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

2008－CSEC－42（21）
2008 / 7 / 25

攻撃履歴を利用したシグネチャ型IDSのDoS耐性の向上

宮澤 僚太† 阿部 公輝†

† 電気通信大学 情報工学科 〒182-8585 東京都調布市調布ヶ丘 1-5-1
E-mail: †{miyazaw-r,abe}@cacao.cs.uec.ac.jp

あらまし シグネチャ型侵入検知システムにおいて，DoS攻撃に対するシグネチャ検索の処理負荷を軽減すること，および，ログ出力のためのIOwait時間を短縮するための手法を提案する．提案手法は次の2つからなる．(1) 検知効率を向上させるため，攻撃の種類であるシグネチャIDと，注目すべき攻撃文字列の位置を示すオフセット値の2つを履歴として記憶し，より最近ヒットしたシグネチャとの照合を優先的に行う．(2) 必要以上のログ出力によるIOwaitを減らすために，アラートが発せられる前に履歴を参照し，同様のアラートが短時間内に出力されていれば，内部処理の段階でログ出力イベントを取り消し，検知回数だけを記録するようにする．提案手法は，標準Snortを用いた評価実験により，一般的なDoS攻撃に対しては約70%，複数種類の同時DoS攻撃に対しては約40%処理負荷を軽減することがわかった．

キーワード 侵入検知，シグネチャ型，DoS攻撃，攻撃履歴，処理負荷

Improving Resistance to DoS using Attack History in Signature-based Intrusion Detection Systems

Ryota MIYAZAWA† and Kôki ABE†

† Department of Computer Science, The University of Electro-Communications, 1-5-1 Chofugaoka, Chofu-shi, Tokyo 182-8585 Japan
E-mail: †{miyazaw-r,abe}@cacao.cs.uec.ac.jp

Abstract We propose a method to reduce processing load for signature matching and IOwait time for log output of signature-based intrusion detection systems (IDSes) against DoS attacks. The method consists of the flowing two ideas: (1)To improve the detection efficiency, holding a history of pairs of signature ID matched with a substring and offset pointing to the substring in attack packets, we perform a prioritized search in the history. (2)To reduce the IOwait time, without issuing a large amount of alerts, we refer to an alert history and cancel the log output events and count the number of detections if the same kind of attacks has been detected within short period. Experimental evaluation using the standard Snort with modifications according to the proposed ideas revealed that processing load of the IDS is reduced by 70% and 40% against a typical DoS attack and multiple kinds of DoS attacks, respectively.

Key words Intrusion detection, signature based, DoS attack, attack history, processing load

1. はじめに

ネットワークに対する脅威としてDoS攻撃がある．DoS攻撃はサーバに対して大量にアクセス要求をしたり，攻撃パケットを送信し続けることで，帯域やサーバの資源を消費させてサービスやネットワークへのアクセスを妨害することを目的としている．このような攻撃を防ぐ手段として，侵入検知システム(Intrusion Detection System,IDS)がある．IDSとは，ネットワーク上の不正なアクセスなどの兆候を検知し，管理者に知らせる機能を持つシステムである．IDSには予め定義されたルールに対応したパケットを検知すると警告を発するシグネチャ型と，統計的学習論的手法によって異常を発見するアノーマリ型がある．

シグネチャ型IDSは，攻撃パケットに含まれる特徴的な文字列（シグネチャ）をあらかじめ辞書に登録しておき，監視するパケットをそれと比較検索することで攻撃を検知するものである．このため，正常なトラフィックを異常とみなす誤検知(false-positive)が起きにくく検知精度が高い．しかし，登録

されたシグネチャ以外は検知できないため，未知の攻撃には対応できない．

アノーマリ型IDSは，DoS攻撃のデータに直接的な攻撃データが含まれていない場合でも，異常トラフィックや振る舞いを解析することで攻撃を検知できる．しかし，アノーマリ型にはfalse-positiveが多いことや，シグネチャ型でしか検出できない攻撃も多数存在するため，IDSの信頼性を保つためにはアノーマリ型とシグネチャ型の両方を組み合わせて用いるのが現実的である．

