IPv6のための異常トラフィック生成プログラムの試作と評価

2011SE206 大平 峻也　2011SE304 山下 瑞樹
指導教員 後藤 邦夫

1　はじめに

近年，インターネットが急速に普及し，日常的に私たちはPCやスマートフォンを利用している．1981年に現在のInternet Protocol version 4(IPv4)の基となる仕様が発表された．それから10年が経過した頃から，このままではIPアドレスが足りなくなるのではないか，という懸念が出始めた．これがIPアドレス枯渇問題である．この問題を根本的に解決するための対策が早急に求められた．そこで考案されたものが，Internet Protocol version 6(IPv6)である．2011年度のWorldwide Infrastructure Security Reportの調査では，回答者からIPv6に対するDistributed Denial of Service(DDoS)攻撃が初めて報告された[4]．IPv6に対応した機器が被害を受けないためにも，脆弱性を把握し対策を講じる必要がある．

そこで，2013年度 榊原，下方の卒業研究「DDoS攻撃検出と対策の実験による評価」[3]はIPv4のみに対応したものであったが，本研究はIPv6のための異常トラフィックを発生する手法を提案し，IPv6固有もしくはIPv4/IPv6に共通する新たなセキュリティリスクの明確化と低減化に貢献することを目的とする．本研究では，悪意を持った攻撃者をAttacker，攻撃対象者をTargetとする．具体的には，一般的なソケットではなく生のソケットを使い，事実上無制約なパケットを生成する．それを実際に送信することでTargetに負荷を与え，通信妨害やサービス妨害を行う．

動作確認・模擬実験には，ネットワークエミュレータのCommon Open Research Emulator(CORE)[5]を使用した．また，COREを使用するにあたり，実験環境にはLinuxを採用した．これは，LinuxのNetwork Namespaceという独立したネットワーク環境を作成でき，なおかつ他のNamespaceや物理環境から干渉を受けにくい点を考慮して決定した．CORE上で動作するHTTPサーバにはJavaScript環境のnode.jsを採用した．

なお，大平はネットワーク構成などシステムの設計，山下は攻撃プログラムの試作を担当する．

2　DDoS攻撃と新たな問題

本節では，DDoS攻撃の概要とIPv6における新たな問題について述べる．

2.1　DDoS攻撃の概要

DDoS攻撃とは，Attackerが複数存在するDoS攻撃のことである．まず，DoS攻撃について説明する．DoS攻撃とは，ネットワーク上のトラフィックを増大させ，回線やネットワーク機器に対して何らかのダメージを与える攻撃のことである．それにより，通信妨害やシステム障害が引き起こされる．場合によっては，システムのダウンやネットワーク機器自体の誤動作，破壊にも繋がることがある．DoS攻撃ならばAttackerが単数なので，比較的対策をしやすい．例えば，特定のIPアドレスに対するアクセス制限や，同一のIPアドレスからのリクエスト回数制限をすれば良い．しかし，DDoS攻撃では数千・数万のAttackerからの一斉攻撃であるため，一つ一つ対応することは非常に難しい．したがって，現状ではDDoS攻撃を完全に防ぐ方法はない．有名企業がTargetとされることが多く，近年の大きな社会問題となっている．サービスの妨害や個人情報の漏洩が主な目的だと考えられる．図1にIPv6ヘッダを示す．

図1　IPv6ヘッダ

2.2　新たな問題

IPv6を利用するにあたり，新たなセキュリティ上の問題が発生してくる．以下は代表的なものである．IPv6を安心して利用できるよう，これらの問題を解決しなければならない．

- 詐称したルータ広告メッセージ (Router Advertisement：RA) を用いた通信妨害及び盗聴
- 不正なRAによる通信妨害
- 詐称した近隣要請，広告メッセージ (Neighbor Solicitation:NS，neighbor Advertisement:NA) を用いた通信妨害

2.3　IP Spoofing攻撃

通常，ホストやネットワーク同士の接続において，ホストは特定のIPアドレスに制限をかけて接続先をフィルタリングする．しかし，IP Spoofing攻撃はAttackerがIPヘッダ部のSrcIPアドレスをTargetのアドレス空間のものに偽装することで，Targetへのアクセス制限を突破する攻撃手法である．AttackerはTargetの接続制限内の特定のIPアドレスになりすますことによって，接続認証が

