既存手法を用いたIPトレースバックの実装とネットワークエミュレータでの評価

2004MT023 平林 裕輝　　2004MT074 小川 貴大
指導教員　後藤 邦夫

1　はじめに

近年問題となっている, 不正アクセス “DoS/DDoS 攻撃” は, 直接的で単純な攻撃であるために, 確実な対策方法が少ない. さらに攻撃パケットの発信源アドレスが偽造された場合, 発信源を正確に特定できないことが対策を困難にしている. この対策方法の一つとして, [1] で挙げられているように, 発信源アドレスの偽造パケットの発信源追跡技術 (IP トレースバック技術) の提案が必要とされる.

IP トレースバックとは, 発信源 IP アドレスが偽造されていたとしても, 攻撃パケットが中継したルータからの報告を元に逆探査することによって発信源を突き止めることである. 発信源を突き止めることができれば, 管理者への通報やパケットフィルタリングの設定などの対処が可能となる. 既存の手法として, リンク検査方式, 逆探知パケット方式, マーキング方式, ロギング方式などがある.

本研究では, これらの既存手法の一つであるロギング方式を実装し, 実際のネットワークでも使える IP トレースバック機構の実現を目的とする. ロギング方式を選択した理由は 3 節で説明する. また, 実際のネットワークでは実験環境の構築などが難しく実験が困難となるため, ネットワークエミュレータを用いて実験する. これにより, ネットワークモデルを自由に変えることができ, 様々な条件下で性能評価できる. 評価基準は, ルータの数や攻撃パケットを流す割合などを変更した時の, それぞれの場合におけるトレースバック成功率やトレースバックに費した時間などを測定し, 実ネットワークで使用できる水準を満たしているかを検証することである.

なお, 平林は主に実験環境構築を担当し, 小川は主にプログラム作成を担当した. また, 性能評価は共同で行った.

2　既存の IP トレースバック方式

本節では, 既存の IP トレースバックの手法を, [1][2] より説明する.

- リンク検査方式
  被害者の最寄りのルータから攻撃の上流となるリンクを特定し, 隣接ルータへと順にたどっていくことで発信源を特定することができる. しかし, 攻撃されている期間中にしか逆探知することができない.
- 逆探知パケット方式
  各ルータを通過するパケットに, 逆探知するために必要な情報を一定の確率で別のパケットにおさめ, 被害者の元へ届ける. それを収集し分析することにより, 発信源を特定することができる. しかし, 別パケットを生成する確率によってはトラフィックの増大などの問題がある.
- マーキング方式
  各ルータにおいて攻撃パケットのある部分に特徴のある情報を一定の確率で埋め込み, その情報を持つパケットを解析することにより発信源を特定することができる. この方式によるトラフィックの増大は発生しない. しかし, 攻撃者が偽造マークを生成した場合, 伝送経路の逆探知が困難になる.
- ロギング方式
  ネットワーク上の要所にある記録装置がパケットを特定するための特徴情報を記録しておき, 被害者側で受信したパケットと記録した特徴情報を照合する. パケットの記録はハッシュ関数を用いて効率良く記録する. 記録装置を保持する手間とコストのかかることが問題点と言える.

3　システムの提案

本節では, 本研究で提案するシステムにおける, 実験環境とネットワークモデルを示す.

3.1　実験環境

本研究では, 実際のネットワークでも使えるトレースバック機構の実現を目的とする. これをネットワークエミュレータ環境で動かす方法は以下の 2 通りである.

1. ネットワークエミュレータのルータ内でのモジュール作成.
2. ネットワークエミュレータ外部で実現し, エミュレータからはトレースバックに必要なデータだけを受け取る.

前者の 1 において, 実際のネットワークで用いる場合, ルータにトレースバック処理を行うためのモジュールを組み込むことができれば再利用可能となるが, 専用 OS のルータでは組み込むことはできないため実現は難しい. 一方で後者の 2 において, トレースバック処理を行うホストはルータとは別のホストであるため, そのまま利用可能であると言える. つまり, 2 の方が実際のネットワーク環境に適していると考えることができる. しかし, リンク検査方式, 逆探知パケット方式, マーキング方式はルータ内で処理を行うものであり, 2 で実現することは困難である. これに対し, ロギング方式はどちらの

場合でも実現できるので, 実装しやすいという観点から他の手法より優れていると言える. したがって, 我々は 2 の方法でも実装しやすいロギング方式を用いることとする.

以上のことを考慮したネットワーク構成は, **図 1** のようにする. ホスト A を攻撃ホスト, ホスト B を被害ホストとし, ホスト A からホスト B への一方向に攻撃パケットを流す. ただし, トレーサはエミュレータ用 PC の外部プログラムとする. 単純な構成であるので実験しやすいという利点もある.