シグネチャ型IDSが行う処理の中で最も重要となるものの一つがアプリケーション層のシグネチャ検索である．シグネチャ検索ではIPアドレスや送信先ポート番号といった固定長のヘッダ部分だけでなく可変長のペイロードも検索対象となるため，その処理負荷は重い．また，DoS攻撃が行われるとIDSのスループットは極端に下がる．さらに，IDSそのものに対するDoS攻撃は，あえて検知されるような攻撃パケットを大量に送信することで，それを監視しているIDSに重い負荷を与える攻撃である．この攻撃は，シグネチャ検索という負荷の高い処理をIDSに強要することの他に，検知後にログを出力させることでCPUに大量のIOwait時間を消費させるという意図ももつ．

本研究ではシグネチャ型IDSにおいて，DoS攻撃に対するシグネチャ検索の処理負荷を軽減すること，および，ログ出力のためのIOwait時間を短縮するための手法を提案する．提案手法は，攻撃の履歴を持つことで次に来る攻撃パケットを予測し，検知処理負荷の軽減とログ出力の効率化を図る．

以下では，第2章で関連研究，第3章で提案手法を述べる．第4章で評価実験の結果を述べ，第5章でまとめる．

2. 関連研究

シグネチャ型IDSの処理負荷を軽減するには，ハードウェア検索エンジンを採用する方法がある[1]．しかしハードウェアは処理は速いが，精度を求めると回路規模が増大する．ソフトウェアは処理は遅いが検出精度は高い．そこで，シグネチャ検索をハードウェアとソフトウェアによる多段処理で行うことが考えられる[2]．しかし，この手法では攻撃パケットの検知は必ずソフトウェア処理に回されるため，攻撃パケットのみを送信するようなIDSに対するDoS攻撃には耐性がない．

代表的なDoS攻撃であるSYNFlood攻撃はサーバに大量の接続要求を出すことでTCPリソースを消費させ，通常のクライアントによるサーバへのアクセスを困難にさせる攻撃である．TCPの3WayHandShakeを行う前に，IPアドレス等を要素として生成した特殊なシーケンス番号によって，セッションを確立しているクライアントをTCPリソースを使用せずに特定する手法がある[3]．この手法は，IDSに対するDoS攻撃ツールとして有名なstick[4]やsnot[5]を悪用し，セッションを確立せずに行う攻撃を未然に防ぐことができる．しかし，NimdaやCodeRedのようにセッションを確立してから感染行為を行うような攻撃を未然に防ぐことはできず，IDSに高い負荷をかける可能性が残る．

DoS攻撃の検出効率を考慮したアノーマリ型IDSの研究[6]では，SYNFlood攻撃やポートスキャンなどのよく知られたDoS攻撃の特徴をあらかじめシステムに登録しておく．そして，監視しているパケットのヘッダ情報を対象ホストごとに時系列順に並べ，そのパケットの振る舞いが，登録されている攻撃の特徴と類似しているかを検査することでDoS攻撃を検出する．この手法は，ペイロード検索を行わないのでシグネチャ型より検知処理負荷は低いと考えられる．しかしSYNFlood攻撃のように特徴がつかみやすい攻撃では検出精度が高いが，特徴がつかみにくいポートスキャンでは精度が低い．したがって，IDSをかく乱する目的で行われるIDSに対するDoS攻撃の多くは正常パケットとしてシグネチャ型検出に渡され，IDSに膨大な処理負荷が生まれてしまう可能性がある．

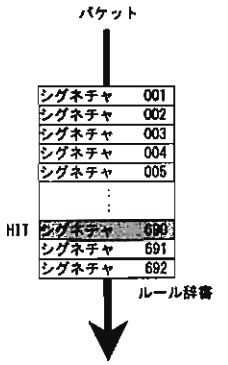

図1 攻撃文字列を検知した後の従来法の様子

Fig. 1 An attack signature matched with a substring of an attack packet in a conventional database of rules.

