第 IV 部

フローベースのネットワーク
トラフィック計測

第 4 部
フローベースのネットワークトラフィック計測

第 1 章　roft ワーキンググループの概要と 2005 年度の活動

roft（Research Of Flow Trend）ワーキンググループは、フローベースのネットワークトラフィック計測手法について議論をするワーキンググループとして、2004 年 3 月に創設された。フローベースのネットワークトラフィック計測とは、Cisco Systems 社の NetFlow[37]、InMon 社 [122] の sFlow[251] といったトラフィック観測技術を用いたトラフィック計測を指す。これらの観測技術を用いることにより、低コストに詳細なトラフィック情報を取得可能であり、特定トラフィックの検出やトラフィックエンジニアリングへの応用が期待されている。roft ワーキンググループでは、具体的な活動目標として、次の 3 つを挙げている。

1. 全パケットダンプに代表される他の計測技術と比較・検討し、フローベース計測の利点と欠点を明らかにする。
2. 観測目的や、観測対象ネットワークに応じた適切サンプリングレートを調査する。
3. 収集したトラフィックデータに対する解析技術について検討し、トラフィック傾向解析、異常トラフィックの自動レポーティング、通信状況に適した高機能トラフィック制御の実現を目指す。

本報告書の第 2 章では、2005 年 WIDE 秋合宿での実験を報告する。本実験では、過去の合宿トラフィック計測実験の経験をベースとし、新たに sFlow を用いたトラフィック計測実験を行った。新規に開発した sFlow2MySQL の性能評価を実環境下で行うとともに、収集されたトラフィック情報の斬新な活用方法に関して議論した。第 3 章から第 5 章にかけては、roft ワーキンググループの活動の成果物である論文、研究会報告への参照を掲載する。

第 2 章　研究会ネットワーク上でのサンプリング計測の運用実験報告

2.1 概要

roft ワーキンググループでは、2004 年 3 月合宿、2004 年 9 月合宿において NetFlow による計測を行い、ネットワークオペレータ支援を行ってきた。今回は、新たに sFlow を用いて実運用環境下で計測実験を行った。NetFlow に比べ、L2 のヘッダ情報からペイロードの一部までを取得することが出来その汎用性が期待される。一方でフローをもとに情報を集約する NetFlow に比べるとセッション情報を追いかけることが難しくなっている。今回は、sFlow による計測を行い、その計測システムの問題点を明らかにするとともに、蓄積されるログデータの活用方法について議論を行った。

2.2 実験目的

本実験の目的は以下のとおりである。

- sFlow2MySQL の動作評価
- 蓄積されるトラフィックデータの活用事例の提示

本実験では、我々のグループが作成した sFlow データグラム収集ソフトウェア、sFlow2MySQL の動作確認および、実運用下での問題点の抽出などを行った。また、トラフィックデータの活用として、新たに取得できるようになった L2 ヘッダ、特に MAC アドレスを活用したデータの提示方法を行った。

2.3 実験内容

2.3.1 計測環境

本実験でのトラフィックデータ収集場所は、対外ルータとユーザセグメントとの境界のスイッチでデータグラムを取得し蓄積を行った。計測環境を図 2.1 に示す。

この場所で計測を行うことによりユーザ間通信以外のすべての通信を取得することができる。この場を利用し、sFlow2MySQL でトラフィックデータを

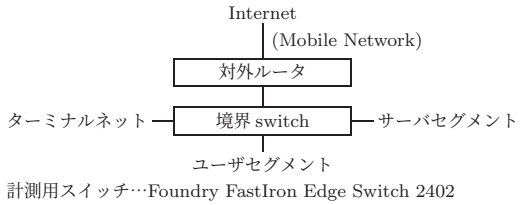


図 2.1.　計測環境

表 2.1.　実験に利用したマシンスペック

CPU:	Pentium4 2.53 GHz
MEM:	1024 MB
HDD:	80 GB 7200 rpm Parallel-IDE ATA/100
NIC:	10/100 対応
DBMS:	MySQL 4.0.13a

取得し、蓄積を行った。実験に利用したマシンスペックは表 2.1 のとおりである。

2.3.2 実験 1：sFlow2MySQL 性能評価

合宿期間中、sFlow データグラムをサンプリングレート 128 で収集を行った。性能評価を行うためにある 1 時間、サンプリングレート 64 で収集を行った。また、ある 5 分間サンプリングレートを 32 に変更して収集を行った。それぞれのサンプリングレートでの収集パケットドロップ率は表 2.2 のとおりである。

サンプリングレートを下げた期間中のトラフィックはおおよそ 2 Mbps であった。このことから、トラフィックが ADSL 程度しか出ていない環境でも本ソフトウェアを利用した場合には、適切にサンプリングレートを設定しない限りパケットドロップが発生してしまうことが分かった。原因としては、データベースの設計の問題が挙げられる。各レイヤのヘッダごとにテーブルを分割しておりデータベースに挿入する際の遅延が問題になっていると考えられる。

表 2.2.　sFlow2MySQL の性能評価

サンプリングレート	結果
128	取りこぼし無し
64	数%の割合でパケットをドロップ
32	10 数%の割合でパケットをドロップ

2.3.3 実験 2：トラフィックデータ提示

今回は単にトラフィック量を表示させるのみではなく、トラフィックデータから BoF の参加者、および BoF 視聴率を予測するという試みを行った。概要としては以下のとおりである。

参加者の定義：

各 BoF 部屋には無線 AP が設置されている。参加者は BoF 中、AP に接続させていると仮定し、AP に接続されている MAC アドレス数を BoF の参加者として定義した。なお、AP に接続されている MAC アドレス数の取得には、同合宿で実験を行った大江さんシステムのご協力をいただいた。

視聴率の定義：

BoF の視聴率は、各 BoF の開催時間中に、あらかじめこちらで決めた IP アドレスをもつサーバ（以下、ブラックリストサーバ）に対して通信を行った MAC アドレスを持つホストを「BoF を聞いていないホスト」として定義し、

$$\text{視聴率} = \text{BoF を聞いていないホスト数/AP 接続ホスト数} * 100$$

として算出した。算出は 10 分ごとに行い、その 10 分間に一度でもブラックリストサーバと通信を行ったホストはその 10 分間 BoF を聞いていなかったとした。本実験でのブラックリストサーバは以下のとおりである。

- `*.2ch.net` の各 IP アドレス
- `*.gree.jp` の各 IP アドレス
- `*.mixi.jp` の各 IP アドレス

上記を挙げた理由は、主に「文字を読む web ページ」を提供しているサーバであることから、アクセス中は記事を読むため BoF を聞いていないだろうと仮定した。

以下の環境で視聴率を割り出した結果、おおむね 90%〜の高視聴率を記録した。グラフとしてはほぼ横ばいのあまりおもしろくない結果となってしまった。

2.4 まとめ

本実験では我々のグループが作成したソフトウェアの評価と、トラフィックデータの活用方法に関して議論を行った。今回、活用事例として BoF の視聴率をトラフィックデータのみを用いて提示した。クロージングプレナリにおいて、部屋の広さやその他の要素を組み合わせて BoF の視聴率を出せば、BoF を取る側からしても部屋を押さえるときの目安にでき、うれしいのではないかというご意見をいただいた。また、データの可視化に関しては、グラフのみ

という結果になってしまったが、netviz ワーキンググループなどを参考にさせていただき、よりおもしろく、興味を持ってもらえるようなデータの提示方法を模索していかなければならないと感じた。

第 3 章　通信先ホスト数の変化に注目した異常トラフィック自動検出手法の提案と評価

本報告書は電子情報通信学会論文誌 Vol.J88-B No.10、pp.1922–1933、2005 年 10 月に掲載された論文をもとにしたものである。

3.1 はじめに

多くの企業や学校で、コンピュータネットワークは日々の活動に必要不可欠なものとなってきている。ネットワークへの依存度が高まるにつれ、ネットワークの障害が組織に及ぼす悪影響も大きくなる。組織のネットワーク管理者は、ネットワークの安定性を維持するため、障害を引き起こすトラフィックを検出し、障害を未然に防止することが求められる。また、障害の結果生じたトラフィックを迅速に検出し、速やかに障害に対処することが求められる。一方、組織は、組織内ネットワークについて、利用ポリシーを定めているのが通常であり、利用ポリシーに反するトラフィックを検出し、該当ユーザに警告することも組織の健全性を保つ上で重要である。

ここで、本論文ではワームによる攻撃と P2P ファイル共有アプリケーションによるトラフィックを異常トラフィックと定義し、検出対象とする。

まず、ワームによる攻撃トラフィックは、障害の結果生じたトラフィックであり、新たな障害を引き起こすトラフィックの典型例である。ネットワーク管理者はワームに感染したホストがないかを常に調査し、感染ホストが発生したら速やかに対処し、感染のまんえんを抑えなければならない。また、P2P ファイル交換アプリケーションの普及により、組織内のユーザが著作物を違法に交換するという事例も後を絶たない。ネットワーク管理者は、P2P ファイル交換アプリケーションの利用状況を把握し、違法な利用がないことを確認することが求められる。これらワームによる攻撃と P2P ファイル共有アプリケーションによるトラフィックは、多くの組織の運営ポリシー上、検出が求められる異常トラフィックの典型であるといえる。

しかし、こうした異常トラフィックの検出は容易ではない。シグニチャ型 IDS（3.2.1.2 参照）を用いた場合、新種のワームを検出できない可能性がある。また、近年の P2P ファイル共有アプリケーションは、特定のポート番号の監視やペイロードに対するパターンマッチでは検出困難である。さらに、IDS は扱えるパケットレートに限界があり、監視対象ネットワークの帯域に制限が生じてしまう。

本論文では、異常トラフィックの共通した特徴を利用し、単位時間あたりのユニークな通信先ホスト数に注目した異常トラフィック検出手法を提案する。そして、提案手法が従来指標の監視では検出不可能な異常トラフィックを、低コストで、シグニチャに非依存で自動検出が可能であることを示す。また、提案手法による異常トラフィック検出の検出精度、および、広帯域への適応性、そして、リアルタイムな異常トラフィック検出への利用可能性を評価する。

3.2 では既存の要素技術と異常トラフィック検出に広く用いられる手法を挙げ、その特徴と問題点を明らかにする。3.3 では、既存手法の問題点をふまえ、優れた異常トラフィック検出手法に必要な機能要件をまとめる。3.4 では提案手法の目的および想定環境とその詳細について述べ、3.5 でその実装と、動作例を示す。3.6 では提案手法の評価と考察をする。3.7 では、評価と考察結果の結果をまとめ、今後の課題を述べる。

3.2 異常トラフィック検出に関する既存手法

本節では、既存の要素技術と、広く用いられている異常トラフィック検出手法を挙げ、その特徴と問題点を明らかにする。

3.2.1 要素技術

本項では、異常トラフィック検出に用いられている要素技術について述べる。

3.2.1.1 トラフィック情報取得技術

トラフィック情報とは、通信トラフィックを構成するパケットの属性情報、パケットの集合体であるコネクションおよびフローの属性情報である。また、トラフィックの流量（転送バイト量や転送パケット数）もトラフィック情報と呼ぶ。異常トラフィック

検出のためには、まず、これらトラフィック情報を取得し、分析する必要がある。本項では、トラフィック情報を取得するために広く利用されている技術として、SNMP（Simple Network Management System）[31]、ミラーリング、NetFlow について述べる。

SNMP は、ネットワーク上の機器が持つ情報の提供方法、およびその取得方法を定めた規格である。ルータやスイッチにおいて各インタフェースを通過した転送バイト量、転送パケット数を取得し、その時間変化を把握するという利用方法が一般的である。MIB-II[175] に規定された情報であれば、ほとんどのネットワーク機器で共通して利用可能であるが、パケットやフローに関する詳細な情報を取得したい場合は、別の技術を利用する必要がある。

ミラーリングではスイッチ上を流れるパケットをコピーして観測用の機器に転送する方法である。この手法を用いることで通常の通信に影響を与えることなく、その通信の全内容を取得することができる。この手法は多くのトラフィックが流れる広帯域リンクを監視した場合にスイッチや観測機器への負荷が高くなるという欠点がある。

NetFlow[37] は Cisco Systems 社 [39] が策定したトラフィック情報取得技術であり、同社のルータ製品の他、アラクサラ社 [5] や Juniper Networks 社 [149]、Extreme Networks 社 [78] 等のルータ、スイッチ製品でも利用できる。NetFlow では、一連の観測パケットを TCP におけるコネクションに近いフローと呼ばれる単位で集約し、その情報を蓄積する。蓄積されたフローの情報は、観測機器へと NetFlow パケットと呼ばれる UDP パケットとして出力される。

NetFlow には多種のバージョンがあり、取得できる属性情報がやや異なる。基本的にはレイヤ 3、レイヤ 4 のヘッダ情報、入出力インタフェース番号、バイト長等が取得可能である。例えば、NetFlow Version 5 では各フローに対して表 3.1 に示した属性情報が取得できる。これらの情報は 1 フローあたり 48 byte とコンパクトであり、長期観測での利用が可能である。

NetFlow は、サンプリングを組み合わせることにより広帯域のネットワークリンク上でもトラフィック情報取得が可能であり、既存のネットワークに変更を加えることなく、低コストでトラフィック情報の取得ができるという利点がある。

表 3.1.　NetFlow version 5 で取得できる情報

ID	information
1	通信元 IP アドレス
2	通信先 IP アドレス
3	次ホップの IP アドレス
4	入力インターフェース番号
5	出力インターフェース番号
6	そのフローに含まれるパケット数
7	そのフローに含まれるバイト量
8	最初のパケットを受信した時間
9	最後のパケットを受信した時間
10	通信元 TCP/UDP ポート番号
11	通信先 TCP/UDP ポート番号
12	IP プロトコルタイプ
13	IP type of service (ToS)
14	TCP フラグの累積論理和
15	通信元 AS 番号
16	通信先 AS 番号
17	通信元 IP アドレスのマスク長
18	通信先 IP アドレスのマスク長

