IPv6に対応したトラフィック異常検知

2008MI017 深谷 利奈　　2008MI061 飯尾 美紀
指導教員　後藤 邦夫

1　はじめに

現在，通信技術の進歩に伴いインターネットをはじめとした各種ネットワークは，私達の生活と切っては切れないほど密接に関係する．その結果，ネットワークは不正アクセスやウイルスの脅威に晒される機会が増加している [6]．これらの未知の脅威に対して侵入検知システム (以後，IDS：Intruction Detection System とする) は有用であり研究が進んでいる．従来のインターネットプロトコルにあたる IPv4 のアドレス資源の枯渇問題により IPv6 アドレスの普及が見込まれるなかで，IPv6 に対応した IDS はまだ少ない．また，IPv6 に移行することによって生まれると予測される未知の攻撃を見つける手段として，パケットを収集しデータを蓄積していく IDS は有効であると考えた．

そこで本研究は，「IPS の実現とネットワークエミュレータ上での評価 [4]」の IDS 部分を参考に，IPv6 に対応したトラフィック異常検知システムの作成をする．さらに，平日・休日ごと，また時間帯ごとのパケットパターンの区分けをもうけ，誤検知や見落としを減らすシステムを目指す．なお，先ほどの論文では，IPv4 に対応した IDS が作成され，Gatekeeper と組み合わせることにより，侵入防止システム (IPS：Intrusion Prevention System) が実現された．

なお，深谷は主にプログラム部分を，飯尾は主にシステム構成を担当した．

2　システムの概要

本研究でのシステム全体の流れについて説明する．

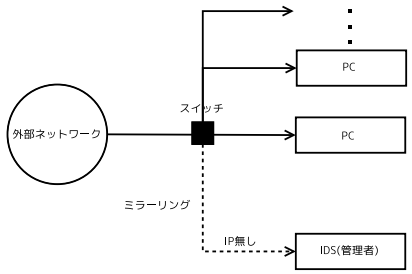

図 1　ネットワーク構成図

図 1 で示したように，外部ネットワークと PC の間にスイッチングハブを挟みミラーリングする．ミラーリングされたパケットを本研究で作成した IDS で解析し，異常なパケットであると判断した場合にアラートを出力する．IDS の侵入検出のアルゴリズムには不正検出と異常検出の二種類がある．不正検出とは，あらかじめ登録されている侵入手口のパターンを検出する方法である．あらかじめ登録されている侵入手口のパターンしか検出ができないので，未知の侵入手口を使った場合には検出ができない．一方，異常検出では急激なトラフィックの増加によって異常検出するので未知の攻撃が検出できる．本研究では通常と異なった振舞を検出する異常検出を採用する．

2.1　異常検出

機械学習型 IDS は，一般に学習フェーズ，検出フェーズの二つのフェーズで構成される．学習フェーズにより攻撃を含まないトラフィックデータから特徴を抽出し，それを 学習ことで通常状態のプロファイルを形成する．検出フェーズではネットワークを流れるパケットデータから特徴を抽出し，その状態から外れたトラフィックが発生した時に異常と判定する．

本研究では学習フェーズ，検出フェーズ共に多くのマイニングアルゴリズムを実装している WEKA[3] を用いる．

2.2　時間帯における区分け

時間帯によりパケット量が大きく変わることが予想される．よって，異常の誤検知や見落としを減らすよう時間における区分けする．

区分け方法はクラスタ分析の手法のひとつである K-means 法により分類する．K-means 法は，まずランダムにクラスタを割り振りデータの各要素の平均を用いクラスタの中心を求める．次に各データとクラスタの中心の最も近い中心のクラスタに割り当て直し，中心を計算しなおす．これをクラスタの割り当てが変化しなくなるまで繰り返す方法である．分類のために使用する属性を以下に示す．

時間における区分け　時間 (HH)，パケット数

時間における区分けは、それぞれ誤差があると考え属性を時間（HH）のみとする．

2.3　検知対象データ

異常を検知する検知対象データの違いで，次の 2 つの検知方法の区分けをもうける．

1. 全体検知
   パケットキャプチャしたパケットの数全体に対してデータマイニングをし，全体のパケット数の変動から異常検知をする．
2. 部分検知
   全体検知とは違い全体のパケット数をみるのではなく，IPv4・IPv6 のバージョン別，TCP・UDP・ICMP 等のプロトコル別のパケット数に対してデー

