Snortを用いた侵入防止システムの構築と侵入検知処理高速化の検討

林 經正† 横山 幹‡ 高原 厚† 岩橋 政宏‡
†NTT 未来ねっと研究所 ‡長岡技術科学大学

近年、ブロードバンド通信サービスの普及にともない、多くのコンピュータがインターネットに常時接続されるようになってきた。これにともない、ネットワークを介した不正侵入の被害を受ける件数が非常に多くなっている。このような不正侵入行為からネットワークシステムを守るために侵入検知システム (IDS) が開発されてきた。しかしながら、IDS が検知するネットワークシステムのトラヒック量が大きくなると、IDS での侵入検知処理が追いつかなくなり検知率が急激に落ちてしまう問題がある。そこで、本稿では、ハードウェア/ソフトウェア協調処理による侵入検知処理の高速化システムを提案し、実ネットワークサービス環境を想定した実験ネットワークで本システムの検知処理能力の評価を行ったので報告する。

Evaluation of Intrusion Prevention System (IPS) with Snort using a high-speed hardware/software detection architecture

Tsunemasa Hayashi† Motoki Yokoyama‡ Atsushi Takahara† Masahiro Iwahashi ‡
†NTT Network Innovation Laboratories ‡Nagaoka University of Technology

In recent years, many people have connected to their computers to the Internet through broadband access lines. In this situation, computers are always running and the number of illegal attacks are increasing. This has becomes a serious problem. In order to prevent the damage from illegal attacks, Intrusion detection system (IDS) has been developed. However, when the bandwidth of background traffic in the network system becomes large, the operation performance of IDS rapidly becomes worse. In this paper, we propose a hardware/software co-operation security system with IDS, and evaluate the system.

1 はじめに

近年、ADSL (Asymmetric Degital Subscriber Line)、FTTH (Fiber to The Home) などのブロードバンド通信サービスの普及にともない、多くのコンピュータがインターネットに常時接続されるようになってきた。これにともない、ネットワークを介したリモートからの不正侵入攻撃で被害を受ける件数が非常に多くなってきている。このような不正侵入行為からネットワークシステムを守るために、過去の侵入手口をデータベース (シグニチャ) 化し、侵入検知を行なう侵入検知システム (Intrusion Detection System: IDS) が開発されてきている [1, 2]。

IDS は、ネットワークを介して接続されるコンピュータ間の通信の振舞いを解析することにより、不正な動きがないか検知することが基本となるが、実際のネットワーク環境下で使用される IDS を用いたセキュリティシステムでは、Ether や IP などのヘッダ情報を元にパケットフィルタリング処理を行い、続いてパケット内のデータ解析、プロトコル状態遷移管理を行なうことになる。つまり、IDS を用いたセキュリティシステムは、ファイヤーウォールなどで用いられているフィルタ処理も必要とする。

IDS による侵入誤検知 (誤検知) を減らすため、また、将来起こりうる未知の攻撃に対応するため、Data Mining 等の技術を用いることにより対応していく技術が検討されている [3, 4, 5] が、実環境の IDS に適用して運用するまでには完成度は高くない。

一方で、実ネットワーク環境下では IDS の処理速度が問題になってくる可能性があることが報告されている [6]。つまり、Web サーバなどの検知対象となるシステムへのアクセス量が大きくなると IDS の処理が追い付かなくなり、検知率が急激に落ちる、または、検知できなくなるという問題である。これは、検知対象となるパケット内のデータ解析やプロトコル状態遷移管理といった IDS の処理を演算プロセッサを用いた逐次処理で行なわなければならないため、単位時間辺りに処理しなければならないパケット数が多くなると、IDS の処理能力が極端に低下するためである。このような IDS の処理速度の問題に対して、IDS の DoS アタック耐性を評価する

