OSS による IPS の実現

桝本　圭†　　原田　季栄†
†E-mail：{masumotok, haradats}@nttdata.co.jp

概要

侵入防止機能とは、コンピュータシステムに対する攻撃が行われた場合に、システムを自動的に防御する機能である。通常、侵入検知システムの補足機能として実現される。侵入防止機能は、システムの安全性を高めるために、人手による常時システム監視を必要としないため、運用コストを削減する利点を提供する。しかし、侵入防止機能が侵入検知システム毎に開発されることや、また侵入防止を行うための設定に応用がきかない課題が残る。ここでは、これらの課題を解決し、侵入防止機能を持つシステムを実現する手法と、提案手法の実装内容について述べる。この手法により、任意の侵入検知システムを無改造で組合せた形式の侵入防止機能が実現可能となる。

１．はじめに

侵入検知システム（Intrusion Detection System：IDS）は、コンピュータシステムに対する侵入行為を検知し、システム管理者に報告する機能を持つ。システム管理者は、IDS の報告を見て、適切な対処を行うことができる。

また近年 IDS は、侵入検知後にコンピュータシステムを防御する機能を備えるようになった。この機能は、侵入防止機能と呼ばれ、例としてファイアウォールを閉じる、アカウントをロックアウトすることがある。侵入防止機能を備えた IDS を侵入防止システム（Intrusion Prevention System：IPS）と呼ぶ。

IDS や IPS の実現例は、オープンソースソフトウェア（Open Source Software：OSS）でも数多く存在する。IDS としては、snort[1]、tripwire[2]、modsecurity[3]等がよく知られており、また IPS としては、snort に侵入防止機能である snortsam[4]を補足する例がある。

これらの OSS に関する特徴は、多くの開発プロジェクトが優れた技術力を持つが、商用製品と比較すると局所的な技術開発になりがちになることである。そのため、例えば通信監視には snort、バッファオーバーフロー対策なら libsafe[5]のように、組合せて用いる。こうすることで、商用製品に劣らない不正侵入対策を安価に行うことができる。

しかし、これらの OSS の IDS に侵入防止機能を付与する場合には、課題が三点残されている。

1. **侵入防止機能に関する互換性の欠如** ― 侵入防止機能は、IDS が生成する警告を入力とする。そのため、警告に含まれる情報や警告形式が異なれば、それに応じた実装を必要とする。すなわち、互換性がないと言える。このため IDS のユーザは、開発コストをかけて、IDS 毎に侵入防止機能を実現しなければならない。
2. **定期的な検知の実施による被害の拡大** ― IDS の監視形態は、常時監視形式と定期検査形式に分類できる。前者の例は、常時通信内容を監視する IDS であり、後者の例は、一定時間毎にシステムに対する変更を検査する IDS である。このうち定期検査形式の IDS は、常時監視形式の IDS よりも強力な検知機能を持つ場合が多く、侵入検知・防止には有用である。しかし、侵入行為発生時から一定時間経過後に検査を行うため、侵入行為発生時にシステムを防御する形式と比較し、被害が大きくなる。そのため、定期検査形式の IDS を用いると、被害総量の把握が難しく、自動的なシステムの防御が困難である。
3. **侵入防止機能の設定に関する拡張性の欠如** ― IPS 導入時にユーザは、「ある警告に対して、特定の機能をもって侵入防止を行う」設定をシステム起動前に行う。それに対し人手による対策では、IDS の警告を見ると、保護対象のシステムの状況や、過去に発生した侵入行為等の事項を考慮して対処を行う。この対比により、従来の IPS の設定では、攻撃を受けたときのシステム状況等が考慮されないことがわかる。これらを考慮した防御を行うためには、IDS や IPS に対して、考慮事項に応じた改造が必要となる。そのため、拡張性が欠けていると言える。また、OSS に一旦改造を施すと、頻繁に発生するバー

†株式会社 NTT データオープンソース開発センタ
Open Source Software Development Center, NTT DATA Corporation.

ジョンアップへの追随が困難となる点からも、設定の拡張性を得ることは重要といえる。

ここでは、これらの課題を解決する「Response Manager（開発コード）」について述べる。Response Managerにより、以下の利点を得ることを目的とする。

1．**任意の IDS を無改造で組合せて IPS を構築可能** ― Response Managerでは、侵入防止機能の互換性を保ち、任意の IDS に適用できる共通の侵入防止機能を提供する。また、この共通防止機能を定期検査型の IDS も使用できるよう工夫を行う。そのため、Response Manager を用いると、異なる IDS を組合せて、IPS を構築可能となる。これにより、例えば通信を監視する IDS が暗号化通信は監視できないが、ローカルホスト内で監視を行う IDS が監視可能等、検知機能の強化が可能であり、システムを防御することも可能である。その他、攻撃者が IDS を回避するような攻撃を行う場合にも、複数の IDS を組合せて侵入行為を検知し、防御することも期待できる。