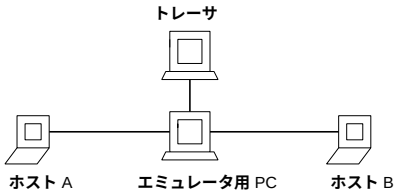

図 1　ネットワーク構成

3.2　ネットワークモデル

本研究では, GINE [3] を用いてトレースバックの実験をする. そのために作成する仮想ネットワークのモデルを**図 2** に示す.

GINE では, ルータの数や配置などが自由に設定でき, MAC アドレスや IP アドレス等の通信に必要な情報も設定できる. また, 前提条件として, トレーサ間の通信は確約されているものとし, 各々のトレーサは, トレーサの IP アドレスや位置関係などの通信に必要な情報はあらかじめ知っているものとする. GINE 内部は, 複数

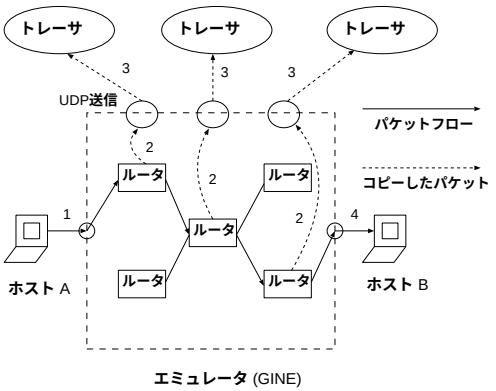

図 2　ネットワークモデル

のルータで構成され, GINE 外部に複数のトレーサが配置されている. トレーサはエミュレータ用 PC と同じ単一ホストで使用ポートを変えることで複数起動できるので, ルータの数に合わせてトレーサの数も増減する.

本システムでは, GINE 内部のルータを変更して, ルータに送られてきたパケットをコピーし, そのパケットをそのままフレームに収め, UDP 送信によって外部トレーサへ通信できるようにした. トレーサでは, GINE 内部のルータから送られてきたデータをフレームから取り出して, 必要な情報を抜き出している.

また, 本システムは通常時と追跡時に分けている. 通常時, 外部トレーサでは, GINE ルータから受け取った情報を受信し, その情報をハッシュ値をキーにして保存し続ける. 追跡時には, 被害ホストから情報を受け取り, 追跡処理を行う. また被害ホスト側では, 通常時には GINE を通して攻撃ホストから送られてくるパケットをキャプチャして, そのパケットのハッシュ値をとり, 保存していく. 追跡時は, 追跡命令をトレーサに送信してやりとりする.

なお, 実際のネットワークでトレーサを使用するときは, パケットのデータはルータから UDP 通信により受け取るか, ルータの隣に設置してパケットキャプチャで入手する, などの方法が考えられる.

4　システムの実現

本節では, 本研究で実装するロギング方式の詳細について [1], [2] を参考にして説明をする.

4.1　ハッシュ値の取り扱い

初めに, ハッシュ値の管理方法について説明する. トレーサや被害ホストではハッシュ値をキーとして通信記録を保存するため, パケットの特定部分を取り出して変換する必要がある. なお, 変換する部分においては, ルータを中継する時にパケット中で値が不変であることが第一条件である. TTL やヘッダチェックサムなどは変わってしまうので, これらを除外した特定部分を**表 1** で示す. なお以下の項目は [2] から引用した. これらの項目をハッシュ値として取り扱っていく.

表 1　ハッシュ値項目

IP ヘッダ	プロトコル, 送信元 IP アドレス 送信先 IP アドレス データ部分の 20Byte
TCP ヘッダ	送信元ポート, 送信先ポート
UDP ヘッダ	送信元ポート, 送信先ポート
ICMP ヘッダ	タイプ, コード

4.2　通常時のシステムの流れ

本システムの通常時の流れは **図 2** となる.
通常時の流れを以下に示す.

1. 攻撃ホストから被害ホストへ向けて, GINE を経由してパケットを送り出す.
2. GINE 内部のルータで, 送られてきたパケットをコピーして, そのコピーしたパケットをそのまま対応トレーサに UDP 通信で送信する.
3. トレーサではパケットを受信し, 送られてきたパケットをハッシュ値に変換する. さらに, ハッシュ値とトレーサと対となるルータの IP アドレス・送信元 MAC アドレスを保存する.

4. 被害ホスト側では, GINE から送り出されてくるパケットをパケットキャプチャ処理によって受信, ハッシュ値と一同のハッシュ値を持つパケットの蓄積数を各々テーブルに保存していく.

4.3 追跡時のシステムの流れ

本システムの追跡時の流れを **図 3** に示す. ここでは, 被害ホストとトレーサ間のみの通信シーケンス図を表し, トレーサが 2 個の場合を想定している.