ためのベンチマークテストの指針が検討されている[7]が、検知対象となるネットワークシステムに流れる全てのトラヒックが攻撃パケットで構成されていることを仮定しているため、DoS アタック耐性以外の処理能力評価には直接用いることができない。そこで我々は、不正侵入攻撃が最も行なわれる条件を想定し、バックグラウンドトラヒックを Web サーバ等へ通常にアクセスするトラヒックとした。この環境下で不正侵入攻撃を行なうことに対する IDS の検知処理能力を評価することとした。本稿では、高バックグラウンドトラヒック時においても、十分に検知処理できる IDS を用いたセキュリティシステムの高速化を検討し、検知結果を元に攻撃元との通信を遮断処理する侵入防止システムの評価を行なったので報告する。

2 侵入検知処理の高速化手法

我々は、IDS として Snort[1] を用いたセキュリティシステムを構築し、侵入検知処理能力を評価すると共に、処理能力の高速化を検討した。Snort は市販 IDS ツールと比較して、高トラヒックやフラグメント化された検知しにくいパケットを用いた侵入攻撃に対しても、高い検知処理能力有することが報告されている [6]。また、最新の侵入被害に対しても対応が早い。Snort は、プリプロセッサの処理後、シグニチャを用いた検索処理を行なう。Snort の処理は、パケット識別によるフィルタリング処理、文字列などのパターンマッチング処理、プロトコル解析処理の 3 つに分けることができる。我々は、IDS 検知処理のうち最も処理負荷が大きくなる原因になる文字列パターンマッチング処理の高速化に着目し、パターンマッチング・アルゴリズム高速化と、パターンマッチングルーチン処理の負荷削減を検討した。また、不正侵入検知後、リモートコンピュータとの通信を遮断する Flex Resp 機能の評価を行なった。

2.1 探索木パターンマッチアルゴリズム

Snort では、パターンマッチに Boyer-Moore アルゴリズム (BM アルゴリズム) [8] を用いている。これは、基本的にシグニチャ内の全ての検索文字列とパケット内データとの間でパターンマッチ処理を行なうもので、検索文字列数が多くなるとマッチング処理量が非常に多くなり、単位時間辺りに処理できるパケット数が少なくなる。

BM アルゴリズムに対して Aho-Corasick Boyer-Moore (ACBM) アルゴリズム [9] は、検索する文字列に対して予め共通文字列を親ノードとする探索木 (検索木) を作成しておき、その探索木に従いマッチングを行っていく手法である。探索木が縮退した場合、マッチングする処理数を削減できるため、BM アルゴリズムよりもマッチング処理数を少なくすることができ、高速処理が期待できる。[1]

2.2 カーネル・パケットフィルタリングの併用

次に我々は、IDS は導入するネットワークシステムに応じて検知するパケットの種類、プロトコルが異なることに注目した。例えば、Web サーバのみのネットワークシステムには、FTP、Radius などのサーバへのアクセスパケットを処理する必要がない。そこで、カーネルでパケットを識別しフィルタリングを行なうことにより、IDS で処理するパケット削減することを検討した。カーネルフィルタとして、iptable を用いた。図 1 に iptable のパケットフィルタリングと SnortIDS の協調動作の構成図を示す。予め iptable のフィルタリングルールに設定したパケットだけが libpcap のキュー (ip queue) に格納される。ipqueue に格納されたパケットだけが Snort で処理されるため、単位時間あたりに処理するパケット数を削減できる可能性がある。

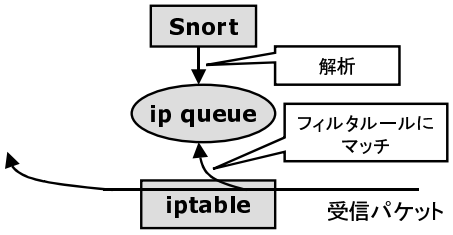

図 1: カーネルでのパケットフィルタと SnortIDS の併用構成図

2.3 ハードウェア/ソフトウェア協調処理

次に我々は、パケットフィルタリングをハードウェアで処理し、パターンマッチ処理をソフトウェアで処理することにより、高バックグラウンドトラヒック状況下でもパターンマッチ処理可能な IDS システムについて検討した。図 2 に試作したネットワーク装置 エッジデバイスを用いたシステムの概略構成