2．**侵入行為が発生した状況に応じて侵入防止を行う IPS の実現が可能** ― Response Managerは、事前に決定した設定だけでなく、過去に発生した侵入行為の履歴や、システム状況に応じて実施する侵入防止を決定する機能を提供する。このため、より攻撃を困難にさせる機能を持つ IPS が構築可能となる。また Response Manager では、侵入行為発生時の状況を判断する処理の内容に依存せずに、任意の処理を複数加えることを目標とする。そのため、高度な分析手法から、システム管理者の知見まで多様な目的の処理を自由に組合わせることができる。その他、新しい管理者の知見が生じた場合や、判断したい状況が増えた場合にも、既存の判断処理に変更を加えることなく IPS に組み込み、機能強化する利点も得られる。具体的に実現した事例や、想定される応用例については４章で述べる。

２．実現方式

２．１　課題の解決指針

前節に述べた三つの課題を、Response Managerは以下の指針により解決する。具体的な方法は２．２節以降で述べる。

1．**検知-防止間の調停** ― 侵入防止機能の互換性を保つため、異なる IDS を用いて内容や情報が異なる警告を受けた場合にも、侵入防止機能への入力を等しく整形するインタフェースを用意する。また、このインタフェースは、警告内に侵入防止機能への入力情報が無い場合には、自動的に補完を行う役割も果たす。このため、警告形式や内容によらず、任意の IDS が侵入防止機能を使用可能となる。

2．**リアルタイムな検知機能の付与** ― 定期検査を行う IDS に対して、リアルタイムに侵入を検知させる機能を用意する。この機能は、従来定期的に実行されていた IDS を、実行すべきタイミングに実行する役割を果たす。このため、定期検査型の IDS を侵入防止目的に適用可能となり、任意の IDS を用いて IPS を構築可能となる。

3．**動的な設定ルールの変更** ― ユーザが事前に行った「ある警告に対して、特定の機能により侵入防止を行う」ルールを初期値として、それらを動的に変更する機能を実現する。この機能は、サイト毎にユーザ任意の設定変更基準をモジュールのように加えることを可能とする。この機能により IPS は、ユーザ任意の変更判定処理に基づき、侵入行為発生時のシステム状況を把握し、侵入防御機能の実施を決定可能となる。

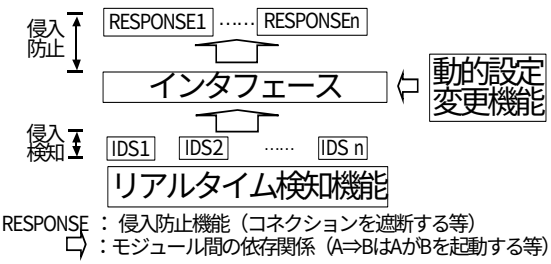

図 1 Response Manager 内部構造

図 1 に、これらの解決指針に基づいた Response Managerの実現イメージを示す。Response Managerは、複数用意した IDS と侵入防止機能の間に位置するインタフェースと、定期検査型の IDS にリアルタイム性を付与するモジュールと、動的な設定変更を行うモジュールを備える。またインタフェースは、警告が発生した場合に実施する侵入防止手段を決めるルールを保持する。適用する IDS と侵入防止機能は任意のものでよい。

また図２は、通常想定される IDS の運用形態における Response Manager の位置付けを示す。各ホストには、ファイルインテグリティチェッカーや設定チェックツール等、ローカルホスト内の侵入を検知するための IDS（ホストベース：HIDS）が複数ある。

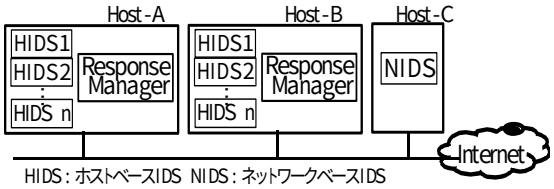

図 2 全体イメージ

通信を監視する IDS（ネットワークベース IDS：NIDS）には別途ホストを用意し、複数ホストを監視させる。この状況で Response Manager は、各ホスト単位に設置され、IDS が侵入を検知した場合に生成する警告を収集し、侵入防止を行う。

３．２　検知－防止間のインタフェース

インタフェースを実現するための要件は、IDS の警告の相違に対応できる一般性である。すなわち、異なる IDS を使用する場合や、IDS のバージョンアップが発生した場合には、警告内容に違いが生じる。このような場合にも、インタフェースは、侵入防止機能への入力を均一にしなければならない。

