修士論文　　　2001年度(平成13年度)

DoS攻撃に対する逆探知機構の設計と実装

慶應義塾大学大学院　政策・メディア研究科
原嶋章介

修士論文要旨　　2001年度(平成13年度)

DoS攻撃に対する逆探知機構の設計と実装

インターネットにおけるDenial of Service(以下，DoS)攻撃とは，標的となるサーバに大量のデータを送信することでサーバのリソースを使い尽くし，サービスを妨害および拒否する攻撃である．インターネットでは，発信元アドレスに正当性の保証がなくてもパケットの送信が可能であり，パケットの辿る経路を発見する機構が備えられていないため，DoS攻撃が脅威となっている．

DoS攻撃を防止するために送信パケットを発信元アドレスによってフィルタリングする方法は，インターネット上の全ルータの協調が前提となるため完全ではない．したがってフィルタリングとは別に，パケットの辿ってきた経路を逆探知する機構も備えることが，DoS攻撃を抑止するために必要と考えられる．

また，最近はDoS攻撃の反射と呼ばれる行為に対する注意が勧告されている．反射では，まず攻撃者が発信元アドレスを被害者と詐称し，被害者と関係ない多数のリモートホストに要求パケットを送信する．すると多数のリモートホストは，攻撃者ではなく被害者に対して一斉に応答するためDoS攻撃となる．反射は，多数のリモートホストを用意する際に面倒なクラック手続きが必要なく，容易に数万単位で攻撃経路を作れるため，大きな脅威である．既存のDoS攻撃に対する逆探知機構は反射に対応していないため，反射にも対応できる逆探知機構は早急に必要とされる．

このような背景で，本研究は反射にも対応できるDoS攻撃逆探知機構(逆探知系RPPM: Reflectable Probabilistic Packet Marking)を実現した．本研究の特徴は，確率的パケットマーク方式と呼ばれる逆探知方式を採用し，既存システムとの親和性を保ちつつ反射にも対応した点である．また，ネットワークとルータへの負荷が少なく，攻撃終了後も逆探知できる．

本論文では，まずDoS攻撃を攻撃形態によって分類し，DoS攻撃に対する関連研究を比較することで，逆探知機構の必然性とパケットマーク方式の有効性を明らかにする．そして，パケットマークの際の反射ホスト問題と符合化問題を明らかにし，それらの問題に対応する逆探知系RPPM設計と実装について詳しく述べる．また，実装した逆探知系RPPMをルータ，反射ホスト，被害者に適用した場合の影響と負荷を測定し，実用上問題がないことを示す．同時に類似研究との機能比較を行い，逆探知系RPPMの有効性を実証する．最後に今後の課題を述べ，全体としての結論を導き本論文を締めくくる．

キーワード:
1 ネットワーク　2 セキュリティ　3 DoS攻撃　4 逆探知　5 反射

慶應義塾大学大学院　政策・メディア研究科
原嶋章介

Abstract of Master's Thesis (Academic Year 2001)

The Design and Implementation of Traceback System against Denaial-of-Service Attacks

Denial-of-Service(DoS) attacks consume the resources of a remote host or network by flooding anonymous packets, thereby denying or degrading service to legitimate users. Although a filtering technique eliminates the capability to forge the source address, its effectiveness depends on universal deployment of filtering. This is too difficult on today's Internet, therefore traceback technologies are necessary to deter DoS attacks.

Moreover, recent analysis of DoS attacks reported further complicated attacks due to the use of *reflectors*. That is, attackers first locate a very large number of reflectors, which can be any Internet servers. Then they send the reflectors forged traffic, purportedly coming from a particular host, the victim. The reflectors will in turn generate traffic to the victim. The net result is that the flood at the victim arrives from numerous reflectors. Consequently, such "laundered" traffic also should be considered as the challenge of traceback technologies.

In this research, Reflectable Probabilistic Packet Marking(RPPM), which is a traceback system against DoS attacks including traffic laundered by reflectors, has been designed and implemented. We have extended the packet marking technique to render reflectors ineffectual so as to trace the origin of the attacks, while ensuring present architectures friendly. At the same time, we have achieved low overhead, low managing cost, and the ability of performance after an attack has completed.

In this paper, DoS attacks are explained and classified by their type. Then several preceding works against DoS attacks are compared and the necessity of a traceback system and advantage of a packet marking scheme are discussed. Next, the reflector's problem and encoding issue on the probabilistic packet marking are addressed, and the design and implementation are described to solve these problems. In addition, it has been proved that the overhead of proposed techniques on existing architectures are very small based on the results of evaluations. Finally, future works and the conclusion of this research are presented.

Keywords:
1 Network 2 Security 3 DoS attacks 4 Traceback 5 Reflector

Keio University Graduate School of Media and Governance
Noriyuki Harashima

目 次

図 目 次

表 目 次

第 1 章

はじめに

1.1　本研究の背景

インターネットにおけるDenial of Service(以下，DoS)攻撃とは，標的となるサーバに大量のデータを送信することでサーバのリソースを使い尽くし，サービスを妨害および拒否する攻撃である．

このようなDoS攻撃として最も有名な例では，2000年2月に，Amazon，Ebay，Etrade，Yahooを含む商用サーバが数時間にわたって業務を妨害された[7]．また，最近の報告では，たった3週間の間に，2,000以上の組織に属する5,000以上のホストが12,805ものDoS攻撃を受けていたことが示されている[19]．

本質的に，インターネットは以下の2つの特徴があるためDoS攻撃の脅威にさらされている．

- 発信元アドレスに正当性の保証がなくてもパケットを送信できる
- パケットの辿ってきた経路を発見する機構を備えていない

DoS攻撃を防止するため，送信パケットを発信元アドレスによってフィルタリングする方法は，インターネット上の全ルータの協調が前提となるため現実的ではない．したがってフィルタリングとは別に，パケットの辿ってきた経路を逆探知する機構も備えることが，DoS攻撃を抑止するために必要と考えられる．

発信元アドレスに正当性の保証を前提としない以上，DoS攻撃は以下の2種類の方法で発信元を撹乱すると考えられる．

ランダム撹乱

発信元アドレスをランダムなアドレスで詐称し，複数のリモートホストから被害者に対して一斉に要求パケットを送信する．

反射撹乱

攻撃者は，発信元アドレスを被害者と詐称し，被害者と関係ない多数のリモート

> ホストに要求パケットを送信する．すると多数のリモートホストは，攻撃者ではなく被害者に対して一斉に応答するため，DoS 攻撃となる．

特に反射撹乱は，多数のリモートホストを用意する際に面倒なクラック手続きが必要なく，容易に数万単位で撹乱された攻撃経路を作れるため，大きな脅威である．このため DoS 攻撃を抑止する逆探知機構は，早急に必要とされる．

1.2 本研究の目的と特徴

本研究では，DoS 攻撃に対する逆探知機構を提供することを目的とする．本研究の逆探知機構によって，侵入検知システムなどで DoS 攻撃を検知した後に，ランダム撹乱や反射撹乱が存在しても実際の発信元を逆探知できる．

本研究の特徴は，確率的パケットマーク方式と呼ばれる逆探知方式を採用し，既存システムとの親和性を保ちつつ反射にも対応した点である．また，ネットワークとルータへの負荷が少なく，攻撃終了後も逆探知できる．

1.3 本論文の構成

本論文では，2 章で DoS 攻撃を攻撃形態によって分類する．また同時に DoS 攻撃に対する関連研究を紹介し，逆探知機構の必然性について言及し，既存の逆探知方式を比較する．3 章では，反射問題を考慮した逆探知を実現する逆探知系 Reflectable Probabilistic Packet Marking (以下，RPPM) について述べる．4 章では，設計に基づいた逆探知系 RPPM の実装の詳細に述べる．5 章では，実装した逆探知系 RPPM の定量的評価と評価結果の考察，類似研究との機能比較を行う．最後に 6 章で今後の課題を述べ，本研究の全体としての結論を導き本論文を締めくくる．

第 2 章

DoS 攻撃に対する関連研究

本章では，まず DoS 攻撃について攻撃形態から分類して説明する．ここでは発信元アドレスの詐称方法からランダム撹乱攻撃と反射撹乱攻撃に分類し，DoS 攻撃の問題を明らかにする．そして，DoS 攻撃に対する関連研究を紹介し，それぞれの関係と有効性について述べる．まず侵入検知システムとフィルタリングを紹介し，本研究の目的である逆探知との関係について言及する．また逆探知については，リンクテスト方式，ログ方式，ICMP 方式，パケットマーク方式の 4 方式挙げ，それぞれの特徴を述べ，比較する．

2.1　攻撃形態

本節では DoS 攻撃について攻撃形態から分類して説明する．ここでは発信元アドレスの詐称方法から，ランダム撹乱攻撃と反射撹乱攻撃に分類する．

2.1.1　ランダム撹乱攻撃

最も一般的な DoS 攻撃を，図 2.1に示す．まず攻撃者 (Master) は，リモートホストに侵入し分身攻撃者 (Slave) を仕立てる．攻撃者は分身攻撃者を遠隔制御し，発信元アドレスをランダムなアドレスで詐称し，被害者 (Victim) に対して膨大な量のパケットを送信する．

この攻撃は実装が容易で，多くのツールがアンダーグラウンドのクラッカーに出回った．有名な DoS 攻撃のツールの中では，TFN[12]，Shaft[9]，Trinoo[10]，Stacheldraht[11] などがこの手法を使う．

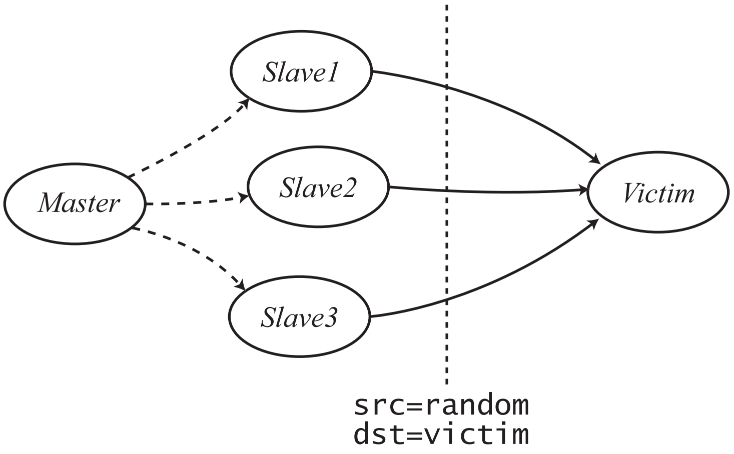


図 2.1: ランダム撹乱 DoS 攻撃

2.1.2　反射撹乱攻撃

さらに攻撃者は，発信元を撹乱するために反射と呼ばれる行為を行う．反射を使った DoS 攻撃を，図 2.1に示す．まず攻撃者 (Master) は，リモートホストに侵入し分身攻撃者 (Slave) を仕立てる．攻撃者は分身攻撃者を遠隔制御し，発信元アドレスを被害者と詐称し，多数の反射ホスト (Reflector) に対して大量の要求パケットを送信する．結果として多数の反射ホストは，攻撃者からの要求パケットに対する応答パケットを，被害者に大量に送信する．

この攻撃の特徴は，あらゆるインターネットサーバが反射ホストになれることである．反射を使わない DoS 攻撃では，分身攻撃者をつくる必要があるため，経路数は多

くても数百が常であった．しかし反射を使った途端，DoS 攻撃の経路数は数万以上に増幅する．またその結果として，反射を使わない場合と比べて，各経路のトラヒック量が少なくても同様の効果をもたらす．

この種類の攻撃は [7, 26, 23] で言及されている．まだこの手法を使った DoS 攻撃のツールは発見されていないが，アンダーグラウンドのクラッカーにはすでに出回っているともいわれる．その攻撃力は飛躍的に向上しているため，早急な対策が必要である．

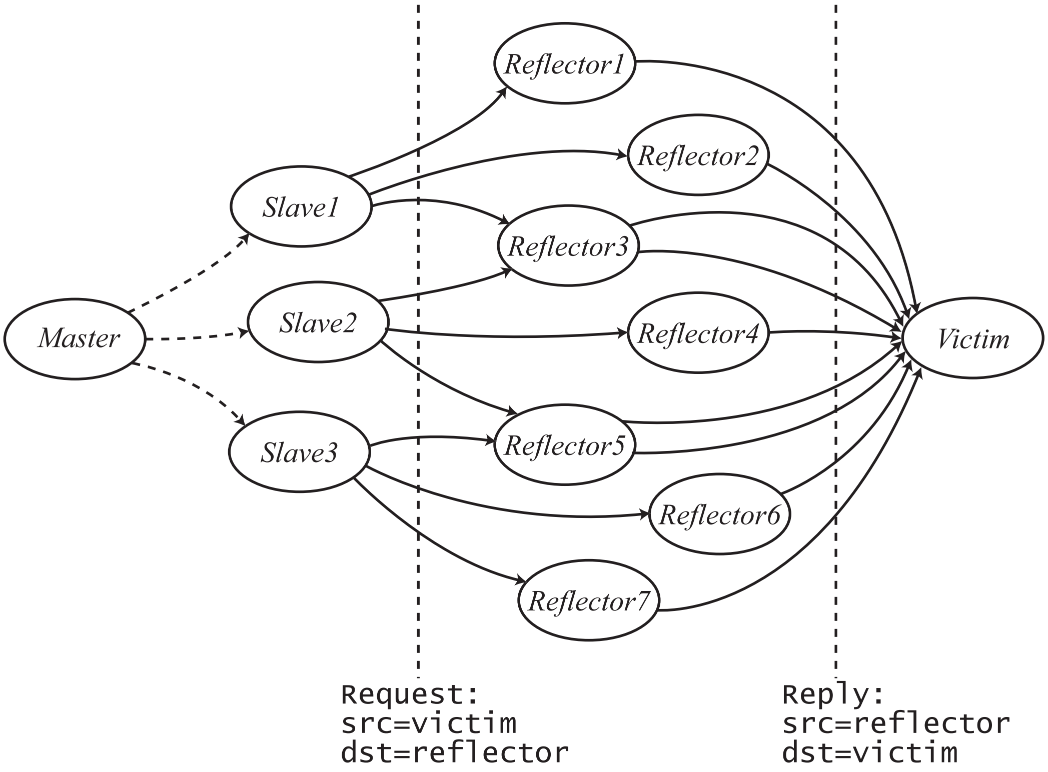


図 2.2: 反射撹乱 DoS 攻撃

2.2 関連研究

本節では DoS 攻撃に対する関連研究を紹介する．まず侵入検知システムとフィルタリングを紹介し，本研究の目標である逆探知との関係について言及する．また逆探知については，リンクテスト方式，ログ方式，ICMP 方式，パケットマーク方式の 4 方式挙げ，それぞれの特徴を述べる．