3.2.1.2 IDS (Intrusion Detection System)

IDS は、狭義では攻撃者によるシステムへの進入路検知するシステムであるが、ワームによる攻撃トラフィックや特定のアプリケーションによるトラフィックの検出にも広く利用される。本論文では、ネットワーク型 IDS およびインライン型 IDS を単に IDS と表記する。

IDS には、シグニチャ型 IDS とアノマリ検出型 IDS があり、シグニチャ型 IDS は、ミスユース型 IDS と呼ばれることもある。シグニチャ型 IDS は、事前にワームや特定のアプリケーション固有のペイロード等の知識を検出シグニチャとして与えることで、該当するトラフィックが観測された際に管理者に通知することができる。この IDS は、シグニチャにマッチする異常トラフィックであれば確実に検出できるという利点がある。アノマリ検出型 IDS は、プロファイル型 IDS と呼ばれることもある。アノマリ検出型 IDS は、本稼働前に学習期間が必要である。学習期間中に観測したトラフィックの状態を正常と仮定し、その状態から外れたトラフィックが発生したときに管理者に通知する。

3.2.1.3 異常認知手法

ネットワーク管理者が、異常トラフィックの発生を認知する手法には能動的手法、受動的手法がある。

能動的手法では、システムが提供する情報に対して管理者が何らかの思考をし、管理者が異常かどうかを判断する。管理者がトラフィック量の推移グラフから、通常量を超えた上昇や下降等を探す方法がこの手法の例である。受動的手法では異常であるか否かの判断主体はシステムであり、システムが異常と判断した出来事のみを管理者に通知する。IDS で用いられる管理者への通知はこの手法にあたる。

能動的手法では、管理者がシステムの出力を定期的に調査しなければならず、管理者への負担が大きい。また、調査間隔を短くすればするほどその負担は増し、リアルタイムな異常検出は難しい。さらに、どの程度の急激な変化を異常とするかは、管理者個人の経験や知識に依存するという欠点がある。受動的な手法は、管理者の負担は小さい。しかし、システムによる異常判断の元となるしきい値の設定が難しい。しきい値を低く設定すると False-Positive 率[1]が上昇する。しかし、逆に高くすると、False-Negative 率[2]が上昇する。False-Positive 率も False-Negative 率も最小となる適切なしきい値の算出には、一般に長期の試行錯誤が必要となる。

3.2.1.4 しきい値算出技術

受動的異常認知手法の実現には、システムが正常／異常の判断機能を有する必要がある。シグニチャ型 IDS のように、トラフィックがシグニチャにマッチするかどうかの判定であれば容易に実現できる。しかし、トラフィックパターンの量的変化に対して正常か異常の判断を下すことは容易ではない。

トラフィックの正常状態と異常状態のしきい値を自動的に算出し、精度の高い通知を実現するために様々な手法が検討されている。文献 [16] では、Wavelet 変換と Holt-Winters 法を用いて異常トラフィックの自動検出を試みている。Wavelet 変換を用いて転送バイト量の時間変化を各周波数成分に分離し、成分ごとの偏差からしきい値を算出する手法と Holt-Winters 法を用いて算出する手法を比較している。いずれの手法も転送バイト量の時間変化を指標としており、転送バイト量に変化を生じさせる現象については高い精度で検出ができている。しかし、検出された現象が必ずしも管理者が必要とする異常の通知であるかどうかは十分に評価されていない。

1　正常状態を異常と通知する現象の発生率
2　異常状態を正常と判断し通知しない現象の発生率

3.2.2 既存手法とその問題点

既存の異常トラフィック検出手法は、大きくわけ、以下の 4 種類がある。

(i) 可視化された転送バイト量、パケット数から、管理者が能動的に異常を発見
(ii) しきい値から外れた転送バイト量、パケット数を異常として受動的に発見
(iii) シグニチャ型 IDS（3.2.1.2 参照）を利用して受動的に発見
(iv) アノマリ検出型 IDS（3.2.1.2 参照）を利用して受動的に発見

(i) および (ii) の手法は、MRTG[194] や FlowScan[84] で実現できる。MRTG は、観測対象ネットワーク上のルータを通過した転送バイト量、転送パケット数を SNMP を用いて取得し、その時間変化をグラフ化できる。FlowScan は、NetFlow を入力として利用することで、さらに利用ポートごとに色分けしたグラフの作成が可能である。P2P ファイル共有アプリケーションは転送バイト量に、ワームの攻撃は転送パケット数に変化をもたらすことがある。したがって、管理者が作成されたグラフに対して能動的な異常認知を行うことで異常トラフィックを発見できる可能性がある。しかし、異常トラフィックが必ずしも転送バイト量やパケット数に識別可能な変化を与えるとは限らない。また、これらの指標はユーザがファイルをダウンロードしただけで大きく変化する。つまり、これらの指標を用いた異常トラフィック検出は精度が低い。FlowScan による利用ポート別の監視も、ランダムなポート番号を使う P2P ファイル共有アプリケーションョンやワームの検出には利用できない。また、(i) の手法は管理者が能動的にシステムの出力を監視しなければならず、運用コストは高い。

(ii) の手法は手法 (i) の通知方法を能動的から受動的に変更したものである。MRTG と FlowScan は、観測値があらかじめ設定した上下限しきい値を外れた際に管理者に通知を送信することができる。受動的異常認知手法の特性として、運用コストは削減し、異常の発生から認知までの時間を短縮できる。しかし、しきい値を管理者が一元的に設定しなければならない。適切な上下限しきい値の設定には、管理者の知識や経験、さらに事前の長期観測データが必要

となり、導入コストは高くなる。また、一元的に設定したしきい値では、トラフィックの周期性等の特徴が考慮されず、早朝のトラフィックでも昼間のトラフィックも同じ基準で判断するという問題点がある。

(iii) および (iv) の手法は、3.2.1.2 で述べた IDS による通知を利用し、異常トラフィック検出する手法である。(iii) では、シグニチャが対応しない未知の異常や暗号化されたペイロードには効果がない。また、(iv) では、正常なトラフィック状態の定義方法としきい値の設定によって False-Positive 率、False-Negative 率は大きく変化する。しかし、False-Positive 率、False-Negative 率を最小にする一般的なアルゴリズムが知られているわけではなく、特定のトラフィックの高精度な検出のためには専門家によるチューニングが必要である。また、IDS は、ミラーリングなどの技術を用い、観測対象トラフィック中のパケットを全て検査するため、対象となるパケットのレートに比例して IDS 装置の負荷は上昇する。つまり、IDS を用いた (iii) および (iv) の手法は、低帯域ネットワークから広帯域ネットワークまで、スケーラブルに対応できないという問題がある。

3.3 異常トラフィック検出手法の機能要件と提案手法の目標

本節では、前節で述べた既存手法の問題点をふまえ、優位な異常トラフィック検出手法に必要な要件を述べる。

3.3.1 機能要件と既存手法

3.2.2 で述べたように、シグニチャを用いた手法や広帯域にスケーラブルに対応できない異常トラフィック検出手法には問題がある。また、転送バイト量やパケット数に注目した手法は検出精度が低く優れていない。導入コストや運用コストが高い手法も問題がある。

この点を考慮すると、異常トラフィック検出手法には次の要件が必要となる。

(a) 低コストで利用を開始可能
(b) 検出シグニチャに非依存
(c) 自動で異常トラフィックを検出可能
(d) 高い精度で異常トラフィックを検出可能
(e) 広帯域環境下で使用可能
(f) リアルタイムに異常トラフィックを検出可能

ネットワークトポロジの変更や高価な機器が不要で、容易に異常トラフィック検出が開始できるという点が要件 (a) である。また、シグニチャ等の事前知識に依存せずに、未知の異常トラフィック検出ができるという点が要件 (b) である。システムが自動的に異常トラフィックを検出することで、管理者の能動的手法に依存せず、特別な経験や知識を持たない管理者でも利用可能という点が要件 (c) である。要件 (d) として、検出精度が悪く、False-Positive や、False-Negative が多発する手法は、信頼できず全く役に立たないため、必然と高精度な必要がある。要件 (e) として、異常トラフィック検出手法は、低帯域環境下でも広帯域環境下でもスケーラブルに対応できることが望まれる。異常の発生に即座に対応するためには要件 (f) も重要となる。

表 3.2.　既存手法と機能要件

		既存手法			
		(i)	(ii)	(iii)	(iv)
機能要件	(a)	○	×	×	×
	(b)	○	○	×	○
	(c)	×	○	○	○
	(d)	×	×	△	△
	(e)	○	○	×	×
	(f)	×	○	○	○

3.2.2 に挙げた既存手法と機能要件を比較すると表 3.2 のようになる。

まず、要件 (a) を見た場合、導入時に適切なしきい値を管理者が算出する必要のある手法 (ii) は優れていない。また、製品の導入やミラーリングのためのネットワークトポロジ変更が必要となる手法 (iii)、(iv) も優れていない。次に要件 (b) を見ると、異常トラフィックの検出をシグニチャに依存している手法 (iii) は不適切である。要件 (c) を見ると、管理者が能動的に異常トラフィックの発見および判断をしなければならない (i) の手法は優れていない。次に、要件 (d) を見ると、(i) および (ii) の手法は、転送バイト量やパケット数に変化を与えるトラフィックしか発見できず、その変化が必ずしも異常トラフィックとは限らないので不適切となる。(iii) の手法はシグニチャにマッチする異常は確実に検出できる反面、未知の異常や暗号化されたトラフィックに対しては効果を発揮できず、△とした。また、高精度な検出には専門家によるチューニングが必要な (iv) の手法も△とした。要件 (e) について述べると、ミラーリン

グ等の手法とともに、すべてのパケットを精査する必要のある (iii) および (iv) の手法は優れていない。要件 (f) について言及すると、異常トラフィックの発生から、管理者がそれを認知するまでに時間を要する (i) の手法は不適切となる。

3.3.2 提案手法の目標

提案手法は、前節に挙げた機能要件をすべて満たす異常トラフィック検出手法を目指す。

提案手法では、既に多くのルータやスイッチに実装されている NetFlow を用いてトラフィック情報を取得する。この特徴により、低いコストで検出を開始できる（要件 (a)）。また。NetFlow はパケットサンプリングと併用できるため、広帯域への適応性を実現する（要件 (e)）。そして、異常トラフィックの一般的な特徴を考慮した指標を用いることで、シグニチャに依存しない検出を実現する（要件 (b)）。そして、Holt-Winters 法を利用し、異常トラフィックと正常トラフィックのしきい値を自動算出することで、要件 (c) を実現する。

高い精度で異常トラフィックを検出可能（要件 (c)）か否か、また、サンプリングレートを下げ、広帯域に適応させた時にその検出制度を維持できるか（要件 (d)）、そして、リアルタイムな異常トラフィック検出を実現できる（要件 (f)）か否かは、3.6 にて明らかにする。

3.4 提案異常トラフィック検出手法

本節では、提案する異常トラフィック検出手法について述べる。

3.4.1 提案手法の対象と異常トラフィックの特徴

提案手法は、3.1 で定義した異常トラフィックの検出が最も必要とされている場所である、企業や学校など組織内のネットワークを対象として、異常トラフィックの検出を目指す。

ここで、3.1 に定義した異常トラフィックの一つ目である、ワームによるトラフィックを考える。文献 [323] では、ワームの検出を目指し、ワームによるトラフィックの特性を様々な角度から分析している。短時間内に多くの新規ホストに通信をしているホストがワームに感染したホストの可能性が高いとしているが、自動検出のためのしきい値設定手法、実トラフィック上での検出精度については考察がなされていない。ワームの一般的な性質を考えると、文献 [323] でも述べられてるとおり、ワームはより多くのホストに感染しようとする性質を持つ。ワームに感染したホストは、他のホストの脆弱性を攻撃パケットにより攻撃し、そのホストに感染する。新たに感染したホストは同様に他のホストの脆弱性を攻撃する。この動作が再帰的に行われることで、ワームの感染は爆発的に広まる。攻撃先ホストの選択方法はワームによって異なり、ランダムに選んだアドレスに対して攻撃するワームもあれば、自ホストのアドレスの周辺アドレスから攻撃を開始するワームもある。しかし、より多くのホストに感染しようと、多数のホストに攻撃パケットを送信する点は共通である。

次に、P2P ファイル共有アプリケーションによるトラフィックを考える。文献 [249] では、P2P ファイル共有アプリケーション固有のペイロードからそのトラフィックの検出を試みている。しかし、近年、ペイロードを暗号化する P2P ファイル共有アプリケーションが増えるにつれ、この手法は効果を失ってきており、ペイロードに依存しない別の検出手法を考える必要がある。一般的な P2P ファイル共有アプリケーションは、P2P ネットワークに接続されているホストのアドレス情報、各ホストが提供している共有ファイルの情報等を各ホストに分散管理している[3]。また、共有されているファイルの実体が複数のホストに分散配置される P2P ファイル共有アプリケーションもある。こうした中、P2P ファイル共有アプリケーションはユーザが要求したファイルを検索し、ダウンロードするために分散管理された情報を収集する必要がある。そのため、P2P ファイル共有アプリケーションは多数のホストと通信をするという性質をもつ。

3.4.2 注目指標

文献 [16] など、IDS 以外の異常トラフィック検出手法および関連研究は、通過トラフィックのバイト流量、パケット流量の変化に注目している。この理由は、歴史的にこれらの値が SNMP 等を用いて低コストで取得ができたからである。しかし、3.2.2 で述べたように、この値に注目することが異常トラフィック検出に最適であるとはいえない。