この要件から実現方法は、IDS と侵入防止機能の差異情報を設定として読込み、それらを参照しながらパターンマッチングを行うこととした。こうすることで、IDS 毎に侵入防止機能を開発せずとも、差異情報の変更により警告の差異に対応可能となる。具体的な設定項目を図3～図5に示す。

図3、図4は、IDS 毎に定義する設定である。複数の IDS を使用する場合は、図3、図4に示す設定を IDS の数だけ用意する。インタフェースでは、ソケット、パイプやファイルモニタを IDS 毎に警告受信用として割当てるため、これらの警告受信用のリソースと、IDS 毎の設定をペアにして使い分ける。

図 3 は、IDS の警告の差異情報を定義する設定（Alert Format Definition：AFD）である。IDS の警告に含まれる情報の種別と、情報を抽出するための正規表現を ’ ” ’（ダブルクォーテーション）で囲んだ形式で記述する。情報種別と正規表現のペアは最大三つまで ’|’（パイプ）で区切ることで記述できる。このペアを複数記述した場合には、前の正規表現のマッチング結果に対して、さらにマッチングを行うことになる。なお、情報種別には攻撃元/攻撃先 IP アドレス、攻撃元/攻撃先ポート番号、攻撃元/攻撃先 MAC アドレス、攻撃者のユーザ ID、攻撃が行われた時間等を記述可能である。

図 4 は、IDS が警告を生成したときに実行する、侵入防止機能を指定するルール（レスポンス・ポリシ）である。

```
[ 設定ルール定義 ]
AFD ::= AFD_RULE
        | AFD_RULE "|" AFD_RULE
        | AFD_RULE "|" AFD_RULE "|" AFD_RULE
AFD_RULE ::= 情報種別 "=" "¥"" 正規表現 "¥""

[ 設定例 ]
SIP = "{[1-9][0-9]{,2}.¥}{3}[1-9][0-9]{,2}
DPORT = "-> .*:[1-9][0-9]{,4}" | "[1-9][0-9]{,4}"
PROTOCOL = "TCP|UDP|HTTP|ARP"
```

図 3　警告の差異情報の定義

```
[ 設定ルール定義 ]
EXR ::= Rule "->" "[" 正数 "]"
RDR ::= Rule "->" "[" 正数 ("," 正数)* "]"
Rule ::= Parts
        | Parts "&&" Parts
        | Parts "&&" Parts "&&" Parts
Parts ::= ( 情報種別 "=" "¥"" 正規表現 "¥"" ) | "*"

[ 設定例 ]
Exception Rules Start
  SIP="192.168.2.[0-9]{1,3}" -> [2]
Exception Rules End
Response Decision Rules Start
  DPORT="80" && PROTOCOL="TCP" -> [1,2]
Response Decision Rules End
```

図 4　レスポンス・ポリシの設定

```
RESPONSE_ID      1            /* Identification No */
RESPONSE_NAME    "FW_CLOSE"
CANCEL_OPS       YES          /* Cancel operation */
CANCEL_RES_ID    6            /* Cancel operation ID */
EXPIRED_TIME     60
COMMAND_LINE     "/sbin/iptables -A INPUT -j -s $SIP -j drop"
                              /* Execute command line */
RESPONSE_ID      2
RESPONSE_NAME    "RST_CONN_TCP"
CANCEL_OPS       NO
COMMAND_LINE     "/sbin/user_made_program $SIP $SPORT"
```

図 5　侵入防止機能用の設定例

このルールは二つあり、実行する侵入防止機能を決定する Response Decision Rules（RDR）と、その例外を指定する Exception Rules（EXR）である。ユーザは Response Decision Rules において、実行する侵入防止機能を ’->’ の後で指定する。この番号は、図5の Response ID であり、侵入防止機能毎に割振られる数値である。また、Exception Rules では、Response Decision Rule の例外を ’->’ の後に指定する。例えば [3] とすれば、Response Decision Rules の2番目までのルールはスキップされ、マッチングの対象外となる。なお、これら二つのルールでは、“ “*” -> [1] “ のようにアスタリスクが記述可能である。この場合、全ての場合がマッチする意味となり、デフォルトルールとして用いることができる。

図5は、侵入防止機能毎の設定である。各侵入防止機能には ID、実行後停止することが必要か否か、実行するために必要な入力形式を記載する。

次に図3～図5を例に、インタフェースの動作を述べる。インタフェースは、IDS から警告を受ける

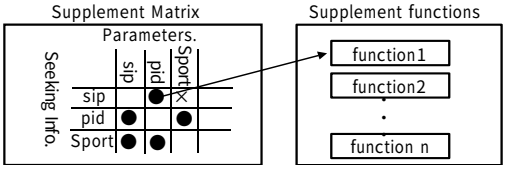

図 6　自動補完に用いる内部構造

と Exception Rule を参照しマッチングを行う。図 4 における Exception Rule の一行目は、SIP（Source IP：攻撃元 IP アドレス）が正規表現にマッチすれば、Response Decision Rule の二行目以前はマッチングの対象外となる。図 4 では、Response Decision Rule の二行目はないため、処理を終了する。