[1] ACBM アルゴリズムを実装した Snort は、Snort version2.X としてリリースされる予定 (2003 年 3 月現在)。

図を示す。エッジデバイスは、ノンブロッキング・ワイヤースピードでで 2.4Gbps の処理能力を持つネットワークスイッチ ASIC (BCM5600)[10] と汎用 CPU を搭載する。この ASIC により、受信パケットの Ether フレームの先頭から 64Byte の bit 一致検索することで、不必要なパケットをフィルタリングする。それ以外のパケットを汎用 CPU で動作する Snort で処理させる。この IDS システムでは、CPU 上の処理結果を元に ASIC を制御することが可能であり、フィルタリングルール、転送パケットのプライオリティ(IP パケットの TOS ビット、802.1p のプライオリティビット) などを変更できる。

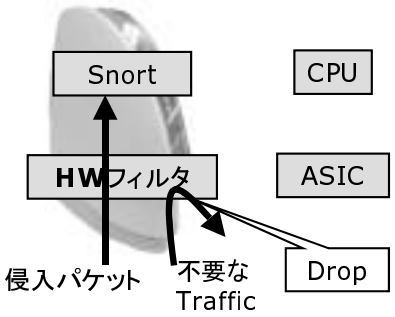

図 2: エッジデバイスを用いた HW/SW 協調動作による IDS 構成

2.4 実験環境、実験方法

図 3 に、今回構築した IDS 評価実験ネットワークの構成を示す。

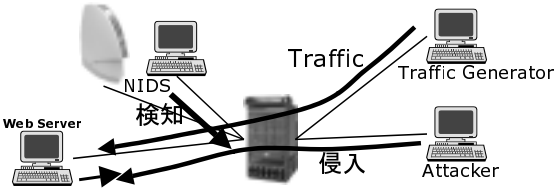

図 3: IDS 評価実験ネットワーク構成図

攻撃対象として Web サーバ (apache) を立ち上げ、トラヒックジェネレータからは TCP アクセスを摸擬したトラヒックを生成し、Web サーバに送信する。このような状況下でアタッカから検知試験に用いる文字列”codered”をデータに格納した侵入 (攻撃) パケットを Web サーバに 100 パケット送信し、トラヒックジェネレータから送出されるバックトラヒック帯域に対する検知率 (送信攻撃パケット 100 に対する検知したパケット数の割合) の測定を行なった。検知率の測定回数は 5 回とし、測定値の最大値と最小値を除いた残り 3 つの値の平均値を測定結果とした。Snort を動作させるシステムは下記に示す 4 種類 (HIDS、NIDS1~3) とし、検知率の測定を行なった。HIDS は、攻撃対象である Web サーバ上に IDS を構築したもので、NIDS1~3 は、ネットワーク型 IDS である。被検索文字列が 7 バイトであるため、各プロセッサのキャッシュヒット率に大きな違いがないと仮定する。また、Snort を用いた本 IDS の侵入検知処理能力と比較するために、Snort と同様にシグニチャを用いる IDS Dragon6.0 を搭載した DM [11] を評価した。

実験環境の詳細を下記に示す。

HIDS (汎用 PC で構成): PentiumIII 1GHz (L1 cache 64KB, L2 cache 256KB), メインメモリ 256MB, OS: RedHat7.3 LINUX

NIDS1 (汎用 PC で構成): PentiumIII 1GHz (L1 cache 64KB, L2 cache 256KB), メインメモリ 256MB, OS: RedHat7.3 LINUX

NIDS2 (試作したエッジデバイス)：Geode GX1 (PentiumII 300MHz 相当、L1 cache 16KB), メインメモリ 64MB, OS: FreeBSD4.1