一方、はじめに定義した異常トラフィックは 3.4.1

3　一部の情報が特定のサーバホストで集中管理されるハイブリッド型と呼ばれる形態もある。

で示した性質を持つため、共通して単位時間あたりのユニークな（重複を除いた）通信先ホスト数（以下、注目指標）が上昇する。この注目指標は、単位時間に観測された重複を除いた通信先ホスト数である。

例えば、表 3.3 に示した始点（source）と終点（destination）を持つ、4 つのトラフィックが観測されたとする。注目指標は観測された重複を除いた通信先ホスト（destination host）が、「192.168.0.1」、「192.168.0.3」の 2 つなので、2 と求められる。

注目指標はファイルのダウンロード、メールのやりとり等、サーバクライアント型サービスによる変動は少ない。組織内のホストが、Web サイトや FTP サイトから巨大なファイルをダウンロードしても、値の変化は 1 である。また、多くの組織内ホストからアクセスされる Web サイトやメールサーバ等が及ぼす影響も（アドレスベースの負荷分散が行われてない限り）たかだか 1 である。

本論文では、異常トラフィックによって大きく変動し、他のトラフィックによる変動が少ない注目指標の時間変化を用いて異常トラフィックの検出を試みる。ただし、本手法は、3.4.1 で示したように、対象組織内の異常トラフィックを発生させているホストを見つける目的での利用を想定する。したがって、提案手法は、上り（対象組織内から組織外へ向かう）トラフィックに対して注目指標を利用する。

アドレス数に注目した既存研究として、文献 [321] や文献 [212] があり、共に DDoS 検知のために送信元アドレスを観測している。ワームや P2P ファイル共有アプリケーションによるトラフィックを発生させている組織内ホストの検出を目的とした本研究とは、その目的が異なる。

3.4.3 指標の周期性と適合モデル

注目指標の時間変化は、観測対象ネットワークを利用する人々の生活周期と深く関連し、周期性を持つ。奈良先端科学技術大学院大学（以下本学）で観測された注目指標の時間変化は図 3.1 のようになる。昼間にピークを迎え早朝が谷となり、土日は平日に比べて値が小さい傾向があることがわかる。また、24 時間周期、1 週間周期に強い周期性が見られることが明らかになった。しかし、周期性があるとはいえ注目指標にはばらつきがある。例えば、先週の注目指標と今週の注目指標は似ている可能性は高いが全く同じではない。こうした傾向はほとんどの組織で同様に見られる。

また、注目指標はアプリケーションには通信継続時間によっても支配され、現在の注目指標は、1 時間前の注目指標よりも 5 分前の注目指標により似ている可能性が高い。こうした特徴を持つ注目指標に対するしきい値計算には Holt-Winters 法が適しており [23]、本手法では Holt-Winters 法と呼ばれる手法を用いる。

Holt-Winters 法は、需要予測分野で実績のある予測手法であり、季節変動のある需要予測に最適であ

表 3.3.　注目指標のカウント例

Flow ID	source (host:port)		destination (host:port)
1	10.0.0.1:1025	→	192.168.0.1:80
2	10.0.0.2:1025	→	192.168.0.1:80
3	10.0.0.1:1026	→	192.168.0.3:80
4	10.0.0.1:1025	→	192.168.0.1:443

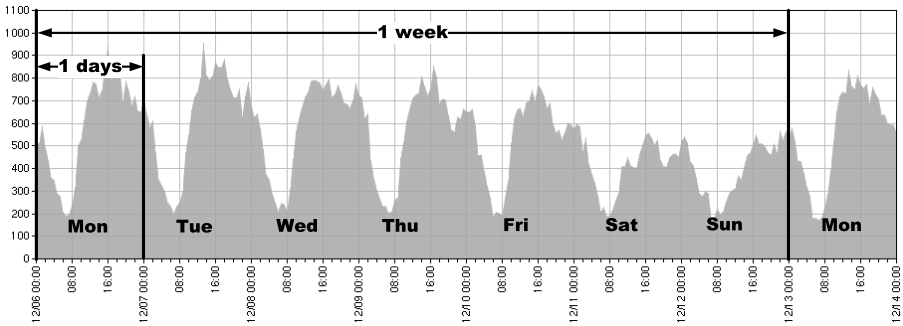


図 3.1.　注目指標の時間変化の周期性

る。過去の観測値に対して各種パラメータと信頼水準を与えることで、次の観測値がある一定の確率以上で収まる範囲（しきい値内）を算出することができる。需要予測分野では周期を 1 年と設定することが多い。しかし、注目指標の周期は、その周期性から考慮し、1 日または 1 週間を周期と設定することが妥当である。ただし、周期を 1 日とした場合、土、月曜日など前日と特性が異なる日の精度が悪化するため、本論文では周期を 1 週間と設定した。

ここで、Holt-Winters 法による予測の信頼水準を高く設定することで、しきい値から外れた値を通常のパターンから外れた異常なトラフィックとして取り出すことが可能となる。

3.4.4 しきい値計算アルゴリズムと異常トラフィック検出

しきい値計算と異常トラフィック検出の具体的なアルゴリズムは次のようになる。

単位時間を N 秒とした場合、現在時刻から過去 M 秒に対して N 秒ごとの注目指標の変動を取得する。取得した値に Holt-Winters 法を適応し、次の N 秒に対する上限しきい値、下限しきい値を設定する。次の N 秒間に実際に観測された注目指標が、設定されたしきい値内に収まらなかった場合。つまり、上限しきい値を上回った場合、または下限しきい値を下回った場合に、異常トラフィック発生の可能性有りとして管理者に通知する。この作業を N 秒ごとに実行し、しきい値内に収まらなかった実測値を通知することで、リアルタイムな異常トラフィック検出を実現する。この過程を図 3.2 に示す。

ここで、M はトラフィック計測を開始してから提案手法が利用可能になるまでの期間となる。あまりにも M を長くすると、運用現場において実用的ではない。しかし、周期変動を正しくとらえるために、設定した周期の数倍以上の長さは必要となる。この点を考慮し、本論文では、M を 28 日（2419200 秒、4 周期）とした。

3.4.5 Holt-Winters 法のパラメータ

Holt-Winters 法によるしきい値設定には、信頼水準 CL、α、β、γ の 4 つのパラメータが必要となり、$0 < CL, \alpha, \beta, \gamma < 1$ である。CL はしきい値内に次の実測値が含まれる確率を示し、1 に近いほど上限しきい値と下限しきい値の差は大きくなる。信頼水準を 1 に近づければ近づけるほど、False-Positive 率が減り、False-Negative 率が上昇する。

α、β、γ は、直近の観測値を重視してしきい値を求めるか、長期の観測値の変動を重視して求めるかを設定するパラメータである。1 に近いほど直近の実測値を重視し、0 に近いほど長期に実測値を重視する。1 に近い値を利用すれば、トラフィックパターンの急激な変化にしきい値が順応しやすくなる。これは、異常トラフィックが発生が継続している状況を許容するしきい値が生成される可能性が高いことになる。逆に、0 に近い値を設定すれば正常なトラフィックパターン変化に対するしきい値の順応が遅れ、結果的に False-Positive 率が上昇する。この点を踏まえてパラメータを調整する必要がある。中でも、α は周期より短い変動を、β は周期より長い変動を、γ は周期ごとの変動をとらえるために利用する。

統計処理ソフトウェアによっては α、β、γ の値を、過去の実測値と過去の予測値の誤差が最小になるよ

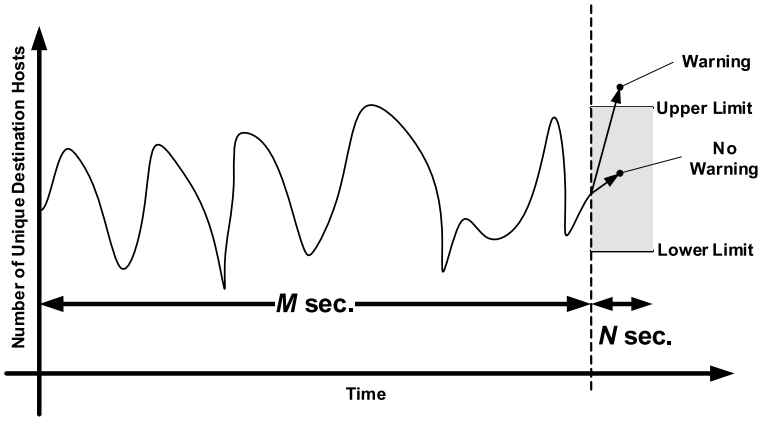


図 3.2. 異常トラフィック検出手法

うに自動算出できるものもあり、自動算出値を利用する方法もある。

3.5 実装

本節では、提案手法の実装について述べる。

3.5.1 実装環境

本システムは、一般的な PC で動作する Linux 上に実装した。使用した PC のスペックは表 3.4 の通りである。データベースサーバとしても利用することから、ディスク容量は余裕を持って確保されている。しかし、実際のフロー属性データの容量は 10 ヶ月分で 10 GB 程度であり、近年の一般的なコンピュータのハードディスク容量を考慮すると、十分、数年以上の長期観測に利用が可能である。

NetFlow パケットから属性情報をデータベースに格納する部分は、再利用可能な個別のソフトウェア（NetFlow2MySQL[198]）として C 言語を用いて実装した。データベースマネジメントシステムとして MySQL[196] を用いる。解析モジュールは Perl を利用して実装し、Holt-Winters 法によるしきい値の算出には、GNU R[95] を利用した。

3.5.2 動作例

提案手法を実測トラフィックに対して適応した例を図 3.3 に示す。注目指標の実測値（Observed Value）に対して、Holt-Winters 法を用いた上限しきい値（Upper Boundary）、下限しきい値（Lower Boundary）を設定し、実測値がしきい値から外れた際に管理者に通知する。リアルタイムの異常トラフィック検出においては、しきい値の算出と実測値との比較は N 秒ごとに実行される。

3.6 評価と考察

本節では、提案手法がその設計目標を満たしており、異常トラフィック検出に利用可能かを評価する。

3.6.1 評価項目とその意義

提案手法は、3.3.1 に述べた要件をすべて満たすことを目標に設計した。要件 (a) および (b) の機能要件は提案手法の設計段階で満たしている。したがって、本節では評価対象としない。ここで、要件 (c) から (e) の目標が達成されているかを評価するため、次の (1) から (4) の評価項目を挙げる。

(1) パラメータごとの False-Positive、
　　False-Negative 率に与える影響
(2) サンプリングレートが False-Positive、
　　False-Negative 率に与える影響
(3) 従来指標に注目した場合との比較
(4) しきい値計算に必要な時間

まず、(1) の評価項目について述べる。提案手法は、3.4.5 に述べたように各種パラメータを必要とする。一度パラメータを与えてしまえば、あとは管理者の判断を必要としない。つまり、妥当なパラメータが決定できれば要件 (c) を満たすことになる。また、要件 (d) を満たすためには、提案手法が多くの異常トラフィック検出でき、かつ、正常なトラフィックを異常と通知することが少ない必要がある。そこで、

表 3.4.　実装環境

CPU	Intel Xeon CPU 2.80 GHz × 2
Memory	4 GB
HDD	400 GB
Network Interface	1 Gbps (BCM5703 Chip)
OS	Mandrake Linux 9.2 (Linux2.4.22)

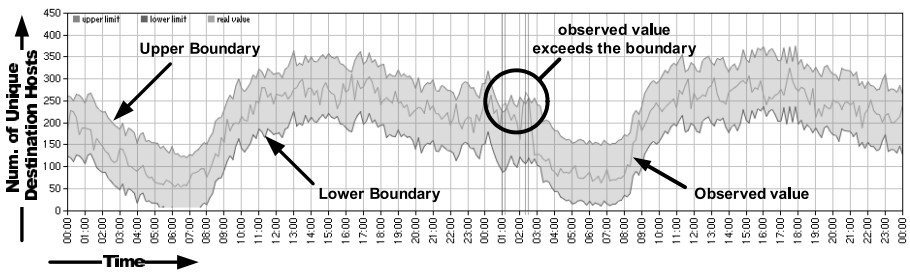


図 3.3.　提案手法の動作例

評価項目 (1) として、単位時間 N と Holt-Winters 法のパラメータごとに提案手法の False-Negative の発生数と、False-Positive 率を評価し、パラメータの設定指針について考察する。同時に、自動算出されたパラメータの利用が妥当かどうかを評価する。ただし、本論文では、他の手法で異常トラフィックが認められなかった時刻に提案手法が異常トラフィック発生の通知を発した場合、提案手法の False-Positive とする。また、他の手法により確実に異常トラフィックの発生と認められた出来事をリファレンスとし、そのリファレンス異常が提案システムによって検出できなかった場合、提案システムの False-Negative とカウントする。

次に、(2) では、パケットのサンプリングレートが、提案手法の False-Positive、False-Negative 率に与える影響を評価する。パケットのサンプリングレートを下げても False-Positive、False-Negative 率に変化が生じなければ、本提案手法はスケーラブルに広帯域に適応可能であり、(e) の要件を満たす。そして、評価項目 (3) では、提案手法で用いた注目指標が妥当性を評価するために、他の指標を用いて異常トラフィック検出を試みた場合と比較する。最後に、評価項目 (4) について述べる。提案手法によるしきい値計算に必要な時間を計測し、要件 (f) に適合したリアルタイムな異常トラフィック検出が実現可能かを評価する。