このようなマッチングを行う際に、ルールの左辺は"SIP=" とあるが、警告内における SIP の記述位置は IDS 毎に異なる。そこで図 3 に示す差異情報を参照し、警告内の SIP を得ている。

次に、Response Decision Rule を参照する。一行目は、DPORT（Destination Port：攻撃先ポート番号）が 80 なら、侵入防止機能 1、2 を実行する意味となる。このルールにマッチすれば、侵入防止機能を実行する。実行時には、図 5 の COMMAND LINE を参照して、任意の侵入防止機能を実行するための形式を得る。RESPONSE ID =1 の欄に$SIP とあるが、$が記載された場合には、警告内から図 3 の設定を参照して情報を得る。すなわち、まず図 4 を参照して防御実施を決定し、侵入防御機能への入力は図 5 を参照しながら整える、その際に逐次図 3 を参照する形式である。

これらの処理を行う上で、IDS の警告内に必要な情報が含まれない場合がある。例えばファイルの改ざんをチェックする IDS の警告には、改ざんを行った攻撃元 IP アドレスは通常含まれない。しかし、改ざんがリモートホストから行われたのなら、攻撃元ホストはアクセス拒否にするために、IP アドレスが必要となる。この場合にインタフェースは、不足している情報を自動的に補完する。

自動補完の実現は、インタフェースが図 6 の内部構造を保持することにより行う。図 6 の Supplement Matrix は、縦軸に補完する情報種類を示し、横軸は補完のために必要な情報種類を示す。Supplement Functions とは、補完を行う機能を果たすものであり、例えば、Linux において PID（攻撃者が攻撃に使用したプロセスの ID）から SIP を求めたい場合、簡易なものであれば netstat –ne でよい。

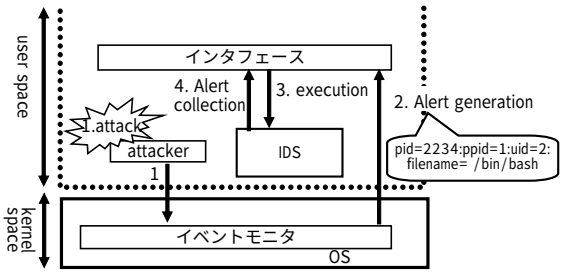

図 7　リアルタイム検知の実現方法

これらの構造を用いた自動補完の流れを述べる。インタフェースは、SIP_PID のように、Supplement Matrix の縦軸と横軸を連結した名称の実行ファイルを読込むことで、Supplement Matrix を起動時に構成する。IDS からの警告を受けた場合、逐次図 3 に示す IDS の警告情報を参照して、必要な情報が警告内に含まれないことを判断する。そこで、図 6 に示す Supplement Matrix を参照する。例えば SIP を補完したい場合、SIP の行を検索しながら補完するために必要な情報を探す。ここでは PID が該当するため、図 3 に示す警告情報内に PID の情報があれば、Supplement Function の入力とすることで SIP が得られる。

３．３　リアルタイム性の付与

定期検査型の IDS に対して、リアルタイム性を付与する機能を実現する上での要件は、IDS のアップデートに対応可能とするため、IDS 自身の改変を伴わないことである。また、このような IDS はホストホストベース IDS が大半であるため、本機能の対象はホストベース IDS としている。

これらの要件に基づく実現方法は、OS 内に発生するイベントを常時監視して、侵入の兆候が発生したことを契機に IDS を実行し、詳細検査後に侵入を検出することである。一般に、OS 内部に高度な侵入検知機能を組込むと、著しい性能劣化となる。そのため、OS 内部では侵入の兆候のみを検出し、その後は OS 外部で検査するこの方法は適当であると考える。

図 7 に、実現方法と動作を示す。OS 内部に設置されたイベントモニタは、リアルタイムに侵入の兆候を検出するため、システムコールを監視する。兆候が発生した際には、インタフェースに対して、図 7 に示す通知を行う。インタフェースは IDS を実行し、警告を収集する。イベントモニタがインタフェースに対して通知するファイル名は“ /bin -> [1,3] ”のように事前にルール定義し、イベントモニタに持たせておく。これは/bin 及び/bin 以下に変更があれば、IDS1 と IDS3 を用いて検査を行う意味となる。