2.2.1 侵入検知システム

侵入検知システム (以下，IDS) は，侵入を検知するシステムである [20, 29]．一般に IDS は，既存の侵入のトラヒックパターンをまとめた，シグニチャと呼ばれるファイ

ルと，実際のトラヒックパターンをバイナリマッチングさせてホストへの侵入を検知する．これは図 2.1，2.2において，攻撃者 (Master) が分身攻撃者 (Slave) を遠隔制御していることを検知するとき最も効力を発揮する．

一方，攻撃者が発信元アドレスを詐称して大量のパケットを送信する場合，IDS で DoS 攻撃自体の検知は容易だが，発信元を特定して原因を突き止めることが困難である．

本研究では，攻撃者による分身攻撃者へのクラック検知と DoS 攻撃自体の検知は IDS の利用を想定し，攻撃経路の逆探知に焦点をおく．

2.2.2　フィルタリング

DoS 攻撃を未然に防ぐ方法として，発信元アドレスの正当性を保証する方法が考えられる．入口フィルタリング [27] では，ルータの設定でパケットの発信元アドレスの矛盾を検査し，矛盾のあるパケットはインターネットに流さないことを提案している．事実，一部の学術機関や企業は，登録された MAC アドレスと IP アドレスの組以外をインターネットに送信させない．

しかし，この手法は発信元アドレスの数が限定されていることが前提なので，巨大なインターネットサービスプロバイダ (以下，ISP) では無理がある．また，多くの ISP のトラヒックが合成された場合，全ての ISP が入口フィルタリングを実行していないと，アドレスの矛盾を検査できない．つまり入口フィルタリングは全 ISP の協調が前提となるのだが，管理コストが大きいため，あまり現実的ではない．

したがって DoS 攻撃を抑止するためには，フィルタリングとは別に，パケットの辿ってきた経路を逆探知する機構も備える必要がある．

2.2.3　リンクテスト方式による逆探知

逆探知の原始的な方法は，手動逆探知である．多くのルータは入力に対するデバッグ機能があるため，管理者は入力デバッグ機能を使って，攻撃パケットを中継している上流ルータを特定する．この作業を再帰的に繰り返すことで攻撃者の発信元に到達する．本研究では，このようにリンクをテストする逆探知をリンクテスト方式と呼ぶ．

しかし手動ではあまりに管理コストが大きいため，[4] では，管理コストのより少ない自動リンクテスト方式を提案した．この方式では，ルータでは入力デバッグを使わずに定期的にリンクに負荷をかける．もし DoS 攻撃が発生すれば，定期的なトラヒックに異変が発生するため，DoS 攻撃を発見できる．そして DoS 攻撃を検知した被害者は，あらかじめ用意されているインターネットマップ [5] を用いて逆探知する．

これらのリンクテスト方式の場合，逆探知が終了するまで攻撃者が攻撃をやめないという前提があるが，実用上は攻撃が終了しても逆探知できる機構が望ましい．本研究では，攻撃が終了して時間が経過しても後から逆探知できる性質を，事後処理性と呼ぶ．

なお既存のリンクテスト方式では反射撹乱攻撃に対応した逆探知機構は存在しない．

2.2.4 ログ方式による逆探知

事後処理性をもつ逆探知方法の一つは，ルータでパケットを中継する際にログを記録することである．本研究ではこれをログ方式と呼ぶ．被害者は，攻撃パケットを隣接する上流ルータのログから再帰的に探索していけば逆探知できる．ただし，ルータで全てのパケットをログに記録するとディスク空間が不足する．したがって，ログ方式では効率的なログ方法が要求される．

CenterTrack[32] では，エッジルータでトラヒックのログを記録し，ログを記録できるエッジルータ間でオーバーレイネットワークを作成することを提案している．CenterTrack では，オーバーレイネットワークを形成するルータでログを記録すれば済むため，全体のログ容量を減らせた．しかし，オーバーレイネットワークの管理コストが非常に大きいため，実用は非現実的である．

またハッシュを利用したログ方式としては，Source Path Isolation Engine(以下，SPIE)[28] がある．SPIE では，パケットに一意なハッシュID をつけることで，ディスク空間を大幅に節約している．ハッシュ関数の入力として，IP 層の発信元や送信先だけでなく，TTL，オフセットフラグ，ID フィールドも使う．したがって IP 層より上位層のデータが等しいパケットでも，ハッシュID をつけるルータによってハッシュID が異なる．この結果 SPIE では，DoS 攻撃のトラヒック量が非常に小さくても細かい粒度で逆探知できる．

なお既存のログ方式では反射撹乱攻撃に対応した逆探知機構は存在しない．

2.2.5 ICMP 方式による逆探知

ログ方式以外に事後処理性をもつ逆探知方法としては，ITRACE[1] がある．ITRACE では，ルータがパケットを中継した際に，低い確率で送信先アドレスに ICMP メッセージを送信する．トラヒック量が大きくなれば，全ての中継したルータからメッセージが届くはずである．本研究ではこのように ICMP を利用した逆探知を ICMP 方式と呼ぶ．

しかし，ITRACE の ICMP メッセージの中継は，ルータの入力デバッグの性能に依存するため，一部のルータでリンクの負荷が高い場合正常に処理されない場合がある．また，ITRACE では送信する ICMP メッセージ分のトラヒックが増やして逆探知するにもかかわらず，ICMP トラヒックを拒否するサーバが多いため，正常に動作しない場合がある．

なお既存の ICMP 方式では反射撹乱攻撃に対応した逆探知機構は存在しない．

2.2.6 パケットマーク方式による逆探知

事後処理性をもつ別の方式として，中継したルータの IP アドレスをパケットにマークする方式が考えられる．本研究ではこれをパケットマーク方式と呼ぶ．IP には，IP Record Route オプション [25] があり，ルータのアドレスを順次オプションヘッダに追加できる．しかしオプションヘッダには 20 バイトという上限があり，9 個のルータアドレスをパケットに追加すればそれ以上アドレスを追加できなくなる．

この問題を解決するパケットマーク方式の改良として，確率的パケットマーク方式[26]が提案された．確率的パケットマーク方式では，被害者を根とするルータ有向グラフの各辺を確率的にサンプリングする．このためサンプリング数が多ければ，経路上の中継ルータの数が増えても固定長でマークできる．幸いDoS攻撃は膨大な量のパケットをもって攻撃するのでサンプリング数は豊富に得られる．

元来パケットマーク方式はルータの負荷を大きくすると考えられていたが，確率的パケットマーク方式では各ルータが自律的にパケットマークを実行すればよいため，管理コストが非常に少なくなる．

[26]は攻撃者が単体であるモデルだったが，[30]は攻撃者が複数の場合に拡張したモデルも提案している．また，[26]では，反射撹乱攻撃に若干言及しており，確率パケットマーク方式を用いた反射撹乱攻撃の解決の可能性を示唆していた．しかし，実際に反射撹乱攻撃に対応した確率的パケットマーク方式の研究は存在しない．

2.3 関連研究の分類

以上をふまえ，関連研究の分類を表2.1に示す．

表2.1: 関連研究の分類

評価項目	IDS	フィルタ	リンク	ログ	ICMP	マーク
逆探知	×	×	○	○	○	○
管理コスト	高	中	中	高	低	低
ネットワーク負荷	低	低	高	低	低	低
ルータ負荷	低	中	高	高	低	低
事後処理性	×	×	×	○	○	○
反射撹乱攻撃対応	×	×	×	×	×	△

本研究ではDoS攻撃の逆探知を目的としている．IDSとフィルタは逆探知を目的としていないので，むしろ逆探知とは共存する関係にある．

また他の4つの逆探知方式については，管理コスト，ネットワーク負荷，ルータ負荷，事後処理性，反射ホスト対応という観点で以下に述べる．

管理コストについては，リンク方式とログ方式では，ルータ間の協調作業が前提となるため管理コストが大きい．しかしICMP方式とパケットマーク方式では，各ルータが自律的に動作するため，全体の管理コストは低い．

ネットワーク負荷については，リンクテスト方式では，余分なトラヒックを発生させるため負荷が高い．

ルータ負荷については，リンクテスト方式では，余分なトラヒックをリンクにかけるため，負荷が高い．またログ方式では，膨大なディスク空間を必要とするため，負

荷が高い．ICMP 方式とパケットマーク方式では，確率的に動作する限り，ルータの負荷はそれほど高くない．

事後処理性については，ログ方式，ICMP 方式，パケットマーク方式が備える．特にログ方式は，DoS 攻撃のトラヒック量が非常に小さくても細かい粒度で逆探知できる先行研究がある．

反射撹乱攻撃については，対応している逆探知方法が未だないが，パケットマーク方式は実用上の可能性を示唆していた．

このように関連研究を分類してみると，パケットマーク方式，特に確率的パケットマーク方式を採用すれば，ネットワークやルータへの負荷が少なく，事後処理性もある，反射撹乱攻撃に対応できる逆探知機構を実現できると考えられる．

2.4 本章のまとめ

本章では，DoS 攻撃を発信元アドレスの詐称方法からランダム撹乱攻撃と反射撹乱攻撃に分類して，DoS 攻撃の問題を明らかにした．また，DoS 攻撃に対する関連研究を紹介し，それぞれの関係と有効性について述べた．まず，侵入検知システムとフィルタリングは，本研究の目的である逆探知とは共存関係にあることを述べた．また，既存の逆探知方式について，リンクテスト方式，ログ方式，ICMP 方式，パケットマーク方式の 4 方式を挙げ，それぞれの特徴を述べ比較した．その結果，反射撹乱攻撃に対応した方式は存在しないが，確率的パケットマーク方式を採用することで，反射撹乱攻撃にも対応できる逆探知機構を実現できる可能性があることを示した．

次章では，以上を参考に，確率的パケットマーク方式を採用した，反射撹乱攻撃にも対応できる逆探知機構の設計の詳細を述べる．

第 3 章

逆探知系 RPPM の設計

本章では，確率的パケットマーク方式を採用した，反射撹乱攻撃にも対応できる逆探知系 Reflectable Probabilistic Packet Marking (以下，RPPM) の設計の詳細を述べる．まず設計に際し，必要な定義と仮定について述べる．次に既存の確率的パケットマーク方式の問題である反射ホスト問題と符合化問題を明らかにし，対応方法について詳しく述べる．最後に，逆探知系 RPPM の基本アルゴリズムであるマークアルゴリズム，反射アルゴリズム，再構築アルゴリズムの詳細を示す．

3.1 設計方針

本研究では，2.2.6節で述べた確率的パケットマーク方式を採用し，既存システムとの親和性を保ちつつ反射にも対応できる逆探知機構を目指す．また，ネットワークとルータへの負荷が少なく，攻撃終了後も逆探知できるよう設計する．そのため，本節では，設計に必要な定義と仮定について述べる．

3.1.1 定義

本節では，DoS 攻撃を逆探知する問題を定義する．図 3.1は，被害者 Vを根とする有向グラフで表現されたネットワークで，潜在的な攻撃者 $A_1..A_4$からの DoS 攻撃を示している．なお，本研究では図 2.1と 2.2における Master と Slave を総じて攻撃者と呼ぶ．また，中間ノード R_iはルータを示し，中間ノード Rf_iは反射ホストを示している．図中の点線で示された矢印は，攻撃者が送信元アドレスを詐称したために撹乱された経路である．$\{A_2, R_2, R_3, R_4, V\}$ はランダム撹乱の経路，$\{A_4, R_6, Rf, R_4, V\}$ は反射撹乱の経路を示している．

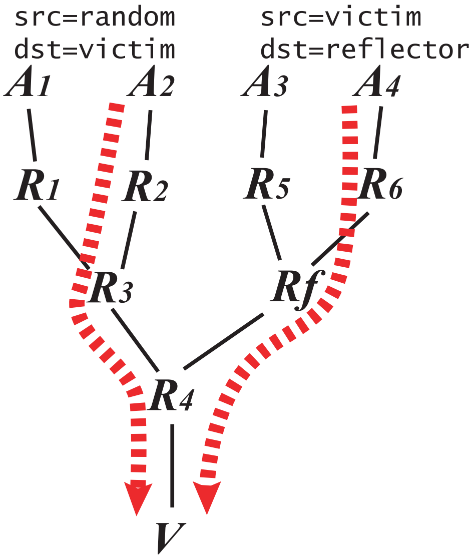


図 3.1: 被害者 Vを根とする有向グラフ

ここで，被害者 Vが攻撃者に対して逆探知を試みる場合，完全逆探知問題 (*exact traceback problem*) を定義できる．完全逆探知問題とは，攻撃者の経路が複数あった場合に，その全てを逆探知できることを目指す．しかし，攻撃者は MAC アドレスすら詐称するかもしれないし，正しいルータアドレスを詐称して攻撃経路を撹乱するかもしれないため，攻撃経路を完全に逆探知するのは困難である．

