

フィルタリングルールの自動更新による IP アドレス詐称対策の効率化

中野 学†

nakano@mlab.jks.ynu.ac.jp

内尾 文隆*

uchio@sys.wakayama-u.ac.jp

松本 勉†‡

tsutomu@mlab.jks.ynu.ac.jp

†横浜国立大学大学院環境情報学府 ‡環境情報研究院
〒240-8501 横浜市保土ヶ谷区常盤台 79-7

*和歌山大学システム工学部
〒640-8510 和歌山市栄谷 930 番地

あらまし IPアドレス詐称問題の有効な対策の一つとして、各々のネットワーク管理者がIPアドレスの詐称されたパケットを、パケットフィルタリングという手段により、外に流さないようにするという方法がある。そのためには、管理対象であるネットワーク内の変更を適切に反映する最新のフィルタリングルールを設定することが重要である。本稿では、ルーティングテーブルを参照して経路情報を利用することによってフィルタリングルールを自動的に更新する方法を提案する。

キーワード ルーティング, IP アドレス詐称, 経路情報, フィルタリング, DoS 攻撃

Effective Prevention of IP Spoofing by Updating Filtering Rule Automatically

Manabu NAKANO†

nakano@mlab.jks.ynu.ac.jp

Fumitaka UCHIO*

uchio@sys.wakayama-u.ac.jp

Tsutomu MATSUMOTO†

tsutomu@mlab.jks.ynu.ac.jp

†Graduate School of Environment and Information Sciences
Yokohama National University
79-7, Tokiwadai, Hodogaya, Yokohama, 240-8501, Japan

*Faculty of Systems Engineering
Wakayama University
930, Sakaedani, Wakayama, 640-8510, Japan

Abstract As one of the effective measures against IP address spoofing, there is a way each network administrator takes care not to pass outside the packet, as which the IP address is misrepresented, by the means of packet filtering. For that purpose, it is important to set up the newest filtering rule that reflects pertinently the change in the network that is the target of management. This paper proposes how to update the filtering rule automatically by using routing announcement with reference to routing tables.

Key words routing, IP spoofing, routing announcement, packet filtering, DoS attack

1. はじめに

今日のように国際的にネットワークが張り巡らされ、電子商取引などが発達しつつある世界においては、コンピュータシステムが常に攻撃の危険にさらされている。毎日のように、どこかの主要なネットワークや大企業のネットワークに接続したコンピュータが誰か不正に侵入されたというニュースが報じられている。しかも、このような侵入では侵入者をなかなか特定できない^[1]。

それぞれのコンピュータシステムの管理者が、これらの問題に対応するためには、膨大なコストや時間が必要となる。そのため、このような問題には効果的に対策を行う必要がある。

本論文では、この中でも DoS (Denial of Service : サービス妨害) 攻撃と併用して行われることのある IP アドレス詐称という問題を取り上げる。IP アドレス詐称の対策として、パケットフィルタリングによって IP ソースアドレスが詐称されたパケットが外部に出さないようにする際に、ルーティングテーブルから経路情報を取り入れて自動的にフィルタリングルールを更新することを提案し、そのための具体的方式を示す。この方式を Perl 言語によって実装して実証した結果についても報告する。

2. ルータの役割

まず、ルータによるパケットの経路制御 (ルーティング) とパケットフィルタリングにつき説明する。

2. 1 ルーティング

送信者の送信したパケットは、ルータ間を転送されながら受信者に届く。ルータがパケットを転送する時は、そのパケットに記載された宛先アドレス (Destination Address) に応じて、次にどのルータに転送すればよいのかを決める処理がルーティングである。ルータは周囲のネットワークの IP アドレスをルーティングテーブルというリストに経路情報として記録し、それに従ってパケットを転送する先を決定する。

経路情報の管理方法には、ルータが他のルータと情報を交換してルーティングテーブルを自動的に生成・更新するダイナミックルーティング (自動経路制御) と、管理者が手作業で設定するスタティックルーティング (静的経路制御) の二通りの方法がある。どちらの方法であっても、ルータは送られてきたパケットを経路情報の書かれたルーティングテーブルを参照して、その中から最適な経路に向けてパケットを転送する^[2]。

