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

斎田佳輝[†] 森島直人[††] 砂原秀樹[††]

コンピュータネットワークの重要性が高まる中，管理者は管理するネットワークがどのように使われているかを把握する必要性が高まってきている．近年の広帯域ネットワークに対応するため，トラフィックを間引きながら取得する計測手法（以下，サンプリング計測という）が用いられている．また，多様化するネットワークトラフィックを解析するために Intrusion Detection System（IDS）などのパケットアナライザが用いられている．それらの技術はそれぞれ独立して用いられることが多く，組み合わせて利用する際の問題点は明らかになっているとは言い難い．そこで，本稿ではまず，トラフィックのサンプリング計測を行いながらも IDS を用いてネットワーク利用傾向を把握したい場合に発生する問題点の洗い出しを行った．本稿では，IDS として snort を利用した．またそれら問題に対して，サンプリング計測データに基づき，正確にネットワーク利用傾向を把握できる手法構築に関して議論した．

The grasping method of network traffic trend with sampling measurement

Yoshiaki Saita,[†] Naoto Morishima[††] and Hideki Sunahara[††]

As computer network is becoming an important infrastructure, administrators need to grasp how their network to be used. We use the method of sampling measurement because of broadband network, packet analizer to analize network traffic that becomes diversified in recent years. These methods are generally used by itself. So we think there are some problem when used in combination. In this paper, We clarify the problems when we grasp the network traffic trend with sampling measurement method. And We discuss how we grasp the network traffic trend.

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 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

† 奈良先端科学技術大学院大学
NARA Institute of Science and Technology
〒 630-0192 奈良県生駒市高山町 8916-5
奈良先端科学技術大学院大学 情報科学研究科
TEL: (0743)72-5163 FAX: (0743)72-5149

†† 奈良先端科学技術大学院大学
NARA Institute of Science and Technology
〒 630-0192 奈良県生駒市高山町 8916-5
奈良先端科学技術大学院大学 情報科学センター
TEL: (0743)72-5148 FAX: (0743)72-5149

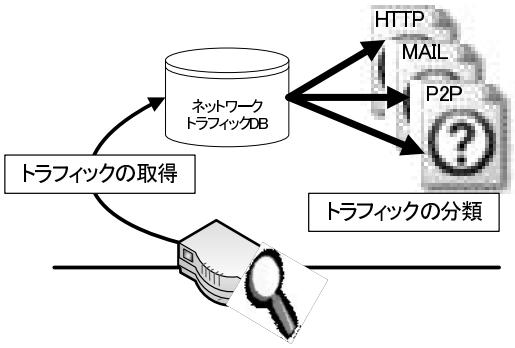

図 1 トラフィックの計測と分類
Fig. 1 traffic monitoring and classification

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．

2. 技術比較と問題点

本章では，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．

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．

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[5] 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．1.5Mbps 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，130GB 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．

2.1.2 サンプリン計測手法

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[1] や InMon 社が提案する sFlow[2] 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．

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．

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．

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．

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[4] 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．

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．

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，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[6)7)8)] 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．

そこで次章では，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たすめの実験を

行い，考察する．

3. 問題点の抽出

本章では，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．

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 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[3] 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 128byte 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 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 に示す．