3.6.2 リファレンス異常

提案手法の False-Negative 率を評価するために、次の 3 つの出来事を異常トラフィックのリファレンス（以下リファレンス異常）として利用する。

(i) ワーム感染の拡大
(ii) P2P ファイル共有アプリケーションの実行 A
(iii) P2P ファイル共有アプリケーションの実行 B

リファレンス異常 (i) は、Sasser と呼ばれるワームに感染した PC が学内に持ち込まれ、学内ホスト約 40 台に感染が広まったケースである。リファレンス異常 (ii) は一人のユーザが P2P ファイル共有アプリケーション WinMX を起動し、ファイルを検索と共有をした事例である。リファレンス異常 (iii) は別のユーザが同様に P2P ファイル共有アプリケーション e-Donkey を使用した事例である。いずれのケースも本提案手法の稼働前に起きた出来事であるが、提案手法を過去の観測データに対して適応し、発見可能であったかどうかを評価する。実際の運用現場において、リファレンス異常 (iii) はワームが攻撃に使用する 445/udp ポートを監視することで発見され、リファレンス異常 (ii) と (iii) はシグニチャマッチング型 IDS で発見された。提案手法においては、使用するポート番号やシグニチャ等の事前知識を用いずにこれらのリファレンス異常の検出を試みる。

表 3.5 に、それぞれのリファレンス異常の継続時間、および、注目指標、転送パケット数、転送バイト量に与えた影響を示す。ただし、注目指標への影響に関しては、$\frac{1}{1000}$ サンプリング[4]に基づいた値である。転送パケット数および転送バイト量はサンプリング後の実測値から実際の流量へ換算している。

3.6.3 Holt-Winters 法のパラメータ評価と考察

まず、False-Negative の評価のため、Holt-Winters 法での CL、α、γ を変化させた場合に、各リファレンス異常が検出できたかどうかを評価する。次に、False-Positive 率の調査のため、他の手法で異常トラフィックが認められなかった時刻に提案手法が異常トラフィック発生の通知を発した率を評価する。多くのリファレンス異常を正常に検出できる、言い換えれば False-Negative 率が低く、正常時を異常と通知しない（False-Positive 率が低い）パラメータが適しているといえる。

- α に対して、40 分以内の実測値に 90%の重みを

表 3.5. リファレンス異常による影響

	リファレンス異常		
	(1)	(2)	(3)
継続時間	24 時間以上	約 20 分	約 6 時間
注目指標への影響（/10min.）	978 hosts	89 hosts	14 hosts
転送パケット数への影響	1.63 kpps	165 pps	52 pps
転送バイト量への影響	637 kbps	141 kbps	252 kbps

4　1000 個に 1 個の割合でパケットをサンプリングし、サンプリングされたパケットを元に作成されたフローの属性情報を利用

表 3.6. Holt-Winters 法のパラメータの評価

ID	N (sec.)	CL	α	γ	リファレンス異常 (i)	リファレンス異常 (ii)	リファレンス異常 (iii)	False-Positive
1	600	A	a	i	○	○	×	3.47%
2				ii	○	○	×	5.90%
3			b	i	○	○	×	4.51%
4				ii	○	○	×	4.17%
5			c	iii	○	○	×	4.17%
6		B	a	i	○	○	×	1.39%
7				ii	○	○	×	2.43%
8			b	i	○	○	×	1.39%
9				ii	○	○	×	1.39%
10			c	iii	○	○	×	1.74%
11	2400	A	a	i	○	×	×	16.67%
12				ii	○	×	×	18.06%
13			b	i	○	○	×	2.78%
14				ii	○	○	×	8.33%
15			c	iii	○	○	×	6.94%
16		B	a	i	○	×	×	6.94%
17				ii	○	×	×	6.94%
18			b	i	○	×	×	2.78%
19				ii	○	×	×	4.17%
20			c	iii	○	○	×	2.78%

置く場合-(a)、50%の重みを置く場合-(b)、自動算出値（3.4.5 参照）を利用-(c)

- γ に対して、1 周期前の同曜日、同時刻に 90%の重みを置く場合-(i)、50%の重みを置く場合-(ii)、自動算出値（3.4.5 参照）を利用-(iii)
- 信頼水準 CL が、99%の場合-(A)、99.9%の場合-(B)
- 単位時間 N を 600 秒（10 分）とした場合、2,400 秒（40 分）とした場合

のそれぞれの組み合わせについて、リファレンス異常が正常に検出できたかどうかと、False-Positive 率について評価した。その結果を表 3.6 に示す。また、パケット単位のサンプリングレートは $\frac{1}{1000}$ とした。表 3.6 中の○は、そのパラメータの組み合わせで該当リファレンス異常が検出できたことを、×は検出できなかったことを示す。

リファレンス異常 (i) はすべてのパラメータの組み合わせにおいて正常に検出できており、提案手法によるワームの感染行為の検出は妥当であるといえる。しかし、リファレンス異常 (ii) は $N = 300$ の時はすべて検出できているが、$N = 2400$ の時は、検出できるパラメータの組み合わせと、検出できない組み合わせがある。リファレンス異常 (i) は、発見され対処されるまで数日の時間を要し、異常状態の継続状態が長時間にわたった。一方、リファレンス異常 (ii) の継続時間は 20 分（1200 秒）程度と短時間であった。この結果、単位時間を 2400 秒とした時のリファレンス異常 (ii) が、注目指標へ与える影響も半減し、検出できなかったと考えられる。その一方で、リファレンス異常 (iii) についてはいずれのパラメータでも検出に成功していない。これは、表 3.5 から分かるように、(iii) の P2P ファイル共有アプリケーションのトラフィックが注目指標に与える変化が小さすぎ、全体量としてみた注目指標をしきい値内に止めたためである。

ここで、α、γ の設定について考察する。3.4.5 の手法で自動算出される α、γ が最適な False-Positive 率をもたらすとは限らない。また、高い False-Positive 率をもたらすわけでもない。一方、この評価結果から、特定の α と γ の値が一般に最適とは言うことはできない。例えば、表 3.6 の ID1 と ID11 は、α と γ の設定方針が同等であるが、ID1 の False-Positive 率は低く、ID11 の False-Positive 率は高い。このように、同等の α と γ を与えても、N が変化すると、False-Positive 率が急激に変化することもある。この点を考慮すると、自動算出値の利用も妥当である。

また、N および、CL の設定について考察する。N が大きいと、短時間の変化しかもたらさない異常トラフィックは検出できていない。この点を考慮すると、対象とする異常トラフィックの一般的な継続時間より短い N を設定する必要があることが分かる。また、ID14 と ID19 の比較で分かるように、高い CL の値は False-Positive 率を下げ、検出できない異常トラフィックを増やすことが分かる。

これらの指針によって N および CL を設定し、自動算出された α、γ を利用することで 3.3.1 に述べた機能要件 (a) を達成することができる。

3.6.4 サンプリングレートごとの評価と考察

次に、N を 10 に、α を (a) に、β を (i) に、信頼水準 CL を (A) に固定し、サンプリングレートを低下させる。3.6.3 で用いたサンプリングレート $\frac{1}{1000}$ を元に、$\frac{1}{2000}$、$\frac{1}{4000}$、$\frac{1}{8000}$、$\frac{1}{16000}$ と低下させた時の、リファレンス異常の検出の可否、および False-Positive 率は表 3.7 のようになった。表 3.7 中の○は、その

表 3.7. サンプリングレートごとの評価

サンプリングレート	リファレンス異常			False-Positive 率
	(i)	(ii)	(iii)	
1/1000	○	○	×	3.47%
1/2000	○	○	×	2.43%
1/4000	○	×	×	1.74%
1/8000	○	×	×	1.74%
1/16000	○	×	×	3.13%

サンプリングレートで該当リファレンス異常が検出できたことを、×は検出できなかったことを示す。

リファレンス異常 (i) はサンプリングレートを下げても検出ができているが、リファレンス異常 (ii) については $\frac{1}{4000}$ 以下のサンプリングレートでは検出できていない。これは、サンプリングレートが下がるにつれ、注目指標の正常時の変動係数が大きくなることが原因であると考える。正常時の変動係数が大きくなると、信頼水準が等しくても、異常トラフィックの発生による注目指標の変化がしきい値内に収まる可能性が高くなるからである。

3.6.5 従来指標との比較

従来から広く用いられている転送バイト量やパケット数の変化に注目し、3.6.3 で使用したパラメータの組み合わせにより Holt-Winters 法を適応した。その結果を、表 3.8 に示す。◎はすべてのパラメータの組み合わせにおいて検出できた指標、○は一部のパラメータの組み合わせにおいて検出できた指標、×はいずれのパラメータの組み合わせにおいても全く検出ができなかった指標を示す。

転送バイト量や転送パケット数を指標に用いた場合、本論文で定義した異常トラフィックは全く検出できていない。提案手法では、リファレンス異常 (i) および、リファレンス異常 (ii) の検出に成功しており、注目指標は妥当であるといえる。転送バイト量やパケット数に注目した場合に検出されたトラフィックは、http や ftp、scp 等によるファイルの転送がほとんどであった。

表 3.8. 従来指標との比較

	リファレンス異常		
	(i)	(ii)	(iii)
注目指標	◎	○	×
転送パケット数	×	×	×
転送バイト数	×	×	×

3.6.6 しきい値計算に必要な時間の評価と考察

リアルタイムな異常トラフィック検出には N 秒間隔でしきい値の計算が必要である。3.5.1 で挙げた実装環境上で、$N = 300$ の時の、単位時間のしきい値計算の所用時間は 0.23 秒、$N = 2400$ の時の所要時間は 0.09 秒以下である。これらの所要時間は、データベースから過去の実測値を抽出する時間を考慮しても十分 1 秒以内に収まる。提案手法は十分リアルタイムなトラフィック検出に用いることができ、機能要件 (f) を満たす。

3.7 おわりに

本提案手法は、3.3 に挙げた機能要件を一部満たすことができた。

まず、従来広く用いられていた転送バイト量やパケット量の監視では発見できなかった、ワームによる攻撃トラフィック、および、P2P ファイル共有アプリケーションによるトラフィックを、検出シグニチャ等の事前知識を用いずに検出することができた。そして、提案手法はその設計において、要件 (a)、要件 (b) を満たすことができた。次に、α、β の自動算出値が妥当であることを示し、CL および、N の設定指針を示した。これにより、提案手法は管理者の知識や経験に依存しない、自動異常トラフィック検出を実現でき、要件 (c) を満たすことができた。そして、提案手法による異常トラフィック検出は、十分短時間で実現できることがわかり、要件 (f) を満たすことができた。

しかし、要件 (d) および要件 (e) の実現性については十分とはいえない。まず、すべての異常トラフィックが検出できるわけではなく、3.6.2 で挙げた、リファレンス異常 (iii) のように提案手法では検出できない異常トラフィックも存在する。しかし、検出できなかった異常トラフィックも注目指標に少なからず影響を与えている。検出できなかった原因は、そのトラフィックによる影響が、全体量としてみた注目指標をしきい値外に押し出すために十分でなかったためである。こうした異常トラフィックの検出にはさらなる工夫が必要となる。同様に、監視対象ネットワークの利用人数が増加すると、1 ホストが注目指標に与える影響が相対的に小さくなり、同様に異常トラフィックの検出が困難になる。このような問題に対処するには、全体量としての注目指標に注目するだけでなく、あらかじめサブネットごとに分割し

た注目指標について提案手法を適応するという応用が考えられる。このような応用手法の実装と評価が今後の課題である。

そして、数パーセント以上の False-Positive 率も実運用への投入に向けた課題である。今後、注目指標だけでなく、通信先ホストのアドレス、使用ポート番号のばらつき等、別の指標を組み合わせて異常の評価をした場合の False-Positive 率を検証していく必要がある。

また、3.6.4 で示したように、サンプリングレートを下げた際の異常トラフィック検出精度が低下している。しかし、$\frac{1}{16000}$ といった低いサンプリングレートでも検出できているリファレンス異常も存在することから、IDS 等の全てのパケットを検査する手法と比較すると、広帯域への適応性は高い。

さらに、3.6.1 で述べたように、本評価では、他の手法で異常トラフィックが認められなかった時刻に提案手法が異常トラフィック発生の通知を発した場合、提案手法の False-Positive としている。しかし、この方法では、他の手法で検出できなかった異常トラフィックが提案手法のみで検出できた場合も False-Positive とされるという問題がある。この問題の解決のために、さらなる解析結果の分析と実地調査の強化が必要となる。

最後に、本論文では P2P ファイル共有アプリケーションによるトラフィックを異常トラフィックと定義した。しかし、提案手法はその特徴から、P2P 型通信を行うアプリケーションによるトラフィックが検出される。検出精度の向上には、検出されたトラフィックが、P2P ファイル共有アプリケーションによるものなのか、それ以外の P2P アプリケーションによるものなのかを判断する必要がある。それに向けた、P2P ファイル共有アプリケーション独自の通信のモデル化が今後の課題となる。

第 4 章　サンプリング計測におけるトラフィック傾向把握手法の提案

4.1 はじめに

コンピュータネットワーク、特にインターネットやイントラネットの生活インフラ化するにつれ、ネットワーク管理者としては管理するネットワークがどのような利用状況下にあるかを把握する必要性が高まってきている。日々ネットワークの利用状況を把握していることにより、将来のネットワーク増強計画や、急激な利用状況の変化からネットワークに対して脅威となりうる問題発見の基盤情報として利用できる可能性がある。

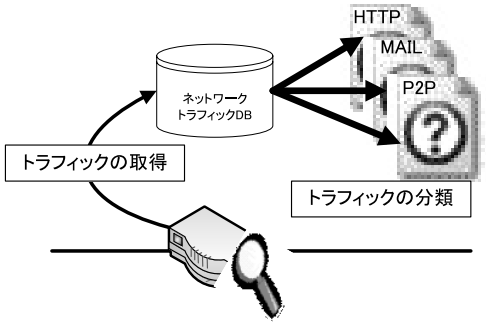


図 4.1.　トラフィックの計測と分類

図 4.1 に示すように、ネットワーク利用傾向を把握するためには、ネットワークトラフィックの取得とその解析および分類は必須である。一方で、ネットワークをとりまく技術は日々進化し、管理する側としては様々な問題が生じてきている。