3. 提案手法

DoS攻撃は短時間に似たような内容のパケットが集中的に観測されるという特徴がある．そこで本研究では，攻撃を受けたときにその内容を履歴に取っておくことで，次回から同様のパケットに対しては処理負荷を下げるため，次の2つの手法を提案する．(1) 検知効率を向上させるため，攻撃の種類であるシグネチャIDと，注目すべき攻撃文字列の位置を示すオフセット値の2つを履歴として記憶し，より最近ヒットしたシグネチャとの照合を優先的に行う．(2) 必要以上のログ出力によるIOwaitを減らすために，アラートが発せられる前に履歴を参照し，同様のアラートが短時間内に出力されていれば，内部処理の段階でログ出力イベントを取り消し，検知回数だけを記録するようにする．

手法 (1) において，攻撃文字列を検知した後の従来法の様子を図1に，提案手法の様子を図2および図3に示した．図は692個のシグネチャを持つルール辞書でパケットを検知している様子である．従来法では履歴という概念がないので，攻撃文字列を検知してもルール辞書には何も手を加えない．しかし提案手法では履歴辞書を新たに加えることで優先レベルを2つに分け，DoS攻撃を受けた時にまず図2のようにルール辞書で攻撃文字列を検知すると，図3に示したように履歴辞書にその情

報を登録し，以降その攻撃は履歴辞書によって検知される．履歴辞書はルール辞書に比べ小さいので，検索時間は短い．履歴辞書で検知されなかったパケットは，従来のルール辞書で検索されるため，検索精度は従来法と変わらない．履歴辞書の履歴保持数は有限であるので，保持数を超えた場合は履歴の古いものから削除される．

近年の IDS では Aho-Corasick 法 [8] による複数文字列の同時探索を行う．これはルール辞書が保持するシグネチャの数が増大し，Boyer-Moore 法 [7] では効率が悪いからである．しか

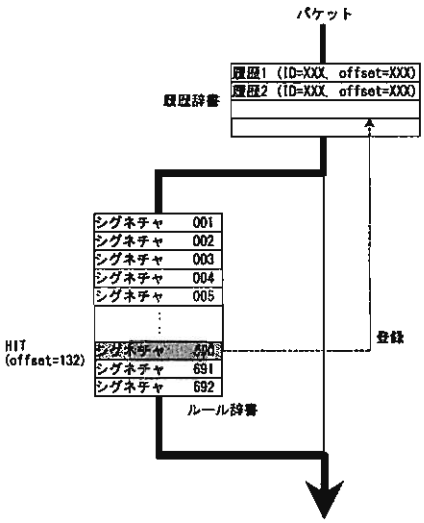

図 2 提案手法 (1): ルール辞書における攻撃文字列の検知と検知情報の履歴辞書への登録

Fig. 2 Proposed method (1): Detection of an attack packet with a database of attack rules and its registration to a history table.

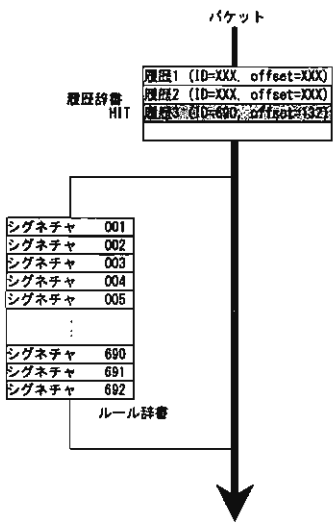

図 3 提案手法 (1): 履歴辞書による検知

Fig. 3 Proposed method (1): Detection with the history table.

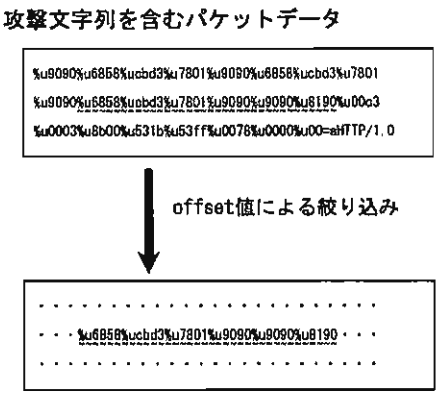

図 4 オフセット値の位置から始まる文字列の照合

Fig. 4 Pattern matching with a string from the position pointed by offset.

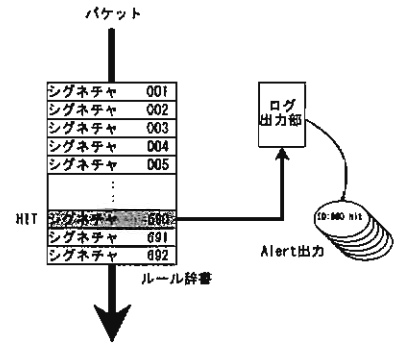

図 5 従来法の DoS 攻撃検知時のログ出力

Fig. 5 Log output when detecting DoS attack in conventional method.

