「マルチメディア，分散，協調とモバイル
(DICOMO2019) シンポジウム」 令和元年7月

複数ウィンドウサイズの効率的監視による バースト形態に依らないDDoS攻撃検出手法の検討

臼崎 翔太郎[1] 油田 健太郎[1] 山場 久昭[1] 朴 美娘[2] 岡崎 直宣[1]

概要：DDoS 攻撃検知では，検知しようとする攻撃のバースト形態に合わせてウィンドウサイズを決定しておかないとその発生の検知が困難となったり，他のバースト形態の DDoS 攻撃の検知精度が悪化することがある．本手法ではインターネットトラフィックを複数のウィンドウサイズで監視することによって，ウィンドウサイズの調整が必要のない統括的な監視を行い，これによってバースト形態に依らない DDoS 攻撃検出の検討を行った．評価実験では，複数のウィンドウサイズの監視によって検知精度が改善されるか，また，同時監視という工夫によって処理性能が低下してしまわないかを調べた．実験の結果，単一のウィンドウサイズを利用した場合よりも複数で監視した時の方が検知精度が高かった．また，計算量は，既存のネットワークトラフィック監視手法と同等程度であり，この改良による処理性能への悪影響は少ないといえる．今後の課題として，高レートでない DDoS 攻撃に対する検知精度の向上が必要である．

Elastic Denial-of-Service Attack Detection Method by Monitoring with Multiple Window Size

SHOTARO USUZAKI[1] KENTARO ABURADA[1] HISAAKI YAMABA[1] MIRANG PARK[2]
NAONOBU OKAZAKI[1]

1. はじめに

不正なトラフィックを送信しサーバをサービス停止状態に追い込む DDoS（Distributed Denial-of-Service）攻撃は，インターネットが社会インフラとなっている昨今ではその検知および緩和処理が重要となっている．DDoS 攻撃は，IoT ボットネットによる攻撃やリフレクション攻撃の流行により，攻撃の規模が 2012 年を境に年々増加しており [1]，2018 年現在，DDoS 攻撃として史上最大規模である 1.3Tbps 以上の攻撃が観測されている [2]．実被害として，2000 年には大手の検索サービスの Yahoo[3] や，DNS のルートサーバ [4] への攻撃が報告されている．また 2016 年には DNS サービスを提供する Dyn 社に対しての攻撃が観測されている [5]．この攻撃により Twitter や Spotify などが一時的に利用できなくなる事態となり，社会的に大きな影響をもたらした．

攻撃被害の緩和のために高精度な DDoS 攻撃検知手法が必要となるが，攻撃検知では，検知しようとする攻撃のバースト形態に合わせてウィンドウサイズを決定しておかないとその発生の検知が困難となったり，他のタイプの DDoS 攻撃の検知に影響を与えたりすることがある．例えば，ウィンドウサイズを小さくした際にはパルス型の DDoS 攻撃 [6][7] を検知できる可能性があるが，ノイズの影響によって通常通信を攻撃だと判定してしまうおそれがある．一方，ウィンドウサイズを大きくすると，ノイズの影響は小さくなり検知精度の向上が見込める．しかしながら，パルス型の DDoS 攻撃などの微小な期間に出現する攻撃の特徴が薄れてしまう．さらに，ウィンドウサイズが大きくなると検知頻度が低下するため，リアルタイムに攻撃の開始や終了を検知できない可能性がある．

本研究ではインターネットトラフィックを複数のウィンドウサイズで並列的に監視することによって，攻撃のバースト形態に合わせたウィンドウサイズ調整が必要のない統括的な監視を行う．これによってバースト形態に依らずに DDoS 攻撃を検出することを目指す．この複数ウィンドウ

[1] 宮崎大学
[2] 神奈川工科大学

© 2019 Information Processing Society of Japan

サイズの同時監視の有用性は，調べた限りではこれまで十分に検討されていない．

この工夫によって，単一のウィンドウサイズのみを監視した時に比べて検知精度が向上するという利点が考えられる．一方，並列的な監視によって処理性能が低下するおそれがある．これらについて，検知精度が向上するか，また処理性能が著しく低下してしまわないかという観点から，実験によって評価する．

本論文は以下の構成となる．第 2 章では DDoS 攻撃の既存の検知手法について述べるとともに問題点を分析する．第 3 章では，提案手法について詳細に述べ，第 4 章では提案手法の性能を調査するための実験環境と実験方法について説明する．第 5 章では実験結果について述べたあとその結果を考察し，第 6 章で結論を述べる．

2. 関連研究

本章では既存の DDoS 攻撃検知手法について述べる．特に，アノマリ型検知手法について，ウィンドウサイズが攻撃検知精度や攻撃検知速度などの性能に影響を与えることを示す．

DDoS 攻撃検知手法は，シグネチャ型検知手法とアノマリ型検知手法の二つのタイプに分類される．

シグネチャ型手法 [8] はあらかじめ既知の攻撃の特徴をパターンとして保持しておき，そのパターンと比較してマッチした時に攻撃として検知する手法である．広く利用されているものに Snort[9] がある．この方法では事前に正しいパターンが登録されているときには非常に高い攻撃検知精度を持つ手法である．この登録されたパターンのことをシグネチャと呼ぶ．しかしながらパケットごとに処理を行う必要があるため，パターンの登録数やパケットが大量に到着した際にはリアルタイム性に欠けたり，機器に負担がかかったりするおそれがあることが指摘されている [10][11]．また，事前にシグネチャを登録しておく必要があるという特徴から未知の攻撃には対応することが不可能であるという点，常に最新のパターンを反映しなければならない欠点も存在する [12]．

アノマリ型検知手法は，ウィンドウサイズという単位ごとに特徴量を計算し，その値が正常状態のものとどれだけ乖離しているかで異常が起きているかを判定する手法である．シグネチャ型検知手法と比較した時のアノマリ型検知手法の利点は，未知の異常を検知できる可能性がある点や拡張が容易である点があげられる．また，変化点検知手法や統計的検知手法に関しては処理効率が高く，リアルタイム性が高い．しかしながら，インターネットトラフィックにおいて正常状態を定義することが難しい点や [13]，パラメータを時々刻々で調整しなければならない点が欠点として挙げられる．さらに，一般的にウィンドウサイズを大きくするとノイズの影響は軽減できるが，検知頻度が減少するため，リアルタイム性が減少する．