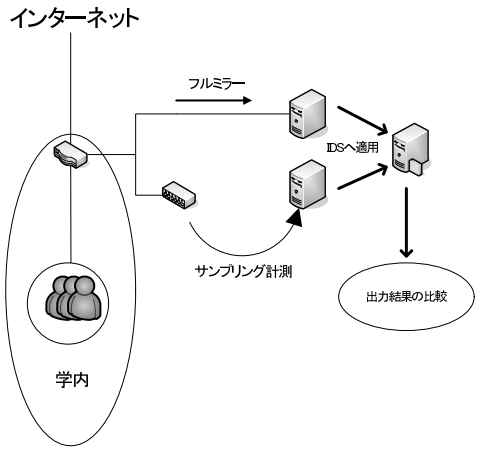

図 2 実験環境
Fig. 2 experimental environment

# 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 snortsnarf の出力結果例
Fig. 4 modification of snortsnarf output

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 5 に示す．

図 5 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

```
[**] [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 3 snort アラートログ例
Fig. 3 example of snort alert log

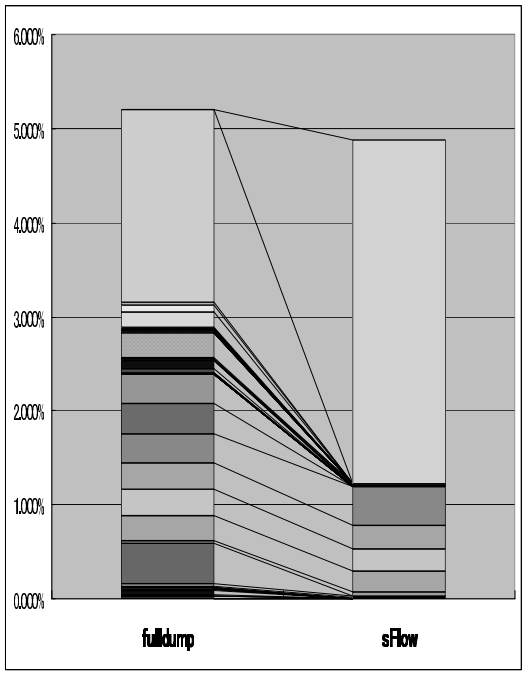

図 5 適用実験結果
Fig. 5 snortsnarf output result

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%

（誤差を同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 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 128byte より大きいパケットの場合，128byte 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 128byte 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，128byte 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，再送さてくるであろう正常な TCP パケット待つ．しかし，サンプリング計測では，128byte 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．

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，3.2 で述べた 3 点であった．

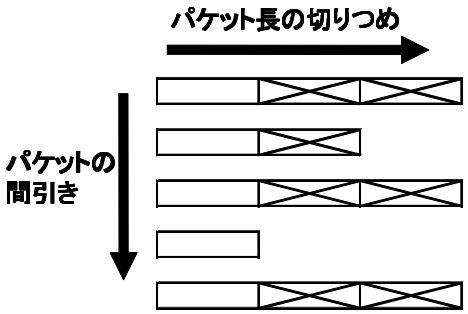

図 6 サンプリング計測の特徴
Fig. 6 feature of sampling method

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 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，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 128byte 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，本章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
- ルールの記述に制約

そこで，次章では，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. 問題点の解決

本章では，3 章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．

4.1 パケットの切りつめに起因する問題

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 128byte 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．

3 章の実験結果から 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 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．

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．

4.1.2 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 128byte 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．

4.1.3 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．

4.1.4 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 8 に示す．図 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．

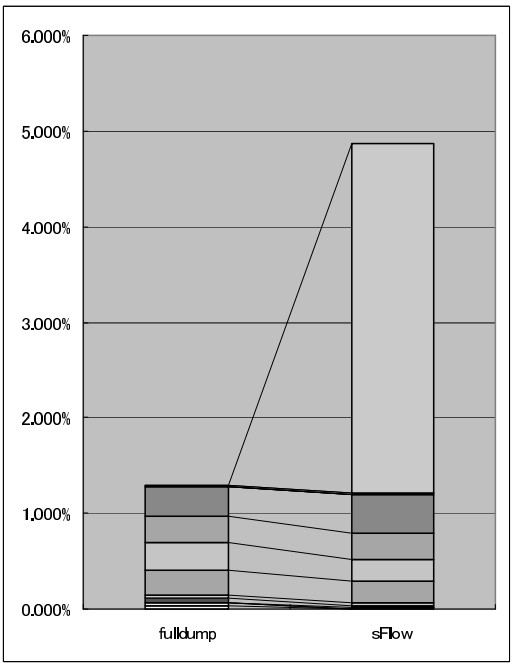

図 7　ヘッダ修正前
Fig. 7　BEFORE header field modification

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．

4.2 トラフィックの間引きに起因する問題

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．

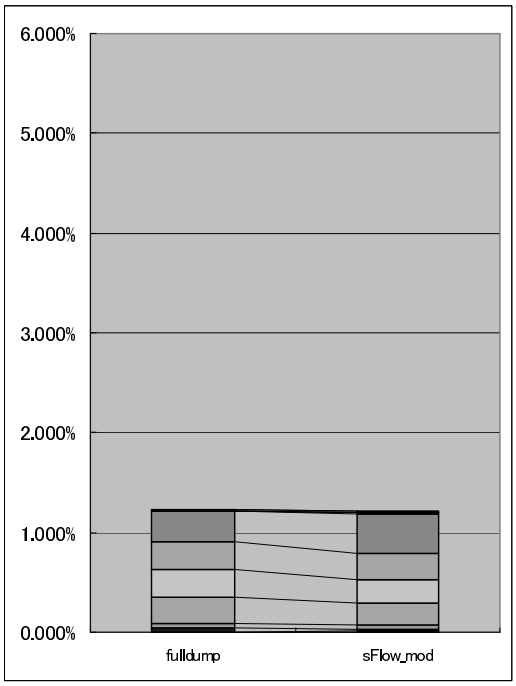

図 8 ヘッダ修正後
Fig. 8 AFTER header field modification

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．

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，本節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

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

図 10 セッション確立検証用ルール
Fig. 10 test rule set for session established test

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 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 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 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 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．

4.2.3 実　　験

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．本節では，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 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．

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 9 セッション確立検証用パケット
Fig. 9 test packet for session established test