ネットワークトラフィック取得は、従来よりトラフィックを全取得する手法（以下、フルダンプ計測という）が用いられてきた。近年では、ネットワークトラフィック増加の問題を解決する手法として、サンプリング計測が登場してきている。サンプリング計測で得られるデータは、パケットが切りつめられていたり独自の形式に集約されたりしているため、設定や利用方法が限られる。その一方で、サンプリングベースの計測手法であることから、広帯域にも対応させることが可能である。

ネットワークトラフィック解析は、従来より通信に利用されている IP アドレスベース、ポート番号ベースで解析する手法が用いられてきた。しかし、ネットワークを利用するアプリケーションの多様化、P2P アプリケーションやユーザが動的に利用ポート番号を指定できるアプリケーションの登場、ウイルスやワームによるなりすまし通信の発生などにより、ネットワークの利用形態と利用ポート番号のかい離が大きくなってきている。このような問題に対してはパケットアナライザや、ネットワークトラフィック解析ツールの一つである IDS などを利用することにより把握できる可能性がある。

それぞれの問題を解決するための手法がそれぞれ

存在する。しかし、それらの解決手法は独立して用いられることが多く、同時に用いた場合に起こりうる複合的な問題に関しては、まだ議論が浅い。

そこで、本稿ではまず、広帯域に対応可能なサンプリング計測により得られたトラフィックログデータを用いて、ネットワークトラフィック解析をする場合に発生しうる問題を洗い出す。そして、それら洗い出された問題に対して、トラフィックログデータに基づき、正確にネットワークトラフィック解析、特に本稿では解析によりネットワーク利用状況を把握するための手法に関して議論する。

4.2 技術比較と問題点

本節では、既存のネットワークトラフィック計測、解析技術の紹介と起こりうる問題点を述べる。

4.2.1 ネットワークトラフィック計測手法

ネットワークトラフィック計測手法は、主にネットワークトラフィックの増加により、様々な手法が登場してきている。

4.2.1.1 フルダンプ計測手法

昔はネットワークに接続されるホストの数も決して多くなく、ネットワークの利用機会も限られていたためネットワークトラフィックの取得が容易であった。そのため、従来より、ネットワークトラフィック取得にはフルダンプ計測が用いられてきた。tcpdump[269] などのツールを利用すれば、一般的なコンピュータでもトラフィック計測が可能である。

しかし、近年のコンピュータ性能の向上、利用目的の多様化、ネットワークに接続されるホスト数の増加により、そのネットワークトラフィックは日々増大してきている。フルダンプ計測では、基本的にすべてのトラフィックを記録するため、保存されるデータ容量が莫大になりがちである。さらに、計測時間に比例してデータ保存領域も増加するため、長期的な計測に用いるには現実的に難しい。特に、近年の数 Gbps オーダーから数十 Gbps オーダーのネットワークリンクの計測に対しては、データ量はもとより、トラフィックの取りこぼしなどが発生する場合がある。1.5 Mbps の ADSL 程度のトラフィックが流れるネットワークトラフィックを 1 日間計測するだけでも、130 GB ものデータ保存領域が必要になる。

また、近年のウイルスや、ワームなど急速に拡散する脅威に対応するためには、ネットワークの一部を計測しているだけでは対応が遅れてしまうことになる。そのために、ネットワークの主要な地点のみならず、多地点でのトラフィック計測を行う必要性が高まってきている。しかし、多地点におけるフルダンプ計測は、コストが膨大になりがちである。

4.2.1.2 サンプリング計測手法

そのような問題を解決する手法として、サンプリングによる計測技術が登場してきている。各ネットワーク機器ベンダにおいても様々な角度からトラフィックを計測できる手法を提案している。Cisco Systems 社が提案する NetFlow[37] や InMon 社が提案する sFlow[251] はその代表例である。

サンプリング計測では、一般的にトラフィックを間引きながら取得する。そのためサンプリングレートを適切に設定することにより、広帯域にも対応可能である。また、取得するトラフィックデータを集約し、トラフィックデータを特定ホストに送信する機能を有するものが多い。そのため、フルダンプ計測に比べ、低コストで多地点を計測することも可能である。

しかし、サンプリング計測により得られたトラフィックログデータは互換性に乏しいものも多く、それぞれの手法に対しては、それぞれ独自の解析手法を用いているのが現状である。

4.2.2 ネットワークトラフィック解析手法

ネットワークトラフィック解析手法も、ネットワークトラフィックの多様化により、様々な手法が登場してきている。

4.2.2.1 IP アドレスベースの解析

以前は、ネットワークを介してサービスを利用する機会が少なかった。また、コンピュータの性能も低かったため、ネットワークサービスを提供するサーバコンピュータは、サービス毎に設置されることが多かった。そのため、どの IP アドレスが通信しているかを知ることでどのようなサービスのトラフィックが流れているかを把握することが可能であった。

しかし、近年のコンピュータ性能の向上により、一台のコンピュータ上で様々なサービスを提供するようになってきた。そのため、IP アドレスのみでト

ラフィック内容を解析することは、事実上不可能となった。

4.2.2.2 ポート番号ベースの解析

各種サーバアプリケーションが利用するポート番号はおおよそ決まっている。特に主要なプロトコルを利用するサービスは、ウェルノウンポートを利用することが慣例となっている。したがって、一つのホスト上で複数のサービスが提供されていても、通信に利用されているポート番号を知ることで、どのようなサービスのトラフィックが流れているかを把握することが可能である。

しかし、近年では P2P アプリケーションのようなウェルノウンポート以外を利用して通信するアプリケーションが増加している。さらに、FTP の Passive モードのように、従来ウェルノウンポートを利用していたサーバアプリケーションにおいても、通信先ホストとネゴシエーションし、動的に利用ポート番号を変更して通信するケースも増えている。そのような状況下では、ポート番号ベースの解析も信頼性を失ってきている。

4.2.2.3 ペイロード捜査による解析

多様化するネットワークトラフィックに対応するため、通信内容、つまりパケットのペイロードを捜査して通信内容を把握する手法が登場してきている。ペイロード捜査には、主にパターンマッチング手法が用いられることが多い。

パターンマッチング手法は、あらかじめ検出用のパターンを記述した定義ファイル（以下、ルールセットという）を用意しておき、そのルールセットと比較しながらトラフィックを解析する。ルールセットに一致するペイロードを持つパケットを受信した場合に、トラフィックを特定する。

パターンマッチング手法は Intrusion Detection System（IDS）などでよく用いられている。IDS は異常なパケットを検出し、管理者に通知するシステムである。オープンソースの snort[260] などが有名である。新たな脅威に対応するため、ルールセットは日々新しいものが出ている。IDS によっては、定期的、自動的にルールセットを更新できるものもある。しかし、ルールに記述されていないトラフィックは基本的に検出できないなどの問題がある。

4.2.2.4 アノマリ解析

ペイロード捜査では、既知なもの以外を検出するのは難しい。一方、アノマリ検出手法は、あらかじめ “ネットワークトラフィックの定常状態” というものを定義しておき、定常状態から外れた場合に異常とする手法である。ルールセットに依存しない検出を行うことができるため、新たな脅威に対しても柔軟に対応できる。しかし、定常状態を定義するのが非常に難しく、誤った定常状態の設定をすると、フォールスポジティブや、フォールスネガティブが増えるなどの問題がある。フォールスポジティブとは、本来異常でないものを異常と検出してしまう現象のことである。誤検知とも呼ばれる。フォールスネガティブとは、本来異常であるものを見過ごしてしまう現象のことである。不検知とも呼ばれる。

4.2.3 新たな問題の発生

増大するネットワークトラフィックの計測、多様化するネットワークトラフィックの解析の問題を解決するための手法がそれぞれ登場してきている。しかし、それらの手法は現在まで独立して用いられることが多かった。しかし、ネットワーク周辺技術が高まるにつれ、トラフィック量が多い地点においても、様々な角度から解析したいという要求が高まってきている。当然、トラフィック量が多くなればなるほど、すべてのトラフィックを取得するのは困難になっていく。しかし、4.2.2 で述べたトラフィック解析手法は、ネットワークトラフィックをすべて取得可能であるという前提で動作するものがほとんどである。

サンプリング計測を利用して特定のトラフィックを検出しようという試み [320, 327, 335] はいくつかある。しかし、基本的にアノマリ検出であり、トラフィック内容まで踏み込んだ解析を試みる手法ではない。

トラフィックが全取得可能な環境を想定しているトラフィック解析手法に対して、サンプリング計測により取得されるトラフィックログデータを用いたい場合、様々な問題が起こりうると考えられる。しかし、そのような問題に関しては、まだ明らかになっているとは言い難い。

そこで次節では、広帯域に対応可能なサンプリング計測を行いながらも、ネットワークトラフィック解析をする場合に発生しうる問題を洗い出すための実験を行い、考察する。

4.3 問題点の抽出

本節では、従来のフルダンプ計測とサンプリング計測により取得されるトラフィックログデータに対し、ネットワークトラフィック解析ツールを適用させた場合、出力される結果にどのような差が出るかを検証する。

本稿では、ネットワークトラフィック解析ツールとしてIDSを用い、その中でもオープンソースのsnortを改造して利用する。

4.3.1 実験

実験は以下の手順で行う。

(1) 学内スイッチにおいて、フルダンプおよびサンプリング計測によりトラフィックを計測、蓄積する
(2) 蓄積されたトラフィックログデータに、それぞれ改造したsnortを適用する
(3) snortより出力されるアラートログから、どのような差が出るかを検証し、その原因の考察を行う

実験ネットワークとして、奈良先端科学技術大学院大学学内ネットワークを利用する。図4.2に示すとおり、学内ネットワークと学外ネットワークの境界スイッチにおいてトラフィックミラーリングを行う。ミラーリングされたトラフィックを2方向に分割し、一方はフルダンプによりすべてのトラフィックを蓄積する。他方ではサンプリング計測によりトラフィックデータを蓄積する。

また、本実験におけるフルダンプとサンプリング計測はそれぞれ、以下の条件で行う。

- フルダンプ
  - tcpdump version3.9.3により取得
- サンプリング計測
  - sFlowを用いて計測
  - sFlow AgentとしてFastIron Edge Switch 2402を利用
  - sflowtool[252] version3.8を用い、pcap形式でパケットデータを記録
  - サンプリングレートは1/512
  - 取得されるパケットデータは先頭128 byteで切りつめられる

本稿で利用するsnortは、インシデントベースでアラートを発生させ記録するシステムである。通常では、どのようなトラフィックが流れたとしても、アラート発生までに関わったトラフィック量を知ることはできない。そこで、アラートを発生させるまでに検査したトラフィックのパケットをカウントする機能を追加した。

そして、アラートが発生した場合、それまでカウントしていたパケットカウントをアラートログに出力するよう修正した。一例を図4.3に示す。

また、アラートログ集計ツールsnortsnarfを、集計結果にカウントの集計をPacketsフィールドとして出力するように改造した。出力結果例を図4.4に示す。

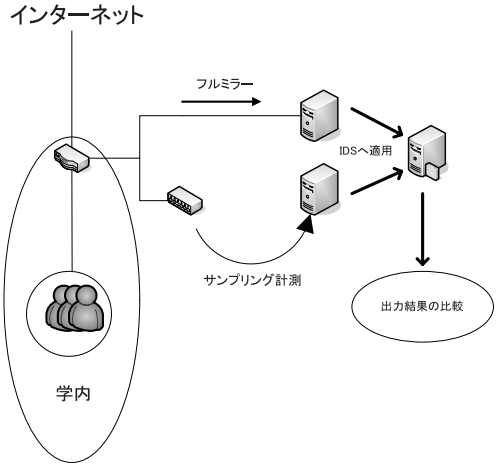
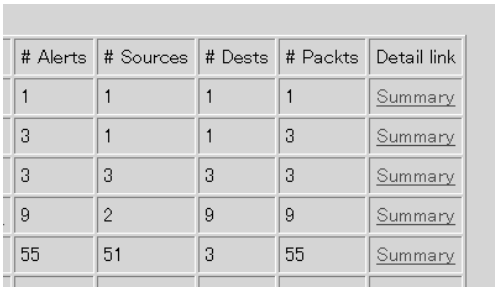


図 4.2. 実験環境

```
[**] [1:1852:3] WEB-MISC robots.txt access [**]
 [Classification: access to a potentially vulnerable web application] [Priority: 2]
09/13-14:42:17.255894 pkts:4 66.196.XXX.XXX:54879 -⟩ 163.221.YYY.YYY:80
TCP TTL:49 TOS:0x0 ID:42513 IpLen:20 DgmLen:244 DF
 ***AP*** Seq: 0x77953A79 Ack: 0x5E597324 Win: 0x16D0 TcpLen: 32
TCP Options (3) =⟩ NOP NOP TS: 244751500 2986396325
 [Xref =⟩ http://cgi.nessus.org/plugins/dump.php3?id=10302]
```

図 4.3. snortアラートログ例



# Alerts	# Sources	# Dests	# Packts	Detail link
1	1	1	1	Summary
3	1	1	3	Summary
3	3	3	3	Summary
9	2	9	9	Summary
55	51	3	55	Summary

図 4.4.　snortsnarf の出力結果例

4.3.2 実験結果

本稿の最終目標はトラフィックの傾向を把握したいということである。したがって、以下結果を百分率で示すこととする。フルダンプおよびサンプリング計測により記録されたトラフィックログデータに snort を適用させた際のアラート発生結果の割合を図 4.5 に示す。