ダイナミックルーティングのプロトコルには、RIP (Routing Information Protocol) や OSPF (Open Shortest Path First) などがある。

2. 2 パケットフィルタリング

パケットフィルタリングとは、パケットのヘッダ情報によってそのパケットを通すか破棄するかを判断する処理である。IP パケットのソースアドレスや宛先アドレス、プロトコル番号、ポート番号などを監視し、管理者の決めるフィルタリングルールに従ってパケットの扱いを決定する。

3. DoS 攻撃と IP アドレス詐称

ここでは DoS 攻撃について説明し、パケットの送信元 IP アドレスを詐称する行為と DoS 攻撃の関係について述べる。

3. 1 DoS 攻撃

ネットワークにおける DoS 攻撃とは、ネットワーク上のサーバが提供するサービスを妨害することで、そのサービスを機能しなくさせる攻撃のことを示す。この攻撃の実行手段は大きく二つに分類できる。一つはサーバが接続されているネットワークやサーバそのものに対して過剰な負荷をかけることで、サービスを提供できなくする方法である。もう一つは、サーバアプリケーションの脆弱性 (セキュリティホール) を突くことで、サーバアプリケーションを正常に機能させなくする方法である。

近年では複数のサーバから DoS 攻撃用エージェ

ントを同時に起動させて攻撃する、DDoS (Distributed Denial of Service : 分散サービス妨害) 攻撃も存在する^[3]。

3. 2 過剰負荷型 DoS 攻撃

過剰な負荷をかける DoS 攻撃について少し詳しく説明する。

サーバは、クライアントからサービスの要求を受けたとき、サービスを提供する等の判断を行い、提供する場合に次の処理に移行する。攻撃者がサーバに何度もサービス要求を行うと、サーバはそのたびに上記の処理を行わなければならないため、サーバに余分な負荷がかかる。攻撃者はサーバがダウンするまで何度も要求を送り続けて負荷を与えることで、サーバにサービスを提供させなくすることができる。

またはサーバが接続されているネットワークに、そのネットワークが処理できるパケット数以上のパケットを送ることで、そのネットワークをダウンさせ、サービスを提供させないようにすることもできる。

このような攻撃に対しては、大量にサービスを要求するホストを IP アドレスで判断し、そこからの接続を遮断する方法が有効である。この方法によって、大量の要求をサーバに処理させず、処理に費やす負荷を削減することができる。

3. 3 IP アドレス詐称

IP アドレス詐称とは、ソースアドレスを偽ったパケットを生成・送信することである。IP アドレス詐称により身元を隠された場合、被害にあったホストから攻撃者を特定するのは困難である。なぜなら、パケットの持っている送信元の情報はソースアドレスだけであるため、IP アドレス詐称を伴う攻撃を受けた際に、その発信元を探索するには攻撃を受けたサーバに繋がる全ての経路について、ログを頼りに一つ一つ遡っていくといった方法が必要とされるからである。

この攻撃方法には、攻撃者が攻撃対象からのレ

スポンスを受け取ることが難しいという欠点もある。それは、ソースアドレスを詐称した要求を通信対象のホストはその要求に対するレスポンスを詐称された IP アドレスの場所に届けてしまうためである。このような攻撃手法が DoS 攻撃と組み合わせやすいのは、DoS 攻撃がレスポンスを必要としない攻撃であるためである。

ネットワークへの過剰な負荷による DoS 攻撃の対策は膨大なサービス要求を行っているホストからの接続を断つことである。IP アドレス詐称がなされたパケットによる DoS 攻撃を受けた場合にこの方法を利用しても、攻撃者はある程度自由にソースアドレスを変更できると考えられるため、効果が十分に現れない。つまり、一つの IP アドレスを遮断しても、攻撃者がそれに気がついた時点で IP アドレスを変更することが可能であるため、また同じことの繰り返しが行われる可能性があり、その一方で、攻撃を受けたホストのレスポンスによる二次的な DoS 攻撃を引き起こす可能性もある。