なお、ホスト内部の侵入行為は、システム設定フ

ァイル等の、重要リソースの改変により検出可能であることが多い。そのため、侵入の兆候とは、事前に設定したファイル、ディレクトリ、シンボリックリンクへの変更としている。

３．４　動的なルール設定の変更

３．４．１ ルール設定の変更とは

機能の実現方法について論じる前に、「ルール設定の変更」を定義する。ここで述べる変更処理とは、特に有用と考えた以下四点である。

1. **侵入防止機能切替え** ー Response Decision Rule で指定した侵入防止機能を切替える。例えばTCPの攻撃はコネクション遮断する設定をしたが、数多く攻撃するIPアドレスに対して重いペナルティを課すために、ファイアウォールの設定変更を行い、一日アクセスを拒否する機能を実行するようにすることである。
2. **侵入防止機能追加** ー Response Decision Rule で指定した侵入防止機能を実行する際に、他の侵入防止機能も合わせて実行する。例えばローカルユーザが何度もシステムの設定ファイルの改ざんを行う場合、設定ファイルを復旧するのみとしていたが、さらに攻撃者の強制ログアウトも行う場合である。

実行期間延長 ー インタフェースは、ある侵入防止機能を実行する際に、図５の EXPIRED_TIME で示す時間を経過すると停止も行うが、その時間を延長する。例えばアカウントをロックアウトする期間を延長する場合に使用する。

時間制約付実行 ーある時刻から、一定時刻が経過するまで、ある侵入防止機能を実行する処理を繰り返し実行する処理である。例えば午前０時から６時まで、あるIPアドレスから来るアクセスを拒否するためファイアウォールを閉じる処理を、日毎に繰返すことである。

３．４．２　ルール設定変更機能の実現

この機能を実現する上での要件は二点ある。一点目に、設定変更基準は簡易なものから高度な分析手法まで組合わせて適用可能とするため、ユーザ任意の判断基準を複数付替え可能とすることである。二点目に、図４の Response Decision Rule を書き換えてしまうと、ユーザが行う初期設定の変更となり、後々ユーザに混乱を招くため、Response Decision Rule の記述自体に変更を加えないことである。

これらの要件に基づき、三つのモジュールにより機能を実現する。各モジュールの関係を図8に示す。

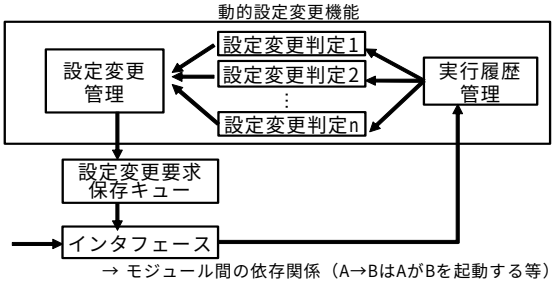

図 8 動的設定変更機能の実現方法

図８の実行管理マネージャは、侵入防止機能の実行状況を監視し、監視状況を変更判定モジュールに振分ける役割を果たす。変更判定モジュールは、設定を変更するタイミングを決定する役割を果たす。ユーザ任意の判定基準を複数加えることが可能である。設定変更決定時には、設定管理モジュールに対して変更要求を出す。設定管理モジュールは、複数の変更判定モジュールが決定した変更内容を管理し、実際の変更処理を行う役割を果たす。具体的な動作を、以下の状況を例として述べる。

状況 ー 自ネットワーク内のホストに対して Port 22（sshd）への調査行為（ポートスキャン）が多発した場合、新攻撃か新ワームの発生が考えられる。このとき、「調査行為発生時に、攻撃ホストからのコネクションを遮断する」設定があるならば、「攻撃ホストからのアクセスを一切拒否する」のように切替え、調査行為自体を禁止したい。また、その後行われるであろう攻撃も禁止したい。さらに調査行為が続くのであれば、通常より長時間アクセス拒否を行い、ssh のサービスを提供するホストの全 IP アドレスの露呈を防ぎたい。

この状況でポートスキャンが行われた場合、インタフェースは、侵入防止機能を実行後に、侵入防止履歴を実行履歴管理マネージャに送る。侵入防止履歴とは、自動補完機能を用いて可能な限り情報を補完した後に得られる Response ID： SIP：攻撃種類を示すメッセージ:…のような情報の羅列である。

実行履歴管理マネージャは、実行履歴を受けると、変更判定モジュール１と２に振分ける。ここでは、両モジュールは以下の動作を行うものとする。

変更判定モジュール １ ー 実行履歴を受ける度に、行われた Port 22 へのポートスキャンを数え、一分以内にある閾値を超えた場合には、攻撃ホストに対して実行する侵入防止機能を変更する要求を出す。