タマイニングをし，バージョン，プロトコルごとに異常検知する．

2.4 IPv6 と IPv4

IPv4 の在庫切れにともないより多くの IP アドレスを使用可能とするために，新たな通信プロトコルとして IPv6 が開発された．IPv6 と IPv4 では，まずアドレス空間がそれぞれ 32 ビットと 128 ビットという大きな違いがある．また, IP ヘッダ形式が完全に異なり, IP アドレス空間も完全に異なるため互換性の問題が存在する．代表的な解決方法に以下の 2 つが挙げられる．IPv6 のパケットを IPv4 のパケットでカプセル化し通信するトンネリング，単一機器に IPv4 と IPv6 という仕様の異なるプロトコルスタックを共存させるデュアルスタックである．上位プロトコルである UDP，TCP は IPv4 と IPv6 で違いはない．しかし，同じく上位プロトコルである ICMP は IPv4 と IPv6 ではヘッダの形式は同じであるが，割り当てられたタイプ，コードが異なる．IPv6 固有の問題として，アドレス空間が IPv4 より広いことで，IPv4 と比べるとパケット分類係数を出すのは容易ではない．また，ICMPv6 においてはマルチキャスト宛のパケットについてもエラーを返すことが許されているため, 外部からの不正なマルチキャストパケットによる, パケットの増幅攻撃に対しても対策が必要である [2].

本研究では, パケット構成の異なる IPv4 と IPv6 に対応したパケットキャプチャを作成する．また，パケット分類係数を取り扱わず，ミラーリングで得られるパケット量の急激な増加にのみ着目し異常検知する．トンネリングについては，スイッチングハブでミラーリングをしパケットの収集するので問題なく扱えると考えた．

3 システムの実現

本研究で用いる IDS を実現するプログラムを C++ と Java で書き，データベースには PostgreSQL を使用する．また，データマイニングのツールとして WEKA を使用し，傾向を付加しながら異常検知の正常範囲を設定する．まずは，表 1 に作成予定のクラスを説明し，次に表 2，表 3 を用いて作成するデータベースを説明する．

3.1 クラスの概要

クラスは表 1 で表した 4 つを作成する．

表 1　クラスの説明

クラス名	説明
Pktcap	パケットをキャプチャし必要な情報を抽出
toDB	パケットデータを pkt テーブルに挿入
CountPkt	pkt テーブルのデータを読み パケット数をカウント パケット数をテーブル weka へ挿入
toWEKA	WEKA へ処理実行を命令

Pktcap クラス，CoutPkt クラス，toWEKA クラス，この 3 つのクラスをスレッドとして並行処理をする．マルチスレッドのために，GNU Common C++ クラスライブラリを利用する．

3.1.1 Pktcap クラス

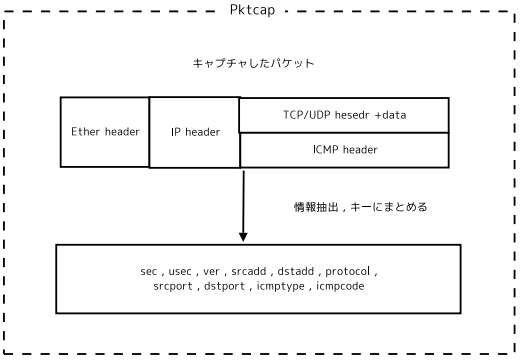

図 2　Pkcap クラスでの情報抽出

Pktcap クラス
Pktcap クラスは lipcap を用いてネットワーク上を流れるパケットを常にキャプチャし，図 2 のように必要なデータを抽出するクラスである．get メソッドで libpcap を使用しパケット採集をする．put_key メソッドでは get メソッドで得たデータを，キーにまとめる．

toDB クラス　データベースへの接続をする．接続ができたら in_pkt メソッドによってキーのデータを pkt テーブルへ格納する．データベースには PostgreSQL を使用する．PostgreSQL に接続するために，PostgreSQL のインターフェースである libpq を使用する．

CountPkt クラス
データベースへと接続をする．接続ができたら，countp メソッドで全体のパケット数，プロトコルバージョン別かつプロトコル番号別の個々のパケット数をカウントする．カウントして得られたデータは in_weka メソッドで weka テーブルにデータを挿入する．このクラスはスレッドとして常に動かしておくが，パケットデータがたまるのを待つため，weka テーブルへのデータ挿入が行われた後 5 秒間スリープする．

toWEKA クラス
pkt テーブルが更新されたか常にチェックする．テーブルが更新されたらそのデータを WEKA へ渡しデータ解析を要求する．また，WEKA で生成されるプロファイルを取得する．WEKA が Java プログラムであるので，WEKA を操作するこのクラスは Java で記述する．