このように正規の IP アドレスを詐称している攻撃者は、実際には全くの無害の組織体から攻撃をしているかのように見せかけることによって、騒動を引き起こすことができる。攻撃を受けているシステムの管理者は、外面上の攻撃元から来る全てのトラフィックをフィルタリングしようとする。しかし、このようなフィルタを追加することによって、悪意の無い正規のエンドシステムに対してサービス妨害を行ってしまう結果を引き起こす。つまり、攻撃を受けているシステムの管理者は、意図的ではなくても、知らず知らずのうちに攻撃者に力を貸すことになることもある。

4. 提案方式

以上の準備のもとに、IP アドレス詐称の対策に関する提案方式を示す。なお本提案では、ルータに繋がる経路のエッジルータ側を支線と称し、逆に外部ネットワークに繋がる側を本線と称している (図 1)。

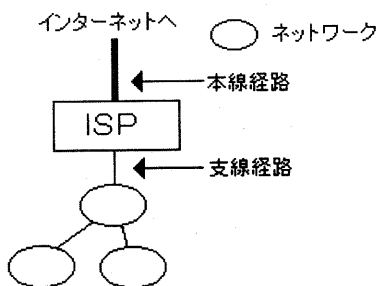


図1：本線経路と支線経路

4. 1 IP アドレス詐称対策へのアプローチ

IP アドレス詐称は、攻撃対象となったサーバが単独で防げるものではない。IP アドレス詐称がなされたパケットでも、パケット自体は構文規則に従っているため、IP アドレス詐称を見分けるのは難しい。別の手段でIP アドレス詐称の事実を認識できない限りパケットの受け取りを拒否するという判断を下すことはできない。このことからIP アドレス詐称を無くすための対策は、そのIP アドレス詐称がなされたパケットが送出される側のネットワークにおいてとることが有効である。

送信側がIP アドレス詐称対策を行った場合、その対策を施したネットワークの下にあるネットワークからはIP アドレス詐称のなされたパケットの流出はほとんど無いと考えられるため、この対策には、どこかでIP アドレス詐称が行われた際に、その詐称元を検索するときの検索範囲を狭める効果もある。

このアプローチによる対策は既に他でも取り上げられているところであるが[4]、本提案ではこれを、経路情報に基づいたパケットフィルタリングの自動化によって効率化しようというものである。

4. 2 経路情報によるフィルタリング

通常のルーティングプロトコルでは、ルータがパケットを転送する際に宛先アドレスだけを見て、それに見合った経路に向けて転送している。しかし、ルータは経路情報と通過するパケットから、パケットがどのネットワークから送られてきたの

か、そしてそのパケットのソースアドレスが何であるかを知ることができる。したがって、自分の支線経路先のネットワークがどのようになっているかを理解できているルータが支線経路から送られてくるパケットのソースアドレスを監視し、そのソースアドレスが支線経路先のネットワークアドレスとは異なる場合には転送を拒否すれば、IP アドレス詐称のなされたパケットが外部ネットワークに流出することを防ぐことができる。

4. 3 IP アドレス詐称対策アルゴリズム

次にルーティングテーブルからパケットフィルタリングのために有益な情報を取り出し、それを元にIP アドレス詐称対策を行うアルゴリズムを示す。

- 1) 対策を行うルータのルーティングテーブルから、支線経路先のネットワークアドレスを抽出する。
- 2) 抽出したIPアドレスをフィルタリングルールに整形し、そのルールをパケットフィルタに設定する。
- 3) 設定されたフィルタリングルールに従ってパケットフィルタはルータを通るパケットのソースIPアドレスを監視し、IPアドレス詐称のなされたパケットを発見して破棄する。
- 4) 以上の動作を繰り返す。