そこで，より柔軟な近似逆探知問題 (*approximate traceback problem*) を定義する．近似逆探知問題では，攻撃者の経路が複数あった場合に，その一部を逆探知できれば，少

しでも攻撃者への手掛かりがつかめたと見なし，近似逆探知問題の解とする．例えば図 3.1においては，$\{R_4, R_3, R_2, A_2\}$ または $\{R_4, Rf, R_6, A_4\}$ のいずれかが逆探知できれば近似逆探知問題の解である．また，もっと精度の悪い場合は，$\{R_4, R_3, R_2, R_1\}$ となるかもしれないが，それも近似逆探知問題の解の 1 つである．なぜならそこには正しい攻撃経路の一部が含まれているからである．

3.1.2 仮定

近似逆探知問題を解決する設計を行うにあたり，以下の点を仮定することで現実に即した設計を行う．

(1) 攻撃者はあらゆるパケットを生成しネットワークに送信できる

(2) 攻撃者は複数のホストからパケットを送信できる

(3) 攻撃者は逆探知されることに無関心である

(4) 攻撃者は各経路において 1 回だけ反射ホストで反射できる

(5) 攻撃者は膨大な量のパケットを送信する

(6) 攻撃経路は短時間に変動しない

(1) から (4) の仮定は，インターネットにおける通信の特徴上，攻撃の可能性と限界を考慮した仮定である．(1) の仮定は，攻撃者はマーク情報も偽造できることを示すため，この点にも対応できるパケットマークを行う必要がある．また (4) の仮定では，反射が攻撃経路上の任意の 1 地点で発生するため，反射によって攻撃経路の情報が失われないようにする必要がある．

(5) の仮定は，確率的パケットマーク方式には必要不可欠の仮定である．3.2.1節で詳しく述べる辺サンプリングは，ルータのアドレスを確率的にサンプリングするため，サンプリング数が多くなければ攻撃経路を再構築できない．しかし，DoS 攻撃は標的となるサーバに大量のパケットを送信し，リソースを使い尽くしてサービスを妨害および拒否するため，攻撃者が膨大な量のパケットを送信するという仮定は容易に受け入れられる．ただし，1 パケットで標的サーバの機能を停止できる ping-of-death[6] のような攻撃はこの限りではない．

(6) の仮定は，インターネットにおける経路変化の挙動は，短時間には変動しないという報告 [22] に基づいている．この仮定によって，攻撃者までの経路を示すサンプリングされたマーク情報を合成して攻撃経路を再構築できる．

3.2 既存の確率的パケットマーク方式の問題

本節では，既存の確率的パケットマーク方式の辺サンプリングについて述べる．そして，既存の辺サンプリングでは反射ホスト問題と符合化問題に対応できていないことを明らかにする．

3.2.1 辺サンプリング

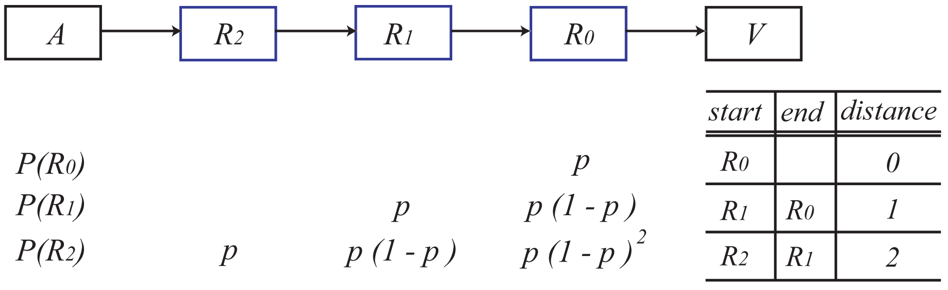
2.2.6節で述べたように，IP Record Route オプションを用いて経路上のルータアドレスを順次オプションヘッダに追加していくと，オプションヘッダの 20 バイトという上限のため 9 個以上のルータアドレスを追加できない．辺サンプリングは，この問題を解決するため，固定長のデータ空間でマークする方法として，[26] で提案された．辺サンプリングは，被害者 V を根とするルータ有向グラフの各辺を確率的にサンプリングする．このためサンプリング数が多ければ，経路上の中継ルータの数が増えても固定長でマークできる．ここで 3.1.2節において，DoS 攻撃は膨大な量のパケットを送信して攻撃すると仮定したため，サンプリング数は豊富に得られる．以下に $start$，end，$distance$ の固定フィールドを使った辺サンプリングの手順を示し，図 3.2に概念図を示す．

(1) 各ルータは確率 p で自身のアドレスを $start$ にマークし，$distance$ を 0 に設定する

(2) 手順 (1) でマークせず且つ $distance$ が 0 の場合，自身のアドレスを end にマークする

(3) 手順 (1) でマークしない場合，常に距離を一つ増やす

(4) 被害者は，サンプリングされた辺 $(start,end)$ を多数集め，距離 $distance$ の小さい順に被害者を根とする有向グラフに挿入し，攻撃経路を再構築する．



start	end	distance
R0		0
R1	R0	1
R2	R1	2

図 3.2: 辺サンプリング

図 3.2が示すように，距離 d におけるルータ R_d が $start$ をマークし，下流のルータに上書きされずに被害者 V に届く確率 $P(R_d)$ は，$p(1-p)^{d-1}$ である．この性質を利用し，最も遠い距離にあるルータがマークしたパケットが被害者に届く確率を計算することで，経路上の各辺を全て集めて経路を再構築するための必要最低限の期待パケット数を求められる．最も遠い距離 d のルータでマークしたパケットを受け取る確率は $p(1-p)^{d-1}$ であるため，d より近くのルータでマークしたパケットを受け取る確率は，それぞれのルータ $p(1-p)^{d-1}$ より大きい．これらの確率事象は互いに独立であるため，全ルータのマークしたパケットを受け取る確率は $dp(1-p)^{d-1}$ となる．このとき受け取

ればよいパケット数は d 種類である．ここで $d(ln(d)+O(1)))$ 個のパケットを集めれば，d 種類のパケットが集まることが，クーポンコレクタ問題 [15] によって知られている．したがって，経路を再構築するために必要な総パケット数の期待値 $E(N)$ は，次の上限で得られる．

$$E(N) < \frac{ln(d)}{p(1-p)^{d-1}} \tag{3.1}$$

例えば p=0.05，d=10 とすれば，$E(N)$ <74 がえられるため，被害者は 75 パケット集めればほぼ経路を再構築できる．

また，攻撃者はマーク情報を偽造して被害者に送信できるが，攻撃者はその経路で最も遠くからパケットを送信するため，距離と実際に集まるパケット数の関係が理論値と大きく異なる．したがって攻撃者はマーク情報を正確に偽造することは困難である．

3.2.2 反射ホスト問題

反射撹乱攻撃については2.1.2節で述べた．本節では，反射撹乱攻撃が確率パケットマーク方式に及ぼす影響について述べる．反射ホストはルータではないため，反射ホストが受信したパケットは反射ホストのアプリケーション層まで到達する．したがって図 3.3が示す通り，攻撃者から反射ホストまでのルータで確率的パケットマーク方式が動作しても，反射ホストがパケットを要求パケットとしてアプリケーション層で処理し，新たに被害者ホストに応答パケットを送信すると，攻撃者から反射ホストまでの経路を示すマーク情報が失われる．つまりアプリケーション層で考えると攻撃を中継しているが，ネットワーク層で考えると攻撃を中継していないため，逆探知に支障を来たす．これらの問題への対処法については，3.3節で述べる．

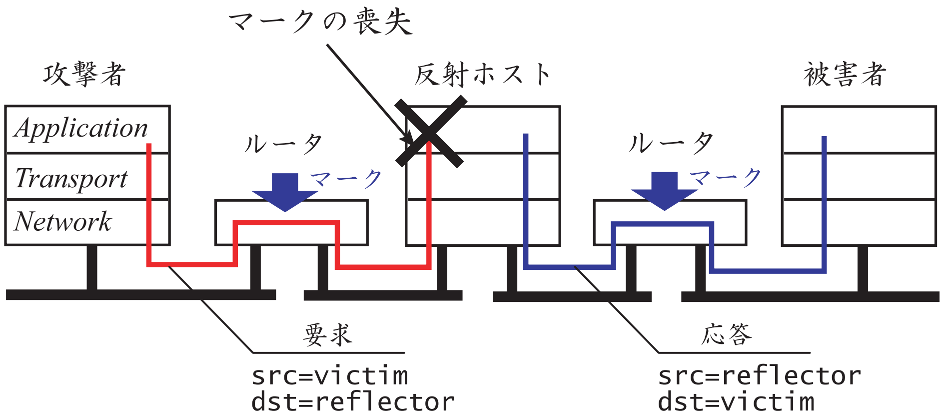


図 3.3: 反射ホスト問題

3.2.3 符合化問題

[26, 30] では，マークした情報を IP ヘッダの ID フィールドに符合化していた．もともと 72 ビット必要な情報量を 16 ビットの空間に詰め込むために，冗長性を持たせて k 個にフラグメント化する．このため，フラグメント化したパケットを組み合わせる計算量は，距離 d でマークされたフラグメントの集合をΨ_d，距離 d におけるルータの集合を S_dとすれば，$O(\sum_d S_d \Psi_d^k)$ となる．ここで複数の攻撃経路を考慮すると場合，以下の点で問題がある．

(1) d が大きいほど $S(d)$ は多くなるため，Ψ_dの組み合わせは一意に定まらず，誤り率が高い．

(2) d が大きいほど $S(d)$ は多くなるため，膨大な量のΨ_dを組み合わせる必要があり，再構築の計算量が大きすぎる．

(3) d が大きいほど $S(d)$ は多くなるにも関わらず，$S(d)$ でマークされるパケットを受信する確率は少なくなる．

(4) マーク情報の有無が判断できないため，マーク情報の活用効率が悪い．

[30] では，(1) から (3) の問題を考慮して，大きなハッシュ空間を使ったフラグメント化を考案し，組み合わせ計算量を $O(\sum_d S_d \Psi_d)$ まで減らした．しかし，各ホストがインターネットマップ [5] を持たなければ再構築できないという条件が付加された．反射ホスト問題を考えれば，各ホストは膨大な反射ホストのインターネットマップを取得しなければならないが，それは管理コストを増加させ，パケットマーク方式の意義がない．

そもそも，ID フィールドを変更すると，フラグメント化した場合にパケットを修復できない．先行研究ではこのフラグメント化問題に対して，インターネットにおけるフラグメント化はたかだか 0.25%である [31] として無視している．

そして最大の問題は，ID フィールドを変更すると ICMP[24] に支障をきたす点である．例えば，`ping` コマンドでは ICMP Echo を送り ICMP Echo Reply を受け取る．その際 ICMP Echo Reply の ID フィールドが変更されていると，応答と見なせない．`traceroute` コマンドも同様であり，ネットワーク管理に欠かせないコマンドが使用できないのは問題である．これらの問題への対処法については，3.4節で述べる．

3.3 反射ホスト問題への対応

3.2.2節で述べた反射ホスト問題を解決するためには，反射ホストで要求パケットのマーク情報を応答パケットにコピーする必要がある．

ここで，反射するべきパケット数は，要求時と応答時で非対称な点に注意する必要がある．例えば，HTTP の GET コマンドはデータ量が少ないが，その応答は非常に

データ量が多い場合がある．このような場合，単純にマーク情報を保存しては動作しない．

考えられる方法は，受信したマーク情報から式 3.1を用いて反射するべきパケット数の期待値を求める方法である．この方法は理論的な反射ができるため，後の攻撃経路の再構築が容易になる．しかし正規分布を近似できるほど大量のパケットを受信できない場合，反射の前後でマーク情報の確率分布が大きく異なる場合が発生する．

そこで，本研究ではマーク情報ごとに要求パケット数の統計をとり，その確率分布にしたがって応答パケットにマーク情報を反射する方法を提案する．この概念を図 3.4に示す．

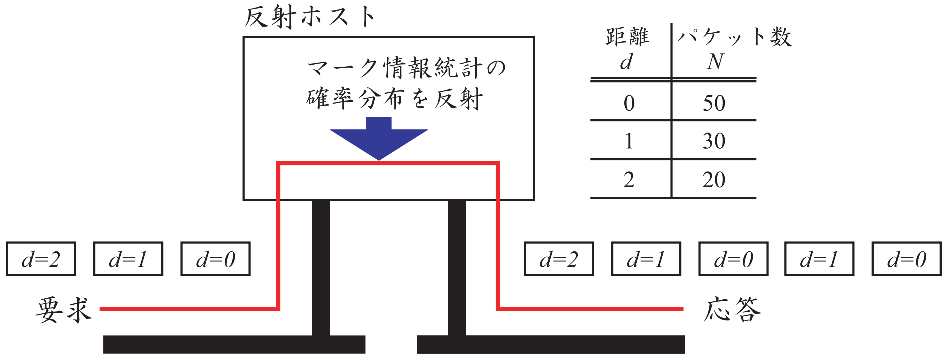


図 3.4: マーク情報の反射

また，マーク情報の反射の際には，要求パケットと応答パケットごとに統計をとる必要があるため，ネットワーク層より上位層のプロトコルヘッダを解析する必要がある．

3.3.1 時系列パケット数の平滑化

マーク情報の確率分布に従ったマーク情報の反射には，期間を設定してパケットの標本をサンプリングし，マーク情報の確率分布を計算する必要がある．各期間ごとの標本パケット数を用いてマーク情報の確率分布を計算すると振幅が大きくなるため，指数平滑法 [13] を用いて標準化する．図 3.5に指数平滑法の例を示し，以下に手順を述べる．