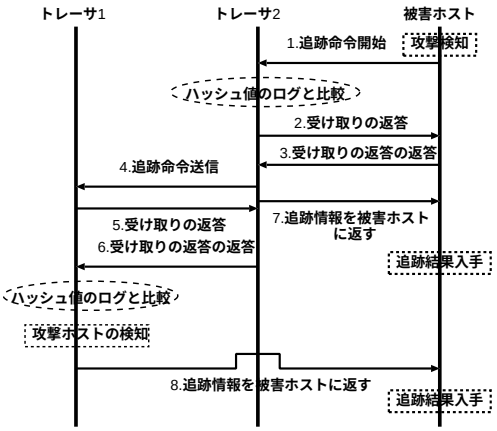

図 3　追跡時のシステムの流れ

追跡時の流れを以下に示す.

1. 被害ホスト側で攻撃を検知して, 貯めていたハッシュ値の中から追跡したいハッシュを選択し, 選択したハッシュ値と被害ホストの IP アドレスを隣接トレーサに送信する.
2. 被害ホストからデータを受けたトレーサは, 送られてきたハッシュ値と通常時に貯めていたハッシュ値とを比較する. 一致すれば, 送信元 MAC アドレスを取り出し, これをもとに次の隣接トレーサにハッシュ値と被害ホストの IP アドレスを送信し, 同時に被害ホストに追跡情報 (ルータの IP アドレス, 送信元 MAC アドレス, 時間, ホップ数) を送信する. 一致しなければ, 失敗メッセージを被害ホストに送る.
3. 中間トレーサでは, 最初のトレーサと同様の処理を行う.
4. 最後のトレーサでは, 受け取ったハッシュ値を比較し, 一致すれば, 成功メッセージと追跡情報を被害ホストに返す. 一致しなければ, 失敗メッセージを送る.

トレーサ決定方法

実ネットワークで本システムを用いる場合, 追跡時の流れ **2** における次のトレーサの決定方法は次のとおりである. 送信元 MAC アドレスとそのルータに対応するトレーサの IP アドレスを設定した対応表を作成する ([2] より引用). この対応表により次のトレーサを決定できる.

5 実験結果

本節では, 本研究で試作したシステムの実験結果 (抜粋) を示す.

5.1 被害ホスト側

被害ホストでの通常処理で, ハッシュ値が配列にたまっていく様子は以下のようになる.

```
./Host3 eth0 192.168.3.1
HASH : a3fa06997a54748a70b0f55fa10dee11
```
↑ 送られてきたパケットのハッシュ値
```
HashData[0]:a3fa06997a54748a70b0f55fa10dee11
            count = 1
```
↑ ハッシュ値を格納，同時にカウントも表示

同様に, 追跡結果は以下のようになる. ここでは, 成功メッセージを表示させている.

```
./Send2 192.168.3.1 a3fa06997a54748a70b0f5
5fa10dee11
```
↑ 追跡したいハッシュ値を引数として実行
```
MESSAGE RECV FROM TRACER
```
↑ 3way ハンドシェイクの確認
```
MESSAGE FROM TRACER
RecvIP : 192.168.4.1
```
↑トレーサの対応ルータの IP アドレス
```
RecvMAC : 00:08:0d:a8:d2:f9
```
↑ 攻撃ホストの MAC アドレス
```
Num of HOP : 3
```
↑ルータのホップ数
```
TimeSub  =  2558
```
↑時間差

5.2 トレーサ側

トレーサ側での通常処理実行結果は以下のようになる.

```
data = a3fa06997a54748a70b0f55fa10dee11
```
↑ハッシュ値データ
```
size 1
```
↑格納されたサイズ
```
update
deleted 0 entries table size 1
```

同様に, 追跡結果は以下のようになる.

```
send to recv message
send to recv message recv!
```
↑ 3way ハンドシェイクの確認
```
hop = 0
Match!!
```
↑ハッシュ値が一致している
```
Key:a3fa06997a54748a70b0f55fa10dee11
smac 00010b0a04e5 IP 192.168.4.3
time 1197960710
```
↑格納されている MAC アドレス，IP アドレス，時間

```
45000054000040003e01b556c0a80101
c0a80301080011e699070000d86d6747
205a020008090a0b0c0d0e0f10111213
1415161718191a1b1c1d1e1f20212223
2425262728292a2b2c2d
```

↑攻撃者のパケットデータ

```
Recv Message get!!
```

6 評価

本節では, 本研究で実験した評価について示す. 主に環境条件を変えたときの追跡完了時間, 成功率, を計測した. 実験には ICMP の ping を 1 秒間に 1 個の割合で用いた. また, 遅延を 30 ミリ秒に設定した. これは実ネットワークの環境に近づけるためである.

