仮想環境を用いた侵入検知システムの安全な構成法

Safe Construction of Intrusion Detection Systems using Virtual Environments

光来 健一 †*† 廣津 登志夫 ‡ 佐藤 孝治 ‡ 明石 修 ‡ 福田 健介 ‡
菅原 俊治 ‡ 千葉 滋 †

† 東京工業大学 情報理工学研究科 数理・計算科学専攻
‡ 日本電信電話株式会社 NTT 未来ねっと研究所

要 旨

侵入検知システム（IDS）は万一侵入された場合に備えてホストを監視するシステムであるが、従来の構成方法では IDS が直接攻撃を受けた場合やアップデートで不具合が生じた場合に機能しなくなる危険性があった。この問題を解決するために、我々は IDS をポートスペースと呼ばれる仮想環境の中で動かす新しい IDS 構成法を提案する。IDS をポートスペース内で動かすことにより、IDS へのポートスペース外部からの攻撃を防ぐことができる。また、新しいバージョンの IDS を別のポートスペースで動かして元の IDS と一時的に共存させることにより、ホストの監視を中断することなく IDS をアップデートすることができる。

1 はじめに

インターネットからの攻撃を防ぐ様々な防御システムが利用されているが、攻撃を完全に防ぐのは難しい。サーバホストはインターネットからアクセスできる通信チャネルを開けておく必要がある上、サーバソフトウェアのバグはなかなかなくならないためである。そこで、万一ホストに侵入された場合に備えて侵入検知システム（IDS）が用いられている。IDS はホストの挙動を監視し、攻撃や侵入の徴候が見られると管理者に通知するシステムである。管理者が早めに攻撃の試みに対する対策を講じたり、侵入経路を割り出して再度の侵入を許さないようにすることができる。

IDS は他の防御システムと同様に常にホストを監視し続けなければならないが、従来の IDS の構成方法では IDS が機能しなくなる危険性があった。1 つは IDS そのものが攻撃を受けた場合である。従来、IDS はサーバホストを監視できるように同じホストや近くのホストに設置されるため、IDS の動いているホストに侵入されると IDS を停止させられる可能性がある。もし、IDS が他のホストのログ収集プログラムや IDS と通信を行う場合には、その通信チャネルを利用して攻撃される恐れもある。別の危険性としては、IDS のプログラムやルールをアップデートする際に、IDS が一時的に停止したり期待したように機能しなくなったりする可能性がある。

そこで、我々は IDS を仮想環境の中で動かす新しい IDS 構成法を提案する。この仮想環境はポートスペースと呼ばれ、独自のファイルシステム空間、ネットワーク空間、プロセス空間を提供する [6, 11]。IDS を動かせるように拡張されたポートスペースからは、そのポートスペースの外側で動くサーバプロセスが利用しているファイルシステムとネットワークを監視することができる。また、ポートスペースはインターネットに直接つながれてはいないが、他のホストのポートスペースと通信するために仮想プライベートネットワーク（VPN）による閉じたネットワークを構築することができる。

このように IDS を構成することにより、従来では IDS が機能を停止していた状況でも、IDS が機能し続けられるようになる。ポートスペースはその外側で動くサーバプロセスから干渉されることはないので、IDS は攻撃を受けたサーバプロセス経由での攻撃を受けずに済む。さらに、ポートスペースはインターネットからも切り離されているので、信頼できない第三者からの直接攻撃も受けない。また、IDS が独立した環境であるポートスペースの中で動作するため、アップデートの際には一時的に古いバー

*kourai@csg.is.titech.ac.jp
†この研究は NTT 在職中に行われた。

ジョンと新しいバージョンを同時に動かすことにより、IDS を停止させることなくスムーズに移行させることができる。

以下、2 章では従来の IDS の構成方法の問題点について述べる。3 章ではポートスペースを用いた我々のアプローチについて述べる。4 章ではポートスペースの実装について述べる。5 章ではポートスペースを用いた IDS 構成法のオーバヘッドを調べた実験について述べる。6 章で関連研究に触れ、7 章で本稿をまとめる。

