要件

IDS 自身が攻撃の対象となった場合、正常に侵入の検知ができない場合がある。このような脅威から IDS を保護するためには、以下の 2 つの保護が非常に重要となる。

重要ファイル保護

IDS においては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となる。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。なぜならこれら 2 つは常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ないほか、例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もある。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。

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一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 保護のための従来アプローチ

IDS 自身のセキュリティを確保する枠組には以下の 2 つのアプローチが考えられる。

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と同じ。

1 のアプローチから IDS の保護機構を提案したものには[3][4]がある。[3]は IDS においてもっと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が、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[4]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上で機能する。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。[3]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、[4]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留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、例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。

そこで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、kernel-level-protection として IDS のための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。カーネル上にアクセス制御の仕組みを実装する理由は 2 つある。

1. アプリケーションレベルでは、保護機構に制限がある。例えば特権ユーザの権限が一旦奪われてしまえば、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毎に異なる保護方法を用いるのではなく、統一的な方法で IDS を保護する。

3. 必須アクセス制御を用いた IDS 保護機能

そこで本研究ではアクセス制御として必須アクセス制御を用いた IDS 保護機能を提案し、実際に Linux を用いて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 保護のための我々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 は通常のプロセスとは違った領域で動作し、通常のプロセスが一切 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。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。つまり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。そこでカーネルレベルでの必須アクセス制御(MAC)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[5][6]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、[7]によって提案された Low Water-Mark Model をベースとした LOMAC と呼ばれる必須アクセス制御方式を利用する。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、inode 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。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。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。そのため、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、これ以上高いレベルでの動作を許可しないようにする。

この方法をそのまま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からである。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、このモデルではアクセス制御を記述できない。そのような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

これらは IDS が利用するため、HIGH_LEVEL に設定すると、他のプロセスは利用できなくなる。しかしながら、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り、IDS の動作を他のプロセスと切り離せなくなる。そこで LOMAC に「限定アクセスモード」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」とは例えば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 8 つのアクセスモードを持つ。

READONLY(r)	読み出し専用
WRITE(w)	書き込み専用
APPEND(a)	追加
CREATE(c)	新規作成
DELETE(d)	削除
LINK(l)	リンク作成・削除
MODIFY(m)	ファイル情報へのアクセス
EXECUTE(x)	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。

サブジェクトとオブジェクトは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。セキュリティレベルは LOW_LEVEL と HIGH_LEVEL 2つのレベルを持つ。

ここで、あるサブジェクト s のセキュリティラベル R_s を $R_s=(l_s, m_s)$とおく。但し、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。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 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]
R_{s1} = (LOW_LEVEL, EXECUTE)
R_{o1} = (HIGH_LEVEL, EXECUTE)
(但し、HIGH_LEVEL > LOW_LEVEL)

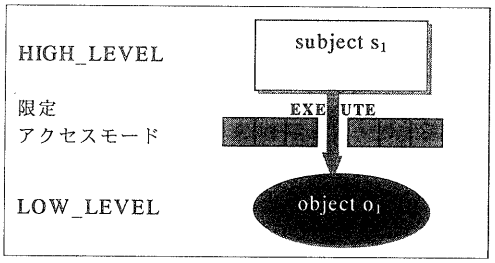

図 1 アクセス制御例 1

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 が行うことはできない。

[例 2]

R_{s2} = (HIGH_LEVEL, READONLY)

R_{o2} = (LOW_LEVEL, READONLY)

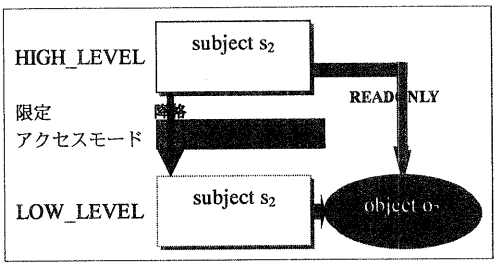

図 2 アクセス制御例 2

s_2 は o_2 よりも高いレベルにあるため、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。この場合 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(LOW_LEVEL, READONLY)となる。

3.3. 実装

我々は拡張した 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。詳細は以下の通りである。

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 inode 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. E-LOMAC (Extended LOMAC)
   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。
4. Label
   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。
   (ア) ログイン時のユーザ ID を監査 ID として保持しておく。
   (イ) ユーザ ID ではなく、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。
   (ウ) 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。
   (エ) 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 Interface
   OLM や ULM のマッピングを変更するためのカーネルインターフェースを提供する。

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 Interface を用いて内部情報を変更することができる。

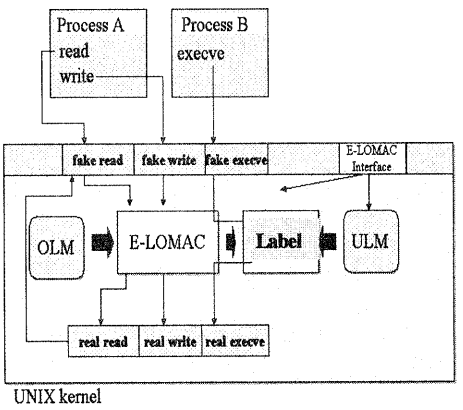

図 3 システム構成

3.3. 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、必須アクセス制御の対象が決定される。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以下に示す。

```
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
```