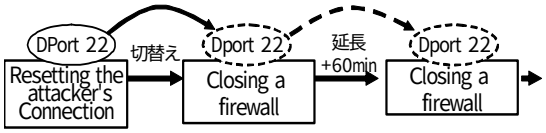

図 9 Configuration Manager が保持する設定

変更判定モジュール 2 ― 一日毎に、行われたポートスキャンを数える。その合計値が、前日までのポートスキャンの日毎の平均値より 20%以上増加していた場合には、Port 22 への攻撃に対して、実行する侵入防止機能を変更する要求を出す。モジュール 1 のみだと、間隔をあけてスキャンを継続的に行われた場合には対応できないため、モジュール2で補強する。

攻撃ホストからのポートスキャンが続いた場合、両モジュールは、設定管理モジュールに対して設定変更要求を出す。想定する状況において、初期値としてポートスキャンに対してはコネクション遮断することになっていることを考えると、要求形式は、「Port 22への攻撃に対してコネクション遮断の侵入防止を行う場合には、設定変更を行う」となる。設定変更後の情報である"ファイアウォールを閉じる"ことに"切替える"意味は含まない。設定変更後の情報は、設定管理モジュールが設定項目として保持する。これは、複数の変更判定モジュールが任意に要求を出したときに発生する矛盾を避けるためである。例えば、ある変更判定モジュールは実行期間延長を要求し、他モジュールは侵入防止機能の切替えを要求すると、実行時間を延長したはずが、侵入防止機能が切替わっている矛盾となる。このような矛盾を避けるため、設定変更後の情報は、設定変更管理モジュールが一箇所に集約し、保持している。

設定変更要求の処理方法を述べる。設定管理モジュールは、これらの要求に対して、図 9 に例示する遷移図に基づき、設定管理を行う。なお、図 9 の遷移図はユーザが行うものであり、設定管理モジュールは起動時に図を読込む。

想定する状況では、設定管理モジュールが、初めに変更判定モジュール 1 の要求を受けると、Dport（Destination Port：攻撃先ポート） 22 の属性を持つオブジェクトが現れ、遷移図に沿って設定変更が行われる。次に、変更判定モジュール 2 からの要求を受けるが、Dport 22 のオブジェクトは既に遷移図上に存在する。この場合には、そのオブジェクトを遷移図に沿って進める。すなわち、Dport 22 のオブジェクトは Closing a Firewall の延長となる。

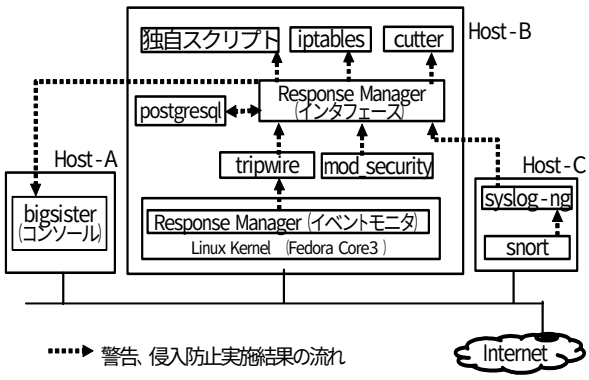

図 10 実装内容

これらの変更処理を行った後、設定管理マネージャは、インタフェースに対して設定変更処理を行う。具体的には、図 8 に示した、インタフェースとの間のキューに変更要求を挿入する。この処理に対してインタフェースは、侵入防止機能を実行する前に、キュー内に変更要求が含まれるか確認することで、実行する侵入防止機能を変更できる。

4．実装と利用例

4．1 実装

これまでに述べた機能について、実装を行った。図 10 に、実装内容を示す。開発言語は C 言語、OS は Fedora Core3 を用いた。イベントモニタは Linux Kernel Module として実現し、OS のソースコード自体の改造を伴わないようにした。また、動的設定変更機能内部では、postgresql[6]を用いた。IDS は snort、tripwire、mod_security を例として適用した。snort が生成する警告は、攻撃先 IP アドレスに応じて、syslog-ng[7]で振分けている。また、侵入防止機能として、iptables[8]、cutter[9]の他、強制ログアウト機能、アカウントロックアウトを行うスクリプトを独自に開発した。その他、管理コンソールとして bigsister[8]を用いている。なお、動的設定変更の例として実装を行ったものは、次節以降で述べる。

4．2 利用例１：改ざんされたリソースの即時復旧

tripwire は、事前取得したファイルのハッシュ値と、現在の値が異なるか定期的に検査する IDS である。ハッシュ値が異なる場合には、該当ファイル名を通知する。特に暗号化通信が行われるサービスを提供する場合、通信を監視する IDS は侵入行為を検知できない点からも、tripwire は有用である。しかし、tripwire は定期的に検査を行うため、例えばホームページが改ざんされた場合、検査を行う前にホームページ閲覧者に対して改ざんが露呈し、システ