2 従来の IDS 構成

攻撃や侵入の徴候を検出する IDS にとって、常にホストやネットワークを監視し続けられることが重要である。ホストの挙動を監視する IDS はホスト型 IDS と呼ばれ、ネットワークパケットを監視する IDS はネットワーク型 IDS と呼ばれる。IDS が停止している間にホストが攻撃されると、攻撃を見逃してしまうことになる。攻撃者は IDS が停止している隙を狙うだけでなく、攻撃によってまず IDS の機能を停止させてからサーバプログラムを攻撃するであろう。このような問題を避けるには、いかなる状況においても IDS が停止しないようにシステムを構成できなければならない。

しかしながら、従来のホスト型 IDS のようにサーバプログラムと同じホストで IDS を動かす構成方法では、IDS が機能しなくなる危険性がある。1 つは IDS そのものに対する攻撃である。IDS のプロセスを停止させられると、それ以降ホストを監視できなくなる。停止させられなくても IDS のルールを変更されてしまうと、以後 IDS に検知されずに攻撃されてしまう。ただルールを知られるだけでも、ルールを回避する攻撃方法を考えられる危険性がある。このような攻撃はインターネットからアクセスされるサーバホストでは容易に起こり得る。例えば、サーバプロセスの利用する通信チャネルから侵入されるかもしれない。もしくは、管理者になりすまされて侵入される可能性もある。また、DIDS [9] などの分散 IDS の場合には、IDS 間で利用する通信チャネルを通して IDS が直接攻撃される可能性もある。

もう 1 つの危険性は IDS のプログラムやルールのアップデートの際に生じる。このようなアップデートは短時間で完了すればほとんど問題ないが、不具合が発生すれば長時間かかることもあり得る。アップデートの間に IDS が止まることになると、その間はホストが無防備になってしまう。かといって、安全のためにホストをネットワークから切り離すとサーバのダウンタイムが長くなる。さらに悪いことに、一見うまくアップデートされたように見えても、気づきにくい問題が発生している場合もある。このような問題を防ぐには IDS をアップデートする前に綿密なテストを行う必要があるが、別個に全く同じ環境を作ってテストするのは大変である。既存の環境でインストールするディレクトリや使用するネットワークポート番号を変更してテストすることも考えられるが、共存させる設定を間違えると既存の IDS に影響を与えかねない。さらには、テスト後に新しい IDS の設定を標準のものに変更する際にミスをする可能性も高い。

一方、ネットワーク型 IDS の場合はサーバプログラムとは別のホストからネットワーク上を流れるパケットを監視することが可能であるが、IDS の動いているホストで管理用のサーバプログラムを動かすとホスト型 IDS と同様の危険性が生じる。

3 アプローチ

IDS が機能しなくなる事態を防ぐために、我々は IDS を仮想環境の中で動かす構成方法を提案する。この仮想環境はポートスペースと呼ばれ、VPN によって作られるマルチホーム・ホストにおいて情報の流れを制御できるようにするために我々が提案、実装しているものである [6, 11]。IDS を動かしてホストを監視できるようにポートスペースを拡張することにより、IDS を攻撃から守り、アップデートを安全に行えるようにする。

図 1 は我々の提案する IDS 構成の全体像を示している。IDS とサーバプログラムは別々のポートスペースで動かされ、IDS の動く監視用ポートスペースはサーバプログラムの動くサーバ用ポートスペースを監視する。従来の実行環境であるベース環境ではネットワークを利用するプログラムは動かさず、サーバプログラムが必要とするネットワークパケットはサーバ用ポートスペースに転送される。監視用ポートスペースは他のホストの監視用ポートスペースとの間に張られた VPN のみを利用する。