まず平滑化定数をαとする (ただし $0\leq \alpha \leq 1$)．また時系列 t における観測した値を $Y(t)$ とし，時系列 t における平滑化された値を $S(t)$ とする．ここで指数平滑法は $S(t)$ を次のように定義する．

$$S(t) = \alpha Y(t) + (1-\alpha)S(t-1) \tag{3.2}$$

また，αと $S(t)$ には以下の関係が成り立つ．

- α=0 ならば，$S(t)$ = $Y(0)$．

e.g.) $\alpha = 0.3$

t	$Y(t)$	$S(t)$
0	4	4.00
1	7	4.90
2	5	4.93
3	11	6.75
4	9	7.43
5	11	8.50
6	7	8.05
7	14	9.83
8	13	10.78
9	15	12.05

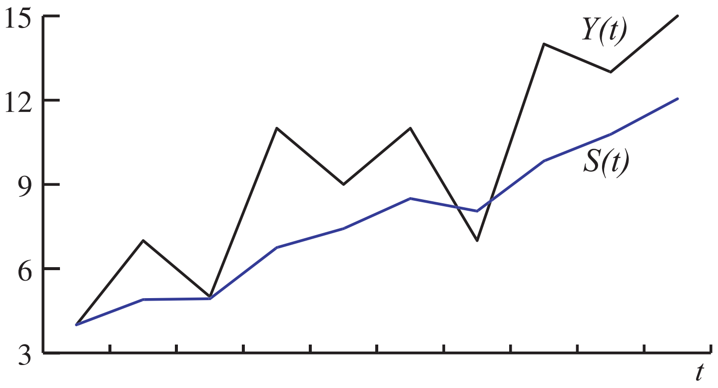


図 3.5: 時系列データの指数平滑化の例

- α=1 ならば，$S(t)$ = $Y(t)$．
- αが大きければ，$S(t)$ は現在の傾向に従う．
- αが小さければ，$S(t)$ は過去の傾向に従う．

一般にαは 0.3 前後で適切に動作するといわれている [2]．本研究において $S(t)$ はパケット数を意味する正の整数なので，$S(t)$ に必要な桁数を Pとすれば，少なくとも$\alpha Y(t) > 10^P$となるよう統計更新周期T(秒) を設定する必要がある．1 秒あたりの入力パケット数を Fとすれば，$\alpha TFp(1-p)^{d-1} > 10^P$ となるため，Tの指標は次のように表せる．

$$T > \frac{10^P}{\alpha Fp(1-p)^{d-1}} \tag{3.3}$$

例えばα=0.3，P=2，F=1000，p=0.05，d=10 の場合，T >11(秒) となる．

3.4 符合化問題への対応

3.2.3節で述べた符合化問題を解決するために，本研究では IP オプション [3] を使うことを提案する．先行研究では，IP オプションはルータの負荷の大きくする愚かな実装とされている．しかし現実問題，以下の理由で IP オプションを使っても問題ない．

- ほとんどのルータは IP オプションを処理できるほど強力になった．
- IP オプションはあまり使われていない (稀に`ping` で IP Record Route オプションを指定する場合がある)．

そこで本研究で提案する IP オプション Reflectable Probabilistic Packet Mark (以下，IP オプション RPPM) のフォーマットを図 3.6に示す．1 バイト目の 1 ビット目の cp フィールドはフラグメント時にこの IP オプションをコピーするか示している．本研究ではコピーしないため 0 となる．1 バイト目の 2 から 4 ビット目の $class$ フィールドでは，パケットの種類を示す．本研究ではコントロールパケットであることを示す 0 となる．1 バイト目の 5 から 8 ビット目の $number$ フィールドでは，IP オプションの種類を示す．本研究では未使用番号 28 を IP オプションの種類として定義する．2 バイト目の len フィールドは，IP オプション長をバイト数で指定する．本研究では 12 バイトの固定長となる．3 バイト目の $flags$ フィールドは未定義だが，IP ヘッダ長を 4 の倍数に揃えるためのパディングとして利用している．

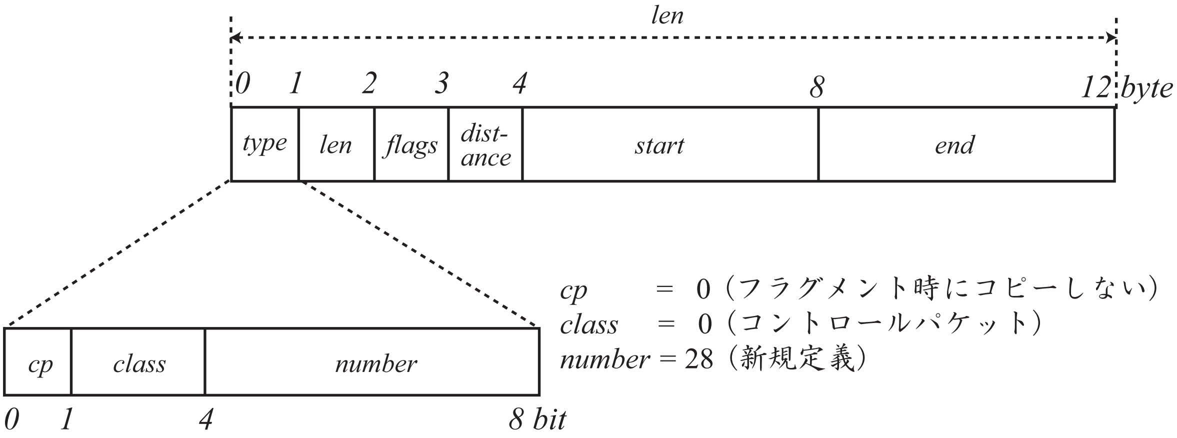


図 3.6: IP オプション RPPM のフォーマット

符合化に IP オプションを使うことに以下のメリットがある．

- ICMP が使えるため，既存システムとの親和性が高い．
- フラグメント化したパケットを組み合わせて検査する処理が必要ないため，計算量は $O(\sum_d S_d)$ まで減る．
- マーク情報の有無が判断できるため，被害者や反射ホストではマークされたパケットだけを対象に集計がとれるため，効率が良い．

3.5 基本アルゴリズム

逆探知系 Reflectable Probabilistic Packet Marking(以下，RPPM) は，反射ホスト問題と符合化問題に対応できる確率的パケットマーク方式を採用した逆探知機構である．逆探知系 RPPM は，マークアルゴリズム，反射アルゴリズム，再構築アルゴリズムの 3 つの基本アルゴリズムから成る．本節では各アルゴリズムの注意点と詳しい挙動および動作場所について述べる．

3.5.1 マークアルゴリズム

反射ホストから被害者までの経路は，発信元アドレスも送信元アドレスも詐称されていない経路であるため，反射ホストより下流のルータでわざわざマークする必要はない．そこで明示的に反射の有無を示すフラグも用意し，反射フラグが上がっていればそれ以上ルータでマークしない最適化が考えられる．しかし攻撃者が反射フラグを悪用した場合，確率的パケットマーク方式が無効となる可能性がある．したがって，マーク情報の反射の有無によって確率的パケットマーク方式の意味論を損なってはならない．以上を考慮したマークアルゴリズムを図3.7に示す．

```
Marking procedure at Router R and Reflector Rf:
  for each packet $w$
    let $x$ be a random number from [0..1)
    if $x < p$ then
      write $R$ into $w$.start and 0 into $w$.distance
    else
      if $w$.distance $= 0$ then
        write $R$ into $w$.end
      increment $w$.distance
```

図 3.7: マークアルゴリズム

マークアルゴリズムはルータおよび反射ホストで動作する．また，反射ホストは反射アルゴリズムの後にマークアルゴリズムを実行する必要がある．挙動内容は既存の辺サンプリングと同じである．各ルータまたは反射ホストは，確率pで自身のアドレスを$start$にマークし，$distance$を0に設定する．もしマークせず且つ$distance$が0の場合，自身のアドレスをendにマークするもしマークせず且つ$distance$が1以上の場合，距離を1増やす．

3.5.2 反射アルゴリズム

反射アルゴリズムでは，要求パケットと応答パケットごとに統計をとる必要があるため，ネットワーク層より上位層のプロトコルヘッダを解析する必要がある．しかし，要求パケットと応答パケットを区別するアルゴリズムは別機構を利用することを想定し，本節のアルゴリズム記述では触れない．以上を考慮した反射アルゴリズムを図3.8に示す．

反射アルゴリズムは反射ホストで動作する．反射ホストは反射アルゴリズムの後にマークアルゴリズムを実行する必要がある．反射アルゴリズムは，$Storing$部と$Reflecting$部からなり，互いに統計Hを共有する．$Storing$部では入力パケットが被害者ホストを示す送信元アドレスをキーに集計される．同じマーク情報のパケットは同エントリに格納され，統計全体の受信パケット数カウントが増加する．$Reflecting$部では出力パ

```
let $H$ be a hashtable
let mark be tuples(start, end, distance, count)
let entry in $H$ be tuples (address, marks, total_count)
Storeing procedure at Reflector Rf:
  for each incomming request packet $w$
    if $H$ contains $w$.source then
      increment total_count of entry($w$.source) in $H$
    if $w$.mark then
      if $H$ contains $w$.source then
        let $e$ be a entry($w$.source) in $H$ where $e$.mark = $w$.mark
        increment count of $e$.mark
      else
        put entry($w$.source, $w$.mark, 0) into $H$
Reflecting procedure at Reflector Rf:
  for each outgoing reply packet $w$
    if $H$ contains $w$.destination then
      let $e$ be a entry($w$.destination) in $H$
      do smoothing with total_count of $e$ and each count of $e$.marks
      let $x$ be a random number from [0..$e$.total_count)
      let total_index be 0
      for each mark $m$ in $e$.marks
        total_index += $m$.count
        if total_index > $x$ then
          write $m$.{start,end,distance} into $w$.{start,end,distance}
```

図 3.8: 反射アルゴリズム

ケットの送信先アドレスが統計 H のエントリと一致すれば，総パケット数のランダムなインデックスにおけるマーク情報を書き込む．

3.5.3 再構築アルゴリズム

反射ホストから被害者までの経路は再構築しない方が効率が良い．しかし明示的な反射の有無のマークで確率パケットマーク方式の意味論を変えると攻撃者に悪用される可能性がある．したがって，再構築アルゴリズムでは反射ホストを受信パケットから推測する必要がある．すなわち，送信元アドレスと $start$ フィールドが一致するパケットがあれば，それは反射ホストからのパケットである．そのパケットの $distance$ フィールドより小さい距離は，通信上詐称されていない経路で，わざわざ再構築する必要はない．しかし，この推測を全面的に信用してはならない．なぜならば，低い確率とはいえ，攻撃者も送信元アドレスと $start$ フィールドを一致させたマークをパケットに埋め込むかもしれないからである．したがって，この推測は参考程度に提供する必要がある．以上を考慮した再構築アルゴリズムを図 3.9に示す．

```
Path reconstruction procedure at vicitim $V$:
  let $G$ be a tree with root $V$
  let edges in $G$ be tuples (start, end, distance)
  let $H$ be a hashtable
  let entries in $H$ be tuples (reflector, distance)
  for each packet $w$ from attacker
    if $w$.start = $w$.source then
      put entry($w$.start, $w$.distance) into $H$
    if $w$.distance = 0 then
      insert edge($w$.start, $V$, 0) into $G$
    else
      insert edge($w$.start, $w$.end, $w$.distance) into $G$
  remove any edge$(x, y, d)$ with $d \neq$ distance from $x$ to $V$ in $G$
  extract path$(R_i..Rj)$ by enumeratin acyclic path in $G$
  show all entries(reflector, distance) in $H$
```

図 3.9: 再構築アルゴリズム

再構築アルゴリズムは被害者ホストで動作する．再構築アルゴリズムの挙動は，反射ホストの一覧を作成する以外，既存の辺サンプリングと同じである．これは反射アルゴリズムによって確率的パケットマーク方式の意味論が失われずに再構築アルゴリズムも記述できていることを意味する．反射ホストの一覧は，入力パケットの送信元アドレスと $start$ フィールドが一致したアドレスの集合である．

3.6 本章のまとめ

本章では，確率的パケットマーク方式による反射撹乱攻撃にも対応した逆探知系 RPPM の設計について述べた．まず設計に際し，近似逆探知問題を定義し，現実的な仮定を設けた．次に既存の確率的パケットマーク方式の問題点である反射ホスト問題と符合化問題を明らかにした．そして反射ホスト問題の対応方法として，反射ホストでマーク情報を反射する方法の詳細を示した．また，符合化問題の対応方法として，IP オプションを利用する方法の詳細を示した．本章の最後では，逆探知系 RPPM の基本アルゴリズムであるマークアルゴリズム，反射アルゴリズム，再構築アルゴリズムについて詳しく述べた．

次章では，本章で述べた設計をもとに実装した逆探知系 RPPM の詳細を述べる．

第 4 章

逆探知系 RPPM の実装

本章では，確率的パケットマーク方式による反射撹乱攻撃にも対応した逆探知系 RPPM の実装の詳細を述べる．まず実装環境と実装概要について述べる．そして，逆探知系 RPPM を構成する共通部，マーク部，反射部，再構築部の実装についてプログラムコードを交えて説明する．最後に，実装した逆探知系 RPPM の運用方法を示す．

4.1 実装環境

逆探知系 RPPM は，以下の理由から，Linux2.4.x 系カーネル [17] に統合された Netfilter と IPtables[18] の拡張として実装した．

- Linux は，ソースコードが公開され，カーネルを変更できる．
- Linux は，サーバ用の OS で 25%のシェアを有しており [16]，適用範囲が広いと考えられる．
- Netfilter と IPtables は，他のソースコードが公開されているフィルタリング機構と比べて，拡張が容易である．