ムの安全性について不信感を持たれてしまう。また、tripwire に常時検査をさせれば、システムに対して多大な性能劣化を招き、現実解とは言いがたい。さらに、tripwire 自身は侵入防止機能を持たないため、管理者が手動でファイルの復旧作業等も行う必要があり、また改ざんされたファイル名のみ通知するため、攻撃者に関する情報は得られない。

Response Manager を用いると、ファイルの改ざんが行われる兆候をリアルタイムに検知し、tripwire に検査を行わせることが可能である。そのため、改ざん行為が露呈する前に、改ざんされたファイルを復旧可能である。また、Response Manager では攻撃の兆候を検知した時のユーザ、プロセス情報も通知する。そのため、プロセス情報から改ざんを行った IP アドレスを自動補完することが可能となる。したがって、攻撃 IP アドレスをアクセス拒否とするようファイアウォールの設定変更を行い、改ざんに失敗した攻撃者が他の手段で攻撃することを防止可能である。また、図 4 に示した Exception Rule に正規の管理者がリモートからアクセスするホストの IP アドレスや MAC アドレスを記載すれば、管理者ホストからのファイルの更新を、侵入防止機能実行の例外とすることができるため、正規の管理者と攻撃者の区別をすることもできる。

４．３ 利用例２：攻撃の頻度に基づいた侵入防止機能の切替え

WWW プロキシを通じて多数のユーザが Web サーバにアクセスする状況を考える。また、プロキシ配下に存在するユーザの一人が攻撃者であり、サーバに攻撃を行うとする。このとき、管理者側では攻撃者にはサービス停止等のペナルティを与えたいが、プロキシ配下にいる一般ユーザへの影響は少なくしたい。

この場合には、攻撃を受ける度に、攻撃者のコネクションのみ切断してしまうことがよい。こうすることで、攻撃を受けたときに、サーバにアクセスしている攻撃者以外はアクセス可能となる。しかし攻撃者の観点からは、亜種攻撃プログラムや、IDS の検知を回避する攻撃を何度も試すことが可能である。そのため、攻撃者には重度のペナルティを与えたい。

そこで、ある一定期間において行われた攻撃頻度に応じて、実行する侵入防止機能を変更するモジュールを動的設定変更機能に加えた。このモジュールは、同じ攻撃元 IP アドレスが、ある一定の閾値以上の攻撃回数を超えると、コネクションを遮断する機能から、攻撃元 IP アドレスからの通信を拒否するようにファイアウォールの設定変更を行う機能に切替える要求を出す。なお、初期設定として、Response Decision Rule には“PROTOCOL=TCP && DPORT=80 -> [1]”のように Web サーバに対する TCP の攻撃であればコネクションを遮断する設定を記述する。

この機能により、攻撃者が早期に攻撃を諦めれば、Response Manager は攻撃を受ける度にコネクション遮断を行う。そのため、プロキシ配下の他ユーザにサービスを提供することができる。また、何度も攻撃手法を変える攻撃者であれば、攻撃回数が閾値を越えた時点で、攻撃元 IP アドレスからの通信をアクセス拒否にする。こうすることで、攻撃者以外のユーザに対するサービス提供妨害を抑えつつ、攻撃を防止する効果がある。攻撃者が複数の踏み台を使用した場合でも、攻撃を繰返す度に履歴は累積するため、攻撃が次第に困難となる効果が期待できる。

なお、ここでは TCP のコネクション遮断から、送信元 IP アドレスによる通信のアクセス拒否を例に実装したが、TCP のコネクション遮断 ⇒ 攻撃元 IP アドレス＋攻撃元ポート番号通信アクセス拒否 ⇒ 攻撃元 IP アドレスによる通信アクセス拒否 ⇒ 通信アクセス拒否時間延長のように、設定変更する項目を増やせば、より攻撃困難とすることも可能であろう。

４．４ 利用例３：攻撃元情報と攻撃時間の特徴を利用した侵入防止機能

ワームの拡散状況は、地域によって異なることが dshield.org[11]よりわかる。また、クライアント PC を対象としたものであれば、ワームの活動は PC の電源が ON であるときに限られる。悪意のあるユーザによる攻撃でも、攻撃ターゲットを探す調査行為は 24 時間行われるが、攻撃は特定の時間に行う場合が考えられる。このように、攻撃元と時間の概念は、攻撃困難なシステムを構築する上で、注目するポイントであると考えた。