3.2 データベースの説明

使用するデータベースについて説明する．データベース内には pkt テーブルと weka テーブルをあらかじめ用意する．

pkt テーブル

pkt テーブルを表 2 に表す．pkt テーブルでは Pktcap クラスで取得したパケットデータを管理する．パケット 1 つ 1 つを管理するために時間を主キーとした．1 秒間に複数のパケットを受信するので，時間データは秒単位ではなくマイクロ秒単位とした．なお，秒数単位での検索をしやすくするために秒 (sec) とマイクロ秒 (usec) を分けて複合キーとした．

表 2　pkt テーブル

属性	説明
sec(主キー)	パケットを受け取った時刻 (秒)
usec(主キー)	パケットを受け取った時刻 (マイクロ秒)
ver	プロトコルバージョン
proc	プロトコル番号
srcadd	送信元アドレス
srcport	送信元ポート
dstadd	宛先アドレス
dstport	宛先ポート
icmptype	ICMP タイプ
icmpcode	ICMP コード

weka テーブル

weka テーブルを表 3 に表す．WEKA に 5 秒毎のパケット量をもとにデータマイニングさせるため，5 秒間のパケット数の合計を，総量とバージョン，プロトコルごとに管理する．weka テーブルでは直接 WEKA へ読み込ませるテーブルとして作成している．また，pkt テーブルと違い 5 秒毎のパケット量を管理するため，主キーは wsec(秒) のみにした．

表 3　weka テーブル

属性	説明
wsec(主キー)	パケット採集時刻
total	パケット総数
icmp4	IPv4 の ICMP パケット数
tcp4	IPv4 の TCP パケット数
udp4	IPv4 の UDP パケット数
icmp6	IPv6 の ICMP パケット数
tcp6	IPv6 の TCP パケット数
udp6	IPv6 の UDP パケット数

3.3 WEKA

本研究では機械学習部分を WEKA で行う．WEKA は Java により実装され，様々なマイニングアルゴリズムが実行できるオープンソースである．WEKA を実行させる方法として，CL，GUI，Java ライブラリの 3 つがある．本研究では Java ライブラリを用い実装する．

主なマイニング手法には，分類学習，数値予測，クラスタリング，相関ルールが挙げられる．本研究では，分類学習の一つであり，データの制約が少ない木構造の C4.5 の WEKA バージョンである J48 を使用する．また，時間帯による区分けではクラスタリングのひとつである K-means 法を使用する．Java ライブラリで WEKA を実行する場合には，WEKA.jar にクラスパスを通す．また，データベースを読み込むために JDBC ドライバもクラスパスを通す．Java ライブラリによる実行方法の構成は以下の通りである．

インスタンス部分

マイニングするデータセットを指定する部分である．ここで学習するデータベースを指定しテーブルを読み込む．データセットに必要なクラスは weka.core である．

マイニング部分

マイニング方法指定部分である．オプション設定もこの部分で行う．今回使用する分類器とクラスタリングに必要なクラスは weka.classifiers, weka.clusterers, weka.core.Instances , experiment.InstanceQuer である．

出力部分

出力を設定する部分である．また，ここでプロファイルを保存する．異常検出をした場合には，異常が検出された時間とそのプロトコルバージョン別のプロトコル番号を Alert 情報としてファイルへ出力させる．