し Aho-Corasick 法の計算量は対象文字列の長さに線形であるため DoS 攻撃を検知する場合，攻撃文字列にたどり着くまで無駄な検索をしてしまう．そこで手法 (1) では DoS 攻撃の検知効率を上げるために，オフセット値 *offset* と攻撃文字列の長さ *length* を記憶し，図 4 のように *offset* から始まる長さ *length* の文字列ひとつを照合対象とすることで，無駄な検索をなくした．

手法 (2) は，DoS 攻撃の検知により大量のログを出力することで膨大な処理負荷が発生してしまうことを防ぐためのものである．従来法では図 5 のようにルール辞書で攻撃を検知するとその結果をログ出力部に送り，同じ警告を各攻撃パケットごとに発していた．手法 (2) では，DoS 攻撃の開始時に警告を発した後，図 6 のように検知された攻撃の情報を履歴辞書に登録する．DoS 攻撃継続中はログを出力せず，検知情報を検知カウンタに送り，攻撃が終了した時点で攻撃回数を出力する．攻撃が複数種類同時に仕掛けられたときは，全ての攻撃情報が検知カウンタで保持され，攻撃が終了したものから出力される．検知カウンタは図 7 に示すように連続する攻撃の種類と回数を記憶している．

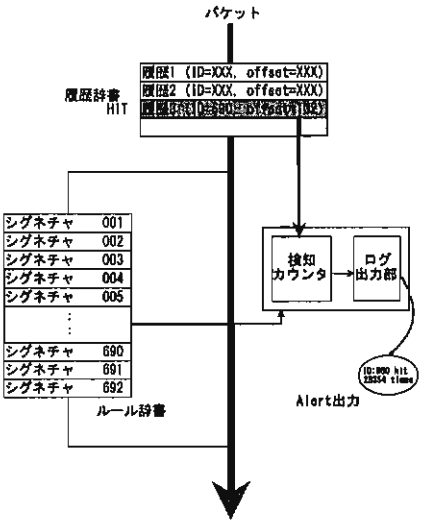

図 6 提案手法 (2) の DoS 攻撃検知時のログ出力

Fig. 6 Log output when detecting DoS attack in proposed method (2).

シグネチャID	攻撃回数
004	1500
690	28354
⋮	⋮

図 7 検知カウンタ

Fig. 7 Detection counter.

4. 評価実験と考察

提案手法の実装は IDS として最も有名なオープンソースソフトウェアである Snort [9] に対してプログラムの変更という形で行った．図 8 に示すように，Snort は監視するパケットに対し，まずプリプロセッサーによる前処理を行い，次に再構成されたパケットを TCP や UDP などのプロトコルごとに分類する．その後，パケット内に攻撃文字列がないかを検知エンジン部で検索し，ルール辞書に登録されているシグネチャと一致する文字列があれば，その結果をログ出力決定部に送り，イベントキューを介して警告を出力する．そうでなければそのパケットは正常と判断され処理を終了する．

提案手法 (1) は，図 8 において，検知エンジン部とシグネチャ構造体の間に検知履歴参照部を挿入し，それを優先的に参照することにより実装する．提案手法 (2) は，ログ出力決定部とイベントキューの間に，検知カウンタを持つ出力履歴参照部を挿入することにより実装する．標準 Snort に提案手法 (1) のみを実装したものを Snort-D，提案手法 (2) のみを実装したものを Snort-L，両手法を実装したものを Snort-DL と呼ぶ．検知履歴参照部，出力履歴参照部で用いる履歴長をそれぞれ h_s, h_l とする．

以下では，まず 1 パケットあたりの攻撃検知処理時間の面で提案手法を評価する．次に DoS 攻撃を受けた時の CPU 使用率を調べる．さらに，複数種類の DoS 攻撃を受けたとき，保持する履歴サイズにより CPU 使用率がどう変わるかを調べる．標準 Snort として Snort-2.7.0.1 をルール辞書 snortrules-pr-2.4 と共に用いる．標準 Snort，Snort-D，Snort-L，Snort-DL を次の環境で実行する．