このアルゴリズムによって、経路情報を基にした、ネットワークの動的な変化に対応したパケットフィルタリングを、管理者の手間を煩わせることなく行うことができる。

5. 提案方式の実装

我々はPC ルータのOSとしてLinuxを、またパケットフィルタリングツールとしてIP Chainを用いて提案方式をプログラミング言語Perlにより実装した。

5. 1 IP Chain と RIP

この実装で使用したパケットフィルタリングツールである IP Chain とルーティングプロトコルである RIP の説明を行う。

5. 1. 1 IP Chain

IP Chain とは、Linux Kernel 2.2.X 以降で利用できるパケットフィルタであり、チェインという仕組みを使って、フィルタリングルールを鎖のように数珠つなぎにしてルールリストを作る。カーネルはパケットを受け取ると、この鎖のようにつながったルールに対して先頭から順番にパケットとルールのマッチングを行い、ルールに書かれた条件に当てはまる場合にそのパケットにルールに書かれた操作を適用する^[5]。

本実装ではこの IP Chain の基本的ルールとして、先に Forward (ルータ内の転送) の許可が出されているもの以外は通さないように設定しておくとする (policy REJECT)。

5. 1. 2 RIP

RIP は BSD 版 UNIX のネットワーキングコードの一部として開発され、「ルート管理デーモン」を表す「routed」というプログラムに組みこまれていた距離ベクトルプロトコルである。メトリックに純粋なポップカウントを使用し、リンク速度は考慮されていない。1988 年 6 月に Charles Hedrik によって RFC-1058 として文章化されている^{[6][7]}。

5. 2 ルール自動設定・更新プログラム

提案方式を実証するために作成したプログラムを本稿末尾の図 7 に示す。このプログラムで実装した具体的方法について説明する。

1) 初期設定と仮定

本実装でルーティングプロトコルには RIP を採用している。そのため、RIP のルーティングテーブルを参照するようにしている。

2) 経路情報の取得

ルーティングテーブルから経路情報を読み込

み、その中から支線経路に含まれる IP アドレスをまとめ、抜き出す。ルーティングテーブルの中の経路情報を走査し、インターフェイス (RIP のルーティングテーブルでは Iface と表示) が支線経路に繋がるイーサカードと同じである経路情報を取り出し、リストに入れる (図 2 参照)。

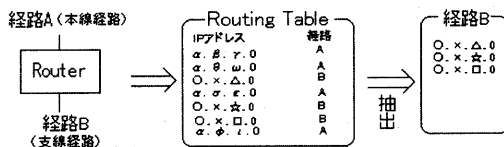


図 2: アドレスの限定

3) ルールへの変換

抜き出してリストに入れた経路情報から対象のイーサカードの先に繋がるネットワークのネットワークアドレス (RIP のルーティングテーブルでは Destination と表示) を取りだし、IP Chain のルールに変換する (図 3 参照)。

なお、本実装のプログラムではフィルタリングルールを「支線経路先のネットワークからのパケットであれば全てを通す」としている。支線経路先のネットワークから送られてくるパケットのサービスをも限定したい場合には、このルールに使用を許可するプロトコルを書き加えることにより行える。

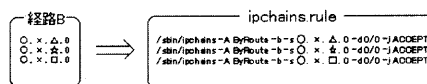


図 3: ルールへの変換

4) フィルタリングルールの更新

最後にルールリストの作成し、IP Chain へのルールの書きこみを行う。

本実装のプログラムでは、「ByRoute」という新しいチェインを作成し、その中に経路情報から作成されたルールを入れている。ルールが書きこまれた瞬間から IP Chain はそのルールを元にパケットフィルタリングを実行する (図 4 参照)。

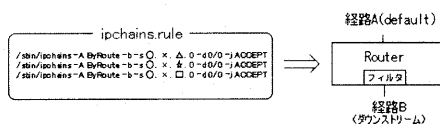


図4：ルール入力後

5. 3 シミュレーション結果

作成したプログラムを実際のネットワークを想定したシミュレーションにより実行した。図5が作成し使用したルーティングテーブルであり、図6がプログラムの実行結果である。ルーティングテーブルから必要な支線経路の経路情報を得て、それをフィルタリングルールとして実際に書きこむことが作成したプログラムにより容易に実行できることが確認できた。

