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

2004－OS－97　(10)
2004／8／1

分散 IDS の実行環境の分離による安全性の向上

光来 健一*　　千葉 滋*　　廣津 登志夫†　　菅原 俊治‡

要 旨

分散 IDS は分散システム全体を監視しなければならないため，分散 IDS を構成する各 IDS は分散システム全体にちらばって配置されている．さらに，各 IDS は分散システム内のホストやネットワークに埋め込まれている．このように，分散 IDS は分散システムから空間的に分離されていないため安全性の問題が生じている．分散 IDS は分散システム内のホストが利用するネットワークを経由したり，サーバや他の分散 IDS のプロセスを経由したりして攻撃を受ける可能性がある．本稿では，分散 IDS を分散システムの他の部分から分離できるようにするために，HyperSpector と呼ばれる仮想分散環境を提案する．HyperSpector はポートスペースと呼ばれる仮想マシンと VPN で構成される．ポートスペースの中で動く IDS はその外側で動いているサーバのファイルシステム，ネットワーク，プロセスを監視することができる．HyperSpector を用いることで，分散 IDS は能動的な攻撃から守られ，受動的な攻撃を受けた場合の被害を HyperSpector 内に限定することができる．

Secure Distributed IDSes Based on Separation of Execution Environments

Kenichi Kourai*　　Shigeru Chiba*　　Toshio Hirotsu†　　Toshiharu Sugawara‡

Abstract

Due to the nature that a distributed IDS has to monitor the whole distributed system, the IDSes constructing a distributed IDS are scattered throughout hosts in the distributed system. In addition, each IDS is embedded into a host and a network in the distributed system. As such, the distributed IDSes are not separated from the distributed system and therefore cause a security problem. A distributed IDS can be compromised via the distributed system. In this paper, we propose a virtual distributed environment called the HyperSpector, which is separated from the rest of a distributed system. The HyperSpector consists of virtual machines called the portspaces and a VPN. The portspace enables an IDS in it to monitor file systems, a network, and processes of servers running in the outside of it. Using the HyperSpector, a distributed IDS is protected from active attacks and damages by passive attacks are confined inside the HyperSpector.

1　はじめに

分散システムでは 1 つのホストが攻撃されるとワームやウィルスなどが急速に広まってしまう．できるだけ早くワームやウィルスの拡がりを防ぐために，分散侵入検知システム（IDS）が必要である．分散 IDS は分散システム内のホスト型 IDS やネットワーク型 IDS を協調動作させる．多くの IDS からの攻撃に関する情報を集めることにより，管理者は分散システム内の攻撃を早期に把握することができる．攻撃を把握することが，ワームやウィルスの被害を抑えるための最初のステップとなる．

分散 IDS は分散システム全体を監視しなければならないため，分散 IDS を構成する各 IDS は分散システム全体にちらばっている．その上，ホストやネットワークを監視するために，各 IDS は監視対象のホストやネットワークに埋め込まれている．このように，分散 IDS は空間的に分散システムから分離されていないため，分散システムを経由して攻撃

*東京工業大学 情報理工学研究科 数理・計算科学専攻, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology.
†日本電信電話株式会社 NTT 未来ねっと研究所, NTT Network Innovation Laboratories, NTT Corporation.
‡日本電信電話株式会社 NTT コミュニケーション科学基礎研究所, NTT Communication Science Laboratories, NTT Corporation.

を受ける可能性がある．攻撃者はインターネットから分散 IDS を直接攻撃できるかもしれないし，分散システム内のサーバホストを経由して間接攻撃を行うことができるかもしれない．また，IDS が監視するデータを注意深く作成することにより，IDS がそのデータを読んだ時に攻撃を成功させることも可能である．

このような問題を解決するために，分散 IDS を分散システムから分離して動作させることができる仮想分散環境 HyperSpector を提案する．HyperSpector は各ホストに配置される仮想マシンとそれらを接続する仮想プライベートネットワーク（VPN）で構成される．我々の仮想マシンはポートスペース [7, 12] と呼ばれ，独立したファイルシステム空間，ネットワークシステム空間，プロセス空間から成る．我々はこれまでに提案してきたポートスペースを拡張し，ポートスペースの外で動かすサーバのファイルシステム，ネットワーク，プロセスを監視できるようにした．