- OS：CentOS4.5 (on VMware Server)
- メモリ：256MB
- CPU：AMD Athlon64 X2 2400MHz
- シグネチャ数：692 (HTTP に関連するシグネチャを選択)
- CPU 使用率の測定に vmstat を使用
- 攻撃パケットの送信に hping2 を使用

攻撃パケットとしては，Snort のルールにヒットするように，Nimda や CodeRed などに現れる文字列を含む疑似的なパケットを生成して用いる．

4.1 パケットあたりの攻撃検知処理時間

標準 Snort，Snort-D，Snort-L，Snort-DL に攻撃パケットを流し，1 パケットあたりの処理時間を測定した．攻撃パケットのデータサイズは，100 バイトとフラグメンテーションを起こさない最大値である 1500 バイトの 2 通りとした．測定は 30 回行い，その結果の平均を求めた．

実験の結果を表 4.1 に示す．パケットサイズが 100 バイトのとき攻撃パケットの処理時間は，標準 Snort を 1 として Snort-D は 0.91，Snort-L は 0.43，Snort-DL は 0.32 となった．検知履歴の記憶と優先参照（提案手法 (1)）により約 10%，ログ履歴の記憶と出力制限（提案手法 (2)）により約 60%，両手法の適用により合計約 70%処理時間が削減されることがわかった．攻撃パケットのサイズが大きくなると処理時間は増加するが，削

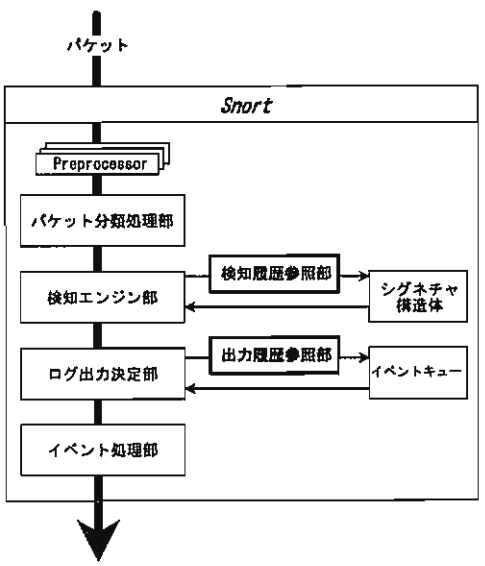

図 8 標準 Snort への提案手法の実装: (1) 検知履歴参照部の挿入; (2) 出力履歴参照部の挿入

Fig. 8 Implementation of proposed methods to standard Snort: (1)Insertion of detection history table; (2)Insertion of log history table.

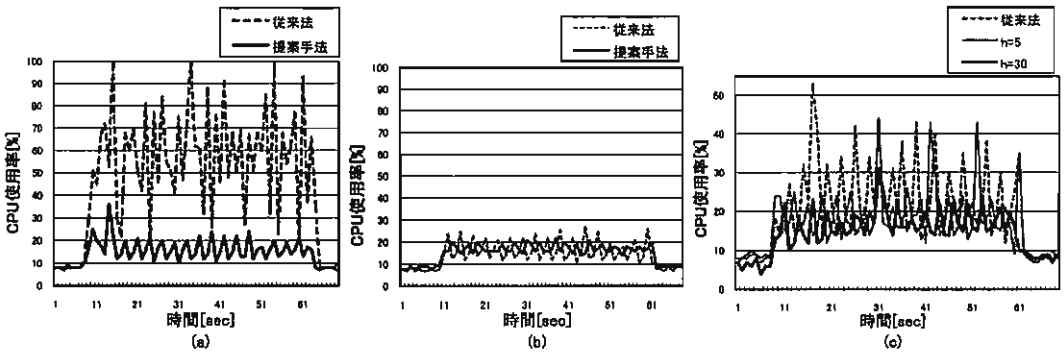

図 9　提案手法 (Snort-DL) と従来法 (標準 Snort) の CPU 使用率: (a) 一般 ($a = 1$) の DoS 攻撃時 (履歴保持数 $h = 5$ とする); (b) 正常時 ($h = 5$ とする); (c) 複数種類 ($a = 20$) の DoS 攻撃時 ($h = 5, 30$ とする)

Fig. 9　CPU usage by Snort-DL and standard Snort: (a)against a DoS attack ($a = 1$) with $h = 5$; (b)for normal packets with $h = 5$; (c)against several kinds of DoS attacks ($a = 20$) with $h = 5, 30$.