なお、行ったシミュレーションでは、想定したネットワークの形に基づいたルーティングテーブルを書き、それに対してプログラムを実行した。このため、本来のルーティングテーブルではなく、自作のルーティングテーブルを参照するようにプログラムを作り変えている。

6. 考察

提案方式は定期的に経路情報を参照することを目的としており、支線経路に新しいネットワークが加わった場合への対処を迅速に行うことを考えるのであれば、頻繁にプログラムを実行することが望まれる。この更新の頻度は、経路情報が更新されるのと同じく、30秒につき1回とすることが望ましい。このように、このプログラムは頻繁に実行されることが想定されるので、なるべく簡潔なものになるよう心がけた。

このプログラムの実行の結果として形成されたパケットフィルタは、支線経路からのパケットのソースアドレスを監視し、支線経路の先に無いネットワークから送られてきたパケットは通過させないため、IPアドレスの詐称されたパケットが漏れ出ることを防ぐことができる。

ただし、この方法はルーティングテーブル内の

経路情報が正しいことを前提としている。経路情報が偽ものである場合にはIPアドレスの詐称されたパケットの流出を提案方式で防げる保証はない。したがって、ルータ間で通信される経路情報自体の偽造や改竄が行えないようにするか、ルーティングテーブルの経路情報が正確であるかどうかの確認を行えるようにすることが求められる。

7. まとめ

複雑なネットワークが張り巡らされている現在では、受信側ホストのみが対策を施すだけではネットワークセキュリティを確立することが難しくなりつつある。そのため、全てのネットワーク管理者は管理するネットワークから他ホストに攻撃が行われることを防ぐことが必要とされている。

しかし、ネットワークを厳格に管理・監視することには、時間やコストがかかり、それは管理者に大きな負担となる。本論文ではそのような負担を軽減するため、IPアドレス詐称と言う問題に対して、ルーティングテーブルの経路情報を利用したパケットフィルタリングルールを作成し、IPアドレスが詐称されたパケットがネットワーク上に流れることを自動的に防ぐ方法について提案した。

本提案ではプログラムとしてその実装を行うことで、ルーティングテーブルから必要な支線経路の経路情報を得て、それをフィルタリングルールとして実際に書きこむまでが容易であることを示し、この方法でIPアドレス詐称の対策が行えることを明らかにした。

参考文献

- [1] Simson Garfinkel, Gene Spafford, UNIX&インターネットセキュリティ, オライリー・ジャパン, Dec. 1998.
- [2] 竹下 隆史, 村山 公保, 荒井 透, 荻田 幸雄, マスタリングTCP/IP 入門編 第二版, オーム社, 1998.
- [3] IPA セキュリティセンター, “DDoS 攻撃に関する情報,” Feb. 2001.
- [4] P.Ferguson, D.Senig, “Network Ingress Filtering,” Request for Comments 2827, May 2000.
- [5] 久米原 栄, ファイアウォール管理者ガイド, ソフトバンク パブリッシング株式会社, Feb. 2000.
- [6] Christian Huitema, インターネット ルーティング, 翔泳社, March 2001.
- [7] C.Hedrick, “Routing Information Protocol,” Request for Comments 1058, June 1988.

```

Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
kozagawa.sys.wa *                255.255.255.255 UH      0      0      0 eth0
133.42.147.0     kozagawa.sys.wa 255.255.255.0   UG      0      0      0 eth0
133.42.147.0     *                255.255.255.0   U       0      0      0 eth0
133.42.140.0     133.42.147.254 255.255.255.0   UG      1      0      0 eth0
133.42.141.0     133.42.147.254 255.255.255.0   UG      1      0      0 eth0
133.42.198.0     133.42.147.254 255.255.255.0   UG      2      0      0 eth0
133.42.196.0     133.42.147.254 255.255.255.0   UG      2      0      0 eth0
133.42.197.0     133.42.147.254 255.255.255.0   UG      2      0      0 eth0
133.42.194.0     133.42.147.254 255.255.255.0   UG      2      0      0 eth0
133.42.195.0     133.42.147.254 255.255.255.0   UG      2      0      0 eth0
133.42.192.0     133.42.147.254 255.255.255.0   UG      2      0      0 eth0
133.42.193.0     133.42.147.254 255.255.255.0   UG      2      0      0 eth0
133.42.254.0     133.42.147.254 255.255.255.0   UG      1      0      0 eth0
133.42.250.0     133.42.147.254 255.255.255.0   UG      6      0      0 eth0
192.163.10.0     192.163.1.254  255.255.255.0   UG      0      0      0 eth1
192.163.100.0    192.163.1.254  255.255.255.0   UG      1      0      0 eth1
192.163.110.0    192.163.1.254  255.255.255.0   UG      1      0      0 eth1
192.163.120.0    192.163.1.254  255.255.255.0   UG      1      0      0 eth1
192.163.200.0    192.163.1.254  255.255.255.0   UG      2      0      0 eth1
127.0.0.0        *                255.0.0.0       U       0      0      0 lo
default          133.42.147.254  0.0.0.0          UG      0      0      0 eth0
default          133.42.147.254  0.0.0.0          UG      6      0      0 eth0
~
(END)