まず, トレーサの数を増やしていった場合における, 成功までにかかる時間を図に表すと**図 4** の様になる. 成功とは, 最終トレーサまで追跡が成功した時をさす. ここでは攻撃パケットのみを 20 秒間流し, トレーサが 10 個までの記録をとった. 次に, 一定間隔における攻撃パケットの割合 (総数) を増やしていった場合における, 成功までにかかる時間は, **図 5** の様になる. なお, 1 倍とは, 一つの端末でパケットを 20 秒間流しており, 2 倍から 5 倍とは, 二つから五つの端末でそれぞれパケットを流していることを意味する. また, トレーサは 3 個の場合を想定している.

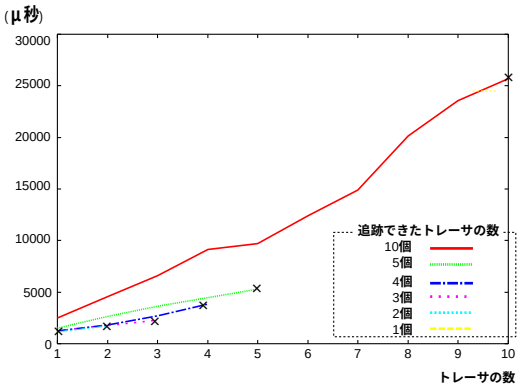

図 4 トレーサの数を変えた場合の追跡完了時間

以上の結果から, **図 4** の場合, 及び**図 5** の場合において, ともに単調増加のグラフであり, 大幅な遅れはみられなかった. また本研究では, 同一 PC によって仮想ネットワーク及び外部トレーサを構成しているため, 負荷がかかり, 追跡完了時間に影響があったと思われるが, 実ネットワークで構成するときは, 通常, 一つのルータに対して一つのトレーサを設置するため, さらに追跡完了時間が短縮する.

また, 成功率 (つまり, 最終トレーサまで追跡でき, 攻撃ホストの MAC アドレスを入手できたかどうか) は常に 100 パーセントであり, パケット損失もみられなかったため, この点でも本システムは実用的であると考えられる.

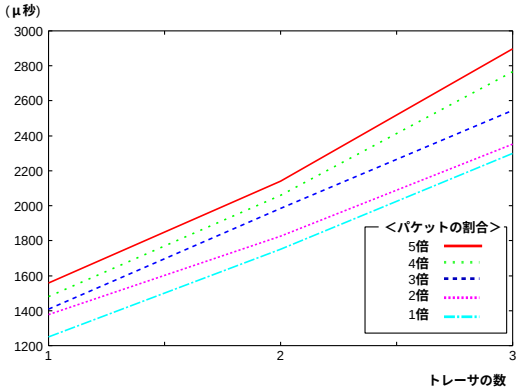

図 5 攻撃パケットの割合を変えた場合の追跡完了時間

7 おわりに

本研究では, 既存手法の一つであるロギング方式の IP トレースバックのシステムを提案し, 実装・評価して, 実際のネットワークでも使える IP トレースバック機構の実現を目指した. 実験の結果, 攻撃ホストの攻撃パケットの情報を検知し, 攻撃ホスト MAC アドレス等の通信情報を入手, 追跡を成功させることができた. また, 本研究で提案したシステムは,

1. 実験環境が容易に設定可能
2. 経路情報を追うことによって攻撃ホストを判断することができる
3. 攻撃中でなくても追跡可能

などの利点がある. また, 本システムでの実験結果の評価を行ったが, 本研究での実験環境では, ネットワークエミュレータと外部ホストを一台の PC として実現したため, ネットワーク構成を複雑にし過ぎると, 一台の PC に多大な負荷がかかってしまう. そのためトレーサの数を 10 個までで実験をした. しかし, 実ネットワーク上で本システムを実現する場合, 通常は一つのルータに一つのトレーサを配置するため, 本システムにみられたような過剰負荷はかからないので, 実ネットワークでの本システムの実現には支障はないと考えられる.

参考文献

[1] 門林 雄基, 大江 将史, IP トレースバック技術, 情報処理学会論文誌, Vol.42 No.12, pp.1175-1180 (2001).

[2] 伊藤 健司, 川本 高弘：既存手法を組み合わせた IP トレースバックの提案と評価, 卒業論文, 南山大学数理情報学部情報通信学科 (2005).

[3] Ihara, A., Murase, S., and Goto, K.: IPv4/v6 Network Emulator using Divert Socket, *Proc. of 18th International Conference on SystemsEngineering(ICSE2006)*, Coventry, UK, pp. 159-166(Sep.2006).