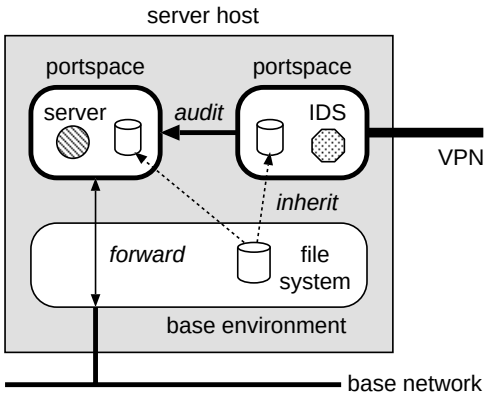

図 1: ポートスペースを用いた IDS 構成

3.1 ポートスペース

ポートスペースはベース環境から独立した仮想的な実行環境である。ポートスペースは独自のファイルシステム空間、ネットワーク空間、プロセス空間を提供する。

- **ファイルシステム空間:** ポートスペース内のファイルやディレクトリにはポートスペース内で動いているプロセスからのみアクセスすることができる。
  ポートスペース毎にファイルシステムを一から構築するのを避けるため、既存のポートスペースのファイルシステムを継承して新しいポートスペースを作成することができる。ベース環境を擬似的なポートスペースと見なすことにより、ベース環境のファイルシステムに含まれるライブラリなどの基本的なファイルを参照することができる。その上で各ポートスペース専用のプログラム等をインストールすることができる。
- **ネットワーク空間:** ポートスペース内では独自のネットワーク設定を行うことができる。そのため、ベース環境と同じ IP アドレスを割り当てることができ、ポートスペース毎に IP アドレスを必要としない。一方、同じ IP アドレスであっても独自のプロトコルコントロールブロックを提供しているため、TCP や UDP のポートを自由に利用することができる。これにより、同じポート番号を使用するプログラムであっても別々のポートスペースで同時に動かすことができる。
- **プロセス空間:** ポートスペース内で動いているプロセスのみにアクセスすることができる。プロセス間通信、共有メモリ、シグナルは同じポートスペース内のプロセス同士でのみ利用可能となる。

3.1.1 監視用ポートスペース

監視用ポートスペースは IDS を動かすために使われる。監視用ポートスペースはサーバ用ポートスペースのファイルシステムを参照することができ、ファイルの属性情報や内容を監視できる。また、サーバ用ポートスペースのネットワークインタフェースも監視することができ、BPF などのパケットフィルタを通してサーバプロセスのやりとりするパケットを取得することができる。監視用ポートスペースは必要に応じて複数作ることができるので、複数の IDS を別々のポートスペースで動かすことが可能である。

監視用ポートスペースは VPN を介して他のホストの監視用ポートスペースと接続され、その VPN のみを利用することができる。ポートスペースは同じホスト上の他のポートスペースやベース環境と通信する手段を持たないため、VPN とこれらのポートスペースは閉じたネットワークを形成する。閉じたネットワークであるため、ポートスペースにベース環境と同じ IP アドレスを割り当てても衝突せずに済む。このネットワーク内で通信を行うために、各ポートスペースは独自のルーティングテーブルを持つ。

3.1.2 サーバ用ポートスペース

サーバ用ポートスペースはサーバプログラムを動かすために使われる。このポートスペースも複数作ることができるので、サーバプログラム毎に異なるポートスペースを使用することもできる。サーバ用ポートスペースはインターネットからアクセスできるように VPN ではなくベースネットワークを利用する。ベース環境と同じ IP アドレスを使って通信できるようにするために、サーバプロセスが使用しているネットワークポートに関して、パケットをベース環境とポートスペースの間で相互に転送する。

3.2 IDS の保護

サーバプロセスが攻撃されて、たとえサーバ用ポートスペースに侵入されたとしても、監視用ポートスペースで動いている IDS は攻撃されることはない。サーバ用ポートスペースは監視用ポートスペースに一切アクセスすることはできないからである。サーバ用ポートスペースは監視用ポートスペース内のプロセスに干渉することはできず、IDS を停止させることができない。ファイルシステムにもアクセスすることはできないので、設定ファイルやログファイルを盗み見られたり改竄されたりすることもない。また、監視用ポートスペースにはサーバ用ポートスペースと同じ IP アドレスが割り振られるため、ネットワーク経由でアクセスすることもできない。