```
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 11 比較実験用ルール
Fig. 11 detection test rule set with fulldump and sampling

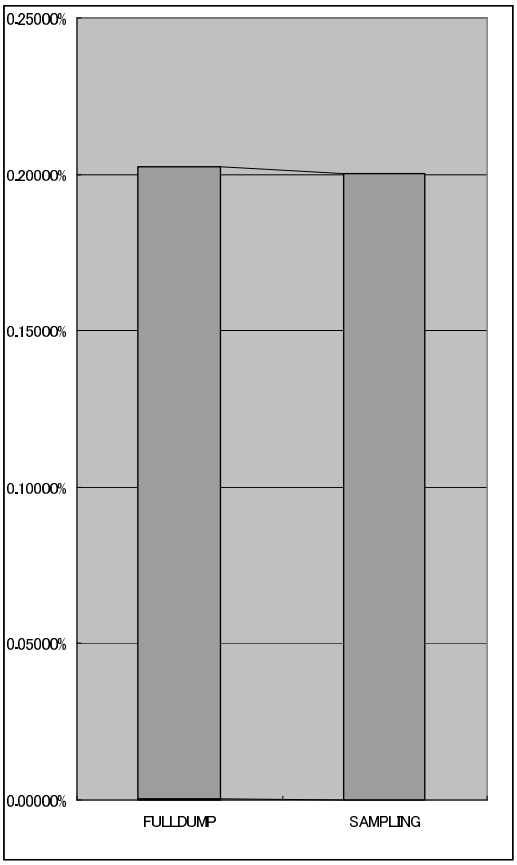

図 12 比較実験結果
Fig. 12 detection test result

4.2.4 実験結果

検出割合の比較結果を図 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．

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．

参 考 文 献

1) NetFlow，http://www.cisco.com/warp/public/732/Tech/nmp/netflow/index.shtml
2) sFlow.org，http://www.sflow.org/
3) sFlow Toolkit, http://www.inmon.com/technology/sflowTools.php
4) snort，http://www.snort.org/
5) tcpdump，http://www.tcpdump.org/
6) 山崎 康広, 下西 英之, 村瀬 勉. サンプリング計測でのパケットロス率推定手法の提案. 電子情報通信学会. Vol.2004, No.154-163, pp.25-30, 2004.
7) 上山 憲昭. サンプリング情報を用いた高レートフローの特定化, 電子情報通信学会, Vol.2004, No.65-82, pp.25-30, 2005.
8) 藤井 聖，中村 豊，藤川 和利，砂原 秀樹. 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，電子情報通信学会, Vol.J88-B, No.10, pp.1922-1933, Oct. 2005.