4.1.1 Netfilter 概要

Netfilter はカーネル空間で動作するパケットフィルタリング機構で，Linux2.4.x 系カーネルから導入された．Netfilter は，IP 層で以下の 5 つのポイントでフック処理を行える．また，このフックポイントを図 4.1と表 4.1示す．

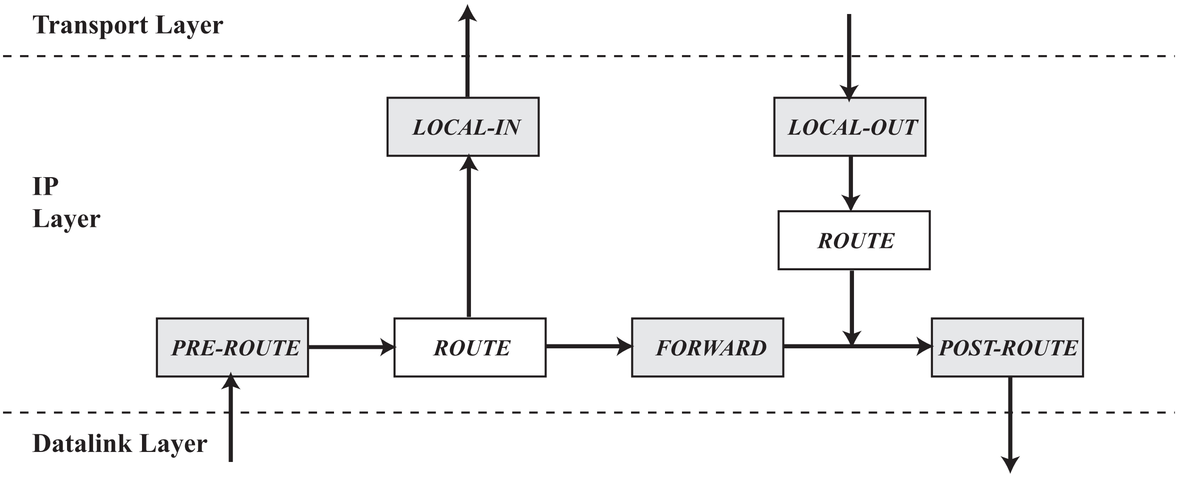


図 4.1: Netfilter の 5 つのフックポイント

これら 5 つのフックポイントでパケットに対する処理を実行するためには，以下に示すテーブルとターゲットを実装する必要がある．

テーブル

テーブルは，フックポイントをターゲットに対して提供する．デフォルトで用意されているテーブルとしては，ファイアウォール用の `filter`，NAT 用の `nat` がある．

独自のテーブルを開発した場合，カーネルモジュールの初期化時に，図 4.2に示

表 4.1: Netfilter の 5 つのフックポイント

フックポイント	説明
`PRE_ROUTING`	IP 層が下位層からパケットを受け取り，IP チェックサムの計算直後
`LOCAL_IN`	ルーティングが終了し，上位層にパケットを渡す直前
`FORWAARD`	ルーティングが終了し，別のインタフェースに中継する直前
`LOCAL_OUT`	IP 層が上位層からパケットを受け取った直後
`POST_ROUTING`	下位層にデータを渡す直前

す `ipt_register_table` 関数を用いて，テーブルを登録する．引数で用いられる `ipt_table` 構造体は，テーブル名，有効なフックポイントを含む．同時に，図 4.2 に示す `nf_register_hook` 関数を用いて，各フックポイントで登録されたターゲットの実行手順も登録する．引数で用いられる `nf_hook_ops` 構造体は，フックポイントで動作する関数のポインタ，実行の優先度を含む．

```
#include <linux/netfilter.h>
#include <linux/netfilter_ipv4/ip_tables.h>
int  ipt_register_table(struct ipt_table *);
int  nf_register_hook(struct nf_hook_ops *);
```

図 4.2: Netfilter のテーブル登録関数

ターゲット

ターゲットは，登録されたフックポイントにおいて，パケットに対して変更を加えたりフィルタリングを行う．デフォルトで用意されているターゲットとしては，パケットにターゲットの適用を免除する `ACCEPT`，パケットを破棄する `DROP` などがある．

独自のターゲットを開発した場合，カーネルモジュールの初期化時に，図 4.3 に示す `ipt_register_target` 関数でターゲットを登録する．引数で用いられる `ipt_target` 構造体は，ターゲット名，パケット処理関数のポインタ，ターゲット使用時のチェック関数のポインタ，ターゲット削除時に動作する関数のポインタを含む．

4.1.2 IPtables 概要

IPtables は，Linux カーネルに組み込まれた Netfilter と協調し，ユーザ空間で動作するパケットフィルタリングツールである．独自に開発した Netfilter のターゲットを

```
#include <linux/netfilter_ipv4/ip_tables.h>
int  ipt_register_target(struct ipt_target *);
```

図 4.3: Netfilter のターゲット登録関数

IPtables で利用するためには，カーネルモジュールの登録とは別に，共有ライブラリでも図 4.3に示す `register_target` 関数でターゲットを登録する必要がある．

```
#include <iptables.h>
void  register_target(struct iptables_target *);
```

図 4.4: IPtables のターゲット登録関数

ターゲットを利用するためのの共有ライブラリをインストール後，図 4.5に示すコマンドを実行する．`-A` オプションによって，任意のフックポイントに複数のターゲットを指定できる．またマッチング条件によって，ターゲットを実行の条件分岐もできる．

```
root# iptables -t <テーブル名> -A <フックポイント> \
      -j <ターゲット名> -m <マッチング条件>
```

図 4.5: `iptables` コマンドの実行

4.2 実装概要

逆探知系 RPPM の実装項目を表 4.2に示す．逆探知系 RPPM は，共通部，マーク部，反射部，再構築部から成る．共通部には `traceback` テーブルと `rppm_ipoptions` オブジェクト，マーク部には `RPPM_MARK` ターゲット，反射部には `RPPM_REFLECT` ターゲット，再構築部には `RPPM_RECONSTRUCT` ターゲットを含む．また，この実装の概要を図 4.6に示す．

4.3 共通部

共通部は，逆探知用ターゲットの実行基盤である `traceback` テーブルと，IP オプション RPPM 操作のライブラリである `rppm_ipoptions` オブジェクトから成る．

表 4.2: 逆探知系 RPPM の実装項目

構成	要素	説明
共通部	`traceback` テーブル	逆探知用ターゲットの実行基盤
	`rppm_ipoptions` オブジェクト	IP オプション RPPM 操作のライブラリ 各ターゲットとリンクして使用
マーク部	`RPPM_MARK` ターゲット	マークアルゴリズムの実行 `POST_ROUTING` で動作
反射部	`RPPM_REFLECT` ターゲット	反射アルゴリズムの実行 `LOCAL_IN` と `LOCAL_OUT` で動作
再構築部	`RPPM_RECONSTRUCT` ターゲット	再構築アルゴリズムの実行 `PRE_ROUTING` で動作

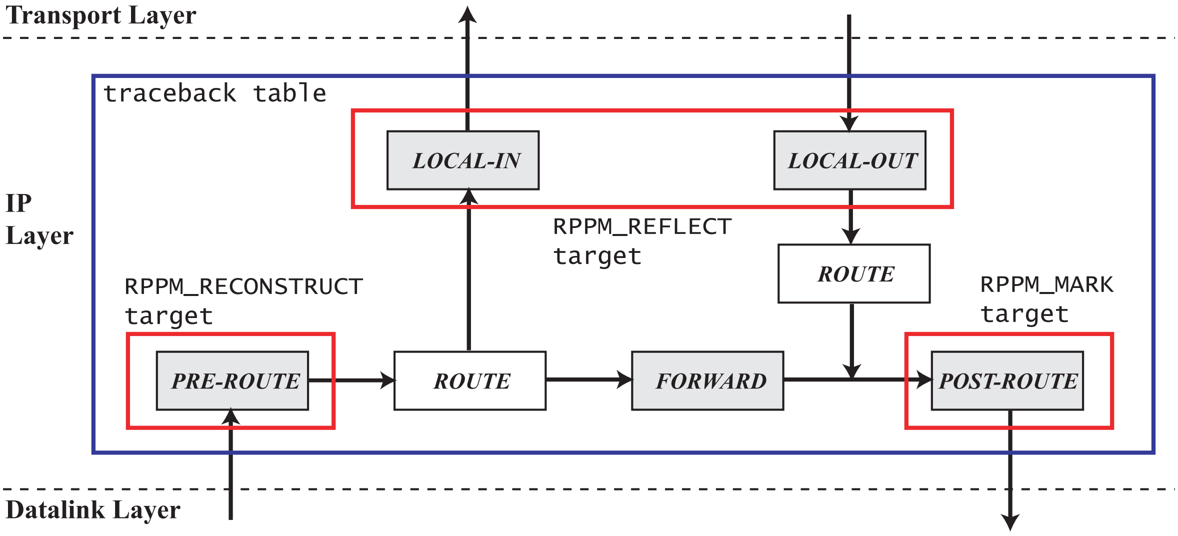


図 4.6: 逆探知系 RPPM の実装概要

4.3.1 traceback テーブル

traceback テーブルは，カーネルモジュール iptable_traceback.o から成り，逆探知用ターゲットを実行する基盤である．

traceback テーブルでは，有効なフックポイントを図 4.7に示す FILTER_VALID_HOOKS マクロで定義した．有効なフックポイントは PRE_ROUTING，LOCAL_IN，LOCAL_OUT，POST_ROUTING の 4 個所である．

```
#define FILTER_VALID_HOOKS ( \
        (1 << NF_IP_LOCAL_IN)     | (1 << NF_IP_LOCAL_OUT)  | \
        (1 << NF_IP_POST_ROUTING) | (1 << NF_IP_PRE_ROUTING) )
```

図 4.7: FILTER_VALID_HOOKS マクロ

また，traceback テーブルの Netfilter 中のフック処理優先度は，図 4.8に示す NF_IP_PRI_TRACEBACK マクロで定義した．フック処理優先度 pri の範囲は-300$< pri <$300 である．pri が大きいほどフック処理は後回しになる．traceback テーブルは，他のテーブルの処理と比較して遅い方がパケットマークを上書きされないので都合がよいため，pri を大きく設定した．

```
#define NF_IP_PRI_TRACEBACK 150
```

図 4.8: NF_IP_PRI_TRACEBACK マクロ

4.3.2 rppm_ipoptions オブジェクト

rppm_ipoptions オブジェクトは，rppm_ipoptions.o から成る．このオブジェクトは，マーク用 IP オプションに関係する操作をまとめており，ターゲット内で IP オプション RPPM を操作する場合にリンクして用いる．

IP オプション RPPM のタイプは，図 4.9に示す IPOPT_RPPM マクロで定義した．これは，フラグメント時にコピーせず，クラスはコントロールパケットで，番号は未使用の 28 を使用する，という意味である．

```
#include "rppm_ipoptions.h"
#define IPOPT_RPPM     (28 | IPOPT_CONTROL)
```

図 4.9: IPOPT_RPPM マクロ

そして，IP オプション RPPM を格納する `ipopt_rppm` 構造体の定義を図 4.10に示す．`type` は上記の `IPOPT_RPPM` が代入され，`len` は 12 バイトとなる．また，`flags` は未定義で，現在は IP ヘッダのパディングとして用いる．

```
#include "rppm_ipoptions.h"
struct ipopt_rppm {
  u_int8_t  type;          /* IP オプション RPPM のタイプ */
  u_int8_t  len;           /* IP オプション RPPM のデータ長 */
  u_int8_t  flags;         /* 未定義のフラグ */
  u_int8_t  distance;      /* 辺サンプリングの距離 */
  u_int32_t edge_addr[2];  /* 辺サンプリングの始点と終点 */
};
```

図 4.10: `ipopt_rppm` 構造体

また，パケットに対して IP オプション RPPM の検査，新規書込，上書，取得，距離増加，終点ルータアドレス追加を行う関数の定義を図 4.11に示す．

```
#include "rppm_ipoptions.h"
/* IP オプション RPPM の検査 */
check_ipopt_t  check_ipopt_rppm(struct sk_buff *);
/* IP オプション RPPM の新規書込 */
void  write_ipopt_rppm(struct sk_buff *, struct ipopt_rppm *);
/* IP オプション RPPM の上書 */
void  overwrite_ipopt_rppm(struct sk_buff *, struct ipopt_rppm *);
/* IP オプション RPPM の取得 */
void  read_ipopt_rppm(struct sk_buff *, struct ipopt_rppm *);
/* 辺サンプリングの終点を追加 */
void  write_ipopt_rppm_edge_end(struct sk_buff *, u_int32_t);
/* 距離を増加 */
void  increment_ipopt_rppm_distance(struct sk_buff *);
```

図 4.11: IP オプションを操作する関数

4.4 マーク部

マーク部は，3.5.1節で述べたマークアルゴリズムを実行する `RPPM_MARK` ターゲットから成る．

4.4.1 RPPM_MARK ターゲット

RPPM_MARK ターゲットは，カーネルモジュール ipt_RPPM_MARK.o と，ユーザ共有ライブラリ libipt_RPPM_MARK.so から成る．また動作するフックポイントは POST_ROUTING である．フックポイント POST_ROUTING で動作することによって，上位層から受け取ったパケットも，IP 層で中継されたされたパケットも同様にマークできる．

実際に RPPM_MARK ターゲットを利用するためには，スーパーユーザで図 4.12に示すコマンドを実行する．RPPM_MARK ターゲットは，マークアルゴリズムの動作のために，確率 prob を引数にとり，範囲は 0<prob<100 である．また，評価用にアドレス address を引数にとれるが，省略すればパケットが送信されるインタフェースの IP アドレスを取得し，自動的に該当インタフェースの IP アドレスをマークする．

```
root# iptables -t traceback -A POSTROUTING \
      -j RPPM_MARK --prob <確率> --address <アドレス>
```