アノマリ型検知手法の中で，高速計算性を持つ方法ながら高い精度を持ち，広く利用されているのがエントロピーベース手法である．エントロピーベース手法は，パケットのヘッダ情報を情報源としてエントロピー値を計算し，それらの増減を監視して攻撃を検知する手法である．利用されるエントロピー値としては，最も有名なシャノンによるエントロピーが多い．一般にエントロピー手法では，送信元 IP アドレス，宛先 IP アドレス，送信元ポート番号，宛先ポート番号，プロトコルといった 5-Tuple の要素が情報源として利用される．この中でも最も利用されるのが，送信元 IP アドレスと宛先 IP アドレスである．小島ら [14] は上記の項目に加えてパケット数，TTL など計 9 つの情報量を利用しているが，最も攻撃検知精度に寄与した成分が前述の 2 つの IP アドレスであったと結論づけている．

エントロピーベース手法の一般的な攻撃検知の仕組みを説明する．エントロピー値には乱雑度が高いほど大きくなり、低いほど小さくなるという特徴があり，これを利用して DDoS 攻撃の特徴を捉えている．例えば，送信元 IP アドレスを情報源とするエントロピー値を H_s，宛先 IP アドレスを情報源とするエントロピー値を H_d とする．DDoS 攻撃時には，大量のホストから攻撃が到来するため大量の送信元 IP アドレスがサンプル内に出現し H_s が増加していく．また，ある特定の宛先に攻撃を行うことから，H_d ではある特定の宛先 IP アドレスの出現回数が極端に多くなるため，乱雑度が減ったとして H_d が減少していく．このように，複数のエントロピー値の変動具合を監視して攻撃の検知を行う．

ここで各エントロピー値の具体的な算出方法を説明する．S を計算対象となるサンプルデータの系列，n_i をシンボル i の出現数とすると，時間 t におけるエントロピー値算出式を以下で表す．

$$H(t) = -\sum_{i=1}^{n} p_i \log p_i \qquad (1)$$

一方でエントロピー手法は、ウィンドウサイズの適切な決め方が問題点となる．外れ値の影響を小さくするために，ウィンドウサイズが時間単位の場合には 1 分程度 [15]，パケット数単位であれば数万程度 [16] のウィンドウサイズが推奨されているが、ウィンドウサイズを大きくすると処理効率が悪くなることが指摘されている [14]．リアルタイム性を向上させるために，一般的にはウィンドウサイズを小さくして検知処理の頻度を多くすること考えられるが，その代わりにノイズの影響が大きくなり検知精度に悪影響を及ぼしてしまう．

このように，アノマリ型の DDoS 攻撃検知手法では，ウィンドウサイズは攻撃検知精度や攻撃検知速度などの性能に影響を与える．

© 2019 Information Processing Society of Japan

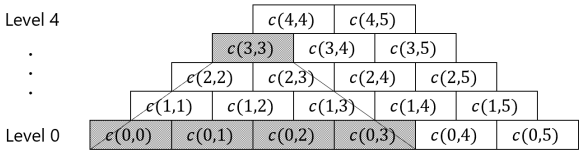

図 1　$L = 5$ の時の Aggregation Pyramid の図．$c(3,3)$ は $c(0,0)$ から $c(0,3)$ までのセルデータを集約した情報を保持している．

3. 提案手法

DDoS 攻撃検知では，検知しようとする攻撃のバースト形態に合わせてウィンドウサイズを決定しておかないとその発生の検知が困難となったり，他のバースト形態の DDoS 攻撃の検知精度が低下することがある．

そこで提案手法では，インターネットトラフィックを複数ウィンドウサイズで並列的に監視することにより，バースト形態に依らない DDoS 攻撃検知を行う．具体的には，(1) まず Aggregation Pyramid と呼ばれるデータ構造 [18] を利用する．この構造は，複数のウィンドウサイズのネットワークトラフィックの特徴量を効率的に集約し，保持することができる．(2) そして，それぞれの特徴量について，マハラノビス距離を用いて，事前学習していたモデルとの距離を計算し，その距離が大きい時に異常が起きていると判定する．この判定を各ウィンドウサイズについて行い，過半数で異常と判定された時に，DDoS 攻撃が発生していると判定する．(3) さらに，現在のトラフィック状況への追従性を向上させるため，モデルは事前学習のみではなく逐次学習も行う．攻撃でないと判定された期間の特徴量のみを学習データとして保存し十分な数だけ集まったら更新する．

3.1 データ構造

本研究のアプローチを実現するためには，複数のウィンドウサイズの特徴量を計算できる必要がある．また，これに加え，インターネットトラフィック監視の文脈では，通常時にも大量データが発生することを考慮して，効率的にデータを保持できるデータ構造が望まれる [19][20][21]．これら二つの特徴を持つデータ構造である，Aggregation Pyramid を本研究では利用する．

Aggregation Pyramid は複数のセルで構成されたピラミッドの構造をしており（図 1），レベル 0 からレベル $L-1$ までの L 個の階層が存在する．ひとつひとつのセルは，所属する階層を l，セルの生成タイミングを t とした時，$c(l,t)$ と表現される．同じ t の値を有するセル，例えば図 1 での $c(0,1)$ と $c(1,1)$ は，同じタイミングで生成される．

最下位セル ($l = 0$) は，最小ウィンドウサイズ W_{min} ごと（ただし，直前のセル生成時刻とその次に到着したパケットの到着時刻の差が W_{min} 以上の時は当該パケットの

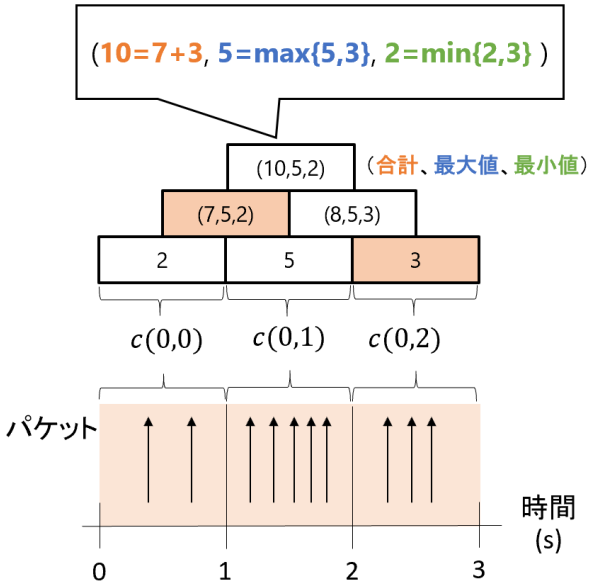

図 2　最小ウィンドウサイズ W_{min}=1.0 秒，特徴量に到着パケット数，集約関数に合計値，最大値，最小値を適用した例．頂点セルの $c(2,2)$ の生成に，$c(1,1)$=(7,5,2) と，$c(0,2)$=3 を利用している．