そこで、過去攻撃を行ったホストが頻繁に攻撃を行う時間内は、事前にシステムを防御する機能を実現し、動的設定変更機能に追加した。この機能は、日毎に攻撃元 IP アドレスに下位 8 ビットのマスクをかけたアドレスをグループとみなし、グループ毎に攻撃回数を集計する。集計処理の結果として、攻撃を多く行う IP アドレスが特に攻撃を多く行う時間帯に、ファイアウォールを閉じる要求を出す。このため、攻撃が多い IP アドレスのグループは、毎日指定された時間帯にアクセス拒否をされるようになる。これは３．４節で述べた時間制約付実行である。

この機能により、攻撃が行われる前に攻撃者をアクセス拒否にすることができるため、新脆弱性や亜

種攻撃ツールが発生して、IDS の検知機能に関する対応が遅れている場合に攻撃抑止効果を発揮し、攻撃困難なシステムを構築することができる。

５．関連研究

侵入防止技術に関しては、これまで様々な研究が行われてきた。侵入防止機能自体の取組みとして、攻撃者を隔離する機能を持つ IPID[12]等がある。また、侵入防止手段を実施することによるシステムへの影響を判定後に侵入防止手段を実施する究[13][14]がある。その他、OS 内でイベント監視を行い、侵入行為の検知状況から最適な侵入防止手段を選択し実行する ADS[15]や、複数種類の IDS の警告に対して分析を行い、誤検知を防ぐ CIDS[16]がある。一方本稿では、任意の IDS を用いながらも侵入防止機能の互換性、動的な設定の変更を実現する基盤を提供することを目的としており、いずれも関連研究とは異なるものである。

おわりに

ここでは、IPS を構築するための共通基盤である Response Manager について述べた。実現方法として、OS内でイベント監視を行い、定期的に検査をする IDS にもリアルタイム性を付与する機能や、異なる IDS の警告内容を整形する機能、また侵入防止機能を実行するための情報を自動補完する機能について述べた。これらの機能により、従来侵入防止機能を持たない IDS や、侵入防止目的には適さない形式の IDS を用い、強力な検知機能を持った IPS を構築することが可能となる。

次に、動的な設定変更機能について述べた。この機能により、従来システムを防御する設定方法が均一であったことに比べ、システムの状況や過去の履歴に応じた侵入防止機能を変更可能となる。さらに、ここで実現した設定機能は、設定変更に関するユーザ任意の判断基準を組込めるため、高度な分析手法から、管理者の知見まで自由に組合わせることが可能である。この機能を用いて、様々なネットワーク環境で得られた知見を集約させ、その中から運用管理能力や自サイトに適した知見を採用して、安全なシステムを構築するために使用可能である。

次に、Response Manager の実装内容について述べ、OSS の IDS を用いた不正侵入対策を実現可能とした。

今後の課題は、ここでは動的設定変更について簡易な例のみ実現したため、より効果的な例を増やしていくことがある。

参考文献

[1] snort, http://www.snort.org
[2] tripwire, http://www.tripwire.org/
[3] modsecurity, http://www.modsecurity.org/
[4] snortsam, http://www.snortsam.net/
[5] libsafe, http://www.research.avayalabs.com/project/libsafe/
[6] postgresql, http://www.postgresql.org/
[7] syslog-ng, http://www.balabit.com/products/syslog_ng/
[8] iptables, http://www.netfilter.org/
[9] cutter, http://www.lowth.com/cutter/
[10] bigsister, http://bigsister.graeff.com/
[11] Dheild.org, http://www.dshield.org
[12] D. Schnackenberg, K. Djahandari and D. Sterne, “Infrastructure for Intrusion Detection and Response”, *DARPA Information Survivability Conference*, vol. 2, pp. 3-11, South Carolina, USA, Jun. 2000.
[13] T. Toth and C. Kruegel, “Evaluating the Impact of Automated Intrusion Response System”, *In Proc of Annual Computer Security Applications Conference*, Las Vegas, Nevada, USA, Dec. 2002.
[14] I. Balepin, S. Maltsev, J. Rowe, and K. Levitt, “Using Specification-Based Intrusion Detection for Automated Response”, *International Symposium on Recent Advance in Intrusion Detection*, Pittsburgh, Pennsylvania, USA 2003.
[15] O. P. Kreid and T. M. Frazier, “Feedback Control Applied to Survivability: A Host-Based Autonomic Defense System”, *IEEE Transactions on Reliability, Vol. 52, No. 3*, Sep. 2003.
[16] Y. S. Wu, B. Foo, Y. Mei, and S. Bagchi, “Collaborative Intrusion Detection System (CIDS): A Framework for Accurate and Efficient IDS”, *In Proc of Annual Computer Security Applications Conference*, Tucson, Arizona, USA, Dec. 2003