```

図5：ルーティングテーブル

```

[root@kozagawa /root]#
[root@kozagawa /root]# /sbin/ipchains -nL
Chain input (policy ACCEPT):
Chain forward (policy REJECT):
Chain output (policy ACCEPT):
[root@kozagawa /root]#
[root@kozagawa /root]# perl program3.pl
ipchains: bad rule (does a matching rule exist in that chain?)
[root@kozagawa /root]#
[root@kozagawa /root]# /sbin/ipchains -nL
Chain input (policy ACCEPT):
Chain forward (policy REJECT):
Chain ByRoute (1 references):
target      prot opt      source            destination      ports
ByRoute     all  -----  0.0.0.0/0         0.0.0.0/0        n/a
Chain output (policy ACCEPT):
Chain ByRoute (1 references):
target      prot opt      source            destination      ports
ACCEPT      all  -----  192.163.10.0      0.0.0.0/0        n/a
ACCEPT      all  -----  0.0.0.0/0         192.163.10.0     n/a
ACCEPT      all  -----  192.163.100.0     0.0.0.0/0        n/a
ACCEPT      all  -----  0.0.0.0/0         192.163.100.0    n/a
ACCEPT      all  -----  192.163.110.0     0.0.0.0/0        n/a
ACCEPT      all  -----  0.0.0.0/0         192.163.110.0    n/a
ACCEPT      all  -----  192.163.120.0     0.0.0.0/0        n/a
ACCEPT      all  -----  0.0.0.0/0         192.163.120.0    n/a
ACCEPT      all  -----  192.163.200.0     0.0.0.0/0        n/a
ACCEPT      all  -----  0.0.0.0/0         192.163.200.0    n/a
[root@kozagawa /root]#

```

図6：実行結果

```

#!/usr/local/bin/perl

### Setting

$under = eth1;

### Read routing table

@routels = `cat /root/badroute`;

### Check IP address

$x = 0;
$y = 0;

while (1){
    $interface = substr(@routels[$x], 72, 4);
    if($interface eq $under){
        @addressls[$y] = @routels[$x];
        $y++;
    }
    $x++;
    if(@routels[$x] lt 1){last;}
}
printf @addressls[1];
### Make rule

$x=0;

while($x < $y){
    $Saddress = substr(@addressls[$x], 0, 16);
    $Daddress = substr(@addressls[$x], 16, 1);
    if ($Daddress eq ""){
        $x++;
        next;
    }
    @rulesls[$x] = "/sbin/ipchains -A ByRoute -b -s ${Saddress} -d 0/0 -j ACCEPT¥n";
    $x++;
}

### Make rule box

system "/sbin/ipchains -N ByRoute";
system "/sbin/ipchains -F ByRoute";
system "/sbin/ipchains -D forward -j ByRoute";
system "/sbin/ipchains -A forward -j ByRoute";

### Write rule

$x=0;
while($x < $y){
    system @rulesls[$x];
    $x++;
}

```

図7. 提案方式を実装したプログラム