到着時）に生成され，パケット量やエントロピー値などのトラフィックの特徴量が直接格納される．なお，t はこの時にインクリメントされる．新しくセルデータが作られると，ピラミッド構造の右側に逐次追加されていく（図 1 の右端のセルを参照）．

一方，上位のレベル $l(0 < l < L)$ のセルデータは，特徴量を $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$，そして集約関数を $\oplus$ としたとき，(2) 式によって集約する．ここで集約関数 $\oplus$ とは，合計，最小値，最大値などがあてはまる．

$$\boldsymbol{x}_{c(l,t)} = \boldsymbol{x}_{c(0,t)} \oplus \boldsymbol{x}_{c(l-1,t-1)} \tag{2}$$

この時，生成に利用されるセルである $c(l-1,t-1)$ と $c(0,t)$ は，ちょうど自分たちの保有するデータの期間が重ならないようになっている．例えば，図 1 で $c(4,4)$ を生成することを考えると，$c(3,3)$ と $c(0,4)$ が集約に利用されるが，どちらも同じ期間は含まれていない．

ここで，集約処理について詳細に述べる．図 2 に，最小ウィンドウサイズ $W_{min} = 1.0$，$L = 3$ の Aggregation Pyramid を示す．保持する特徴量を最小ウィンドウサイズ W_{min} 以内に到着したパケット数とし，集約関数に合計，最大値，最小値を適用して，それぞれのデータをセルで保持している．いま，図 2 の頂点セルにあたる，$c(2,2)$ のデータを生成することを考える．$c(l-1,t-1)$ と，$c(0,t)$ にあたるセルが，図 2 でオレンジ色に塗られた $c(1,1)$ と $c(0,2)$ である．$c(1,1)$ が持つデータが，合計，最大値，最小値の順に（7,5,2），$c(0,2)$ の持つデータが 3 なので，$c(2,2)$ の保持するデータは $(7+3, \max\{5,3\}, \min\{2,3\}) = (10,5,2)$ となる．これらの値は，$c(0,0)$ から $c(0,2)$ の間のパケット

© 2019 Information Processing Society of Japan

数の合計値，最大値，最小値を意味する．このようにして，レベル l セルは，l 個分の期間の特徴量を保持している．

適用する集約関数の種類を考えれば，少なくとも $l \times W_{min}$ の期間について，合計値，最大値，最小値等を効率よく監視できる．利用するデータは 2 つだけであるため，1 つのセルデータの更新にかかる計算量は $O(1)$ となる．この処理はピラミッドの階層数 L だけ行われるため，最終的な計算量は $O(L)$ となる．

また，最低限の時系列データを構築する上で必要なセルデータは L だけである．Aggregation Pyramid はピラミッド構造となっているため本来であれば $L(L+1)/2$ のセルデータが必要となるが，例えば，$t+1$ のセルデータを生成するのに必要なデータは，図 3 の青く塗った箇所のみである．したがって，これらのセルデータさえ保持するようにすれば，新しいセルデータを生成し続けることができる．本手法ではメモリ効率を考慮してピラミッドデータをすべて保持せず，L 分だけ保持するようにする．

ここまで単一の時系列の Aggregation Pyramid を説明してきたが，トラフィックの異常を見つける際には，単一の時系列での監視は少量のパケットしか発生しない異常の検知が困難となる [11]．そこで，トラフィックの送信元 IP アドレスにランダムハッシュをかけ，トラフィックをサブトラフィックに分割するようにする．具体的に，key を送信元 IP アドレスを符号なし整数に変換した値，a と b をそれぞれ $(0 < a < key)$，$(0 \leq b < key)$ の範囲から決まる乱数，H をハッシュの種類の最大値とすると，(3) 式 [20][22] でハッシュ ID を生成し，そのハッシュ ID に対応する Aggregation Pyramid を更新するようにする（図 4）．なお，実験では性能を一定に保つため，$a = 1$，$b = 0$ に設定する．

$$h = (a \times key + b) \bmod H \tag{3}$$

保持するデータ量としては $O(H \times L)$ と増えるが，更新処理に関してはサブトラフィック空間で新たにパケットが到着したときのみである．したがって，更新処理の計算量は変わらず $O(L)$ であり，効率よくデータ構造を更新することができる．

まとめると，データ更新にかかる時間計算量およびデータ保持にかかる空間計算量は，それぞれ $O(L)$ と $O(H \times L)$ となり，既存手法 [19][20][21] と比較しても複数ウィンドウサイズの監視による処理効率への悪影響は少ないと言える．この考察は 5.2.1 節で詳細に述べる．

ここで，セルデータの更新処理のアルゴリズムについてまとめる．あるハッシュ IDh に対応するパケットの到着時刻の系列が，到着時刻を昇順にして $\boldsymbol{a_h} = \{a_{h1}, a_{h2}, \ldots, a_{hi}, \ldots a_{hn}\}$ であるとき，データ構造の更新処理は Algorithm1 の手順で行われる．

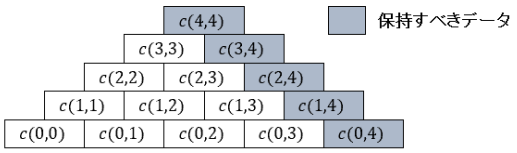

図 3　Aggregation Pyramid の形成に必要なデータ量

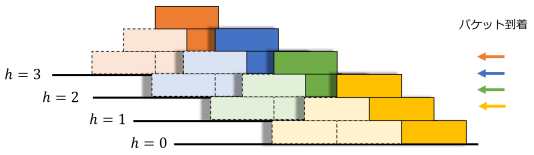

図 4　$L = 3$，$H = 4$ の時の Aggregation Pyramid の例．保持するデータは実線部分のみ．

Algorithm 1 Aggregation Pyramid の生成方法

```
t ← 0
while 次のパケットが存在する do
  MAXLV ← max(t, L − 1)
  h ← 送信元 IP アドレス mod H
  if a_hn − a_h0 < W_min then
    特徴量 x_tmp の計算のためのトラフィックデータ収集
  else
    for l = 0 to MAXLV do
      if l = 0 then
        a_h0 から a_hn−1 までのデータを用いて x_tmp を計算
        x_c(0,t) ← x_tmp
      else
        x_c(l,t) ← x_c(0,t) ⊕ x_c(l−1,t−1)
      end if
      攻撃検知処理 (Algorithm2 参照)
    end for
    a_h0 ← a_hn
  end if
  t++
end while
```

3.2　DDoS 攻撃検知処理

攻撃検知では，事前実験で DDoS 攻撃検知に有用な特徴を見せた，「パケットレート」，「フロー ID によるエントロピー値」，「フローサイズ分布の最大値」を特徴量として利用することにする．