NIDS3 (米国 MRV 社 OptiSwitch (LightReef) [12] 上の AP-CPU で構成): PowerPC 366MHz (L1 cache 32KB, L2cache 32KB), メインメモリ 128MB, OS: Debian3.0α LINUX

Attacker (汎用 PC で構成): PentiumIII 550MHz, メインメモリ 128MB, OS: RedHat7.3 LINUX

Traffic Generator SmartBits [13]。ハードウェアトラヒックジェネレータ

実験で使用した攻撃・検疫ツールを以下に示す。

Nmap フリーウェアのポートスキャナ、ステルススキャンなどの各種ポートスキャンを実行可能

CASL [14] 、ネットワークパケットを自由に編集し、プロトコル処理シーケンスを任意に変更してリモートホストと通信が可能

SmartWindow SmartBits を制御するアプリケーション。任意のパケットを作成・送信し、トラヒック帯域のコントロールが可能

3　検知率の評価

3.1　従来手法の評価

まず、既存の Snort の検知率を測定した。図 4 はシグニチャ数 689、図 5 はシグニチャ数 1 でのバックグラウンドトラヒックの大きさに対する Snort IDS の検知率の結果である。

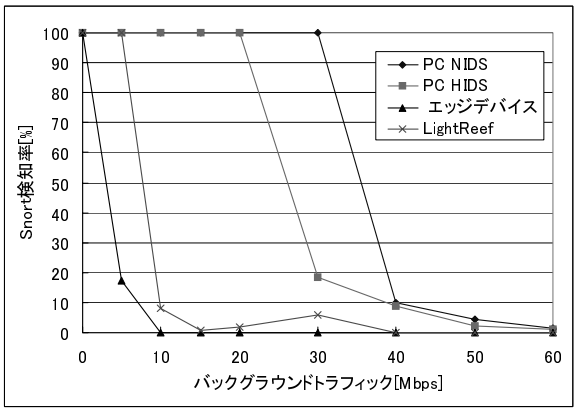

図 4: BM アルゴリズム Snort のパフォーマンス（シグニチャ数 689）

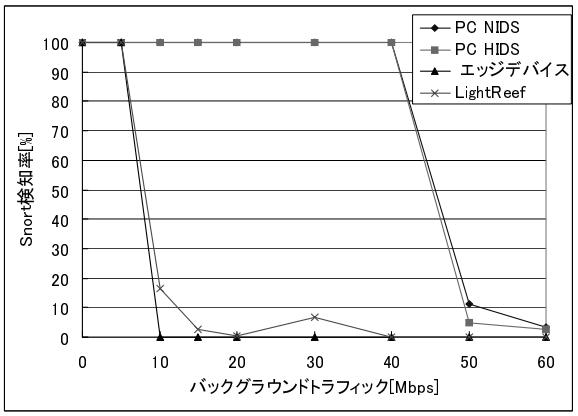

図 5: BM アルゴリズム Snort のパフォーマンス（シグニチャ数 1）

この結果から、シグニチャ数が多いと検知シーケンス処理が多くなり、相対的にパフォーマンスが落ちることがわかる。今回実験に用いた汎用 PC に 2 つの物理インターフェースを搭載し、ソケットインタフェースを用いたソフトウェアフォワーダを動作させると、単一方向 (上り、または下りのパケットのどちらか) のパケットフォワードを行なう場合、100Mbps を越える処理能力を持っていることから、バックグラウンドトラヒックが、30 ～ 50Mbps 前後で急激に検知率が低下するのは、汎用 PC の PCI バスなどの物理的な構成によるものではなく、IDS のシーケンス処理の負荷が大きいからと考えられる。

NIDS に対して HIDS は、バックグラウンドトラヒックに対するカーネルでのパケット処理 (TCP SYN パケットに対し SYN/ACK パケットを返す処理) や、Web サーバアプリケーションにより、CPU リソースが奪われるため、IDS のパフォーマンスが低下する。