HyperSpector を用いることにより，分散 IDS を能動的な攻撃から保護することができる．HyperSpector の外からの攻撃は分散 IDS に到達することはできない．攻撃者が分散システム内のサーバの攻撃に成功したとしても，それを経由して HyperSpector 内の分散 IDS を攻撃することはできない．また，HyperSpector は分散 IDS が受動的な攻撃を受けた後，それ以上被害が拡がるのを防ぐことができる．1 つの分散 IDS が受動的な攻撃を受けたとしても，他の分散 IDS は別の HyperSpector 内で動いているため，攻撃者は他の分散 IDS に対して能動的な攻撃を行うことができない．受動的な攻撃を受けた場合は分散 IDS が停止してしまうが，同じ機能を持った異なる分散 IDS を同時に動かしておくことで対処できる．

以下，2 章では分散 IDS の構成から生じる安全性の問題について述べる．3 章では分散 IDS を分散システムから分離する HyperSpector について述べ，4 章ではその実装について述べる．5 章では HyperSpector を使う場合の IDS の性能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 分散 IDS の安全性

分散 IDS はホスト群とネットワークから成る分散システムを監視するために，分散システム全体をカバーしなればならない．そのため，分散 IDS を構成する各 IDS は分散システム全体にちらばっており，その上，各 IDS は分散システム内のホストやネットワークに埋め込まれている．ホストのファイルシステム，プロセス，ネットワークを監視するホスト型 IDS は監視対象のサーバと同じホスト上に配置される．このタイプの IDS は同じホスト上で動いているサーバや他の IDS から分離されておらず，ファイルシステム，プロセス，ネットワークを共有している．他方，ネットワークパケットを監視するネットワーク型 IDS は監視対象のネットワーク上のホストに配置される．このタイプの IDS もサーバが利用するネットワークから分離されておらず，ネットワークを共有している．

分散 IDS が空間的に分散システムから分離されていないために安全性の問題が生じている．攻撃者は IDS を攻撃して警告を出される前に停止させれば，侵入を検知されることなく攻撃を続行できる．攻撃の方法には能動的な攻撃と受動的な攻撃の二種類がある．能動的な攻撃とは，攻撃者が分散 IDS に対して直接アクセスする攻撃である．分散 IDS において，各 IDS はお互いに通信するためのネットワークポートを持つ．このネットワークポートは分散 IDS 内でのみアクセス可能であるべきだが，分散システムの外側のインターネットからアクセス可能になっているかもしれない．この場合，分散 IDS が直接攻撃にさらされる．

IDS のネットワークポートがファイアウォール等によって守られていて直接攻撃されないとしても，攻撃者は間接攻撃を行うことができる．攻撃者はまずインターネットからアクセスできる分散システム内のサーバを攻撃し，そのサーバを経由して分散 IDS を攻撃することができる．例えば，IDS のネットワークポートは分散システム内からはアクセス可能であることが多いため，サーバ経由で攻撃できるかもしれない．攻撃者は IDS のプロセスを停止させることができるかもしれないし，IDS のポリシーファイルを書き換えることができるかもしれない．

一方，受動的な攻撃とは，攻撃者が IDS を攻撃するためのデータを作り，IDS がそのデータを読んで攻撃が成功するのを待つという攻撃である．ネットワーク型 IDS については，攻撃者が注意深く作ったパケットをサーバに送ることによって，そのパケットを監視している IDS を攻撃することができる．このような攻撃の例はフォーマット文字列の脆弱性を

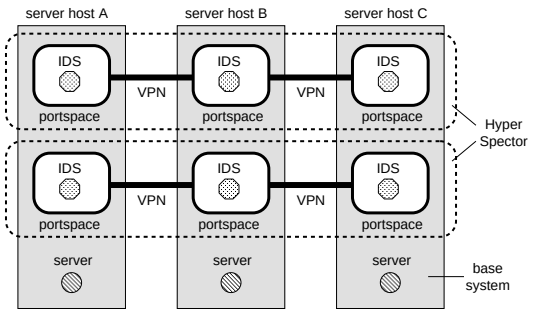

図 1: HyperSpector による分散 IDS の分離

利用する攻撃などがある．このタイプの攻撃はインターネットから直接行うこともできるし，サーバを攻撃してから間接的に行うこともできる．ホスト型 IDS については，攻撃者はサーバホストのファイルの内容や属性を書き換えることによって，ファイルの整合性を検査する IDS を攻撃することができる．