フロー ID とは，ここでは IP ヘッダの「送信元 IP アドレス」，「宛先 IP アドレス」，「送信元ポート番号」，「宛先ポート番号」，「プロトコル」フィールドの 5-tuple で区別される ID である．DDoS 攻撃検知に一般的に利用される送信元 IP アドレス・宛先 IP アドレスではなく，フロー ID のエントロピー値を採用した理由は，パケットの送信元 IP アドレスあるいは宛先 IP アドレスが同じであった場合にも，ポート番号の分散具合を監視したいためである．DDoS 攻撃では，パケットの送信元ポート番号や宛先ポート番号をランダム化することが確認されている [24]．

フローサイズとは，あるウィンドウ内に出現したフロー ID のカウント数であり，フローサイズ分布はそのカウン

© 2019 Information Processing Society of Japan

ト数をビンにしたヒストグラムを表す．DDoS 攻撃ツールを用いる場合には送信元 IP アドレスや送信元ポート番号がランダム化されて送られることがあるため，通常時に比べて新規に出現するフローが多くなり，フローサイズが 1 のパケットが大量に出現すると考え，フローサイズのカウント数の値を DDoS 攻撃検知に利用することにした．最大値を利用した目的は，通常時と DDoS 攻撃時の特徴の差を際立たせるためである．通常時には同様のフロー ID が出現しやすいことから値が小さくなり，DDoS 攻撃時には新規に出現するフローサイズが多くなることから値が極端に大きくなるようにしている．

「パケットレート」「フロー ID によるエントロピー値」「フローサイズ分布の最大値」の計算方法について，順を追って説明する．

まずパケットレートを説明する．ある期間に生成されたセル $c(l,t)$ が持つパケット数を $n_{c(l,t)}$，期間の長さを $g_{c(l,t)}$ とした時に，パケットレート $r_{c(l,t)}$ は (4) 式で計算される．

$$r_{c(l,t)} = \frac{n_{c(l,t)}}{g_{c(l,t)}} \tag{4}$$

ここで，$n_{c(l,t)}$ はパケット数に対して，$g_{c(l,t)}$ は合計到着間隔に対して，集約関数として合計を適用したとき，それぞれ (5) 式と (6) で計算される．

$$n_{c(l,t)} = n_{c(0,t)} + n_{c(l-1,t-1)} \tag{5}$$

$$g_{c(l,t)} = g_{c(0,t)} + g_{c(l-1,t-1)} \tag{6}$$

次に，フロー ID によるエントロピー値 $e_{c(l,t)}$ は，i をサンプルデータ中で観測されたフロー ID，$p(i)$ を計算対象となるサンプルデータ内でのフロー IDi の出現率とすると，集約関数に合計を適用して (7) 式で計算される．ただし，文献 [15] と同様に，$0 \log 0 = 0$ として扱う．

$$e_{c(l,t)} = \begin{cases} -\sum_{i=1}^{n} p_i \log p_i & (l=0) \\ e_{c(0,t)} + e_{c(l-1,t-1)} & (otherwise) \end{cases} \tag{7}$$

最後に，フローサイズ分布の最大値 $f_{c(l,t)}$ は，$\boldsymbol{X}$ を計算対象となるサンプルデータ内における，フローサイズ分布とすると，集約関数に最大値を適用して以下の式で計算される．

$$f_{c(l,t)} = \begin{cases} \max \boldsymbol{X} & (l=0) \\ \max\{f_{c(0,t)}, f_{c(l-1,t-1)}\} & (otherwise) \end{cases} \tag{8}$$

提案手法では，マハラノビス距離によって，$r_{c(l,t)}$，$e_{c(l,t)}$，$f_{c(l,t)}$ の 3 つの特徴量を統合して異常度を評価する．マハラノビス距離とは，(9) 式で与えられる距離であり，事前学習したモデルとの距離を測っている．この値が大きくなればなるほど，通常時に比べて異常度が大きいと言える．

$$M = (\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \tag{9}$$

ここで Σ^{-1} は分散共分散行列の逆行列で，精度行列とも呼ばれる．精度行列を掛けあわせることによって，各データの分散の違いを考慮して正規化していることになる．これによって，各特徴量について，異常検知の寄与率が平等になるように統合している．

マハラノビス距離を M，異常度の閾値を d とすると，(10) 式を満たすときに攻撃トラフィックが到達していると判定する．

$$M > d \tag{10}$$

特徴量が正規分布に従うならば，ホテリングの T^2 法 [23] により，マハラノビス距離の閾値 d を χ^2 分布から決定することができるが，トラフィックデータは分布を推定するのが困難であることが知られており [13]，現在は経験的に定めることとしている．

ここまで説明してきた攻撃検知処理の手順を Algorithm2 に示す．攻撃検知処理はセルの生成時に行う．ただし，パラメータとしてステップ幅 S を導入し，レベル l が 0 あるいは S の倍数のセルでのみ処理を行うこととする．したがって，検知処理回数は $L/S + 1$ となり，ステップ幅 S によって処理効率と検知精度のバランスを調整できる．ステップ幅は乗数としても指定することができ，ある整数 a を乗数モードとして指定したときの検知処理の回数は $\log_a L$ 回となる．

最終的に，閾値を超えた数 cnt が，検知処理回数の過半数である $L/2S$ 以上の時に攻撃の発生を判定し，その時に観測されたクライアントを，被疑クライアントとして判定し，そのフロー ID をマークする．

3.3 トラフィックデータ学習方法

異常であるということを検知するためには，通常時の特徴量の平均・分散を学習しておかなければならない．システムを稼働させるうえで必要となる事前学習では，通常時に観測されたデータを利用して学習する．しかし，事前学習のみを利用する場合，現在のトラフィック状況への追従性が低下するおそれがある．そこで，事前学習だけでなく，逐次学習も行う．逐次学習には，攻撃と判定されなかったセルデータのみを利用する．各ウィンドウサイズでそれぞれ学習用データを収集し，データが十分な数蓄積された後に，特徴量の平均・分散を再計算する．

4. 評価実験

本章では，提案手法の性能評価を行う．単一のウィンド

© 2019 Information Processing Society of Japan

Algorithm 2 攻撃検知方法

```
t ← 0
while 次のパケットが存在する do
  cnt ← 0
  MAXLV ← max(t, L − 1)
  for l = 0 to MAXLV do
    if l = 0 ∨ l mod step = 0 then
      M ← (x_{c(l,t)} − μ)^T Σ^{-1} (x_{c(l,t)} − μ)
      if M > d then
        cnt++
      end if
    end if
  end for
  if cnt ≥ L/2S then
    W_min 以内に到着したフロー ID を被疑クライアントとして判定
  end if
  t++
  cnt ← 0
end while
```