3.2　探索木パターンマッチ型 IDS の検知率

次に、NIDS1 に対して ACBM アルゴリズムを適用し、検知率を測定した。結果を図 6 に示す。

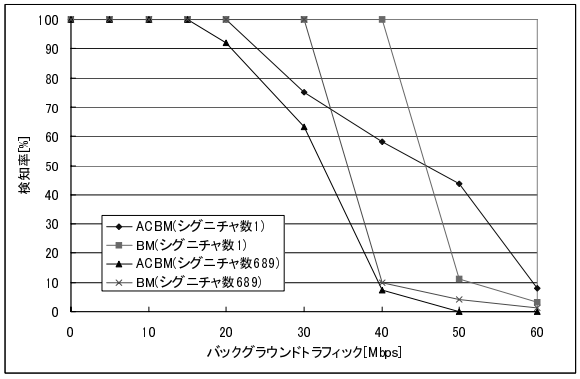

図 6: BM と ACBM の検知率評価 (NIDS1/汎用 PC)

ACBM アルゴリズムを適用した際に探索木が縮退することが期待できる場合においても、BM アルゴリズムによる検知率より低下している。ACBM アルゴリズムの探索木が縮退して検索処理数が少なくなるよりも、ノード分岐処理が大きくなっていると考えられる。また、本実験では、侵入パケットに格納されている文字列が 7 バイトと小さく、IP パケットに格納されている位置も全て同じことから、非常に単純なパターンマッチング処理となっている。このため、BM アルゴリズムを用い、CPU のキャッシュにヒットする処理の方が処理ステップ数が短くなっている可能性がある。

3.3　カーネル・パケットフィルタと協調動作する IDS の検知率

次に、カーネルでのパケットフィルタと IDS との協調動作による高速化を評価した。LightReef 上の NIDS3 に、iptable によるカーネル・パケットフィルタと IDS を協調動作させたシステムの検知率測定結果を図 7 に示す。

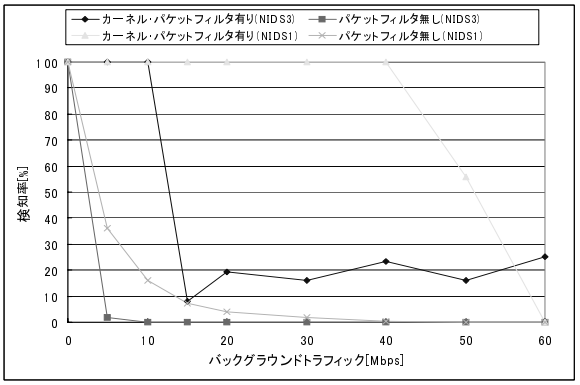

図 7: カーネルでのパケットフィルタと協調動作する Snort IDS(NIDS1, NIDS3) の検知率、シグニチャ数 1277)

80 ポートへの TCP アクセスするパケットを iptableでフィルタリングし、Snort で処理するパケットの数を減らすことにより、検知処理パフォーマンスは向上した。60Mbps の高バックグラウンド状況下においても、侵入パケットを検知していることがわかる。しかし、CPU パフォーマンスの小さい NIDS3 では、20Mbps 以上の高バックグラウンドトラフィック下での検知率は低下している。測定結果にバラツキが多く (測定平均に対して、各測定結果は 50%くらいのバラツキがあった) システムが不安定である。一方、CPU パフォーマンスが高い NIDS1 では、バックグラウンドトラヒックが 40Mbps まで 100%の検知率である。これは、パケットフィルタリングにより、IDS での処理負荷が軽減されたためである。しかし、50Mbps を越えるバックグラウンドトラヒック下では、極端に検知率が低下していくことがわかる。これは、カーネルでのフィルタリング処理の負荷が非常に高くなり、インターフェースカードからカーネルへの DMA 転送、カーネルから IDS への DMA 転送が追い付かなくなっているからだと考えられる。