図 4.5 各々のアラート数の割合、およびアラート内容を比較、検証した。グラフ中央付近ではフルダンプ、サンプリング計測ともに同程度の割合で検出できているものが存在することが見て取れる。同程度の割合で検出されたアラート名とフルダンプ、サンプリング計測による割合は以下のようなものであった。

- BAD-TRAFFIC same SRC/DST
  0.002% : 0.004%
- BAD-TRAFFIC IP Proto 103 PIM
  0.002% : 0.004%
- ICMP Destination Unreachable Communication Administratively Prohibited
  0.036% : 0.037%
- ICMP Echo Reply
  0.310% : 0.404%
- ICMP PING *NIX
  0.0280% : 0.229%
- ICMP PING BSDtype
  0.0280% : 0.262%
- ICMP PING NMAP
  0.0003% : 0.008%
- MS-SQL version overflow attempt
  0.0261% : 0.229%
- SNMP request udp
  0.0008% : 0.008%

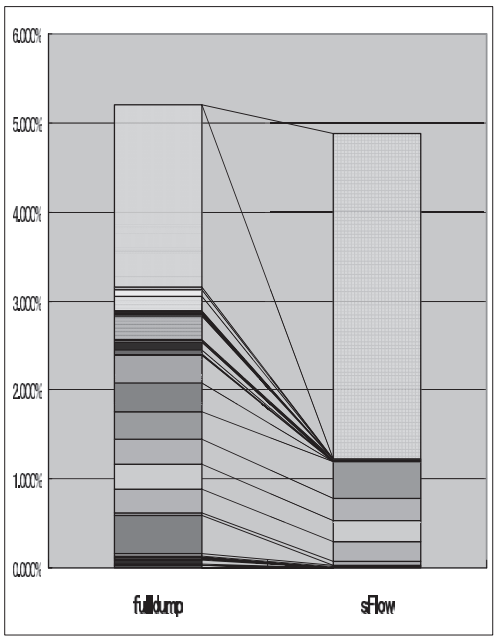


図 4.5.　適用実験結果

（誤差をどの程度許容するかという議論はもちろん必要であるが、本稿では割愛する。）

一方、フルダンプでは検出されているが、サンプリング計測ではまったく検出されていないアラートも多数見受けられる。逆にサンプリング計測にのみ大量に発生しているアラートも見受けられる。

そこで、それぞれの原因を追及するために snort をデバッグモードで動作させ、snort 内部におけるパケットの処理方法を追った。

その結果、以下のような問題点として分類できた。

一部の UDP パケットが不正パケットとして検出

サンプリング計測側にのみ “Short UDP packet, length field 〉 payload length” というアラートが大量に発生している。（図 4.5 サンプリング計測側最上段）サンプリング計測により取得されたトラフィックログデータは、パケット長が 128 byte より大きいパケットの場合、128 byte に切りつめられる。それにより、切りつめられたパケットに関しては、そのパケットのヘッダに記録されているパケット長フィールドと、実際記録されているパケット長が異なっている。そのため、取得されたパケットが snort の解析ルーチンにより不正なパケットとして検出されてしまう。

TCP ルールで不検出

実験結果のアラート内容を調べた結果、サンプリング計測側は TCP パケットによって検出されるルー

ルではまったくアラートが発生していなかった。サンプリング計測により記録されたトラフィックログデータ中にも確かに TCP パケットは含まれている。アラート対象になるパケットも確かに含まれている。

snort の動作検証の結果、パケットヘッダのチェックサムフィールドと実際に計算して求められるチェックサムが異なる TCP パケットは検査対象になっていないことが判明した。

パケットヘッダのチェックサムフィールドは、パケットのヘッダとペイロードを利用して計算される。パケット長が 128 byte より大きいパケットは、一部の UDP パケットが不正パケットとして検出される問題同様、128 byte にまで切りつめられてしまうため、チェックサムフィールドも実際記録されているパケットから算出されるものとは異なってしまう。そのため、snort は、チェックサム異常のパケットが取得されたと認識してしまう。

本来、TCP 通信はパケットが壊れていた場合、再送要求により正常なパケットが再送されることが期待される。snort も同様に、壊れた TCP パケットを観測した場合は、そのパケットは異常パケットとして解析ルーチンに回さず破棄し、再送されてくるであろう正常な TCP パケットを待つ。しかし、サンプリング計測では、128 byte より大きいパケットは切りつめられるため、正常なパケットであったとしても snort 内部でのチェックサム再計算結果と、記録されたパケットのチェックサムフィールドは一致しない。そのため、壊れた TCP パケットとして認識されてしまい、解析対象にならないことが判明した。

セッション情報を利用するルールで検出不可

snort のルールセット中に、“flow:established” という記述がある。これは、TCP のセッション情報を利用した検出を行うための記述方法である。snort のルールセットから攻撃パケットを生成する snot のようなツールによる、単発的な攻撃を無視するための策である。

ルールセットを調査したところ、TCP に関するルールのほぼすべてにこの記述がなされていた。フルダンプのトラフィックログデータであれば、正確にセッションを追うことができるため問題はほぼ発生しない。しかし、サンプリング計測では、取得されるパケットは間引かれている。そのため、セッションを正確に追うことが事実上不可能である。そのため、TCP のセッション情報を利用した検出には工夫が必要である。

4.3.3 問題点のまとめ

サンプリング計測で取得したトラフィックログデータに snort を適用させた場合に発生した問題点は、4.3.2 で述べた 3 点であった。

ここで、これらの問題をサンプリング計測の特性から考察してみる。サンプリング計測により得られるトラフィックログデータには、図 4.6 に示すような、2 つの大きい特性がある。パケットの切りつめとサンプリングによるパケットの間引きである。それぞれの特性と、4.3.2 の結果を比較する中で、それぞれの問題点はサンプリング計測で得られるトラフィックログデータの特性にのみ由来していることがわかった。

パケットの切りつめにより、得られるトラフィックデータの最大パケット長が 128 byte に切りつめられる。また、パケットの間引きにより、TCP に関してセッションが追えなかったり、ルールセットの記述に制約がでたりする。

したがって、本節で洗い出された問題点は、サンプリング計測の「パケットの切りつめ」に起因する問題、「トラフィックの間引き」のみに起因する問題として集約可能であると考える。

まとめると、以下の通りとなる。

(1) パケットの切りつめに由来
- 一部の UDP パケットが不正パケットとして検出
- TCP ルールで不検出

(2) トラフィックの間引きに由来
- TCP ルールで不検出
- セッション情報を利用するルールで検出不可

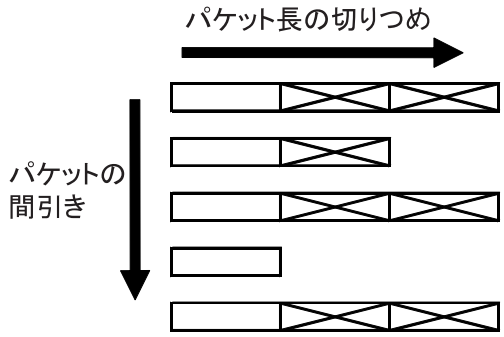


図 4.6. サンプリング計測の特徴

● ルールの記述に制約

そこで、次節では、それぞれの集約した問題点に対処し、正常に snort に解析させることができるようにするための手法とその評価実験、考察に関して述べる。

4.4 問題点の解決

本節では、4.3 節で述べたそれぞれの問題点を解決するための手法と実装、およびその考察を述べる。

4.4.1 パケットの切りつめに起因する問題

4.4.1.1 概要

サンプリング計測により取得されるトラフィックログデータは最大 128 byte に切りつめられることになる。そのため、パケット長フィールドの値と実パケット長の不一致や、チェックサムフィールドと実パケットから算出されるチェックサムとの不一致が起こる。そのため、UDP パケットが異常として検出されたり、TCP パケットが解析されないなどの問題が発生した。

4.3 節の実験結果から UDP パケットに関連するアラート部分のみを抽出した結果を図 4.7 に示す。サンプリング計測のみに多数のアラートが発生していることが分かる。

また、パケットの切りつめにより、トラフィックログデータに記録されているパケットから算出されるチェックサムの値と、パケットヘッダのチェックサムフィールドの値も異なる。そのため、snort は、チェックサム異常のパケットが取得されたと誤認識してしまい、そのパケット、特に TCP パケットを検査ルーチンに回さず破棄してしまう。

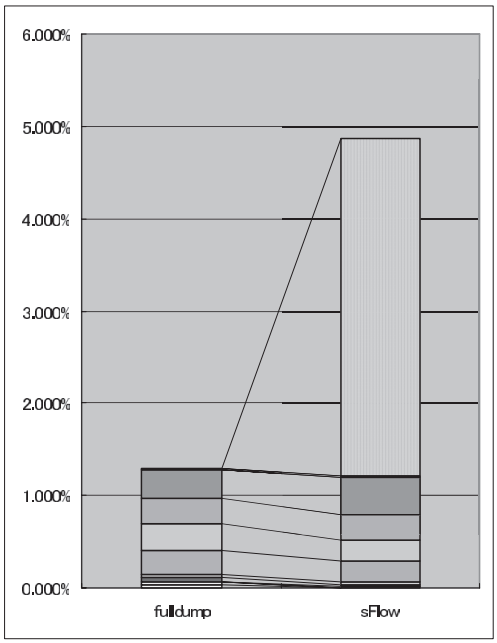


図 4.7.　ヘッダ修正前

これらの誤検知と不検知を解決するため、サンプリング計測により記録されたトラフィックログデータのパケットヘッダのパケット長フィールドおよびチェックサムフィールドを再計算し、実パケットから求められる値に修正した。

4.4.1.2 実装

実装は C 言語で libpcap を利用して実装を行った。サンプリング計測により取得されたトラフィックログデータを順に読み込む。記録パケット長が 128 byte より大きい場合、実パケット長になるようにヘッダのパケット長フィールドを修正した。また、同様にチェックサムの再計算を行い、チェックサムフィールドの値を修正した。

4.4.1.3 実験

ヘッダ情報修正前のトラフィックログデータと修正後のトラフィックログデータにそれぞれ snort を適用させ、そのアラート出力結果を比較した。また、切りつめられたパケットの snort 内部における扱いを調査した。

4.4.1.4 結果

ヘッダ長フィールドを修正トラフィックログデータに、snort を適用させた結果を図 4.8 に示す。図 4.7 と比較すると、不正パケットとして検出されていた部分が無くなっていることがわかる。その他検出に関して比較調査した結果、ヘッダのパケット長フィールド修正にともなう不具合は特に見られなかった。

また、チェックサムフィールドを再計算したパケットが snort 内部において、正常に解析ルーチンに回っていることを確認した。このことから、切りつめられたパケットに対して正常なパケット検査処理を行わせることに成功したといえる。

4.4.2 トラフィックの間引きに起因する問題

4.4.2.1 概要

snort の検出ルールの一つに TCP のセッション情報を利用する記述方法がある。これはトラフィック

フローを把握して、どのフローが攻撃トラフィックかを特定するためである。また、snot のような snort のルールから攻撃パケットを生成するような単発的な攻撃トラフィックを無視するという理由もある。フルダンプのようにすべてのトラフィックを取得可能な手法でトラフィックデータの蓄積をした場合は、すべてのセッション状態を把握することができる。

一方で、サンプリング計測により取得されるパケットは間引かれて記録される。そのため基本的にセッション状態を把握することは不可能であるといえる。そこで本稿ではこの解決策としてセッション情報を考慮せずに解析を行った場合にどのような結果になるかを実験、比較した。

本研究の最終目標はトラフィック傾向を把握するということである。詳細に攻撃などを検出したい場合は、セッション情報を考慮しない検出を行うと、不具合が発生する可能性が考えられる。しかし、単に傾向を把握したいという目標から、セッション情報は考慮しなくても深刻な不具合は発生しないのではないかと考えた。

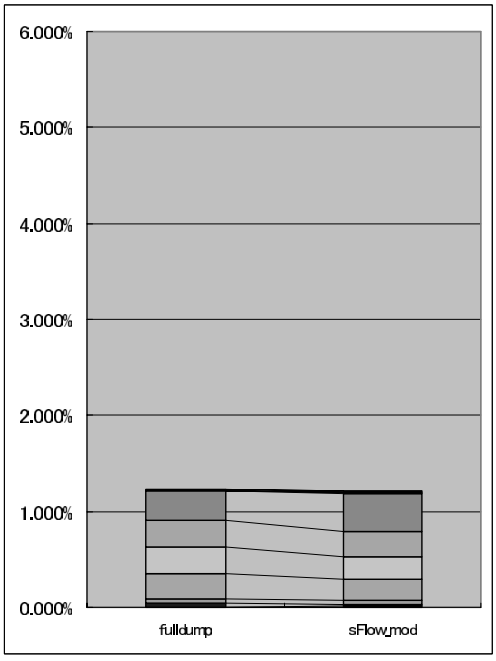


図 4.8. ヘッダ修正後

4.4.2.2 予備検証

セッション情報を利用したルールオプションは flow ディレクティブ以外に存在しない。しかし snort 内部でどのような状態保持を行っているかが不明であった。そこで、本項の実験を行う前に予備検証として snort 内部でセッション情報がどのように作成、保持されているかを調査した。

snort のソースファイルを調査した結果、snort 内部では以下の 5 つの属性を組としてセッション情報を作成していることが判明した。

- 送信元 IP アドレス
- 送信先 IP アドレス
- 送信元ポート番号
- 送信先ポート番号
- プロトコル番号

これらの情報を利用してハッシュを作成しセッション情報を作成する。セッション情報の中には、現在セッションがどのような状態にいるかを保持している変数が存在する。

セッションが確立した状態（flow = established）になるには、以下の条件が満たされた場合であることが分かった。

- TCP を利用したパケットである
- ACK フラグの立ったパケットが両 IP アドレスから観測された

この事象の検証用に、数秒間 HTTP 通信を tcpdump を用いて、フルダンプによるトラフィック計測を行い、その中から図 4.9 に示すパケットを抽出した。そして、それらのパケットに snort を適用させ、図 4.10 に示したルールで検出されるか否かを実験により検証した。