このような受動的な攻撃はサーバが利用するデータを監視しなければならない IDS にとって避けるのが難しい．そのため，同じ機能を持った異なる分散 IDS を同時に動かしておき，監視の機能が完全に停止するのを防ぐという対策が考えられる．攻撃者が受動的な攻撃によって 1 つの分散 IDS を停止させてからサーバを攻撃したとしても，別の分散 IDS がその攻撃を検出することができる．しかし，攻撃者は 1 つの分散 IDS の受動的な攻撃に成功した後，他の分散 IDS を間接的に攻撃することができる．そのため，分散 IDS を多重化しても十分に安全であるとは言えない．

3 HyperSpector

分散 IDS を分散システムから分離するために，我々は HyperSpector と呼ばれる仮想分散環境を提案する．HyperSpector はそれぞれの分散 IDS を独立して動作させることを可能にする．HyperSpector は分散システムから各ホストとネットワークの IDS に関連する部分だけを集めたものである．HyperSpector は図 1 のように仮想マシンと VPN で構成される．ホスト型 IDS はサーバホストに作られる仮想マシン内で動かされ，ネットワーク型 IDS は専用ホストまたはサーバホストの中の仮想マシン内で動かされる．それらの仮想マシンは VPN とだけ接続され，VPN 以外からのネットワークアクセスを受けつけない．

3.1 ポートスペース

我々の仮想マシンはポートスペース [7, 12] と呼ばれ，ファイルシステム空間，ネットワークシステム空間，プロセス空間から成る．これらの名前空間は他のポートスペースやポートスペースを動かすベースシステムの名前空間から独立している．各 IDS のプロセスはポートスペースが提供しているファイルシステムやネットワークシステムを使う．

- **ファイルシステム空間** ポートスペースは分離されたファイルシステムを提供し，管理者は IDS のプログラムやポリシーファイルを用意することができる．また，他の IDS と通信するために VPN や DNS などのネットワーク設定ファイルを用意することもできる．このファイルシステムはポートスペースの中で動くプロセスからのみアクセスを許される．
- **ネットワークシステム空間** ポートスペースは HyperSpector 毎に提供される VPN のみと接続される．そのため，ホスト内の他のポートスペースやベースシステムと同じ IP アドレスを使用することが可能である．管理者はポートスペース毎に異なる IP アドレスを割り当てる必要がない．さらに，同じ IP アドレスを使っていても，ネットワークポートの使用は他のポートスペースやベースシステムと競合しない．
- **プロセス空間** ポートスペースはプロセス間のインタラクションをその内部に制限する．プロセスはプロセス間通信，共有メモリ，シグナルを使って同じポートスペースで動いているプロセスとのみメッセージのやりとりをすることができる．

3.2 監視機構

我々が文献 [7, 12] で提案したポートスペースは外界から閉じた実行環境を提供するものであった．我々はポートスペース内の IDS が外で動いているサーバを監視できるようにポートスペースを拡張した．この監視機構により，既存のユーザレベルの IDS をそのままポートスペースで動かすことができる．この監視機構はポートスペース内の IDS がサーバの使用するリソースを監視することを可能にする．

- **ファイルシステム** サーバのファイルシステムはポートスペース内の専用ディレクトリにマップされる．IDS はそのディレクトリにアクセス

することによって，通常のファイルシステムと同じようにサーバのファイルシステムを監視することができる．

- **ネットワーク** ポートスペースはホスト全体に対するパケットフィルタを提供しており，IDS はホストが受け取る全てのパケットを監視することができる．これはホスト全体にわたる攻撃を検出するのに役立つ．また，サーバの行う Unix ドメインソケットの通信を盗み見ることができるため，サーバが発行した syslog メッセージをポートスペース内の syslogd が受けとることができる．
- **プロセス** IDS はサーバプロセスが発行するシステムコールをトレースすることができる．