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ラベルを定義するためのラベルである。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。

4. 評価と考察

本章では実装した機能の評価を行い、本機能の利点と欠点について考察する。

4.1. IDS 保護評価

本機能を導入した場合の効果を測定するために、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調査を行った。実験では以下の 3 種類の IDS を用いた。

1. IDA (IPA)
   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一つの判定エンジンで解析する[8]。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ーとして、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 (FUNDP)
   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[9]。RUSSEL 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。基本的に一つのマシン上で動作する。
3. AAFID (COAST)
   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侵入検出システムである[10]。各ターゲットマシン上には、侵入事例毎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ーと呼ばれるデーモンに侵入の情報を受け渡す。モニター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。

これらの IDS を図 4 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。

① IDS プロセスの停止
   setuid されたプログラムをバッファーオーバーフローさせ特権モードを取得すると同時に IDS のプロセスを kill コマンドを用いて停止させる。
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ーオーバフローさせて特権モードを取得すると同時に 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。
③ ログファイルの改竄
   setuid されたプログラムをバッファーオーバフローさせて特権モードを取得すると同時に 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。
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 のログファイルのモードを書き込みモードに変更し、削除してしまう。

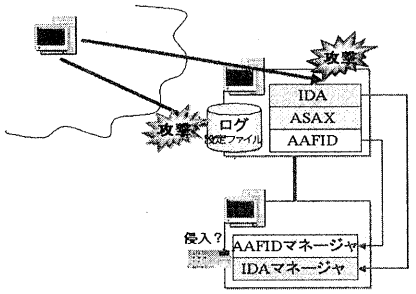

図 4 実験環境

4.2. テスト結果

提案した機能を利用しなかった場合のテスト結果を表 1 に、利用した場合の結果を表 2 に示す。IDS 保護機能を利用しなかった場合、IDA と AAFID は擬似攻撃①によって侵入を検出する前にプロセスとして終了してしまった。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。擬似攻撃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。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。擬似攻撃③によってログを改竄した場合、IDA 以外は侵入を検出することができた。擬似攻撃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。

	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 1 テスト結果（IDS 保護機能なし）

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。

	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

表 2 テスト結果（IDS 保護機能利用）

以上の実験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。
- IDS 保護機能によっても攻撃③を避けることはできなかった。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正常に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。

4.3. 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 2 に示す。ベンチマーク 1 は execve を 1000 回繰り返し、subject 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、1000 バイトのデータを書き込み、クローズするものを 100000 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 II 333Mhz、メモリー 128M、OS:RedHat5.2 である。

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

表 3 ベンチマーク結果

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伺える。

4.3. 考察

本機能を利用する利点は以下のようである。

1. カーネルレベルでの保護
   カーネル内部のアクセス制御によって保護を行うため、例え特権ユーザ権限を奪取されても保護機構が正常に働く。
2. 統一的な保護アプローチ
   ログファイルの保護、IDS プロセスの保護に対してアクセス制御という統一した方法が適用可能である。
3. 保護メカニズムのオーバーヘッド
   アプリケーションレベルでの保護に比べて全体にアクセス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。

逆に以下のような問題点もある。

1. 設定の複雑さ
   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が、限定アクセスモードの導入によって設定がより複雑になり、ポリシーの記述には熟練度が必要となる。

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 自身に対する致命的な攻撃からある程度身を守ることが可能であることを示した。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枠組みによって、さまざまな IDS のために利用することが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するまた実環境で IDS とともに運用し、その評価および改良を行う。

参考文献

[1] http://www-rnks.informatik.tu-cottbus.de/~sobirey/ids.html

[2] B. Mukherjee, L.T. Herberlein, and K.N. Levitt, "Network intrusion detection.", IEEE Network, vol.8 no.3, pp.26-41,May/June 1994.

[3] 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, p.13-18, 1999.

[4] S.Bhattacharya, and Nong Ye, "Design of robust, survivable intrusion detection agent", in Proceedings of the 1st Asia-Pacific Intelligent Agent Technology Conference (IAT'99), pp.274 - 278, 1999.

[5] D.E. Bell and L. La Padula, "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)

[6] L. Badger, D.F. Sterne, D.L. Sherman, K.M. Walker and S.A. Haghighat, "A Domain and Type Enforcement UNIX Prototype", proceedings oh the Fifth USENIX UNIX Security Symposium Salt Lake City, Utah, June, 1995.

[7] Tim Fraser, "LOMAC - Low Water-Mark Mandatory Access Control for Linux", The 8th USENIX Security Symposium Washington D.C., August, 1999.

[8] M. Asaka, M. Tsuchiya, T. Onabuta, S. Okazawa, and S. Goto,"Local Attack Detection and Intrusion Route Tracing," IEICE Transaction on Communications, Vol.E82-B No.11, pp.1826-1833, November 1999.

[9] A Mounji, B Le Charlier, "Continuous Assessment of a Unix Configuration Integrating Intrusion Detection and Configuration Analysis", In Proceedings of the ISOC'97 Symposium on Network and Distributed System Security San Diego, California, 1997.

[10] J.S.Balasubramaniyan, J.O.Garcia-Fernandez, D.Isacoff, E.Spafford,and D. Zamboni, "Architecture for Intrusion Detection using Autonomous Agents," COAST Technical Report, COAST Laboratory, Purdue University,1998.