3.4 ハードウェア/ソフトウェア協調処理 IDS の検知率

次に、ハードウェアパケットフィルタと IDS との協調動作による高速化を評価した。ASIC のハードウェアフィルタにより、80 番ポートに TCP アクセスするパケットをフィルタリングした場合の検知率を図 8 に示す。60Mbps の高バックグラウンドトラフィック下においても、100%の検知率を実現できることを確認した。このことから、パケットフィルタリングをハードウェアで処理し、パケット内データのパターンマッチング処理をソフトウェアで処理することにより、高トラヒック時に対応できる IDS を構築できることを確認した。

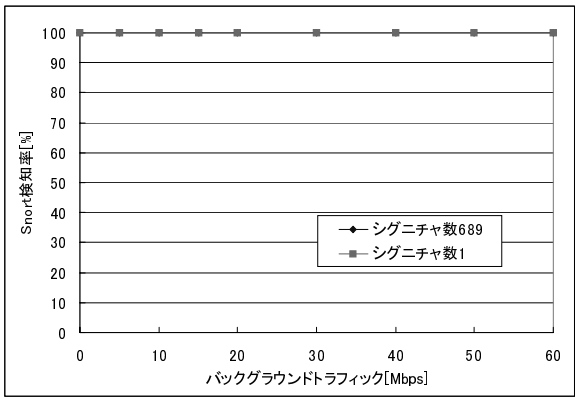

図 8: HW/SW 協調動作による Snort の検知率

3.5 DM シリーズの評価

次に、Dragon 6.0 を搭載した DM1000 の NIDS の検知率を測定した。DM1000 は 2 つの CPU によるマルチプロセッシング構成を取り、使用する Linux kernel はマルチプロセッシング対応の SMP 版である。下記に DM1000 の構成を示す。

DM1000 CPU: Intel XEON 1.8GHz × 2 (L1 cache: 16KB, L2 cache: 512KB), メインメモリ 1GB, OS: Linux (kernel-2.4.9 SMP 版, ベースは RedHat), ポート数 3 (10/100/1000BaseT)

検知率の測定結果を図 9 に示す。

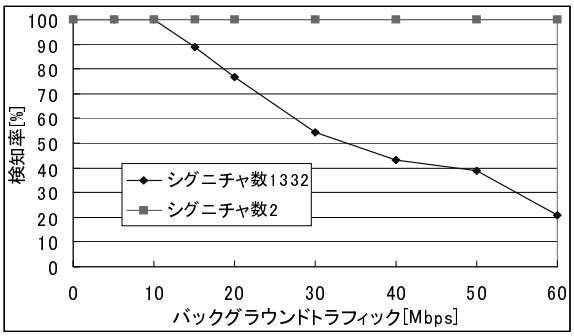

図 9: DM1000 の NIDS パフォーマンス

シグニチャ数が 2 の場合、バックグラウンドトラヒックが増加しても検知率が 100% である。しかしながらシグニチャ数が 1332 と多くなると、検索文

字列数が多くなるため、バックグラウンドトラヒックが高くなるにつれて検知率が下がる。このことからも、ソフトウェアだけの処理だけでは、高バックグラウンドトラヒック時に侵入検知処理が十分機能しなくなる可能性がある事がわかる。

3.6 FlexResp 機能の評価

最後に、Snort の FlexResp 機能による通信強制遮断機能の評価を行なった。FlexResp は、侵入検知結果を元に、目的ホストに対して TCP RST パケット等を送信することで、通信を強制遮断する。アタッカからのポートスキャンに対し、Web アクセス以外の TCP ポート 22 と 23 において通信強制遮断を試みた。実験結果を表 1 に示す。

表 1: Nmap でのポートスキャン結果

状態	通常時		遮断時	
TCP ポート番号	22	23	22	23
TCP connection scan	開	閉	開	閉
TCP half scan	開	閉	開	閉
FIN scan	開	閉	閉	閉
NULL scan	開	閉	閉	閉
Xmas scan	開	閉	閉	閉