表 1　実験環境

OS	Ubuntu 16.04.1
メモリ	16GB
クロック速度	3.6GHz
開発言語	C++
ライブラリ	libpcap

ウサイズの監視に比べて複数で監視した時により検知精度が向上するか，またこの工夫によって，処理性能に対して悪影響が発生しないかを確認することが目的である．そこで，提案手法を性能評価用のデータセットに適用し，検知精度と処理性能を調査する．実験環境は表 1 に示す．本章では，まず利用するデータセットを述べたのち，検知精度と処理性能の調査方法について詳しく説明する．

4.1　利用したデータセット

4.1.1　DARPA2000

DARPA2000 は MIT Lincoln Laboratory が人為的に作成した DDoS 攻撃のデータセットである [25]．このデータセットは 2000 年のものであるため古いものとなっているが，DDoS 攻撃検知の研究では評価に広く利用されており，既存手法との比較を行えるという点で有用である．DARPA2000 は組織に侵入しホストをボット化した後に外部組織に DDoS 攻撃を仕掛けるまでのキャプチャデータが保存されており，DDoS 攻撃の他にも，IP スキャン，侵入などの攻撃パケットが観測されている．具体的に，DARPA2000 では以下の 5 段階のシナリオを想定している．

Phase1　リモートホストからターゲット組織に対して IP スキャンを行い，稼働しているホストを調査

Phase2　Phase1 で調査したアクティブなホストに対し sadmind の有無を確認

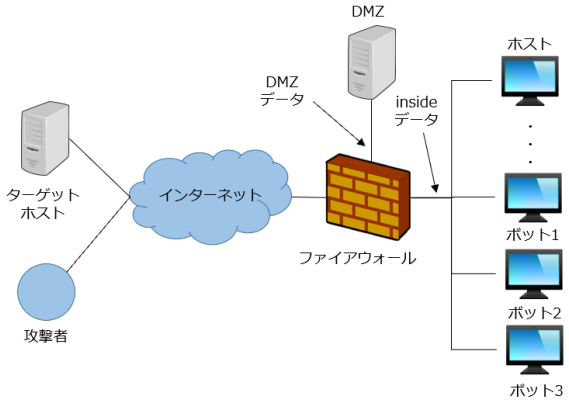

図 5　DARPA2000 のネットワーク図（文献 [14] を参考に作成）

Phase3　sadmind の脆弱性を利用し，システムに侵入

Phase4　3 台のホストに DoS 攻撃ツールの mstream をインストールしボット化

Phase5　ボットに外部組織に対する DDoS 攻撃の開始を指示

DARPA2000 ではターゲット組織のファイアウォールの内部で観測された inside データと外部で観測された DMZ データが提供されている (図 5)．本研究では総パケット量の多い inside データ（総パケット数 649,787）を利用し，Phase5 の DDoS 攻撃部分のみを攻撃とみなして実験を行う．

4.1.2　CICIDS2017

DARPA2000 には二つの問題点がある．高レートな DDoS 攻撃であるため攻撃の判別が行いやすいことと，データセットが古いことである．そこで，高レートではない DDoS 攻撃が含まれ，かつ比較的新しいデータセットでの評価を行うため，もうひとつ CICIDS2017 を利用した．

CICIDS2017 は CIC(Canadian Institute for Cybersecurity) による IDS の性能評価のためのデータセットである [26][27]．取得期間は 2017 年 7 月 3 日（月）午前 9 時から 7 月 7 日（金）の午後 5 時までのデータとなっており，月曜日から金曜日まで，曜日ごとに分割されて提供されている．月曜日には攻撃が含まれないが，火曜日から金曜日までのデータには人為的に作成された攻撃トラフィックが記録されている．

本研究では DDoS 攻撃の含まれる金曜日のデータを利用して，攻撃検知性能を調査する．DDoS 攻撃では外部の 3 台の Windows8.1 マシンから DDoS 攻撃のツール Low Orbit Ion Canon により攻撃が行われている（図 6）．この攻撃のパケット量は平均 1600pps 程度であり，高レート攻撃 [28] にはあたらない．金曜日のデータには，他にも ARES[29] というツールを用いたボットネット通信と nmap によるポートスキャンが観測されている．なお，本システムはインバウンドパケットを監視対象にすることを想定しているため，実環境での性能を確かめられるようインバウンドパ

© 2019 Information Processing Society of Japan

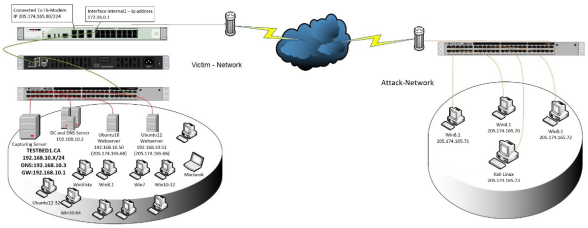

Figure 1: Testbed Architecture.

図 6 CICIDS2017 のネットワーク図 文献 [26] より引用

ケットのみをデータセットから取り出して実験を行った．

本データセットの特徴的な点として，攻撃ホストの送信元 IP アドレスがすべて 172.16.0.1 に NAT 変換されていることが挙げられる．そのため，本手法のようにヘッダデータのみを対象に攻撃を解析する手法では，DDoS 攻撃としての特徴が出ず，DoS 攻撃の様態となっており，DDoS 攻撃検知精度を評価するにはやや不向きなものとなっている．しかしながら，本研究では監視対象を送信元 IP アドレス単位ではなくフロー ID 単位で見ているため，IP アドレスが同じでも影響は少ないと考え，利用している．

4.2 検知精度

本研究の検知精度は前述の DARPA2000 と CICIDS2017 のデータセットを用いて評価する．ここでは検知精度の指標を説明した後，実験の手順を説明する．

4.2.1 指標

攻撃を正しく攻撃として検知できた数を TP(True Positive)，検知されたパケットのうち実際に攻撃でなかった数を FP(False Positive)，通常パケットを正しく通常パケットとして検知できた数を TN(True Negative)，攻撃パケットのうち正しく検知できなかった数を FN(False Negative) としたとき，次式で示す適合率 Pr と再現率 Rc，F 値 F，検知率 Dr を検知精度の指標として利用する．

$$Pr = \frac{TP}{TP + FP} \tag{11}$$

$$Rc = \frac{TP}{TP + FN} \tag{12}$$

$$F = \frac{2 \times Pr \times Rc}{Pr + Rc} \tag{13}$$

$$Dr = \frac{TP + TN}{TP + TN + FP + FN} \tag{14}$$

Pr が高ければ高いほど通常パケットを攻撃だと誤判定することが少なく，Rc が高ければ高いほど攻撃を見逃すことが少ない攻撃検知手法となる．実験結果の考察にはこの二つを用いる．F は Pr と Rc の調和平均で算出されるもので，両者が同等程度に高い場合に高くなる値である．Dr は，攻撃を攻撃として，正規パケットを正規パケットとして正しく判定できる割合である．F と Dr は，主に既存手法との性能比較のために用いる．