監視用ポートスペースは IDS がリモートホストから直接攻撃されるのを防ぐこともできる。監視用ポートスペースの利用するネットワークは VPN のみであり、任意のリモートホストからアクセスするインタフェースを持たない。そのため、監視用ポートスペースにアクセスできるのは信頼できるリモートの監視用ポートスペースのみである。監視用ポートスペースでは公開するサーバプログラムを動かす必要はないので、このような閉じたネットワークのみで十分である。

監視用ポートスペースでベース環境のファイルシステムを継承している場合、ベース環境に侵入されるとファイルを書き換えられてしまい、書き換えられたファイルを参照する IDS が正しく機能しなくなる可能性がある。ベース環境でネットワークサービスを動かしたり、内部に信頼できないユーザがいるなど、ベース環境に侵入される可能性がある場合には、継承を用いずに監視用ポートスペースを作成する必要がある。ファイルシステムを継承しなければ、ベース環境からも一切の攻撃を行うことはできない。

このように監視用ポートスペースは外部からの攻撃に対して堅牢であるが、万一、IDS が停止させられた場合に備えて、IDS を多重化することができる。それぞれの IDS を管理ポリシの異なる別々の監視用ポートスペースで動かすことにより多重化を行う。このような多重化により、1 つの監視用ポートスペースへの攻撃に成功されて IDS が停止させられても、他の監視用ポートスペースの IDS は動き続けることができる。この際に、監視用ポートスペースは独立した実行環境を提供するため、IDS 同士がネットワークやファイルシステムにおいて衝突することはない。そのため、同じ IDS を全く同じ設定で 2 つ動かすことも可能である。

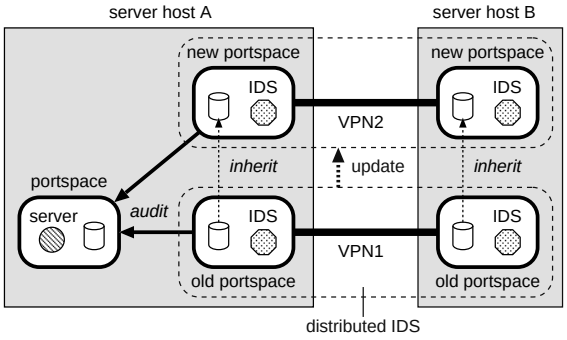

図 2: 継承を用いた分散 IDS のアップデート

3.3 IDS のアップデート

監視用ポートスペースを用いて複数の IDS を同時に動かせるという特徴を利用すると、IDS のアップデートを安全に行うことができる。IDS のプログラムやルールをアップデートした新しい監視用ポートスペースを用意し、アップデート前の IDS と同時に動かす。仮にアップデート後の IDS がうまく動いていなかったとしても、アップデート前の IDS が正しく動いているため安全性は損なわれない。そして、アップデート後の IDS が正しく動いていることが確認できたら、アップデート前の IDS を停止させ、その監視用ポートスペースを破棄すればよい。ポートスペースを VPN でつないだ分散 IDS の場合も同様に、分散 IDS 全体をひとまとまりと考えてアップデートを行うことができる。

IDS のルールなどアップデート部分がわずかな場合には、ポートスペースの継承を用いることでより簡単にアップデートを行うことができる。IDS が動いている監視用ポートスペースを継承して新しいポートスペースを作り、継承したファイルシステムの中の必要な部分だけを変更する。その後、新しいポートスペースの中で IDS のプログラムを実行すれば、変更が反映された IDS が起動される。アップデートに問題がなければ、継承元のポートスペースは破棄せずに、そこで動いている IDS だけを停止さ

せればよい。図 2 は継承を用いて分散 IDS を一括してアップデートする様子を示している。これを繰り返すと継承関係が深くなり性能に影響を及ぼすので、適宜、ベース環境を直接継承するポートスペースを作った方がよい。