攻撃対象となった Web サーバでは、Web アクセスに使用する 80 番ポート以外に、22 番ポートが開いていて 23 番ポートが閉じている (表 1 中の通常時)。FlexResp 機能を動作させると、両ポートとも閉じているように振舞わなければならない。しかし、この実験結果から、TCP connection scan、TCP half scan に対して、22 番ポートが開いていることをアタッカに伝えてしまった。これは、アタッカからの TCP SYN パケットに対し、FlexResp 機能による TCP RST パケットが送信される前に、カーネルの処理で SYN/ACK パケットが送信されてしまうからである。OS 上のアプリケーションレベルで実行される Snort だけでは、完全な FrexResp 機能をリモートに対して行なうことができない。この対処法として、カーネルとの協調動作が必要だと考える。

4 まとめ

IDS を用いたセキュリティシステムを構築する場合、CPU のソフトウェア処理だけでは、検知率の高い安定したシステムを構成できない。IDS を導入するネットワークシステムに応じて、検知に不必要なパケットをハードウェアフィルタリングすることにより、高バックグラウンド時にでも安定した IDS を構築できることを確認した。本稿で提案したシステムでは、エッジデバイス上のソフトウェアからハードウェアフィルタの設定を変更することが可能である。例えば、エッジデバイスを通過するパケットのプライオリティを変更することもできる。このようにハードウェアフィルタの設定を運用環境に応じて変更することで、よりセキュアなネットワークが構築可能となる。

謝辞

ネットワークセキュリティの情報や、貴重なアドバイスを頂いた NTT 情報流通プラットフォーム研究所の岡田 浩一氏、NTT 未来ねっと研究所の谷井 一人氏、NTT-AT の平塚 政樹氏、NTT サイバーソリューション研究所の倉橋 孝雄氏、(株) 日立国際電気サービスの岡本 陽児氏に感謝いたします。

参考文献

[1] The Open Source Network Intrusion Detection System http://www.snort.org.

[2] ENTERASYS Networks. Dragon http://www.enterasys.com/.

[3] Wenke Lee, S.J. Stolfo, P.K. Chan, E. Eskin, Wei Fan, S. Miller, M. andHershkop, and Junxin Zhang. Real time data mining-based intrusion detection. *DARPA Information Survivability Conference & Exposition II*, Vol. 1, pp. 89–100, 2001.

[4] Sung-Bae Cho. Incorporating soft computing techniques into a probabilistic intrusion detection system. *Systems, Man and Cybernetics, Part C, IEEE Transactions*, Vol. 32, pp. 154–160, 2002.

[5] R. Brooks, N. Orr, J. Zachary, and C Griffin. An interacting automata model for network protection. *Information Fusion*, Vol. 2, pp. 1090–1097, 2002.

[6] 伊藤良孝, 平井伸幸. Ids の検知能力を検証する. *Nikkei Internet Technology*, 2002.9.

[7] T. Champion and M.L. Denz. A benchmark evaluation of network intrusion detection systems. *Aerospace Conference, 2001, IEEE Proceedings.*, Vol. 6, pp. 2705–2712, 2001.

[8] BOYER R.S. and MOORE J.S. A fast string searching algorithm. *Communications of the ACM.*, No. 20, 1979.

[9] B. Commentz-Walter. A string matching algorithm fast on the average. 1979.

[10] BroadCom Corp. *StrataSwitch*. http://www.broadcom.com.

[11] 販売: 住友商事株式会社、製造: 横河電機株式会社. http://www.sumitomocorp.co.jp/jyohodenki/dragon/.

[12] OptiSwitch-Z Family (Modular L2/3/4 Switch/Routers) http://www.mrv.com.

[13] SmartBits network performance analysis system http://www.spirentcom.com/.

[14] Custom Attack Simulation Language http://www.sockpuppet.org/tqbf/TopLevel/CASL.html.