3.3 攻撃への対処

HyperSpector は分散 IDS を能動的な攻撃から保護する．HyperSpector の外からの攻撃は分散 IDS には到達できない．HyperSpector 内の分散 IDS は VPN のみを利用し，外部ネットワークには接続されていないためである．攻撃者が分散システム内のサーバを攻撃できたとしても，それらを経由して分散 IDS を攻撃することはできない．攻撃者は IDS プロセスを見ることができないため，IDS を停止させることはできない．IDS の使うファイルシステムにもアクセスできないため，IDS のポリシーを書き換えることもできない．IDS の使う VPN にもアクセスできないため，IDS のネットワークポートへの攻撃もできない．

HyperSpector は分散 IDS が受動的な攻撃を受けた後，それ以上被害が拡がるのを防ぐ．HyperSpector は分散 IDS を能動的な攻撃から保護するので，受動的な攻撃を受けた分散 IDS が他の分散 IDS を能動的に攻撃することはできない．よって，同じ機能を持った異なる分散 IDS を同時に動かしておけば，監視を継続することができる．異なる分散 IDS が同じ受動的な攻撃を受ける可能性は低いと考えられる．また，攻撃者はサーバのファイルシステム，ネットワーク，プロセスに影響を与えることはできない．その一方で，攻撃者はポートスペースの監視機構を使ってサーバのファイルシステム，ネットワーク，プロセスの情報を盗み見ることはできるが，HyperSpector の外部にその情報を送信することはできない．

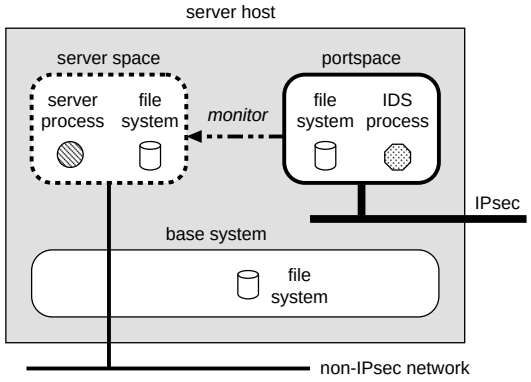

図 2: サーバホスト内の 3 種類の実行環境

4 実装

我々は HyperSpector を FreeBSD 4.9 をベースにした Persona OS に実装した．Persona は VPN を実現するために IPsec を利用する．

4.1 サーバホストの構成

各ホストは図 2 のように 3 種類の実行環境から構成される．

- **ベースシステム** 従来の OS の実行環境であり，IPsec を使わない従来のネットワークに接続される．この実行環境ではサーバは動かさず，管理者はコンソールからのみログインすることができる．
- **ポートスペース** IDS を動かすために使われる仮想的な実行環境であり，他の実行環境から完全に独立している．他のホストのポートスペースとの間に張られた IPsec トンネルを通してのみ通信を行うことができる．
- **サーバスペース** サーバを動かすために使われる仮想的な実行環境であり，ポートスペースに似ている．違いはインターネットからのアクセスを受けつけるためにネットワークをベースシステムと共有していることである．サーバスペースは HyperSpector を実現するのに必ずしも必要ではないが，サーバを閉じ込めてより安全に実行するために用いている．

4.2 ポートスペース

ポートスペースは `mkportspace` システムコールによって作られ，そのシステムコールを発行したプロセスとその子孫がその中で動く．実装の詳細につい

ては文献 [7, 12] で述べられている．

4.2.1 ファイルシステムの仮想化

ポートスペースは union ファイルシステムを用いて独立したファイルシステムを構築する．union ファイルシステムは既存のファイルシステムの上にディレクトリを配置し，上の層にのみ変更が反映されるというファイルシステムである．各ポートスペース毎に専用のディレクトリを用意し，切り替えて用いることで仮想化を実現できる．