4.2.2 実験手順

実験では，DARPA2000 と CICIDS2017 に対して提案手法を適用し，適合率 Pr と再現率 Rc，F 値 F，検知率 Dr を算出する．パラメータ設定では W_{min} を 1.0 秒に，ハッシュサイズ $H = 10$ に固定した．ここで，複数のウィンドウサイズによる監視の効果を測定するため，レベル 0 のみを利用した場合 ($L = 1$) と，レベル 30 および 60 まで利用した場合 ($L = 30, L = 60$) で比較を行うことにした．なお，マハラノビス距離計算に用いる初期値の平均・分散は，それぞれのデータセットの通常時のデータを用いて計算した．DARPA2000 の場合は同じ組織で観測された DARPA1999[25] というデータセットの，DARPA2000 と同じ曜日で攻撃の含まれないデータを利用した．そして，CICIDS2017 は攻撃の含まれない月曜日のデータを利用した．なお，学習用データセットは評価用のものと同じくインバウンドパケットのみを取り出して利用した．逐次学習は 10,000 パケット溜まった時に行うようにした．

4.3 処理性能

処理性能を評価する目的は，複数ウィンドウサイズの並列監視による，処理性能への悪影響が小さいことを確認するためである．実験環境やプログラムの実装方法によって処理性能は変わるため，公平な比較は行いづらい．そこで計算量に着目して評価する．

また，最後に，DDoS 攻撃のパケット量の規模に比べて，実験環境における提案手法のパケット処理数がどの程度追従できるかを評価するため，DARPA2000 と CICIDS2017 に対して提案手法を適用し平均スループット [pps] を算出した．ここで，平均スループットは 10 回実験を行った時の平均を算出したものである．

5. 実験結果

5.1 検知精度

5.1.1 DARPA2000 における検知精度

表 2 に検知精度の結果を示す．$L = 1$ の場合に比べて，$L = 30$,$L = 60$ の場合の方が全体的に検知精度が向上している．ウィンドウサイズを大きくし，異常と判定した回数が，検知処理回数の過半数以上の時に攻撃として判定しているため，レベル 0 セルのみではノイズデータも攻撃として検知するところを，複数のウィンドウサイズ監視によりその影響を軽減できていると考えられる．

$L = 30$，$L = 60$ の場合，どちらも Pr が 0.93 以上，Rc が 0.95 以上となっている．$L = 30$ での $Dr = 0.995$ という結果は，DARPA2000 を精度評価に利用した文献 [31] のものと同程度の性能であり，また F の結果である 0.980 も，文献 [14] での最大 0.986 と同程度の性能であることから，十分な検知精度といえる．

© 2019 Information Processing Society of Japan

表 2　$W_{min} = 1.0$ での DARPA2000 における検知精度

L	S	H	d	Pr	Rc	F	Dr
1	1	10	100	0.904	0.950	0.950	0.988
30	10	10	100	0.961	0.980	0.980	0.995
60	10	10	100	0.933	0.950	0.941	0.987

表 3　$W_{min} = 1.0$ での CICIDS2017 における検知精度

L	S	H	d	Pr	Rc	F	Dr
1	1	10	30	0.639	0.999	0.779	0.929
30	10	10	30	0.674	0.987	0.800	0.945
60	10	10	30	0.704	0.997	0.825	0.945

$L = 30$ の場合と $L = 60$ の場合とでは，$L = 30$ の方が検知精度が高くなっているが，これは，DARPA2000 の DDoS 攻撃部が 5 秒程と短いことと，パケット量が攻撃開始直後に大きく増加し，攻撃終了直後に大きく減少することが原因だと考えられる．この状況では，結果的にウィンドウサイズを小さくした方がすばやく攻撃の開始や終了を捉えることができ，FN と FP を少なくできたのだと考えられる．DARPA2000 のようなパルス型の攻撃に対しては，「異常判定回数が過半数以上」という単純な判定方法ではなく，階層ごとに判定の価値を重みづけするなどして，攻撃の開始と終了の有効な判定方法を検討する必要がある．

5.1.2　CICIDS2017 における検知精度

表 3 に検知精度の結果を示す．結果としては，Rc が最大 0.999 であるが，Pr が最大で 0.704 となっており，DARPA2000 での結果に比べて低くなってしまっていることが分かる．この結果は，DDoS 攻撃の検知自体は行えているが，それ以外の通信を DDoS 攻撃だと誤判定する割合が多かったことを意味する．既存手法として，検知器に ID3 を利用した場合 [26] に Pr，Rc，F がすべて 0.98 を超えていることから，精度としては十分なものではなく，高レートでない DDoS 攻撃に対しては検知性能が課題となる．

DDoS 攻撃だと誤検知してしまった通信は，主にポートスキャンと Windows Update であった．

DDoS 攻撃とポートスキャンではむしろポートスキャンの方が異常度が高いという結果になっていた．この原因としては，DDoS 攻撃よりもポートスキャンの通信量の方が大きかったことが考えられる．ポートスキャンではポート番号を逐次変えていくことによって新規フロー ID が増加するため，「フロー ID によるエントロピー値」と「フローサイズ分布の最大値」が DDoS 攻撃と同様に変化度が大きくなることは想定しており，ポートスキャンとの区別は「パケットレート」の違いによって可能であると考えていた．しかしながら，ポートスキャンと同じ規模の DDoS 攻撃の場合にはその違いが見えない．そのため，今回の実験のように，ポートスキャンと同じ規模の DDoS 攻撃に合わせて閾値を調整すると，ポートスキャンも DDoS 攻撃として誤検知してしまうと考えられる．

通信量が大規模な Windows Update を DDoS 攻撃として判定してしまっていた点に関して，各特徴量の変化度と最終的な異常度を観察すると「フロー ID によるエントロピー値」と「フローサイズ分布の最大値」の変化度は少なかったものの，「パケットレート」の変化度が大きかったため，最終的な異常度が大きくなっていたことが分かった．マハラノビス距離を用いているため，特徴量のうち 1 つでも変化度が大きくなってしまうと，他の特徴量の変化度が少なくても最終的な異常度が大きくなってしまう場合があり，今回の実験結果もそのケースであったと考えられる．今回の実験では Windows Update であったが，フラッシュクラウド [32] に代表されるような，パケットレートの大きい他の通常通信も DDoS 攻撃として誤判定する可能性がある．