4 実装

この章では、実行環境を仮想化するポートスペースの実装および、サーバ用ポートスペースを監視する機構の実装について述べる。

4.1 ファイルシステムの仮想化

ファイルシステム空間を多重化するために、`/.filespace/<id>/`（`<id>`はポートスペースの ID）をポートスペース専用に割り当てる。このディレクトリはポートスペース外部から不正に書き換えられないようにするために、ポートスペース内部でのみ名前検索が許可される。ファイルシステムを継承する場合は BSD の union ファイルシステムを用いて、ポートスペース専用のディレクトリを親のポートスペースのファイルシステムの / の上に透過的にマウントする。ファイルシステムを継承した子ポートスペースでのファイルの読み出しは親ポートスペースのファイルシステムから行われ、変更、作成、削除は子ポートスペースのファイルシステムにのみ反映される。子ポートスペースのファイルシステムには変更の差分だけが保存されるため、IDS が変更されたファイルだけをチェックすることも容易にできる。

ファイルシステムを継承しない場合は、ポートスペース専用に割り当てられたディレクトリをポートスペースのルートディレクトリにする。この場合、ファイルシステムを一から作成しなければならないが、親ポートスペースのファイルシステムを参照時コピーすることによって比較的容易に作成することができる。まず、ファイルシステムを継承する場合と同様にマウントする。次に IDS やサーバプログラムを動かして参照されるファイルを親ポートスペースから子ポートスペースのファイルシステムにコピーする。参照されるファイルをコピーし終えたらアンマウントする。

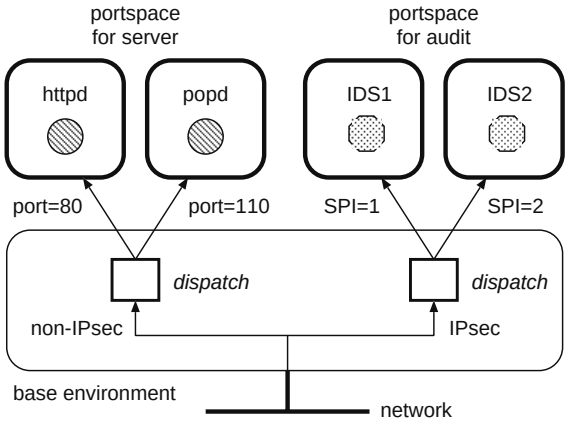

図 3: パケットのディスパッチ

4.2 ネットワークの仮想化

ポートスペースは独自のネットワークインタフェース・リスト、ルーティングテーブル、プロトコルコントロールブロック・リスト、IPsec のデータベースなどを保持する。

4.2.1 監視用ポートスペース間の通信

監視用ポートスペース同士はポートスペース間に確立される IPsec の通信路を用いて通信を行う。全てのポートスペース間に IPsec の通信路がメッシュ状に確立されているとは限らないので、パーソナルネットワーク内ではそのネットワークトポロジに応じた経路制御を行う。そのため、全てのポートスペースがルータの役割を果す。各ポートスペースは IPsec の通信路を設定する時に相手のポートスペースへの経路を動的に生成する。経路情報はベースネットワーク同様に RIP などのプロトコルを使って交換する。

パケットを送る際には、まずポートスペースのルーティングテーブルを検索し、経路が見つかれば、次に IPsec のセキュリティポリシーを検索する。IPsec 通信を許可するポリシーが見つかれば、ポートスペース間で確立されているセキュリティ・アソシエーション（SA）を用いて IPsec トンネルモードの処理を行い、パケットをカプセル化する。このパケットを相手ホストのカーネルが受け取ったら、図 3 のようにパケットの IPsec ヘッダのセキュリティパラメータ・インデックス（SPI）をキーにして配送すべきポートスペースを検索する。SPI は IPsec の SA に