表 1　パケットあたりの攻撃検知処理時間

Table 1　Processing time of detecting an attack packet.

	パケット長			
	100byte		1500byte	
	msec	ratio	msec	ratio
標準 Snort	0.64	1.00	0.80	1.00
Snort-D	0.59	0.91	0.68	0.85
Snort-L	0.27	0.43	0.31	0.39
Snort-DL	0.21	0.32	0.24	0.30

減率はあまり変わらない.

4.2　DoS 攻撃を受けた時の CPU 使用率

DoS 攻撃に対する検知処理負荷を調べるため，次の 3 つの実験を標準 Snort と Snort-DL に対して行う.

(1) 一般的な DoS 攻撃は単一種類のパケットを連続して送信することで行われる．この時の CPU 使用率を測定する.

(2) 提案手法のオーバーヘッドを調べるために，正常なパケットにおいても CPU 使用率を測定する.

(3)IDS に対する DoS 攻撃の場合，複数種類の DoS 攻撃を受けることが考えられるため，複数種類の DoS 攻撃を受けているときの CPU 使用率を測定する.

実験 (1) と (2) ではパケットのデータサイズは 500 バイトとし，10μ 秒ごとに 1 つのパケットを送信する．トラフィック量は 400Mbps となる．Snort-DL における履歴保持数は $h_s = h_t = 5$ とする．実験 (3) では，IDSwakeup [10] を参考にし，50μ 秒ごとに 100 バイトの攻撃パケットを送信する．トラフィック量は 16Mbps となる．また IDS に対する DoS 攻撃として多くの通常パケットを送ることで処理負荷を上げる手段も存在するので，攻撃パケットと同数の正常パケットも同時に送信する．どの実験においてもパケット送信時間は 50 秒とする．測定は攻撃パケットの送信時間の前後 10 秒を含む計 70 秒間にわたって行った．実験 (3) では履歴の最大保持数 h_s，h_t の値と攻撃種類数 a の関係が CPU 使用率にどのように影響するかも調べる．ここでは $h_s = h_t = h$ とする．$h < a$ と $h > a$ の 2 つの場合を調べるために，$a = 20$ とし，$h = 5$ と $h = 30$ の 2 つの条件で実験を行った.

実験 (1)，(2)，(3) の結果を図 9 の (a)，(b)，(c) に示す．(a) から一般的な DoS 攻撃検出時には提案手法は処理負荷を大きく軽減することが分かる．(b) から提案手法のオーバーヘッドは無視できる程度であることがわかる．また (c) から IDS の混乱を狙って加えられる複数種類の DoS 攻撃に対して，提案手法は有効であることがわかる．履歴保持数 h が攻撃種類数 $a = 20$ より小さい場合 ($h = 5$) と比較し，大きい場合 ($h = 30$) の方が，測定時間全体を通して CPU 使用率は低いことがわかる.

4.3　攻撃種類数と履歴保持数の関係

4.2 の実験 (3) では $h < a$ と $h > a$ の 2 つの条件で CPU 使用率を測定した．ここでは攻撃種類数に対し履歴保持数を変えたときに CPU 使用率がどのように変わるかを調べるために a を 20，30，40 とし，それぞれに対して h を 0 から 50 まで変化させて実験を行う．攻撃を 20 秒間加えその間の平均値を求める実験を 10 回行い，10 回分の平均値，最大値，最小値を求める．$a = 20$，30，40 に対する結果をそれぞれ図 10 の (a)，(b)，(c) に示す.

これらの図から $h \leq a$ のときは履歴数 h を増やすと処理負荷も低減していくことがわかる．例えば $a = 20$ のとき $h = 0$ では (標準 Snort に対応)，CPU 使用率は約 35% であるが，$h \geq 20$ とすると約 20%に減少する．すなわち攻撃種類数以上の履歴を確保していれば，処理負荷は 40%程度改善される.

5.　ま　と　め

本研究では，履歴を考慮することで DoS 攻撃の検知効率を向上させる手法を提案した．評価実験により，一般的な DoS 攻撃に対しては約 70%，複数種類の同時 DoS 攻撃に対しては約 40%処理負荷が低減することがわかった.