図 4.12: RPPM_MARK ターゲットの利用

4.5 反射部

反射部は，3.5.2節で述べた反射アルゴリズムを実行する RPPM_REFLECT ターゲットから成る．

4.5.1 RPPM_REFLECT ターゲット

RPPM_REFLECT ターゲットは，カーネルモジュール ipt_RPPM_REFLECT.o と，ユーザ共有ライブラリ libipt_RPPM_REFLECT.so から成る．また動作するフックポイントは LOCAL_IN と LOCAL_OUT である．

RPPM_REFLECT ターゲットを利用するためには，スーパーユーザで図 4.13に示すコマンドを実行する．フックポイント LOCAL_IN では，パケットマークの統計をとるために，引数 smooth と cycle をとる．smooth は平滑化定数で，範囲は 0<smooth<100 である．cycle は統計更新周期 (秒) で，範囲は cycle>0 である．なお，平滑化定数および統計更新周期の詳しい指標は3.3.1節で述べた．また，フックポイント LOCAL_OUT では引数は必要ない．

RPPM_REFLECT ターゲットを利用する反射ホストにとって，攻撃者が被害者と詐称して送信する要求パケットと，反射ホストが被害者に送信する応答パケットは，悪意のない 1 組のコネクションにみえる．したがって反射ホストでは，攻撃者が詐称した発信元アドレスをキーとして，ルータにマークされた情報を組にして統計をとればよい．RPPM_REFLECT ターゲットでは，発信元アドレスとマーク情報の組をハッシュを使って

```
root# iptables -t traceback -A INPUT  \
      -j RPPM_REFLECT --smooth <平滑化定数> --cycle <周期>
root# iptables -t traceback -A OUTPUT \
      -j RPPM_REFLECT
```

図 4.13: `RPPM_REFLECT` ターゲットの利用

格納する．各ハッシュエントリは図 4.14に示す `rppm_reflect_htable_element` 構造体に格納される．

```
#include "ipt_RPPM_REFLECT_core.h"
struct rppm_reflect_htable_element {
  u_int32_t  ipv4_addr;                 /* 発信元アドレス */
  struct rppm_reflect_htable_distance {
    struct rppm_reflect_htable_path {
      u_int32_t  edge_addr[2];          /* ルータアドレス */
      u_int32_t  count[2];              /* パケット数 */
    }  path[MAX_PATH_HASH];
  }  distance[MAX_DISTANCE];
  unsigned int  total_count[2];         /* 全体パケット数 */
  unsigned long last_jiffies;           /* 前回更新時間 */
  struct ipt_rppm_reflect_target_info  *targinfo;
};
```

図 4.14: `rppm_reflect_htable_element` 構造体

`RPPM_REFLECT` ターゲットでは，コネクションの応答パケットに対してマーク情報を反射する．したがって，IP 層より上位層のコネクションを追跡する必要がある．Netfilter では，`ip_conntrack` モジュールからコネクション追跡ライブラリが提供されている．実際には図 4.15に示す `ip_conntrack_get` 関数でパケットのコネクション状態を調べられる．`ip_conntrack_info` 列挙型変数が `IP_CT_IS_REPLY` より大きければコネクションは上位層のリプライを示す．

4.6 再構築部

再構築部は，3.5.3節で述べた再構築アルゴリズムを実行する `RPPM_RECONSTRUCT` ターゲットから成る．

```
#include <linux/netfilter_ipv4/ip_conntrack.h>
struct ip_conntrack *  ip_conntrack_get(struct sk_buff *,
                                        enum ip_conntrack_info *);
```

図 4.15: ip_conntrack_get 関数

4.6.1 RPPM_RECONSTRUCT ターゲット

RPPM_RECONSTRUCT ターゲットは，カーネルモジュール ipt_RPPM_RECONSTRUCT.o と，ユーザ共有ライブラリ libipt_RPPM_RECONSTRUCT.so から成る．また動作するフックポイントは PRE_ROUTING である．

実際に RPPM_RECONSTRUCT ターゲットを利用するためには，スーパーユーザで図 4.17 に示すコマンドを実行する．RPPM_REFLECT ターゲットでは，収集期間 period を引数にとり，収集期間ごとに再構築に必要なパケットの集計を 0 でリセットする．

```
root# iptables -t traceback -A PREROUTING \
      -j RPPM_RECONSTRUCT --period <収集期間>
```

図 4.16: RPPM_RECONSTRUCT ターゲットの利用

RPPM_RECONSTRUCT ターゲットでは，被害者を根とする有向グラフを再構築し，深さ優先探索で各経路を表示する．また，表示には Linux の proc ファイルシステムを用いる．これは，パケットはカーネルモジュール内で集計されるため，集計結果だけをユーザ空間にコピーすればメモリを節約できるからである．実際には，図 4.17に示すように/proc/net/rppm_reconstructed_path を開くと再構築された経路の一覧と潜在的な反射ホストの一覧が表示される．

```
root# cat /proc/net/rppm_reconstructed_path
[0] 192.168.1.2 [1] 192.168.1.1 [2] 192.168.2.2
[0] 192.168.1.3 [1] 192.168.1.1 [2] 192.168.2.2
Rf: 192.168.1.2(d=0)
Rf: 192.168.1.3(d=0)
```

図 4.17: 再構築された経路の表示

4.7 運用方法

本節では，ルータ，反射ホスト，被害者のそれぞれで，逆探知系 RPPM の実装を運用する方法について述べる．なお，本実装は，`http://www.ht.sfc.keio.ac.jp/~nick/rppm/` からダウンロードできる．

ルータ

ルータでは，`RPPM_MARK` ターゲットを用いて図 4.12に示すコマンドを実行し，確率的パケットマークを行う．なお，ルータ間で確率引数は統一することが望ましい．

反射ホスト

反射ホストでは，`RPPM_REFLECT` ターゲットを用いて図 4.13に示すコマンドを実行し，マーク情報を反射する．また，`RPPM_MARK` ターゲットを用いて図 4.12に示すコマンドも実行し，確率的パケットマークを行い，擬似的にルータの役割をすることで逆探知を容易にする．反射ホストは，攻撃者に狙われるだけで，あらゆるインターネット上のサーバが候補になれる．特に DNS サーバ，Gnutella ホストは注意が勧告されている [7, 23]．

被害者

被害者では，`RPPM_RECONSTRUCT` ターゲットを用いて図 4.17に示すコマンドを実行し，多数の受信パケットから攻撃経路を再構築する．例えば収集期間に 1 日を設定し，IDS などで DoS 攻撃を検知した場合，当日の再構築された経路の一覧と反射ホストの候補を閲覧できる．

4.8 本章のまとめ

本章では，確率的パケットマーク方式による反射撹乱攻撃にも対応した逆探知系 RPPM の実装の詳細を述べた．まず実装環境と実装概要について述べ，逆探知系 RPPM を構成する共通部，マーク部，反射部，再構築部の実装についてプログラムコードを交えて説明した．本章の最後では，実装した逆探知系 RPPM の運用方法を示した．

次章では，実装した逆探知系 RPPM の定量的評価と評価結果の考察，類似研究との機能比較を行う．

第 5 章

逆探知系 RPPM の評価

本章では，逆探知系 RPPM に対して，定量的手法と定性的手法を用いて評価し，有効性を実証する．定量的評価では，まず反射アルゴリズムが確率的パケットマーク方式に及ぼす影響について測定し，確率的パケットマーク方式の意味論を損なわないことを示す．また実装した各機能が実際に要する時間を測定し，基本性能を把握する．さらに近似逆探知問題の解空間を把握し，現実的であることを示す．定性的評価では，関連研究の中から特に類似する確率的パケットマーク方式を採用した研究を取り上げ，機能性の観点から比較評価を行う．

5.1 定量的評価

本節では，逆探知系 RPPM に対して，定量的手法を用いて評価し有効性を実証する．まず，反射アルゴリズムが確率的パケットマーク方式に及ぼす影響について測定し，確率的パケットマーク方式の意味論を損なわないことを示す．また，実装した各機能が実際に要する時間を測定し，今後の実装の最適化の参考とする．

5.1.1 反射アルゴリズムが確率的パケットマーク方式に及ぼす影響

本節では，反射アルゴリズムの実装が確率的パケットマーク方式に及ぼす影響を定量的に評価する．反射アルゴリズムは，確率的パケットマーク方式の意味論に影響を及ぼさないことが望ましい．すなわち，反射の有無に関わらず，被害者を根とする有向グラフ上の辺をサンプリングするために必要なパケット数が変化しないことを，測定で定量的に示す必要がある．

測定環境と測定方法

反射アルゴリズムの影響の測定に利用したホストを表 5.1に示す．また，表 5.1で示したホストを使用して構築した測定環境を図 5.1に示す．なお，攻撃者 A，ルータ R，反射ホスト Rf，被害者 Vの間は 100BaseTX のネットワークで接続され，外部との接続はない．

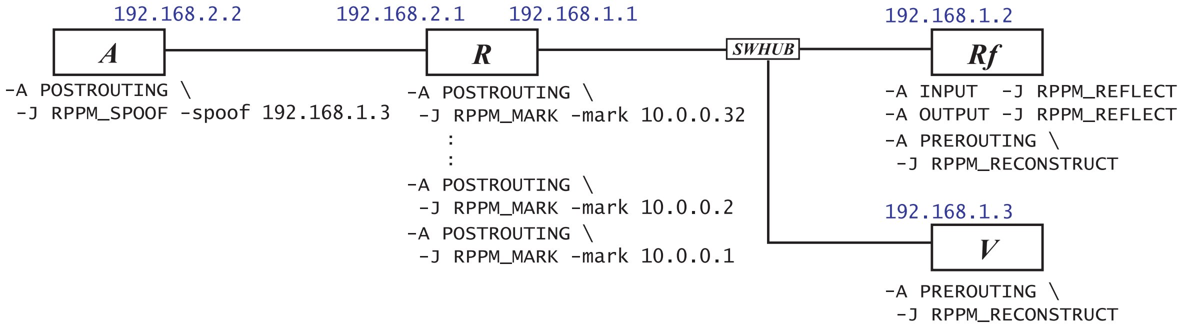
表 5.1: 反射アルゴリズムによる影響の測定環境

ホスト	CPU	メモリ	カーネル	Netfilter
攻撃者 (A)	Pentium 200MHz	64Mbyte	ver.2.4.12	ver.2.4
ルータ (R)	AMD-K6 350MHz	96Mbyte	ver.2.4.12	ver.2.4
反射ホスト (Rf)	Pentium 133MHz	32Mbyte	ver.2.4.12	ver.2.4
被害者 (V)	Celeron 400MHz	128Mbyte	ver.2.4.12	ver.2.4

この環境で，距離 $d(1 \leq d \leq 32)$ の経路の再構築に必要なパケット数を反射の前後で測定する手順を以下に述べる．なお最大距離を 32 とした理由は，ほとんど全てのインターネット上の経路は 32 ホップ以下である [8] という報告に基づく．

(1) まず攻撃者 A は，評価用に作成した `RPPM_SPOOF` ターゲットを用いて送信元アドレスを被害者 Vと詐称し，`ping` コマンドで ICMP メッセージを毎秒 1000 パケット送信する．

(2) ルータ Rは，`RPPM_MARK` ターゲットを用いて擬似的に 32 ホップ分のルータのマークアルゴリズムを実行する．この際 `RPPM_MARK` ターゲットの引数`prob` は 5 で統一する．



```
-A POSTROUTING \
 -J RPPM_SPOOF -spoof 192.168.1.3
```

```
-A POSTROUTING \
 -J RPPM_MARK -mark 10.0.0.32
   :
   :
-A POSTROUTING \
 -J RPPM_MARK -mark 10.0.0.2
-A POSTROUTING \
 -J RPPM_MARK -mark 10.0.0.1
```

```
-A INPUT  -J RPPM_REFLECT
-A OUTPUT -J RPPM_REFLECT
-A PREROUTING \
 -J RPPM_RECONSTRUCT
```

```
-A PREROUTING \
 -J RPPM_RECONSTRUCT
```

図 5.1: 反射アルゴリズムによる影響の測定環境

(3) 反射ホスト Rf は，`RPPM_REFLECT` ターゲットを用いて A からの攻撃を V に反射する．この際 `RPPM_REFLECT` ターゲットの引数 `smooth` は 30 とする．また，`cycle` の値は，式に α=0.3，P=2，F=1000，p=0.05，d=32 を代入して $T>33$(秒) が得られるため，少し多めに `cycle` は 40 秒とする．
また，反射前のトラヒックに対して，評価用の `RPPM_RECONSTRUCT` ターゲットを用いて距離 $d(1 \leq d \leq 32)$ の経路の再構築に必要なパケット数を測定する．

(4) V は，距離 d のルータがマークした後の Rf が反射したトラヒックに対して，評価用の `RPPM_RECONSTRUCT` ターゲットを用いて距離 $d(1 \leq d \leq 32)$ の経路の再構築に必要なパケット数を測定する．

測定結果と考察

上記の測定環境と測定方法で，各距離において反射前後に 1000 回ずつ測定し，それぞれの平均と 95%信頼区間の上限を求めた．この結果のグラフを図 5.2に示す．

図 5.2から反射前のトラヒックで再構築に必要なパケット数の 95%信頼区間の上限値と，反射後のトラヒックで再構築に必要なパケット数の 95%信頼区間の上限値とはほぼ同じ値を示していることがわかる．同様に，平均値もほぼ同じ値を示していることがわかる．