図 4.9 に示したパケットは、すべてに TCP のフラグ、ACK、PUSH のビットが立っている。

その結果、最初まったくセッション情報が保持されていない状態での GET コマンドを含むパケット（1 パケット目）ではアラートは発生しなかった。1 パケット目、2 パケット目解析後は、ACK フラグの立ったパケットが双方向に流れているため、snort 内部でセッションが確立した状態になる。そして、GET

No.	Time	Source	Destination	Protocol	Info
1	0.000000	163.221.170.103	163.221.170.104	HTTP	GET / HTTP/1.1
2	0.001664	163.221.170.104	163.221.170.103	HTTP	Continuation or non-HTTP traffic
3	0.003246	163.221.170.103	163.221.170.104	HTTP	GET /apache_pb.gif HTTP/1.1

図 4.9. セッション確立検証用パケット

```
alert tcp any any -⟩ HTTP_SERVERS$HTTP_PORTS
(msg:"GET COMMAND";flow:from_client,established;
 content:"GET"; nocase;)
```

図 4.10.　セッション確立検証用ルール

コマンドが含まれている 3 パケット目で、初めてアラートが発生することを確認した。

4.4.2.3 実験

4.4.1 でパケット長フィールド、チェックサムフィールドを修正してきた。このトラフィックログデータ使用し snort を適用させる。snort の標準状態ではセッション情報を考慮した検出を行う設定になっている。本項では、セッションを取り扱う snort のプリプロセッサである stream4 を無効化した状態でどのような検出結果になるか実験した。

デフォルトのルールセットでは、検出数が多種にわたり比較しにくくなるため、今回の実験用に図 4.11 に示すルールを用意した。トラフィックログデータの中から HTTP 通信の、さらに GET、POST メソッドを含んだパケットを検出するルールである。

```
alert tcp any any -⟩ HTTP_SERVERS$HTTP_PORTS
(msg:"GET COMMAND";flow:from_client,established;
 content:"GET"; nocase;)
alert tcp any any -⟩ HTTP_SERVERS$HTTP_PORTS
(msg:"POST COMMAND";flow:from_client,established;
 content:"POST"; nocase;)
```

図 4.11.　比較実験用ルール

4.4.2.4 実験結果

検出割合の比較結果を図 4.12 に示す。フルダンプによって取得されたトラフィックログデータ中に、GET メソッドを含むパケットは 0.20191%含まれていた。POST メソッドを含むパケットは 0.00051%含まれていた。

サンプリング計測によって取得されたトラフィックログデータ中に GET メソッドを含むパケットは 0.20024%含まれていた。POST メソッドを含むパケットは検出することができなかった。

以上の結果より、ルールの記述方法次第では、TCP に関するルールにおいても正常に検出可能であることが判明した。しかし、stream4 プリプロセッサを無効にしていることから、ステートフルな解析がまったくできないのは当然ながら、セッション情報を蓄積しないため、フラグメントされたパケットの解析もまったく行えない。snort はセッション情報を利用してフラグメントされたパケットを一定時間蓄積し、リアセンブルして解析できるような仕組みになっている。この機能がまったく利用できないため、新たな解決手法の確立が必要になると考えられる。

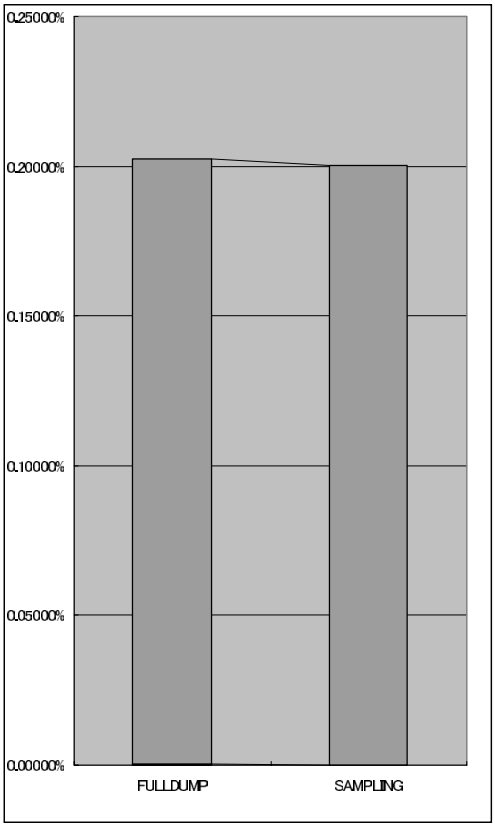


図 4.12.　比較実験結果

4.5 まとめと課題

本稿では、サンプリング計測とパケットアナライザの一つの IDS、特に snort に注目し、それらを組み合わせてトラフィック解析を行う場合に起こりうる問題点を洗い出し、原因を考察した。そして、それらの問題に対して、問題を解決できうる手法を提案し、実装、考察した。

洗い出された問題点には対応できたが、実験をしていく中で新たな問題も発生してきた。新たに発生した問題点は以下の 2 つである。

(1) ヘッダフィールド修正に依存する問題
(2) IP フラグメントされたパケットに関する問題

ヘッダフィールド修正に依存する問題は、ルールの記述においてヘッダ情報を利用、特に今回のパケット

長フィールドやチェックサムフィールドを利用する場合に問題が発生するおそれがある。例えば、dsize オプションを利用する場合に発生すると考えられる。dsize オプションはパケットのペイロード長を検査するルールである。今回の実験においては、アラート数に差は見られなかったが、今後、オプション等に関する扱いの検証も必要になってくると考えられる。

IP フラグメントに関する問題は非常に解決が困難であると考える。snort 内部では、フラグメントされたパケットは一定期間メモリ上に蓄積する。フラグメントパケットをすべて受信した後、パケットの再構築を行い再度解析を試みる。しかし、サンプリング計測により記録されたトラフィックログデータは、そのパケットが IP フラグメントされたものであったとしても、後続するフラグメントパケットを観測できる確率はきわめて低い。そのため、ルールの記述方法を工夫し問題の解決を図る、もしくは取得したパケット情報から前後のトラフィックを予測するなどの新たな解析手法の確立が必要になってくると考えられる。しかし、サンプリングレートに大いに依存するところがあり、今後把握したいトラフィック傾向に依存する適切なサンプリングレートの設定に関する議論も必要である。

また、先に述べた問題以外に、snort を利用してトラフィックの傾向をつかむことができるようなルールの記述手法の確立が必要である。本稿ではサンプリング計測により記録されたトラフィックログデータを、いかに IDS、snort に特化して解析させることが可能かという議論に終始してしまった。しかし、最終目標は IDS をトラフィックアナライザとして利用し、現状監視を行っているネットワークがどのような利用傾向にあるかを把握することにある。本稿の実験では HTTP トラフィックの中から GET メソッド、POST メソッドを含んだトラフィックのみを抽出するという簡単なルールしか記述することができなかった。ルールセットの開発は、トラフィック傾向を把握するためには最大の課題である。

第 5 章　sFlow によるネットワークトラフィック監視システムの提案

本報告書は DICOMO2005 に提出された論文「sFlow によるネットワークトラフィック監視システムの提案」をもとにしている。

5.1 はじめに

コンピュータネットワークの広帯域化にともない、ネットワーク上で利用されているアプリケーションの多様化が進んでいる。これにともない、健全で安全なネットワーク環境を提供する必要性が増しつつある。健全で安全なネットワークを維持するためには P2P ファイル共有ソフトによる違法性の高いコンテンツ転送の発見、ウイルスによる被害拡大の防止など要求が多様化している。ネットワーク運用者がこれらの通信を検出するには、ネットワーク型 Intrusion Detection System（IDS）を用いるのが一般的である。

ネットワーク型 IDS は通常、すべてのパケットを取得し、解析することによりネットワーク管理者が意図しないパケットかどうかを判断する。しかしネットワークの広帯域化、多様化によりネットワーク上を流れるパケット一つ一つを解析する事は困難になりつつある。こうした背景の中、各ネットワーク機器ベンダにおいては様々な角度からトラフィックを観測できる技術を提案、提供している。本学においては、2003 年 5 月より Cisco Systems 社が提案する NetFlow[37] を利用したネットワークトラフィック監視システムを構築し、実際に運用している [326]。さらに 2005 年 3 月より、InMon 社が提案する sFlow[251] を利用した計測を開始した。本稿では、sFlow を利用したネットワーク監視システムについて述べる。

以下 5.2 節では既存技術の問題点について述べる。5.3 節では我々が現在運用している sFlow を利用したネットワーク監視システムの設計について述べる。5.4 節では、我々の研究グループで開発した NetFlow を利用したネットワーク監視システムへの適応について述べる。5.5 節では NetFlow を利用したネットワーク監視システムの問題点とその解決に向けた基礎実験について述べる。

5.2 既存技術

本節では、既存のネットワークトラフィック計測の技術を紹介する。

5.2.1 MRTG

Multi Router Traffic Grapher（MRTG）[194] はルータや計測機器より取れるデータを可視化するツー

ルである。長期にわたり計測が可能であり、記録するデータを集約するため、保存に必要な領域も少ない。また、簡単に導入できるため、広く利用されている。しかしながら、過去のデータほどその粒度は荒く、詳細なデータの分析は困難である。

5.2.2 tcpdump

Tcpdump[269] は一般的なコンピュータで動作するネットワーク計測ツールである。導入されたコンピュータのネットワークインタフェースに到達するパケットを表示・記録保存できる。すべてのパケットを記録するため、保存されるデータ容量が大きくなる。特に、近年の数 Gbps オーダーから数十 Gbps オーダーのネットワークリンクの計測に対しては、パケットの取りこぼしなどが発生する場合もある。また、計測時間にあわせたデータ保存領域が必要になり、長期的な計測に用いるには現実的に難しい。

5.2.3 NetFlow

NetFlow は Cisco Systems 社が提案するサンプリングベースのネットワークトラフィック計測技術である。同社のルータや各ベンダからも対応機器が販売されており、それらを導入すれば利用できる。ルータに実装されているため、ネットワークの基幹部分での計測が可能である。NetFlow の情報は、ルータのルーティング用のキャッシュから生成している。このため機器への負荷を低く抑えている。また、NetFlow ではトラフィック情報を TCP のコネクションの概念に近い独自の “フロー” と呼ばれるデータに集約し、トラフィック情報を報告する。広帯域に対応可能である反面、取得できるデータが Layer 2 から Layer 4 までのヘッダ情報の一部に限られるため、トラフィックの解析には工夫が必要である。

5.2.4 sFlow

sFlow は InMon 社が提案するトラフィック計測技術である。Foundry Networks 社やアラクサラネットワークス社のルータなどに実装されている。sFlow の情報は、ルータを通過するパケットを、あらかじめ設定されたサンプリングレートで観測する。観測されたパケットはルータで変更されることなく記録される。つまり、観測されたパケットデータは実質 tcpdump のデータと同等のものといえる。観測されたパケットは、sFlow データグラムというパケットに格納されルータから送信される。sFlow データグラムには、sFlow のヘッダ、および、サンプリングされたパケットの上位 256 byte が格納されている。また sFlow データグラムには、上記のデータ以外にも、Simple Network Management Protocol（SNMP）を利用したときに取得できるルータの各統計情報も送信するため、SNMP エージェントを利用せずに機器の実トラフィック量などを知ることもできる。

NetFlow では、各 Layer のヘッダ情報の一部しか取得することができない。そのため、その通信内容を含めたデータ解析ができなかった。しかし、sFlow ではルータを通過するパケットをそのまま記録するため、既存の解析手法に適応するなどの汎用性が期待できる。

5.3 sFlow を利用したネットワーク監視システムの設計と構築

本節では、sFlow を利用したネットワーク監視のシステムの設計に関して述べる。

5.3.1 用語定義

本項では、本稿で用いる用語の定義について述べる。sFlow を利用したトラフィック計測ができるルータを、エージェントと呼ぶ。エージェントが送信する sFlow に関するパケットを sFlow データグラムと呼ぶ。エージェントからの sFlow データグラムを受信し、解析・蓄積する機器をコレクタと呼ぶ。

5.3.2 収集データ

本システムにおいて、我々はエージェントが送信する sFlow データグラムのうち、観測パケット情報が含まれるデータグラムを取得・解析し、データベースへ保存する。機器情報が含まれる sFlow データグラムは収集対象としていない。収集される sFlow データグラムのフォーマットを図 5.1 に示す。また、各項目の説明を表 5.1 に示す。

sFlow データグラムのバージョンは 2、4、5 が存在する。本システム上で利用されているエージェントが送信する sFlow データグラムのバージョンは 2 および 4 である。収集対象としている sFlow データグラムでの両者の差は無いため、両バージョンとも問題なく収集できる。

我々は、sFlow データグラムの各項目をデコードし、データベースへ蓄積するためのソフトウェアを

開発した。sFlow データグラムにはデータグラム中に含まれる観測パケットが実際観測された時刻は記録されていない。そのため本ソフトウェアでは、コレクタが sFlow データグラムを受信した時刻の記録を行う。その時刻を、記録されているパケットが発生した時刻とし、データベースへ保存する。この時、実際のパケットがルータを通過した時刻と、データベースに記録される時刻には若干のずれが生じる。しかし、そのずれはたかだか 1 秒未満である。