ネットワークや IDS 自身に対する DoS 攻撃を受けると，IDS

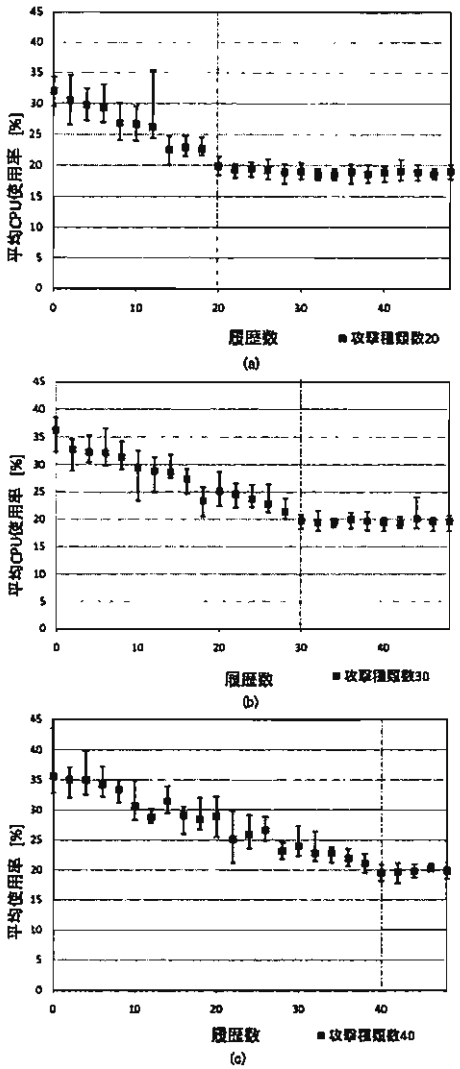

図 10 攻撃種類数 a と履歴保持数 h を変えたときの平均 CPU 使用率: (a)$a = 20$; (b)$a = 30$; (c)$a = 40$

Fig. 10 Average CPU usage when the number of DoS attack kinds and the length of attack history are varied: (a)$a = 20$; (b)$a = 30$; (c)$a = 40$.

の処理負荷だけでなくログを解析する管理者にも膨大な負荷がかかる．そこでログを機械的に解釈し，攻撃の種類や数などをグラフィカルに表示することにより，管理者の作業量を大幅に減らすことも重要である．提案手法 (2) は IOwait 時間や CPU 負荷を軽減するが，グラフィカルに表示するだけの情報量は提供できない．DoS 攻撃に対し，IOwait 時間や CPU 負荷を増加させず，管理者が正確に判断できるだけの情報を提示することは，今後の課題である．

謝辞 本研究は，一部，日本学術振興会科学研究費補助金 (基盤研究 (C)18500048) による．

文　献

[1] H. Song, T. Sproull, M. Attig, and J. Lockwood, "Snort offloader: a reconfigurable hardware NIDS filter," Proc. International Conference on Field Programmable Logic and Applications, pp.493-498, 2005.

[2] 小林礼明, 阿部公輝, "Karp-Rabin 法を用いたシグネチャ型 IDS の性能コスト評価," 情報処理学会研究会報告, Vol.2007, No.48, pp.73-78, 2007.

[3] 伊藤大輔, 泉裕, 斉藤彰一, 上原哲太郎, 國枝義敏, "TCP セッション管理による DoS 耐性の考察," 情報処理学会研究会報告, Vol.2001, No.111, pp.183-190, 2001.

[4] http://www.eurocompton.net/stick/

[5] http://www.stolenshoes.net/sniph/index.html

[6] S. Pukkawanna, V. Visoottiviseth, and P. Pongpaibool, "Lightweight Detection of DoS Attacks," Proc. 15th IEEE International Conference on Networks, pp.77-82, 2007.

[7] R. S. Boyer and J. S. Moore, "A fast string seaching algorithm," Commun. ACM, Vol.20, No.10, pp.762-772, 1977.

[8] A. V. Aho and M. J. Corasick, "Efficient string matching: An aid to bibliographic search," Commun. ACM, Vol.18, No.6, pp.333-340,1975.

[9] http://www.snort.org/

[10] http://www.hsc.fr/ressources/outils/idswakeup/index.html.en