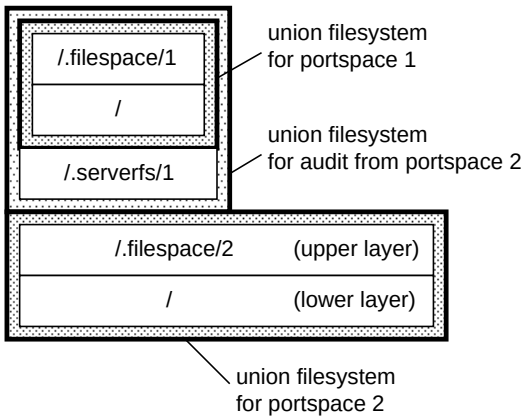

図 4: 監視用ポートスペースのファイルシステムの構造

割り当てられた 2 つのホスト間で一意の値である。

4.2.2 ポートスペース・トランスレーション

ベース環境はサーバ用ポートスペースとの間で相互にパケットを転送するために、ポートスペース・トランスレーションという機構を用いる。この機構はポートスペースとベース環境を別々のホストと考えると NAPT [10] と同様のことを行う機構である。ベース環境はポートスペース・トランスレーションのための変換テーブルを持ち、ベース環境のポート番号と特定のポートスペースのポート番号の間の相互変換を行うためのルールを保持する。ベース環境が受信したパケットが変換ルールにマッチした時は、図 3 のように対応するポートスペースにパケットを転送する。一方、ポートスペースから送信されたパケットが変換ルールにマッチした時は、変換ルールに応じてパケットヘッダの送信元ポート番号を書き換え、ベース環境からパケットを送信する。変換ルールにマッチしない時は一時的な変換ルールを作り、パケットを送信する。

4.3 監視機構

監視用ポートスペースからサーバ用ポートスペースのファイルシステムを監視できるようにするために、サーバ用ポートスペースのファイルシステムの / を監視用ポートスペースの `/.serverfs/<id>`（`<id>`はサーバ用ポートスペースの ID）に読み込み専用で union マウントする[1]。これにより、監視用ポートスペースからは、継承されているファイルシステムも含めてサーバ用ポートスペースからアクセス可能なファイルシステム全体にアクセスすることができる。図 4 は監視用ポートスペースのファイルシステムの構造を示している。通常、ポートスペース間にまたがるファイルシステムのアクセスは不可能なので、このマウントはカーネル内で行われる。なお、サーバ用ポートスペースのファイルシステム全体ではなく、変更の差分（`/.filespace/<id>`）だけを union マウントするようにすれば、監視すべきファイルを大幅に減らすことができる。

監視用ポートスペースからサーバ用ポートスペースのネットワークを監視できるようにするために、ベース環境のネットワークインタフェースを監視用ポートスペースからアクセスできるようにする。サーバ用ポートスペースにはポートスペース・トランスレーションを用いて特定のポートへのパケットしか配送されないが、IDS はポートスキャンなどシステム全体に対する攻撃も監視する必要があるので、ベース環境に届く全てのパケットを監視できるようにする。ポートスペース内で動くプロセスは BPF などのパケットフィルタにベース環境のネットワークインタフェースを設定することでパケットを監視することができる。

5 実験

我々の提案する新しい IDS 構成法のオーバヘッドを調べる実験を行った。実験には PentiumIII-S 1.4GHz の CPU、512MB のメモリ、Intel Pro/100+ の NIC を搭載した PC を 2 台使用し、スイッチを介して 100baseT のイーサネットで接続した。OS にはポートスペースを実装した FreeBSD 4.7 を用いた。

5.1 ウェブサーバ

ウェブサーバ thttpd [7] をサーバ用ポートスペースで動かした時のオーバヘッドを調べた。サーバ用ポートスペースで動かすためのオーバヘッドには、主にポートスペース・トランスレーションと union

[1]機能的にはエイリアスを作る null マウントでよいのだが、union ファイルシステムの上に null マウントを行うと `getdirentries` システムコールなどが再帰的にディレクトリエントリを検索してくれない。