4.2.2 ネットワークシステムの仮想化

ポートスペースは独自の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（PCB），VPN データベースを用意することでネットワークの仮想化を行う．同一ホスト内の全ての実行環境が同じ IP アドレスを使えるようにするために，ネットワークパケットは IPsec のセキュリティパラメタインデックス（SPI）によって振り分けられる．SPI は IPsec トンネル毎に一意の値を持つ．IPsec でない通信はベースシステムによって処理される．また，ポートスペース毎に PCB を持つことにより，同じ IP アドレスを持つポートスペース間でネットワークポートの衝突を避けることができる．

4.2.3 プロセスの仮想化

ポートスペース内のプロセスはほとんどの点において通常のプロセスと同じである．独立したアドレス空間を持ち，ホスト内で一意なプロセス ID を持つ．違いは，仮想化されたファイルシステムと仮想化されたネットワークシステムのみを使うことと，同じポートスペース内のプロセスしか見ることができない点である．

4.3 サーバスペース

サーバスペースの実装はネットワーク部分を除いてポートスペースと同じである．サーバスペースはベースシステムと同じネットワークをベースシステムと同じ IP アドレスで使用する．これを可能にするために，サーバスペースへのネットワークパケットはサーバが使用している TCP または UDP のポート番号によって振り分けられる．サーバが使用するポート番号はサーバの起動時にカーネル内のテーブルに登録され，ホストがパケットを受け取るとそのテーブルを元に適切なサーバスペースに配送される．

4.4 監視機構

4.4.1 ファイルシステムの監視

サーバスペースのファイルシステムを監視するために，各サーバスペースのルートディレクトリを各ポートスペースの `/.serverfs` に読み出し専用で union マウントする．ポートスペースからサーバスペースのファイルシステムを直接指定することはできないので，このマウントはカーネル内で行われる．このマウントにより IDS が監視するパスが従来のパスから変更されるので，管理者は IDS のポリシーファイルを変更する必要がある．

サーバスペースのファイルシステム（union ファイルシステム）の上の層だけを各ポートスペースの `/.dserverfs` に union マウントすることもできる．このマウントにより，IDS はサーバスペースのファイルシステムの変更部分のみを検査することができる．

4.4.2 ネットワークの監視

ホストに送られた全てのパケットを監視するために，ベースシステムのパケットフィルタ（bpf）デバイスをポートスペースにマップする．ベースシステムの bpf デバイスは`/dev/bpf255` からデバイス番号が小さくなる順で予約され，ポートスペースの `/dev/bpf0` からデバイス番号が大きくなる順にマップされる．

サーバスペース内の Unix ドメインソケットに送られたメッセージをポートスペースの中から盗み見するために，`mkdup` システムコールを提供している．このシステムコールは指定されたパス名を持つ Unix ドメインソケットをサーバスペース内に作り，そのソケットとポートスペース内で指定されたパス名を持つ Unix ドメインソケットを結びつける．サーバスペース内の Unix ドメインソケットに送られたメッセージは複製され，ポートスペース内の Unix ドメインソケットの受信バッファに入れられる．

4.4.3 プロセスの監視

サーバスペース内のプロセスが発行するシステムコールをトレースするために，ポートスペースからサーバスペースのプロセス ID を取得することができる．このプロセス ID を用いて `ptrace` システムコールを使うことが許されている．安全のためにプロセスの停止などを行うことはできない．