改善案として，ポートスキャンと DDoS 攻撃の区別に関しては「パケットレート」以外に，低レート攻撃に対して反応が大きくなる特徴量を追加する必要があると考えられる．また通常通信との違いを区別するためには，3 つの特徴量の全てにおいて変化度が高い時に DDoS 攻撃だと判定するような仕組みを導入する必要がある．

5.2　処理性能

5.2.1　計算量

3.1 節で述べた通り，データ更新に関わる時間計算量は $O(L)$ であり，空間計算量は $O(H \times L)$ である．これらの計算量は，Sketch と呼ばれる，ネットワークトラフィックの効率的な縮約方法と同等程度であり [19][20][21]，この結果から，複数ウィンドウサイズ監視による処理性能への悪影響は小さいものであると言える．

計算量が同等程度である Sketch に比べて Aggregation Pyramid が優れる点は，精度良く特徴量を計算できる点である．Sketch はランダムハッシュによって振り分けられる符号なし整数型のカウンタ情報から特徴量を推定するためにエラー率が存在する．一方，Aggregation Pyramid は，ウィンドウサイズを W_{min} に指定した時に計算される特徴量については，精度を落とすことなく計算することができる．

メモリ量に関しては，Sketch は符号なし整数型によるカウンタしか持たないためメモリ量を推定できる一方，Aggregation Pyramid は他の情報も保持するため，そのメモリ量は特徴量の選択次第となる．なお，今回の実験では，$L = 60$，$H = 10$，保持するデータバイト数＝「パケット数:int」＋「合計到着間隔:int」＋「フロー ID エントロピー値:double」＋「フローサイズ分布の最大値:int」の計 20B となっていることから，メモリ消費量は $60 \times 10 \times 20 = 12000 = 12$kB となっている．文献 [19] が 928kB，文献 [21] が 200kB から

© 2019 Information Processing Society of Japan

表 4　DARPA2000 におけるスループット [pps]

ウィンドウサイズ	平均スループット [pps]	標準偏差
1	161676	0
30	107784	0
60	79939.7	2694.6

表 5　CICIDS2017 におけるスループット [pps]

ウィンドウサイズ	平均スループット [pps]	標準偏差
1	159664.9	4571.9
30	126820.4	3262.5
60	102586.6	2333.70

1MB 以下となっており，現段階においてはメモリ使用量は優れているといえる．

5.2.2　スループット

DARPA2000 におけるスループットの結果を表 4 に，CICIDS2017 におけるスループットの結果を表 5 に示す．

実験環境においては，複数ウィンドウサイズの監視にあたり，最大 1 秒あたり 12 万パケットを処理することができる．文献 [33] によれば，2017 年の第一四半期に観測された DDoS 攻撃では，ピーク時に 90Mbps の規模であったため，規模の大きい DDoS 攻撃を検知するのに対して，十分な処理性能であるとはいえない．しかしながら，この結果は，あくまで別環境であるため純粋な比較は行えないが，文献 [19] の最大約 15 万パケットという結果には劣っているものの，文献 [20] の最大約 9 万パケットという結果とは同程度の性能となっている．

Aggregation Pyramid の階層数を増やすに従って，平均スループットが低下している．これは頂点 L を増やすと検知処理を行う回数が増えるためであり，この量は，ステップ幅 S などを用いて調整が必要であると考える．

6.　まとめ

DDoS 攻撃は，IoT ボットネットによる攻撃やリフレクション攻撃の流行により，攻撃の規模が 2012 年を境に年々増加しており，高精度でリアルタイム性の高い検知手法が望まれる．攻撃被害の緩和のために高精度な DDoS 攻撃検知手法が必要となるが，攻撃のバースト形態に合わせてウィンドウサイズを決定しないと攻撃検知が困難になるおそれがある．ウィンドウサイズを小さくした際にはパルス型の DDoS 攻撃を検知できる可能性があるがノイズの影響があり，逆に大きくした場合はノイズの影響が小さくなる代わりに攻撃の影響が平滑化され，リアルタイム性が減少してしまう．このように，ウィンドウサイズの適切な選択が必要になる．

本研究ではインターネットトラフィックを複数のウィンドウサイズで並列的に監視することによって，攻撃のバースト形態に合わせたウィンドウサイズ調整が必要のない統括的な監視を行い，これによってバースト形態に依らない DDoS 攻撃を検出する手法を検討した．これを実現するデータ構造として Aggregation Pyramid を利用し，効率的な集約処理を行って複数のウィンドウサイズでの同時監視を可能にした．異常検知では，各ウィンドウサイズでマハラノビス距離による攻撃検知を行い，異常判定回数が過半数を超えた時に攻撃の発生を判定するようにした．

DARPA2000，CICIDSI2017 に対して，レベル 0 のみを利用した場合と複数のウィンドウサイズを監視した場合で検知精度を評価した．複数ウィンドウサイズでの監視の方が全体的に検知精度が向上しており，レベル 0 セルのみではノイズデータも攻撃として検知するところを，複数のウィンドウサイズ監視により，その影響を軽減できたと考えられる．また，既存手法と比較した検知精度は同等程度となっており，十分な検知精度を持っていることが分かる．CICIDS2017 では再現率 Pr が最大で 0.704 となっており，高レートでない攻撃に対しても検知はできているものの，DDoS 攻撃でない通信を DDoS 攻撃だと判定する誤検知が多いという結果になった．この誤検知は，CICIDS2017 での DDoS 攻撃は高レート攻撃にはあたらない攻撃であるためにポートスキャンと区別できないことや，ひとつの特徴量のみの変化度が大きい時にも異常度が高くなることが原因であると考えられる．

処理性能を計算量とスループットで評価した．計算量は，大規模データトラフィックの処理に対して有効な Sketch というデータ構造と同等程度であり，複数のウィンドウサイズ監視による処理性能への悪影響は少ないと言える．既存の Sketch 手法と提案手法の違いを考えると，Sketch 構造のメモリ量は 100kB 以上であるのに対し，提案手法は 12kB のみであり，この点で優れている．スループットは平均 1 秒辺り 10 万パケットであり，純粋な比較はできないものの，既存手法と同等程度であった．

今後の課題として，低レート DDoS 攻撃に対しても有効な特徴量や検知器を利用して，検知精度を向上させる必要がある．また，今回の実験ではパケットキャプチャファイルをロードして攻撃検知を行っているが，現実の運用としてはオンラインキャプチャによって攻撃を検知していくため，オンライン実装時の性能を調査する必要がある．

謝辞

本研究は JSPS 科研費 JP17H01736，JP17K00139，JP18K11268 の助成を受けたものです．

参考文献

[1] Arbor 12th Annual World Infrastructure Security Report, 2017.