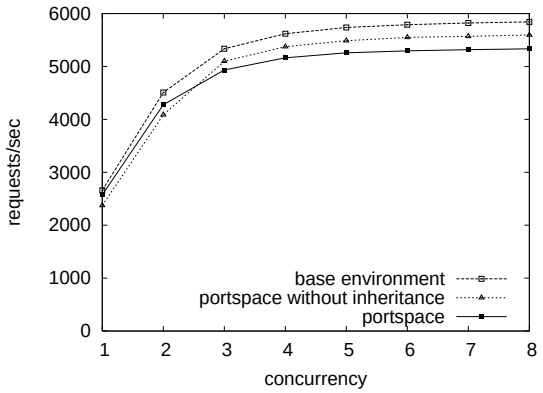

図 5: ポートスペースが thttpd の性能に及ぼす影響

ファイルシステムのオーバヘッドがある。ベンチマークプログラムには ApacheBench [1] を用い、リクエストの並行度を変えながらウェブサーバのスループットを測定した。測定は以下の環境下で thttpd を動かして行った。

- ベース環境
- ポートスペース
- ベース環境を継承しないポートスペース

実験結果は図 5 のようになった。ポートスペース内で thttpd を動かした時のオーバヘッドは並行度が 1 の時は 3.3%だが並行度が増すにつれて次第に大きくなり 9.5%に達する。一方、ファイルシステムを継承しない場合は、並行度が小さい時のオーバヘッドはより大きくなるが、並行度が増すと 4.3%程度になる。ファイルシステムを継承しない場合には union ファイルシステムのオーバヘッドがなくなるため全体のオーバヘッドも減ると考えられるが、並行度が小さい時にオーバヘッドが大きくなる原因は現在調査中である。

5.2 Tripwire

Tripwire [4] はファイルの改竄を監視する IDS である。Tripwire は前もってホスト上にあるファイルの完全性を検証しておき、定期的に完全性をチェックして変更がなされた場合にはそれを通知する。我々は Tripwire が以下の環境下でシステムの全ファイルの完全性をチェックするのにかかる時間を測定した。

- ベース環境で動く Tripwire がベース環境のファイルシステムをチェック

	時間（秒）
ベース環境を使用	80
ポートスペースを使用	91

表 1: Tripwire のチェックにかかる時間

	ドロップ率（%）
ベース環境を使用	7.6
ポートスペースを使用	8.5

表 2: Snort のパケットドロップ率

- 監視用ポートスペースで動く Tripwire がサーバ用ポートスペースのファイルシステムをチェック

Tripwire のバージョンは 2.3 であり、チェックの対象となったファイルおよびディレクトリの数はいずれの場合も 52,489 個であった。サーバ用ポートスペースを使用する場合にはベース環境から継承したファイルも含む。

実験結果は表 1 のようになった。監視用ポートスペース内で動く Tripwire がサーバ用ポートスペースのファイルシステムをチェックするためのオーバヘッドは 15%であった。ただし、サーバ用ポートスペースの union ファイルシステムになされた変更部分だけをチェックするように改良すれば、ベース環境のファイルシステムをチェックする場合と比べても、大幅な性能改善が期待できる。

5.3 Snort

Snort [8] はネットワーク上のパケットを監視する IDS である。Snort は取得した一連のパケットを攻撃パターンに照らし合わせて攻撃を検出する。我々は大量の UDP パケットを送りつけた時に Snort がどの程度パケットをとりこぼすかを測定した。測定は Snort をベース環境で動かした場合と監視用ポートスペースで動かした場合とで行った。Snort のバージョンは 1.8.7 であり、デフォルトの 1,333 個の検出ルールを使用した。送りつけた UDP パケットは毎秒約 2,200 パケットであり、約 24Mbps の帯域を使用した。