したがって，反射アルゴリズムは 3.2.2節で述べた反射撹乱攻撃の際のマーク情報を喪失する反射ホスト問題を防ぐと共に，反射の有無によって確率的パケットマーク方式の意味論にほとんど影響を及ぼさないことが示せた．

5.1.2 各機能の基本性能測定

本節では，逆探知系 RPPM で実装したマーク部，反射部，再構築部の各機能が実際に要する時間を測定し基本性能を把握し，今後の実装の最適化の参考とする．

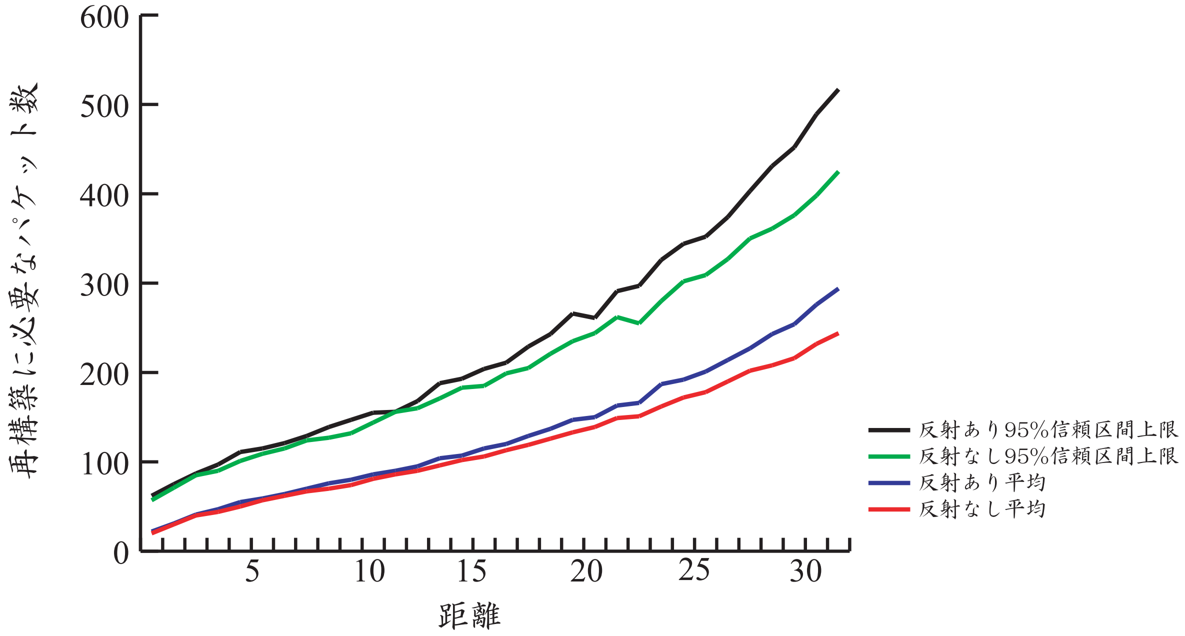


図 5.2: 反射の有無による再構築に必要なパケット数の変化

測定環境と測定方法

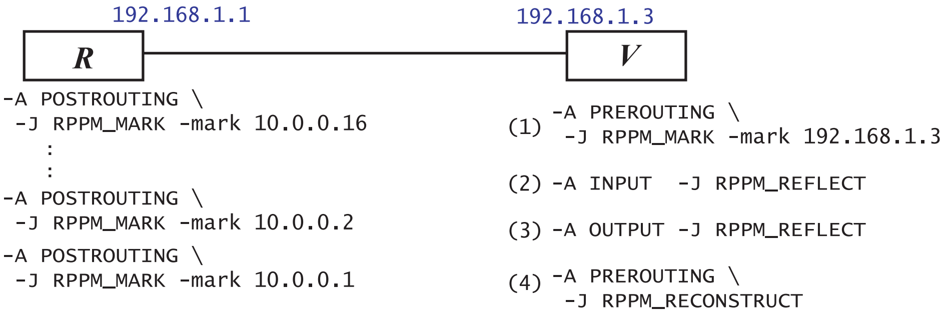
基本性能の測定に利用したホストを表 5.2に示す．また，表 5.2で示したホストを使用して構築した測定環境を図 5.3に示す．なお，疑似ルータ Rと測定ホスト Vの間は 100BaseTX のネットワークで接続され，外部との接続はない．

表 5.2: 各機能の基本性能測定環境

ホスト	CPU	メモリ	カーネル	Netfilter
疑似ルータ (R)	AMD-K6 350MHz	96Mbyte	ver.2.4.12	ver.2.4
測定ホスト (V)	Celeron 400MHz	128Mbyte	ver.2.4.12	ver.2.4

この環境で，Vが経路上の距離 d=16 の地点で `RPPM_MARK` ターゲット，`RPPM_REFLECT` ターゲット，`RPPM_RECONSTRUCT` ターゲットを，それぞれ適用した場合の基本性能を測定する手順を以下に述べる．

まず，Rは Vに対して，`RPPM_MARK` ターゲットを用いて擬似的に 16 ホップ分のルータのマークアルゴリズムを実行し，`ping` コマンドを利用して 1000 パケット送信する．この際`MARK` ターゲットの引数`prob` は 5 で統一する．なお距離を 16 とした理由は，ほとんどのインターネット上のホストにとって距離の最頻値平均は 16 ホップである [8] という報告に基づく．



```
-A POSTROUTING \
 -J RPPM_MARK -mark 10.0.0.16
   :
   :
-A POSTROUTING \
 -J RPPM_MARK -mark 10.0.0.2
-A POSTROUTING \
 -J RPPM_MARK -mark 10.0.0.1
```

```
(1) -A PREROUTING \
     -J RPPM_MARK -mark 192.168.1.3
(2) -A INPUT  -J RPPM_REFLECT
(3) -A OUTPUT -J RPPM_REFLECT
(4) -A PREROUTING \
     -J RPPM_RECONSTRUCT
```

図 5.3: 各機能の基本性能測定環境

(1) `RPPM_MARK` ターゲットでは，各出力パケットごとの全体処理時間と，マークするために内部で利用する以下の 5 種類の処理に要する時間を測定する．

- 確率 p でマークすることを決定し，受信パケットにマーク情報が存在しないため，新規に $start$ フィールドを書き込む処理
- 確率 p でマークすることを決定し，受信パケットにマーク情報が存在するため，$start$ フィールドを上書する処理
- 確率 p でマークしないことを決定し，受信パケットにマーク情報が存在するため，終点ルータアドレスを追加する処理
- 確率 p でマークしないことを決定し，受信パケットにマーク情報が存在するため，距離のみを増加する処理
- 確率 p でマークしないことを決定し，受信パケットにマーク情報が存在しないため，何もしない処理

引数`prob` は Rと統一して 5 とする．

(2) `RPPM_REFLECT` ターゲット (入力) では，各入力パケットごとの全体処理時間と，統計をとるために内部で利用する以下の 3 種類の処理に要する時間を測定する．

- マーク情報のないパケットを受信したので，統計全体の受信パケット数を増加する処理
- マーク情報のあるパケットを受信し，新たに統計のエントリを追加する処理
- マーク情報のあるパケットを受信し，既存の統計のエントリを更新する処理

引数`smooth` は 30 とする．また，`cycle` の値は，式にα=0.3，P=2，F=1000，p=0.05，d=16 を代入して T >14(秒) が得られるため，少し多めに`cycle` は 15 秒とする．

(3) `RPPM_REFLECT` ターゲット(出力)では，各パケットごとの全体処理時間と，マーク情報を反射するために内部で利用する以下の2種類の処理に要する時間を測定する．

  - 総パケット数のランダムなインデックスにおけるマーク情報が存在し，マークする処理
  - 存在せずマークしない処理

(4) `RPPM_RECONSTRUCT` ターゲットでは，以下の2種類の処理に要する時間を測定する．

  - 各入力パケットごとに集計するための全体処理時間
  - `/proc/net/rppm_reconstructed_path` を開いたときの出力を作成するカーネル内部の処理時間

測定結果と考察

上記の測定環境と測定方法で各ターゲットごとにPentiumカウンタを利用して1000回ずつ測定し，それぞれの平均と標準偏差を有効数字3桁まで求めた．この結果を表5.3に示す．

表 5.3: 基本性能の測定結果

ターゲット	測定項目	平均値 (microsec)	標準偏差 (microsec)	標本数
RPPM_MARK	新規マーク	5.47	0.656	20
	マーク上書	1.04	0.292	30
	終点ルータ追加	0.837	0.335	51
	距離増加	0.916	0.297	482
	マークなし	0.477	0.135	417
	全体	0.833	0.762	1000
RPPM_REFLECT(入力)	カウント増加	2.22	0.609	456
	エントリ追加	4.21	0.854	16
	エントリ更新	4.13	0.803	528
	全体	3.29	1.19	1000
RPPM_REFLECT(出力)	マークなし	570	550	396
	マークあり	618	558	604
	全体	599	555	1000
RPPM_RECONSTRUCT	パケット収集	2.25	0.587	1000
	再構築	372	20.6	1000

表 5.3から，特に `RPPM_REFLECT` ターゲットの出力で時間を要していることがわかる．この処理時間は，マーク情報の反射の有無にかかわらず処理時間を要するため，統計から総パケット数のランダムなインデックスにおけるマーク情報を検索するのに時間を要していることがわかる．このオーバーヘッドを最適化するためには，`RPPM_REFLECT` ターゲットの入力で統計情報を集計する際に，さらにメタ情報を加えることが考えられる．

また，`RPPM_MARK` ターゲットの新規マークがマークの上書きより多く時間を要する理由は，Linux の IP 層の実装で IP オプションを処理するためには，一度パケットのデータをコピーする必要があるためである．しかし，新規マーク処理自体の発生確率が低いため，問題にならない．

5.1.3 近似逆探知問題の解空間

本節では，近似逆探知問題の解空間を定量的に把握する．また，近似逆探知問題と完全逆探知問題の解に必要なパケット数を比較し，近似逆探知問題の解が現実的であることを示す．

測定環境と測定方法

近似逆探知問題の解空間を定量的に測定するにあたり，膨大な経路数を実機で再現するのは困難であるため，表 5.4に示すホスト上のシュミレーションで評価する．

表 5.4: 逆探知問題の解空間の測定環境

CPU	メモリ	カーネル
Celeron 400MHz	128Mbyte	ver.2.4.12

シュミレーションでは，距離 d=64，経路数 m を 10000 から 100000 まで 10000 ずつ増加させた際の，完全逆探知問題の解に必要なパケット数 N_eと近似逆探知問題の解に必要なパケット数 N_aの変化を測定する．d=64 とした理由は，逆探知系 RPPM の確率的パケットマーク方式は，5.1.1節の評価によって反射の有無に関わらず動作することが示せたため，攻撃者が反射を 1 度用いれば，最悪の距離は 32 の倍である 64 となるためである．

測定結果と考察

上記の測定環境と測定方法で，各 m ごとに 100 回ずつ測定し，それぞれの平均を求めた．このシュミレーションの結果を図 5.4に示す．

ここで，m=1 の場合 $N_e = N_a$となり，N_aを 100 回測定した平均が 206 であった．そこで，N_eと N_aをそれぞれ切片 206 で線形近似すると，式 5.1と式 5.2が得られた．ただし $m \geq 1$ である．

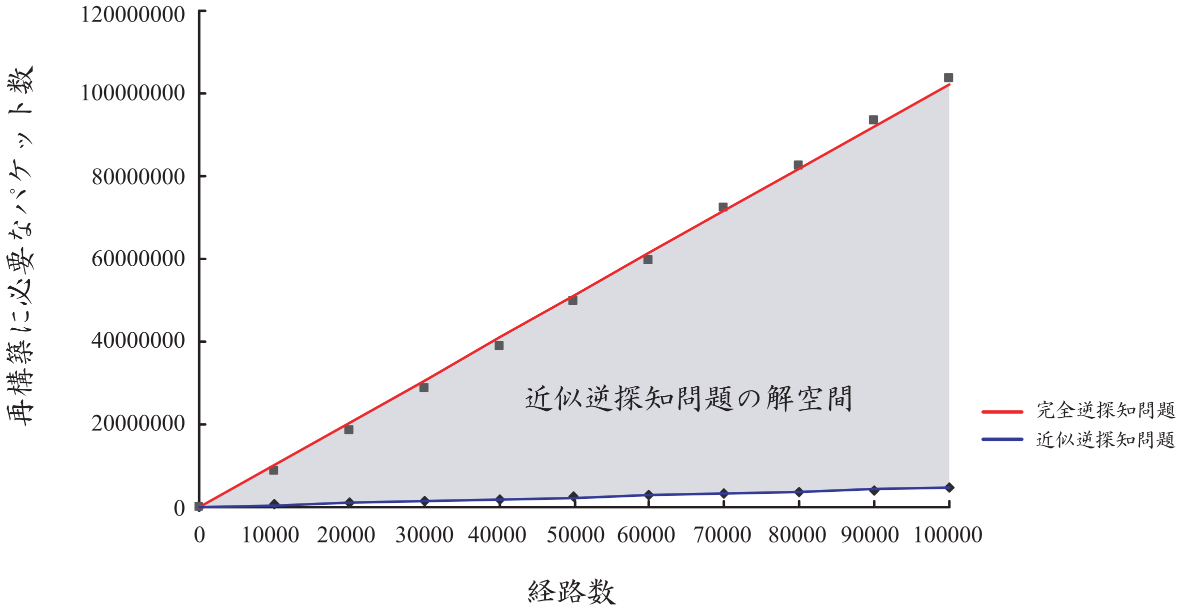


図 5.4: 近似逆探知問題の解空間の測定結果

$$N_e = 1020m + 206 \tag{5.1}$$
$$N_a = 47m + 206 \tag{5.2}$$

m が小さいときにはこのような単純な線形増加はしないが，m が大きくなれば，ほぼ線形に増加する．そして，N_e と N_a の近似直線に囲まれた領域が近似逆探知問題の解空間であるといえる．