[2] Kottler, S.: February 28th DDoS Incident Report. available from ⟨https://githubengineering.com/ddos-incident-report/⟩ (accesed 2019/01/15).

[3] Garber, L.: Denial-of-Service Attacks Rip the Internet,

© 2019 Information Processing Society of Japan

Computer, pp. 12–17 (2000).
[4] Vixie, P., Sneeringer, G., and Schleifer, M.: Events of 21Oct2002, available from ⟨http://c.root-servers.org/october21.txt⟩ (accessed 2017/02/07).
[5] 中田 敦: DNS サービスの「Dyn」に大規模 DDoS 攻撃、Twitter などが影響受けダウン (オンライン)，入手先 ⟨https://tech.nikkeibp.co.jp/it/atcl/news/16/102203079/⟩
[6] Zeifman, I.: Attackers Use DDoS Pulses to Pin Down Multiple Targets, available from ⟨https://www.imperva.com/blog/pulse-wave-ddos-pins-down-multiple-targets/⟩ (accessed 2019/05/07).
[7] DDoS-GUARD: Hidden threat of Pulse Wave DDoS attacks, available from ⟨https://ddos-guard.net/en/info/blog-detail/hidden-threat-of-pulse-wave-ddos-attacks⟩ (accessed 2019/05/07).
[8] Osanaiye, O., Choo, K-KR. and Dlodlo, M.:Distoributed denial of Service(DDoS) resilence in cloud: Review and concenptual cloud DDoS mitigation framework, Journal of Network and Computer Applications 67, pp.147–165 (2016).
[9] Snort Homepage, available from ⟨https://www.snort.org⟩ (accessed 2017/02/08).
[10] Mahjabin, T., Xiao, Y., Sun, G. and Jiang, W.: A survey of distributed denial-of-service attack, prevention, and mitigation techniques, International Journal of Distributed Sensor Networks, Vol. 13, No. 12, pp. 1550147717741463 (2017).
[11] 福田健介：インターネットバックボーントラフィックにおける異常検出，コンピュータソフトウェア，Vol.30, No.22, pp. 23–32 (2013).
[12] Modi,C., Patel, D., Borisaniya, B., Patel, H., and Patel, A., Rajarajan, M.: A Survey of intrusion detection techniques in Cloud, Journal of Network and Computer Applications, Vol. 36, pp.42–57 (2013).
[13] Paxson, V. and Floyd, S.: Wide area traffic: the failure of Poisson modeling, IEEE/ACM Transanction Network, Vol. 3, No. 3, pp. 226–244 (1995).
[14] 小島 俊輔, 中嶋 卓雄, 末吉 敏則: エントロピーベースのマハラノビス距離による高速な異常検知手法, 情報処理学会論文誌, Vol.52 No.2 pp.656–668 (2011).
[15] Nychis, G., Sekar, V., Andersen, D. G., Kim, H. and Zhang, H.: An empirical evaluation of entropy-based traffic anomaly detection, in *the Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pp. 151–156 (2008).
[16] Feinstein, L., Schnackenberg, D., Balupari, R. and Kindred, D.: Statistical Approaches to DDoS Attack Detection and Response, Proceeding of DARPA Information Survivability Conference and Exposition, Vol.1, pp.303–314 (2003).
[17] 原田 薫明, 川原 亮一, 森 達哉, 上山 憲昭, 廣川 裕, 山本 公洋: 異常トラヒック発生検出および終了判定手法，電子情報通信関係学会, 信学技報, IN2006-133, pp.115–120 (2006).
[18] Zhang, X., Shasha, D.: Better Burst Detection, ICDE'06 Proceeding of the 22nd International Conference on Data Engineering, pp.146–149 (2006).
[19] Jing, X., Yan, Z., Jiang, X., and Pedrycz, W.: Network traffic fusion and analysis against DDoS flooding attacks with a novel reversible sketch, Information Fusion, Vol. 51, pp. 100–113 (2019).
[20] Wang, C., Miu, T., Luo, X., and Wang, J.: SkyShield: A Sketch-Based Defense System Against Application Layer DDoS Attacks, IEEE Transactions on Information Forensics and Security, Vol. 13, No. 3, pp. 559–573 (2018).
[21] Yang, T., Jiang, J., Liu, P., Huang, Q., Gong, J., Zhou, Y., Miao, R., Li, X., and Uhlig, S.: Elastic sketch, in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication - SIGCOMM '18*, pp. 561–575 (2018).
[22] Chi, L., and Zhu, X.: Hashing Techniques, ACM Computing Surveys, Vol. 50 No. 1, pp. 1–36 (2017).
[23] 井出 剛，杉山 将，異常検知と変化検知，講談社，機械学習プロフェッショナルシリーズ (2015).
[24] jgamblin: Mirai BotNet, available from ⟨https://github.com/jgamblin/Mirai-Source-Code⟩ (accessd 2019/05/08).
[25] MIT：DARPA Intrusion Detection Evaluation Data Set, available from ⟨https://www.ll.mit.edu/ideval/data/index.html⟩ (accessed 2017/01/25).
[26] Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A.: Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization, 4th International Conference on Information Systems Security and Privacy (ICISSP), Purtogal, January (2018).
[27] Intrusion Detection Evaluation Dataset (CICIDS2017), Canadian Institute for Cybersecurity, available from ⟨http://www.unb.ca/cic/datasets/ids-2017.html⟩ (accessed 2018/10/22).
[28] Moore, D., Shannon, C., Brown, D., Voelker, G. and Savage S.: Inferring Internet denial-of-service activity, ACM Transactions on Computer Systems, Vol. 24, No. 2, pp. 115–139 (2006).
[29] sweetsoftware: Ares, available from ⟨https://github.com/sweetsoftware/Ares⟩ (accessed 2019/05/08).
[30] Özçelik, İ. and Brooks, R. R.: Deceiving entropy based DoS detection, Computers & Security, vol. 48, pp. 234–245 (2015).
[31] Lee, S., Kim, D., Lee, J. and Park, J.: Detection of DDoS attacks using optimized traffic matrix, Computers & Mathematics with Applications, Vol. 63, No. 2, pp.501–510 (2012).
[32] Yu, S., Zhou, W., Jia, W., Guo, S., Xiang, Y., and Tang, F.: Discriminating DDoS Attacks from Flash Crowds Using Flow Correlation Coefficient, IEEE Transactions on Parallel and Distributed Systems, Vol. 23, No. 6, pp. 1073–1080 (2012).
[33] VERSIGN: VERISIGN DISTRIBUTED DENIAL OF SERVICE TRENDS REPORT, Vo. 4, No. 1 – 1st Quarter, 2017 (2017).

© 2019 Information Processing Society of Japan