実験結果は表 2 のようになった。ポートスペースを使用した場合、パケットを 0.9%ほど多くとりこぼしている。

6 関連研究

攻撃によって IDS が機能しなくなる状況を防ぐために、IDS を OS カーネル内で動かす方法がある [3, 5]。カーネル内で動く IDS はユーザプロセスとして動く IDS とは違い、ホストに侵入されても比較的停止させられにくい。しかしながら、IDS のルールを設定するシステムコール等のインタフェースが存在すると、侵入者にルールを変更され IDS を無効化されてしまう可能性がある。また、カーネル内の IDS が他のホストのログ収集プログラムや IDS と通信する場合、その通信チャネルから攻撃を受け、カーネルごと乗っ取られる可能性がある。さらに、IDS のプログラムをアップデートする際にはシステム全体を停止させる必要がある。アップデートしやすいように IDS がカーネルモジュールとして提供されていたとすると、侵入者にモジュールを停止される恐れが出てくる。

別の構成法として、サーバプログラムを仮想マシンの上で動かす方法が提案されている [2]。IDS とサーバプログラムを分離することによって、仮にサーバプログラムの動いている仮想マシン内に侵入されたとしても、仮想マシンの外で動いている IDS に干渉することはできない。しかし、IDS が他のホストと通信する場合、その通信チャネルを介して攻撃を受ける可能性がある。我々の構成法とはサーバプログラムだけでなく IDS もシステムのその他の部分から分離している点が異なる。

7 まとめ

本稿では既存の IDS をポートスペースと呼ばれる仮想環境の中で動作させる新しい IDS 構成法を提案した。IDS を監視用ポートスペースの中に隔離し、VPN のみを使わせるようにすることで、IDS の安全性を確保することができる。また、IDS のアップデートの際には複数の監視用ポートスペースを用いて一時的に複数の IDS を動かすことにより、監視に空白の時間ができないようにすることができる。

今後の課題としては IDS に対する DoS 攻撃への対処が挙げられる。DoS 攻撃を受けるとネットワーク型 IDS はパケットを処理しきれなくなる可能性があり、その間にサーバプロセスが攻撃されるかもしれない。このような攻撃に対しては一般的な DoS 攻撃対策に加えて、パケットをとりこぼさないようにするために監視用ポートスペースの処理の優先度を上げるなどの対処も必要になると考えられる。

参考文献

[1] Apache HTTP Server Project, : Apache HTTP Server Benchmarking Tool, `http://www.apache.org/`.

[2] Garfinkel, T. and Rosenblum, M.: A Virtual Machine Introspection Based Architecture for Intrusion Detection, in *Proceedings of the Network and Distributed Systems Security Symposium* (2003).

[3] Huangang, X.: LIDS Project, `http://www.lids.org/`.

[4] Kim, G. and Spafford, E.: The Design and Implementation of Tripwire: A File System Integrity Checker, in *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pp. 18–29 (1994).

[5] Ko, C., Fraser, T., Badger, L. and Kilpatrick, D.: Detecting and Countering System Intrusions Using Software Wrappers, in *Proceedings of the 9th USENIX Security Symposium* (2000).

[6] Kourai, K., Hirotsu, T., Sato, K., Akashi, O., Fukuda, K., Sugawara, T. and Chiba, S.: Secure and Manageable Virtual Private Networks for End-users, in *Proceedings of the 28th Annual IEEE Conference on Local Computer Networks*, pp. 385–394 (2003).

[7] Poskanzer, J.: Tiny/turbo/throttling HTTP Server, `http://www.acme.com/software/thttpd/`.

[8] Roesch, M.: Snort – Lightweight Intrusion Detection for Networks, in *Proceedings of the 13th USENIX System Administration Conference* (1999).

[9] Snapp, S., Smaha, S., Teal, D. and Grance, T.: The DIDS (Distributed Intrusion Detection System) Prototype, in *Proceedings of the Summer USENIX Conference*, pp. 227–233 (1992).

[10] Srisuresh, P. and Holdrege, M.: IP Network Address Translator (NAT) Terminology and Considerations, RFC 2663 (1999).

[11] 光来健一, 廣津登志夫, 佐藤孝治, 明石修, 福田健介, 菅原俊治, 千葉滋：VPN とホストの実行環境を統合するパーソナルネットワーク, ソフトウェア科学会論文誌 コンピュータソフトウェア：掲載予定.