また，近似逆探知問題では，m=100000 のとき N_a =4590000 であり，1 秒あたりの入力パケット数 F=1000 とすれば，約 1.2 時間の攻撃で少なくとも 1 つ経路を再構築できることを示している．これに対して，完全逆探知問題では，m=100000 のとき N_e=104000000 であり，F=1000 とすれば，約 28 時間の攻撃がないと全経路を再構築できないことを示している．実際の DoS 攻撃は数時間で終わることが多い [7] ので，完全逆探知問題の解は現実的でないことがわかる．したがって，近似逆探知問題は完全逆探知問題より現実的であるといえる．

5.2　定性的評価

本研究は，反射ホストに対応できる逆探知機構の設計と実装を目的としている．その際確率的パケットマーク方式を採用し，ネットワークとルータへの負荷を少なくし事後処理性をもたせた．

本節では，同様に確率パケットマーク方式を採用している類似研究[26](表中 PPM) と[30](表中 ARPPM) を取り上げ，定性的機能比較評価を行う．評価項目および評価結果を表 5.5に示す．

表 5.5: 類似研究との比較

評価項目	**PPM**	**ARPPM**	RPPM
反射ホスト対応	×	×	○
複数経路対応	×	○	○
管理コスト	低	高	低
経路再構築の計算量	$O(\sum_d S_d \Psi_d^k)$	$O(\sum_d S_d \Psi_d)$	$O(\sum_d S_d)$
符号化する場所	IP-ID	IP-ID	IP オプション
パケット毎の追加データ量	0byte	0byte	12byte
ICMP との親和性	×	×	○
公開実装の有無	×	×	○

類似研究との最も大きな差違は，PPM や ARPPM では反射ホストを定義と仮定から除外した設計を行っているが，本研究では反射ホストに対応した点である．反射ホストの攻撃の脅威が明らかな現在，大きな差違である．

また，PPM では辺サンプリングを提案して，複数の攻撃経路に対応しないことで，16bit の IP-ID フィールドに 72bit の辺サンプリングの情報をフラグメント化して符合化できた．ただし，複数の攻撃経路を考慮すると，経路再構築の計算量は$O(\sum_d S_d \Psi_d^k)$となり，計算量が大きく誤り率が高いことを 3.2.3節で述べた．一方，APPM ではIP-ID フィールドを利用したまま複数経路に対応するために，より大きいハッシュ空間を使ったフラグメント化を提案し，経路再構築の計算量を $O(\sum_d S_d \Psi_d)$ まで減らしたことを 3.2.3節で述べた．しかし，APPM ではインターネットマップを使うという前提条件を付加したため，管理コストを増大させた．このように先行研究では符号化の際に IP-ID フィールドにこだわったため，PPM では複数経路への対応を犠牲にし，APPM では管理コストを犠牲にした．

本研究では，符合化に IP オプションを利用し，パケット毎の追加データ量を 12byte に増やすことで複数経路に対応した．また，インターネットマップを前提としないため，管理コストは少ない．そして，経路再構築の計算量は$O(\sum_d S_d)$ まで減らせた．さらに，IP-ID フィールドへの符合化をやめた結果として，ICMP との親和性ができた．IP オプションの利用によってルータの負荷が若干増加するが，現在ほとんどのルータは IP オプションを処理できるほど強力になってきたため問題ない．

最後に，PPM や APPM は設計とシュミレーションによる評価だけであり，実際に動作する公開実装は存在しない．

以上をふまえると，本研究は反射ホストに対応した逆探知機構の設計と実装を実現したといえる．

5.3 本章のまとめ

本章では，逆探知系 RPPM の基本性能特性を把握し，測定結果およびその考察を今後の実装最適化の一助とするため，定量的および定性的量側面から評価を行った．

定量評価においては，反射アルゴリズムが既存の確率的パケットマーク方式の意味論を損なうことなく，反射ホスト問題に対応できることを示した．また，実装した各ターゲットの基本性能を把握した結果，`RPPM_REFLECT` ターゲットは最適化の可能性を示した．そして `RPPM_MARK` ターゲットは，新規にマークする際の負荷が若干高いが，事象の発生確率が低いため，実用上問題ないことも示した．さらに，膨大な量の攻撃経路数のシュミレーションでは，近似逆探知問題と完全逆探知問題の解に必要なパケット数を比較し，近似逆探知問題の解空間が現実的であることを示した．

一方，定性評価においては，逆探知系 RPPM の有する反射ホスト問題への対応と符合化問題への対応を加味し，既存システムとの親和性や公開実装の有無などの観点から，逆探知系 RPPM は反射ホストに対応した逆探知機構の設計と実装を実現したといえた．

第 6 章

おわりに

6.1 今後の課題

本研究では，反射に対応できる逆探知機構の設計と実装を目的として，逆探知系 RPPM を設計，実装，評価した．今後一層実用性を向上させるためには，以下の課題に取り組む必要がある．

再構築した経路の検証

本研究では，パケットを中継するルータは全て信頼することを仮定したが，実際には攻撃者がルータを不正に制御する可能性もある．逆探知の効率上オーバーレイネットワークを提案する研究 [32] があるが，むしろルータでパケットにマークする過程で改竄防止するためにオーバーレイネットワークは必要であると考える．パケットを中継する際に，ルータのアドレスをマルチパーティプロトコル [14] を利用して暗号化してマークすれば，n 台のルータのうち t 台までの不正ルータがいたとしても，攻撃経路を復元できる．

経路再構築計算の委譲

本研究では，DoS 攻撃を逆探知するには事後処理性が必要であることに触れたが，リアルタイムに逆探知できればなお良い．リアルタイムに逆探知することが難しい理由は，被害者が DoS 攻撃を受けている場合，ネットワーク割込が多発して負荷が高くなり，被害者の計算能力が低下するからである．ここで，IDS などで DoS 攻撃を検知してから被害者のネットワーク資源を浪費し尽くすには若干のタイムラグがある．この若干のタイムラグを利用して，より計算能力の高いホストに攻撃経路の再構築計算を委譲することも考えられる．

IPv6 への対応

本研究では，DoS 攻撃の逆探知機構を，最も普及している IPv4 に焦点をあてて実装した．なぜなら普及しているからこそ DoS 攻撃の対象となるからである．しかし今後の IP は v4 から v6 に移行すると考えられる．実際 IPv6 用の DoS 攻撃ツールは発見されている [21] ため，今後は v6 にも適応した DoS 攻撃の逆探知機構も実装する必要がある．

6.2 まとめ

本論文では，反射にも対応できるDoS攻撃逆探知機構の設計と実装について述べた．

まずDoS攻撃を発信元アドレスの詐称方法からランダム撹乱攻撃と反射撹乱攻撃に分類して，DoS攻撃の問題を明らかにした．そしてDoS攻撃に対する関連研究を紹介し，侵入検知システムとフィルタリングは本研究の目的である逆探知と共存関係にあることを述べ，逆探知の必然性を述べた．同時に既存の逆探知方式としてリンクテスト方式，ログ方式，ICMP方式，パケットマーク方式の4方式を比較し，確率的パケットマーク方式が反射撹乱攻撃に対応できる逆探知機構を実現できる可能性があることを示した．

設計に関しては，近似逆探知問題を定義し，現実的な仮定を設け，既存の確率的パケットマーク方式の問題点である反射ホスト問題と符合化問題を明らかにした．そして反射ホスト問題の対応方法として，反射ホストでマーク情報を反射する方法の詳細を示し，符合化問題の対応方法として，IPオプションを利用する方法の詳細を示した．

定量評価においては，反射アルゴリズムが既存の確率的パケットマーク方式の意味論を損なうことなく，反射ホスト問題に対応するために動作することを示した．また，実装した各ターゲットの基本性能を示し，今後の最適化の可能性を示した．さらに，膨大な量の攻撃経路数のシュミレーションでは，近似逆探知問題の解が現実的であることを示した．そして定性評価においては，逆探知系RPPMの有する反射ホスト問題への対応と符合化問題への対応を加味し，既存システムとの親和性や実装の有無などの観点から，逆探知系RPPMは反射ホストに対応した逆探知機構の設計と実装を実現したことが立証された．

今後は，再構築した経路の検証，経路再構築計算の委譲，IPv6への対応などを考慮し，より実用性の高い研究として進める．

謝辞

本研究を進めるにあたり，絶えず懇切丁寧な御指導を賜わりました，慶應義塾大学環境情報学部教授徳田英幸博士に深く感謝致します．また，本論文の副査として貴重な助言を頂いた，慶應義塾大学環境情報学部教授武藤佳恭博士，同学部教授萩野達也博士に深く感謝致します．

慶應義塾大学大学院政策・メディア研究科助教授西尾信彦博士，同研究科助教授戸辺義人博士，同研究科専任講師望月祐洋博士には，折りに触れ貴重な示唆や助言，御指導を頂きました．また，慶應義塾大学大学院政策・メディア研究科博士課程に在籍の中澤仁氏には，本論文執筆にあたり絶えざる励ましと御指導を賜りました．そして，慶應義塾大学徳田・村井・楠本・中村・南研究会の諸先輩方ならびに同研究会KMSF研究グループ，MCng研究グループ，ECN研究グループの方々には，本研究に関する様々な議論をして頂きました．

ここに，深い感謝の意を表します．

平成14年 1月 24日

参考文献

[1] S. M. Bellovin. *ICMP Traceback Messages*, 3 2000. Internet Draft: draft-bellovin-itrace-00.txt.

[2] B. L. Bowerman and R. T. O'Connell. *Time Series and Forecasting.* Duxbury Press, North Scituate, Massachusetts, 1979.

[3] R. Braden. *Requirements for Internet Hosts – Communication Layers*, 10 1989. RFC1122.

[4] H. Burch and B. Cheswick. Tracing anonymous packets to their approximate source. In *USENIX Systems Administration Conference (LISA)*, pp. 319–327, 2000.

[5] B. Cheswick and H. Burch. The Internet Mapping Project. `http://cm.bell-labs.com/who/ches/map/index.html`.

[6] Computer Emergency Response Team. *CERT Advisory CA-96.26 Denial-of-Service via pings*, 1996. `http://www.cert.org/advisories/CA-96.26.ping.html`.

[7] Computer Emergency Response Team. *CERT Advisory CA-2000-01 Denial-of-Service Developments*, 2000. `http://www.cert.org/advisories/CA-2000-01.html`.

[8] Cooperative Association for Internet Data Analysis. Skitter Analysis. `http://www.caida.org/tools/measurement/skitter/`.

[9] D. Dittrich. *An analysis of the "Shaft" distributed denial of service tool.* `http://staff.washington.edu/dittrich/misc/ddos/shaft.analysis.txt`.

[10] D. Dittrich. *The DoS Project's "trinoo" distributed denial of service attack tool.* `http://staff.washington.edu/dittrich/misc/ddos/trinoo.analysis.txt`.

[11] D. Dittrich. *The "stacheldraht" distributed denial of service attack tool.* `http://staff.washington.edu/dittrich/misc/ddos/stacheldraht.analysis.txt`.

[12] D. Dittrich. *The "Tribe Flood Network" distributed denial of service attack tool.* `http://staff.washington.edu/dittrich/misc/ddos/tfn.analysis.txt`.

[13] G. Everette and S. Jr. Exponential smoothing: The state of the art. *Journal of Forecasting*, 4(1):1–38, 1985.

[14] R. Gennaro, M. O. Rabin, and T. Rabin. Simplified VSS and Fast-track Multiparty Computations with Applications to Threshold Cryptography. In *ACM PODC '98*, 1998.

[15] L. Holst. On birthday, collector's, occupancy and other classical urn problems. *ISI International Statistical Review*, 54:15–27, 1986.

[16] International Data Corporation. `http://www.idc.com/`.

[17] The Linux Kernel Archives. `http://www.kernel.org/`.

[18] The Netfilter Project. `http://netfilter.samba.org/`.

[19] D. Moore, G. M. Voelker, and S. Savage. Inferring Internet Denial-of-Service Activity. In *USENIX Security Symposium*, 2001.

[20] Packet Monster. `http://www.sfc.keio.ac.jp/~keiji/backup/ids/pakemon/`.

[21] Packet Storm. `http://packetstorm.widexs.nl/`.

[22] V. Paxon. End-to-End Routing Behaivor in the Internet. *IEEE/ACM Transactions on Networking*, 5(5):601–605, 1997.

[23] V. Paxon. An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks. *ACM Computer Communiation Review*, 31, 7 2001.

[24] J. Postel. *Internet Control Message Protocol*, 9 1981. RFC792.

[25] J. Postel. *Internet Protocol*, 9 1981. RFC791.

[26] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical Network Support for IP Traceback. In *ACM SIGCOMM 2000*, pp. 295–306, 2000.

[27] D. Senie. *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing*, 5 2000. RFC2827.

[28] A. C. Snoeren, C. Partridge, L. A. Sanchez, and C. E. Jones. Hash-Based IP Traceback. In *ACM SIGCOMM 2001*, 2001.

[29] Snort. `http://www.snort.org/`.

[30] D. Song and A. Perrig. Advanced and Authenticated Marking Schemes for IP Traceback. In *IEEE INFOCOM 2001*, 2001.

[31] I. Stoica and H. Zhang. Providing guaranteed services without per flow mangagement. In *ACM SIGCOMM 1999*, pp. 89–94, 1999.

[32] R. Stone. Centertrack: An IP Overlay Network for Tracking DoS Floods. In *USENIX Security Symposium*, 2000.