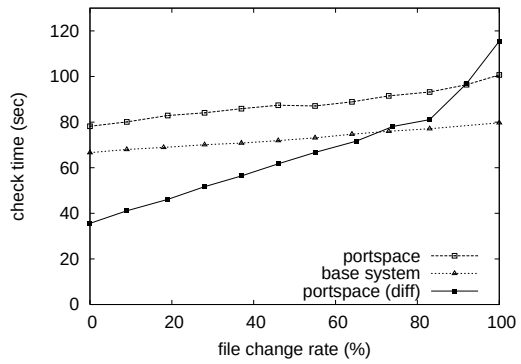

図 3: ファイル変更率毎の tripwire の検査時間

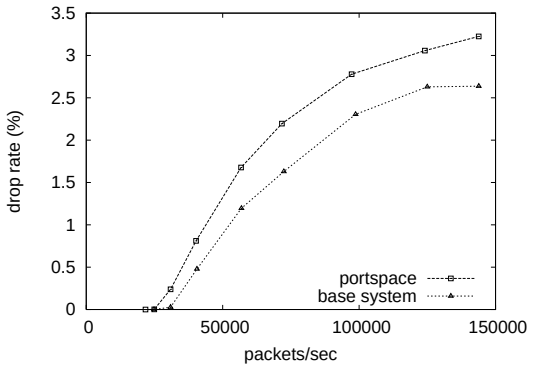

図 4: 負荷毎の snort のパケットドロップ率

5 実験

我々のシステムにおいて IDS とサーバはそれぞれポートスペースとサーバスペースのオーバヘッドを被る．そのオーバヘッドを調べるために，我々はいくつかの IDS とサーバの性能を測定した．この実験では Pentium 4 3.0GHz の CPU，1GB のメモリ，Intel Pro/100+のネットワークカードを備えた PC を 2 台用意し，100Base-T のスイッチングハブで接続した．

5.1 ポートスペースのオーバヘッド

5.1.1 Tripwire

Tripwire [6] はファイル情報をあらかじめデータベースに保存しておき，定期的にファイルに対する変更を検査する．我々はポートスペース内で動いている Tripwire が，サーバスペースのファイルシステムの整合性を検査するのにかかる時間を全ファイルに対する変更率を変えながら測定した．比較のために，Tripwire をベースシステムで動かした場合についても測定した．また，サーバスペースのファイルシステムの変更部分だけを検査するように改良した Tripwire をポートスペース内で動かした場合の性能も測定した．

図 3 はその結果を示している．Tripwire をポートスペース内で動かすことにより，17%〜26%のオーバヘッドを被っている．ファイルの変更率が高くなるほどオーバヘッドも大きくなっている．一方，ファイルシステムの変更部分だけを検査した場合，ファイルの変更率が 70%を越えるまでは検査にかかる時間が短くて済んでいる．特に，ファイルの変更率が低い場合は約半分の時間しかかかっていない．

5.1.2 Snort

snort [10] はネットワークパケットを盗み見し，攻撃パターンを格納したシグネチャデータベースと比較することで攻撃を検出する．我々はポートスペースで動いている snort の性能を測定した．性能の指標としては，snort によるパケットドロップ率を用いた．実験には snort 2.1.2 および標準の 1,779 個のルールを用いた．snort に様々な負荷を与えるために，snort の動いているホストに 1 バイトの UDP パケットを様々なレートで送った．比較のために，snort をベースシステムで動かした場合についても測定を行った．

図 4 はこの実験の結果を示している．1 秒に約 15 万パケット送った場合のドロップ率は，snort をポートスペースで動かした場合は 3.2%，ベースシステムで動かした場合は 2.6%であった．このように，ネットワーク負荷が高い場合，我々のシステムを使うことによって 0.5%程度のオーバヘッドを被っていることが分かる．

5.1.3 Truss

truss は FreeBSD 標準のシステムコールをトレースするコマンドである．侵入検知を行うわけではないので厳密には IDS ではないが，システムコールをトレースするオーバヘッドを見積もるために truss を用いて実験を行った．この実験では，ウェブサーバ thttpd 2.23beta1 [11] の発行するシステムコールを truss によってトレースし，リモートホストで ApacheBench [1] を使って性能を測定した．truss を動かす環境を変えて以下の 2 通りの実験を行った．