3.4 処理の流れ

システムの処理の流れを図 3 に示す．

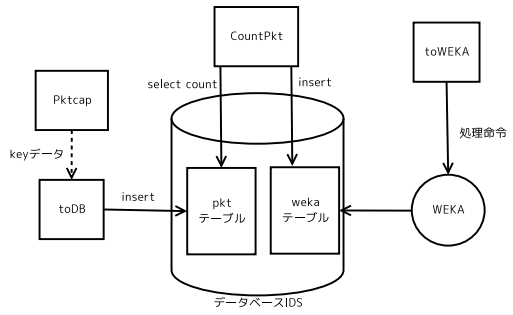

図 3　処理の流れ

Pktcap クラスではクラスメソッドの get でパケット

データの収集をし，必要な情報を抽出する．put_key メソッドでは抽出したデータをポインタにまとめ，キーとして返す．toDB クラスの in_pk メソッドでは，受け取ったキーのデータをメソッド in_pkt を用いて pkt テーブルへ挿入する．pkt テーブルに格納されたデータは CountPkt クラスで，5 秒ごとのパケット量を調べる．cpkt メソッドによってパケット量をカウントし，カウントしたデータを同じくクラスメソッド in_weka で WEKA が読み込む weka テーブルへと挿入される．toWEKA クラスで WEKA がへテーブル weka を読み込むよう指示を出す．WEKA では，時間によりクラスタリングをし，その後トラフィック異常検知をする．パケット量が予想範囲外である場合には，異常検出情報をファイルへ出力する．その後，データマイニングによってパケット量の増減傾向を予測し，パケット量の増減予測範囲を更新する．また，異常を検知した場合には，異常と判定された属性と，その属性の含まれる行の時刻データを出力する．

4 実験方法

本研究ではネットワークエミュレータとして Goto ’ s IP Network Emulator(以後，GINE とする)[5] を用いて実験する．GINE とは，NameSpace を作成することによって現実的なネットワーク構成を模倣することが出来るネットワークエミュレータである．GINE 上で NameSpace を 3 つ作成し，それぞれを外部ネットワーク，IDS，内部ネットワークと見立て環境を構築する．

4.1 事前学習データと評価データ

事前学習のためのデータとして，1999 DARPA Intrusion Detection Evaluation Data Set[1] を用いる．このデータセットは 1999 年 MIT の LINCOLN 研究所が作成した評価用データで，5 週間分のデータが公開されいる．1 週目と 3 週目のデータは攻撃を含んでいない．しかし，DARPA のデータセットは現在のトラフィックの性質とは大きく異なるため，評価データとしては適当とは言い難い．そこで，深谷，飯尾両者の自宅ネットワークでパケットの採集をし利用する．採集期間は 9 月から 11 月末日までである．自宅で採集したパケット Snort を用い攻撃を除き，学習データとして利用する．パケットまた評価データとして，自宅ネットワークで採集した 12 月分のパケットデータを利用する．

4.2 自宅でのパケットデータ収集方法

自宅で収集したパケットは，スイッチで 1 ポートに全ポートのパケットを出力するよう設定し，そのポートに PC を接続し tcpdump で収集した．tcpdump を端末で起動するときに，オプション [-r] をつけキャプチャしたパケットログをファイルへ記述する．収集したパケットの例を次に示す．

自宅収集パケット例

```
08:45:37.194114
IP web.setup.1900 >
239.255.255.250.1900: UDP，length 268
08:46:18.396991
IP6 fe80::23a:9dff:fed2:bdf8 >
ip6-allnodes: HBH ICMP6，multicast listener
queryv2 [gaddr ::]，length 28
```

5 おわりに

Pktcap クラスのパケットを取得するメソッド get と，マイクロ秒の時間を取得するメソッド get_time は完成した．また, toDB クラスの pkt テーブルへデータを挿入するメソッド in_pkt も完成した.CountPkt クラスにおけるパケットをカウントするそのデータを挿入する in_weka メソッドの SQL 文発行箇所実行箇所の記述ができている．WEKA のクラスタリングの実行もできた．

しかし，CountPkt クラス toWEKA クラスが未実装となっており本研究では実験評価をするまでに至らなかった．また，現在のプログラムでは，テーブルへ入れているデータを削除する機能がなく，データが膨大になってしまうので，削除機能が必要である．

参考文献

[1] MIT Lincoin Laboratiry: 1999 DARPA Intrusion Detection Evaluation Data Set (accessed Sep. 2011). http://www.ll.mit.edu/mission/communications/ist/index.html.

[2] 高橋睦美：ネットワーク構造の転換期？ IPv6 も「現実の問題」に (accessed Sep. 2011). http://www.atmarkit.co.jp/fnetwork/tokusyuu/59interop11/01.html.

[3] The University of Waikato: WEKA (accessed Sep. 2011). http://www.cs.waikato.ac.nz/ml/weka/.

[4] 伊藤遼平，嶋田伊吹：IPS の実現とネットワークエミュレータ上での評価，2010 年度卒業論文，南山大学情報通信学科 (2010).

[5] Sugiyama, Y. and Goto, K.: Design and Implementation of a Network Emulator using Virtual Network Stack, *Proc. of the Seventh International Symposium on Operations Research and Its Applications (ISORA2008), Lecture Notes in Operations Research,* Vol.8, pp.351–358 (2008).

[6] 警察庁：情報技術解析平成 22 年報 (accessed Sep. 2011). http://www.npa.go.jp/cyberpolice/detect/pdf/H22_nenpo.pdf/.