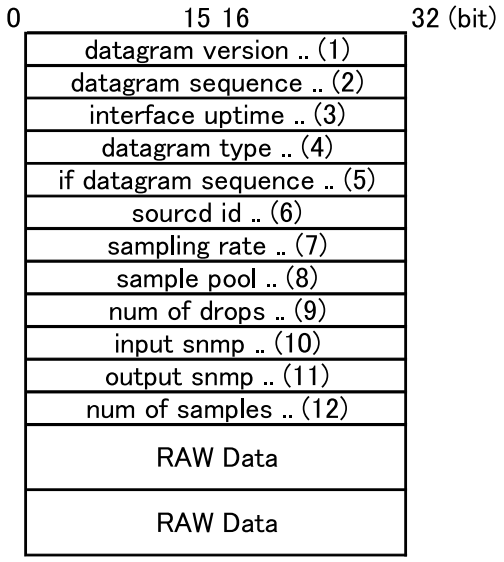


図 5.1. sFlow データグラムフォーマット

表 5.1. 各項目の説明

No	説　明
1	sFlow データグラムのバージョン
2	データグラムの通し番号
3	sFlow データグラムが作成されたインタフェースが起動してからの経過時間（現在時刻ではない）
4	sFlow データグラムの種類
5	計測機器に複数のインタフェースがある場合の、各々のインタフェースが作成した sFlow データグラムの通し番号
6	sFlow データグラムが作成されたインタフェース情報
7	サンプリングレート
8	取得対象となったパケット数
9	取りこぼしたパケット数
10	取得パケットが受信されたインタフェース番号
11	取得パケットが送信されたインタフェース番号
12	sFlow データグラム中に含まれる観測パケットの数
以下、取得されたパケット上位 256 byte までの RAW データ（イーサネットフレーム、FDDI フレームなど）	

5.3.3 データベース設計

本項ではデータベースの設計について述べる。本学のネットワークは、イーサネットにより構成されているため、データベーステーブルは以下 8 つを作成した。

a) header テーブル
　表 5.1 の 1、2、3 のデータを格納する。5.2.4 項で述べたように、sFlow データグラムにはルータにより観測されたパケットデータ以外に、計測機器自身の情報が含まれるデータグラムもある。それらすべての sFlow データグラムに共通してこの 3 項目のヘッダが付加されている。今後の拡張を考慮して、別テーブルとした。

b) flow sample テーブル
　表 5.1 の 5 から 11 までの項目を格納している。

c) ethernet ヘッダテーブル

d) IPv4 ヘッダテーブル

e) IPv6 ヘッダテーブル

f) TCP ヘッダテーブル

g) UDP ヘッダテーブル

h) payload テーブル

c から h のテーブルへは sFlow データグラムに含まれるパケットの各ヘッダおよびペイロード部を保存する。それらのデータは、データベースへ蓄積するソフトウェアで独自にデコードされる。

5.3.4 計測環境

本学では、学外との境界に HITACHI の GS4000 を導入している。このルータは sFlow エージェントとして動作する。学内から学外への通信はこのルータを経由することになり、学内からの通信、学外か

表 5.2. コレクタのスペック

CPU	Pentium III 1.2 GHz
Memory	512 MB
Hard Disk	80 GB
NIC	100 Mbps
OS	Slackware Linux 10.1
RDBMS	MySQL 4.0.23a

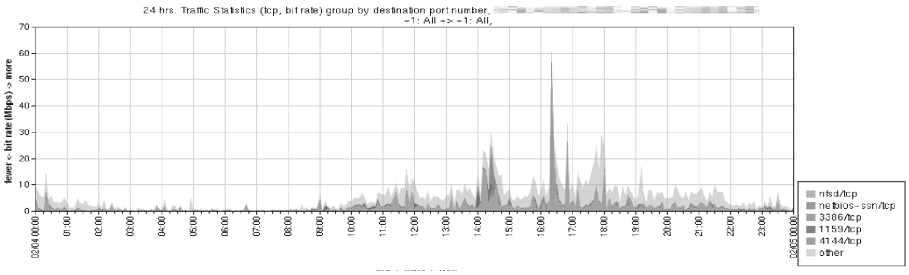


図 5.2.　ポート別トラフィック量の推移

らの通信を観測することができる。また、学内にも数台、sFlow に対応した機器を設置しており、各々 sFlow エージェントとして設定されている。

sFlow コレクタには一般的なコンピュータを利用している。構成を表 5.2 に示す。

5.4 NetFlow を利用したネットワーク監視システムへの適応

本節では、本学で 2003 年 5 月より行っている NetFlow を利用したネットワークトラフィック監視システム [326] に sFlow を利用したネットワークトラフィック監視システムの適応を述べる。

sFlow を利用したネットワーク計測によって、データを 1 ヶ月間貯めた結果、レコード数はおよそ 240 万レコードであった。データ容量はおよそ 700 MB であった。その 1/4 程度は payload 部のデータである。1 年間後のデータ容量の概算は 10 GB 程度であることから、sFlow を利用したネットワーク計測でも長期間の計測が可能であるといえる。

図 5.2 は sFlow を利用したネットワーク監視システムで得られたデータを、NetFlow を利用したネットワーク監視システムで利用しているトラフィック可視化ツールに適応させた例である。NetFlow を利用した計測により蓄積しているデータベースのテーブル構成とは異なるものの、取得できるデータは、Layer 2 から Layer 4 のヘッダと同等のものである。そのため SQL クエリを若干変更させるのみで容易に適応させることができる。

図 5.3 は、2005 年 5 月 2 日に TCP 6667 ポートを利用して通信しているホストで、通信パケット数の多い順に抽出した例である。6667 ポートは通常 IRC で利用するポートである。しかし bot[21] の操作に利用される事も多い。通信数が極端に多いホストは bot の感染疑いがある。図 5.4 は bot が活動している疑惑のあるホストを検出した例である。図中

src addr	dstport	pkt_count	estimation
163.221.	6667	48	393216
163.221.	6667	12	98304
163.221.	6667	9	73728
163.221.	6667	5	40960
163.221.	6667	5	40960
163.221.	6667	5	40960
163.221.	6667	3	24576
163.221.	6667	2	16384
163.221.	6667	2	16384
163.221.	6667	1	8192
163.221.	6667	1	8192
163.221.	6667	1	8192

図 5.3.　bot 感染疑いホストの検出

src addr	dstport	pkt_count	estimation
163.221.	1433	388	3178496

図 5.4.　bot 活動疑惑ホストの検出

の pkt_count は実際観測されたパケット数であり、estimation はサンプリングレートから求めた実際の通信パケット数の予測値である。

実際、図 5.3 で通信パケット数 1 位に抽出されたホストは、w32/sdbot.worm.gen という bot に感染していた。しかしその他のホストに関しては false positive であった。

5.5 NetFlow を利用したネットワーク監視システムの問題点とその解決

前節で述べたように、NetFlow を利用したネットワーク監視システムへの適応は可能であった。しかし、このシステムの問題点は、ヘッダ情報のみを用いて異常検出を行うため、本来異常ではない通信も異常と判断してしまう false positive の率が高いことにある。そこで、前節で述べたような異常の疑いのある通信データに対して IDS やウイルススキャンプログラムを適応することができれば、false positive

率の低下が期待できる。

本節においては、sFlow を利用したネットワーク監視システムで蓄積されるデータを IDS の一つである snort[260] へ適応させるための基礎実験と結果について述べる。

5.5.1 実験環境

本実験の実験ネットワークは図 5.5 に示すように、仮想内部ネットワークを定義し、その内部のノードに対して異常パケットを送信する。

エージェントには、Foundry Networks 社の FastIron Edge Switch2402 を用い、仮想ネットワーク側のイーサネットインタフェースで sFlow を計測するように設定した。

次に、異常パケット生成マシンのスペックを表 5.3 に示す。コレクタは表 5.2 に示したスペックのものと同じものを利用する。

異常パケットの生成には、hping2[110] を利用する。このツールは様々なパケットを生成することができ、firewall の設定のテストにも利用される。

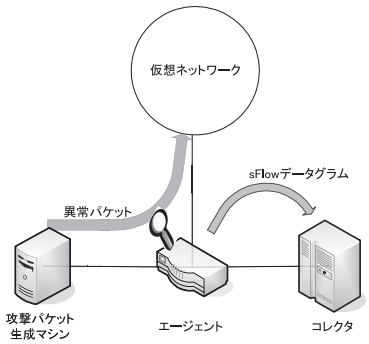


図 5.5. 実験トポロジ

表 5.3. 異常パケット生成マシンスペック

CPU	Pentium4 2.53 GHz
Memory	1024 MB
Hard Disk	80 GB
NIC	100 Mbps
OS	Linux
ソフトウェア	hping2

IDS には、snort version 2.20 を用いる。snort は、フリーで利用することができ、事前に用意されているルールパターンに該当するトラフィックをリアルタイムに検出することができる。

今回の実験では、snort のルールパターンは 2004 年 11 月現在の最新版を利用した。それらのルールパターンに加え、特定の文字がパケットに含まれた場合にアラートを発生させるルールを、実験用に独自で追加した。図 5.6 に追加したルールを示す。

5.5.2 実験方法

hping2 を利用し、仮想ネットワーク内のクライアントに対して以下の 4 つのパターンのパケットを送信する。

1. TCP 宛先 port 番号 0 に対するパケット
2. 1 と同様のパケットで、パケット長が 256 byte より大きいパケット
3. ペイロード部分に特定の文字列を含めた、パケット長 256 byte 以下であるパケット
4. 3 と同様のパケットで、パケット長が 256 byte より大きいパケット

sFlow により記録されるパケットはその仕様上、256 byte に切りつめられる。そのため、本実験ではパケット長が 256 byte 以下の場合と 256 byte より大きい場合に分けて実験を行った。

以上 4 つの場合に分け、それぞれのパケットを一定時間送信し、エージェントに sFlow データグラムを送信させる。コレクタのデータベースに十数レコードが蓄積されたところで送信を停止する。データベースにアラートの対象となるパケット情報がいくつ蓄積されたかを確認した後、データベースの情報からパケットを復元し snort に適応させる。

これらのパケットを直接 snort に適応させた場合、パターン 1、2 のパケットは、“BAD-TRAFFIC tcp port 0 traffic” として検出される。パターン 3、4 は独自に追加したルールにより検出される。

```
alert tcp $EXTERNAL_NET any <> $HOME_NET 81
(msg:"appeared THE ALERT!!"; flow:stateless; content:
"alert_alert"; classtype:misc-activity; sid:9999; rev:4;)
```

図 5.6. 独自に追加したルール

5.5.3 実験結果

表 5.4 に実験結果を示す。パターン 1 から 3 のパケットでは、データベースに蓄積された異常パケットに該当する該当レコード数と snort に適応させた場合のアラート数は一致した。しかしパターン 4 のパケットではアラートは発生しなかった。

以上の結果から、ヘッダのみを捜査するルールでの検出では、sFlow によりパケットが途中で切られていたとしても検出に問題がないことがわかった。しかし、ペイロードを捜査するルールでは、そのまま適応したとき、パケットサイズが 256 byte より大きい場合、検出することができなかった。理由としては、パターン 4 のパケットデータをデータベースより復元すると、そのパケットの IP ヘッダに記録されているデータ長と、実際 snort に適応されたデータ長が異なる。snort はこのパケットをフラグメントされたパケットと判断してしまうようである。このため snort は、本来、後に続くべきパケットの到着を待つため、判断を見送る。しかし、実際はフラグメントが行われていないので、待ってもパケットが到達することはない。そのため、最終的に判断できずにアラートが発生しないと考えられる。

そこで、再度ヘッダ情報の修正を行ってパターン 4 の情報を復元し snort に適応させた。IP ヘッダのデータ長を実際のデータベースに記録されているデータ長（TCP ヘッダ長 + ペイロード長）に変更し復元した。この場合では、データベースに記録されたアラート対象のレコード数である 12 のアラートが発生した。

本節では、蓄積されたデータを既存 IDS に適応させられることを示すことができた。また、データに若干修正を加えるのみで、正常にアラートが発生しなかったレコードに対してもアラートを発生させることができることを示せた。

表 5.4.　実験結果

実験パターン	アラート対象レコード数	実アラート数
パターン 1	13	13
パターン 2	11	11
パターン 3	12	12
パターン 4	12	0

5.6 おわりに

本稿では、我々が開発した NetFlow を利用したネットワーク監視システムに対し sFlow を利用したネットワーク監視システムが適応可能であることを示した。また、NetFlow を利用したネットワーク監視システムの問題点を補うために、sFlow で取得可能なパケットデータを利用した IDS による異常トラフィック検知を行うための基礎実験を行った。これを既存の解析手法と組み合わせることで、問題点であった false positive 率の減少が期待できる。

しかし、本稿においては、単一パケットを捜査するだけでアラートを発生させることができる単純なルールを適応させたのみで、セッション情報を利用するルールでは評価していない。今後 bot などの新しい脅威への対応が必要である。

課題として、

- セッション情報を利用するルールへの適応手法の確立
- 蓄積されたデータの復元手法毎、計測サンプリングレート毎による検出可能性、精度の調査、データ操作による副作用の検証
- 他の検出手法に適応可能な、汎用性のあるデータ復元手法の構築

などが挙げられる。

第 6 章　おわりに

roft ワーキンググループは、まもなく設立から 2 年が経過する。この中で、フローベーストラフィック計測の認知度上昇への貢献、取得したトラフィック情報の解析手法や実運用への応用事例の公開について、一定の成果を出すことができた。今後は、よりファンダメンタルな部分として、目的に応じたサンプリングレートの調査などを重点的に行っていく必要があると考えている。