- truss をポートスペースで動かし，thttpd をサーバスペースで動かす

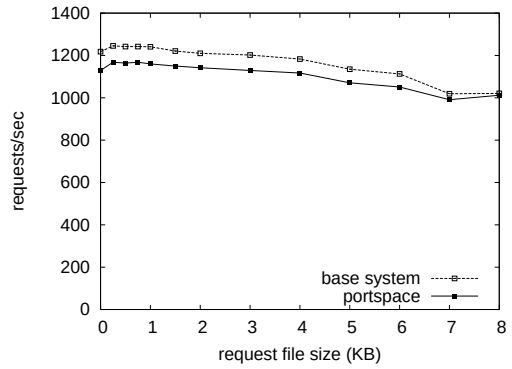

図 5: truss を用いた場合の thttpd の性能

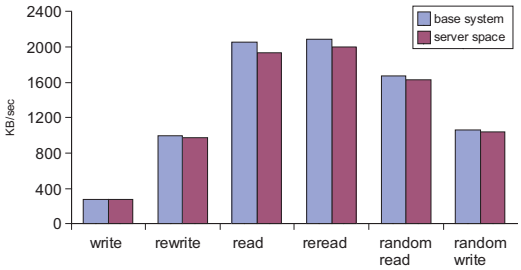

図 6: サーバスペースのファイル I/O 性能

- truss と thttpd をベースシステムで動かす

実験結果は図 5 のようになった．truss をポートスペースで動かした場合，ベースシステムで動かした場合と比べると，0.8%〜7.3%のオーバヘッドを被っている．ポートスペースでトレースするオーバヘッドはほとんどないと考えられるので，このオーバヘッドの大部分はサーバスペースのオーバヘッドであると考えられる．

5.2 サーバスペースのオーバヘッド

5.2.1 マイクロベンチマーク

まず，iozone ベンチマーク [4] を用いてファイル I/O の性能を測定した．この実験では，新しいファイルへの書き込み（write），既存のファイルへの書き込み（rewrite），既存のファイルの読み出し（read），最近読まれたファイルの読み出し（reread），ランダム読み書き（random read/write）の性能を測定した．ファイルサイズは 1MB，レコードサイズは 1KB とした．実験結果は図 6 のようになり，union ファイルシステムのオーバヘッドは最大で 6%程度であることが分かる．

次に，netperf ベンチマーク [9] を用いてネットワーク性能を測定した．netperf は TCP/IP および UDP/IP の往復のレイテンシとスループットを測定することができる．レイテンシの測定で送信するパケットのデータサイズは 1 バイトとした．また，スループットの測定における送信バッファのサイズは TCP で 32KB，UDP で 9KB とし，受信バッファのサイズは TCP で 56KB，UDP で約 40KB とした．表 1 の実験結果から，サーバスペースがネットワーク性能に及ぼす影響はほとんどないことが分かる．

表 1: サーバスペースにおけるレイテンシ（μsec）およびスループット（$Mbps$）

	レイテンシ		スループット	
	TCP	UDP	TCP	UDP
ベースシステム	185.1	179.0	93.84	96.28
サーバスペース	185.7	179.8	93.84	96.28

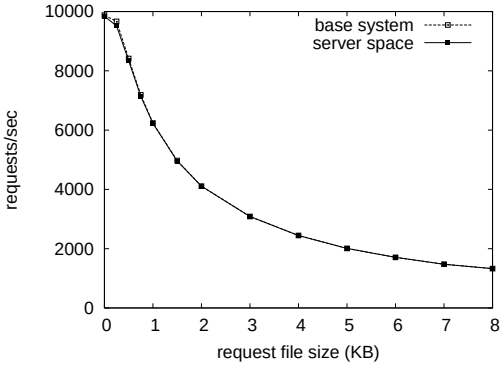

図 7: サーバスペースにおける thttpd の性能

5.2.2 ウェブサーバ

サーバスペースで thttpd を動かした時の性能をリモートホストで ApacheBench を用いて測定した．図 7 からウェブサーバの性能はサーバスペースで動かすことによるオーバヘッドをほとんど受けていないことが分かる．特に，リクエストされるファイルサイズが小さい時は最大で 1%程度のオーバヘッドがあるが，ファイルサイズが大きくなるとオーバヘッドは 0.1%以下になっている．

6 関連研究

LIDS [8] などのカーネル内 IDS はサーバから分離されており，サーバ経由で攻撃するのは難しい．しかし，完全に分離されているわけではなく，カーネル

内 IDS がファイルシステムから IDS のデータベースを読んだりログファイルを書いたりすると，プロセスから影響を受けるかもしれない．また，他のホストのカーネル内 IDS やプロセスと通信していると，その通信チャネルから攻撃されるかもしれない．

SODA [5] はサーバや OS を仮想マシン内に閉じ込め，Kernort と呼ばれるネットワーク型 IDS を仮想マシン内のゲスト OS のカーネルに埋め込む．カーネル内 IDS と違い，Kernort はログデータを仮想マシン外のホスト OS のファイルシステムに格納するため，ログデータは仮想マシンに対する攻撃の影響を受けない．しかし，Kernort 自体は仮想マシン内にあるため，サーバから完全には分離されておらず，攻撃を受ける可能性がある．

ReVirt [2] はサーバと OS を仮想マシン内で動かし，ロガーをホスト OS 上で動かす．仮想マシンの実行に関するすべてのイベントは仮想マシンのハードウェアレベルでログが取られ，ロガーによってホスト OS 上のファイルシステムに格納される．そのログデータに基づいて実行をリプレイすることで攻撃を解析することができる．ログは自動的に取られるので，ロガーは仮想マシンから完全に分離されている．しかし，ホスト OS 上のロガーに対する攻撃には対処できない．

Livewire [3] は ReVirt と似ているが，ホスト OS 上で動く IDS が仮想マシンを能動的に監視することができる．Livewire の IDS はハードウェアレベルの状態を監視したり，ゲスト OS の構造の知識を用いてゲスト OS の状態を監視したりできる．Livewire は完全に IDS をサーバから分離しているが，ReVirt 同様に IDS に対する攻撃には対処できない．

これらのシステムがサーバを分離しようとしているのに対し，我々のシステムは IDS を分離しようとしている．このような設計は IDS 同士が通信しなければならない分散 IDS では IDS を守るために重要である．また，これらのシステムでは専用の IDS を開発しなければならないが，我々のシステムでは既存の IDS を利用することができる．これは実際に運用する場合に重要である．

7 まとめ

本稿では，分散 IDS の安全性を向上させることができる HyperSpector を提案した．HyperSpector はポートスペースと呼ばれる仮想マシンと VPN から成る独立した仮想分散環境である．ポートスペースは独立したファイルシステム空間，ネットワークシステム空間，プロセス空間を提供する．分散 IDS は HyperSpector の中で動かされ，分散システムのその他の部分から分離される．HyperSpector を用いることにより，分散 IDS は能動的な攻撃から守られ，受動的な攻撃の被害を HyperSpector 内部に限定することができる．

今後の課題の 1 つは，受動的な攻撃への対処で分散 IDS を多重化する際に，負荷を制御できるようにすることである．例えば，ファイルシステムの整合性を検査する IDS を複数動かす場合，うまくスケジューリングすることによってシステムの負荷を下げることができるかもしれない．別の課題としては，DoS 攻撃への対処も挙げられる．ホストが DoS 攻撃を受けると分散 IDS の機能がほとんど停止してしまい，攻撃の検出が遅れる場合がある．我々は性能の点からも HyperSpector を分散システムから分離できるようにする必要があると考えている．

参考文献

[1] `http://www.apache.org/`.

[2] Dunlap, G., King, S., Cinar, S., Basrai, M. and Chen, P.: ReVirt: Enabling Intrusion Analysis through Virtual-machine Logging and Replay, in *Proc. of Symp. on Operating Systems Design and Implementation*, pp. 211–224 (2002).

[3] Garfinkel, T. and Rosenblum, M.: A Virtual Machine Introspection Based Architecture for Intrusion Detection, in *Proc. of Network and Distributed Systems Security Symposium*, pp. 191–206 (2003).

[4] `http://www.iozone.org/`.

[5] Jiang, X., Xu, D. and Eigenmann, R.: Protection Mechanisms for Application Service Hosting Platforms, in *Proc. of Symp. on Cluster Computing and the Grid* (2004).

[6] Kim, G. and Spafford, E.: The Design and Implementation of Tripwire: A File System Integrity Checker, in *Proc. of Conf. on Computer and Communications Security*, pp. 18–29 (1994).

[7] 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 *Proc. of Conf. on Local Computer Networks*, pp. 385–394 (2003).

[8] `http://www.lids.org/` .

[9] `http://www.netperf.org/` .

[10] Roesch, M.: Snort – Lightweight Intrusion Detection for Networks, in *Proc. of System Administration Conference* (1999).

[11] `http://www.acme.com/software/thttpd/` .

[12] 光来, 廣津, 佐藤, 明石, 福田, 菅原, 千葉：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, コンピュータソフトウェア, Vol. 21, No. 1, pp. 2–12 (2004).