容易にできてしまう．そのため，Attacker が Target へ侵入できる可能性が高まる．また，Target 側のシステムのログにも偽装した SrcIP アドレスが記録されるため，Target は Attacker を特定することは難しい．さらに，この攻撃手法を DoS 攻撃にも応用することができる．実際に，この手法を併用した攻撃が占める割合は大きい．DDoS 攻撃に使用される場合，Attacker は偽装した SrcIP アドレスを大量に生成するため，IP Spoofing 攻撃を単体で使用する場合に比べ，Attacker を特定することがより困難になる．また，特定の IP アドレスにアクセス制限を施すといった対策が無効となる．図 2 に IP Spoofing 攻撃の一例を示す．

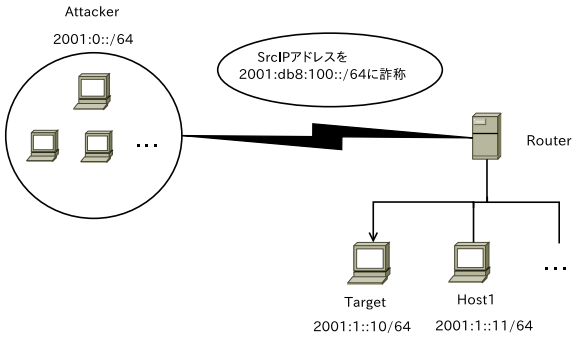

図 2　IP Spoofing の一例

他に，DDoS 攻撃での実験も考慮したネットワーク構成にした．さらに，IP Spoofing 攻撃をするにあたり，Target 側のアドレス空間になりすますための Host を作成した．ルーティングについては，構築したネットワークがシンプルであるため，デフォルトもしくは静的ルーティングを設定した．また，本研究では内部からのアクセスを不正アクセスとして扱わない．つまり，内部からの攻撃は考慮しないものとする．

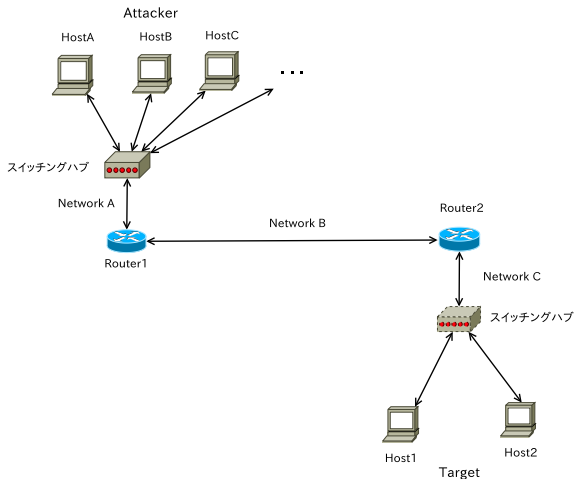

図 3　CORE で構築したネットワーク構成図

3　システムの概要

本節では，本研究で使用するネットワークエミュレータ CORE の概要とそのネットワーク構成について述べる．また，ネットワークで使用するサーバ環境構築のための Node.js[2] について述べる．

3.1　CORE

CORE とは，仮想マシンを用いて仮想的にネットワークを構築できるエミュレータである．Python モジュールのトポロジを描画するための GUI と，スクリプトネットワークエミュレーション用の軽量仮想マシンで構成されている．使用する上での利点は，低コストで現実に近い結果を再現することが可能なこと，単純かつ使いやすいよう GUI が作られているため操作が容易なことである．また，仮想デバイスを用いるため，個人が実世界で再現することが困難であるようなシチュエーションやデバイス構成であってもテストすることが可能である．さらに，今回は行わなかったが，高度なカスタマイズもできる．しかし，CORE に実装されていないデバイスを利用することができないという欠点もある．

3.2　ネットワーク構成

本研究では，DDoS 攻撃を実現するために，Host A B C のように Attacker を複数用意する．Host 1 を Target，Host 2 を TCP Connect のための IP Spoofing 攻撃の際に偽装に使用される Host とする．CORE によって作成したネットワーク構成を図 3 に示す．本研究では，TCP SYN Flood，UDP Flood，Ping6 Flood など DoS 攻撃の

Attackers 側の IP アドレスの割り当て

```
Host: 2001:0::10/64
Router(内側): 2001:0::1/64
Router(外側): 2001:3::1/64
```

Target 側の IP アドレスの割り当て

```
Host: 2001:1::10/64
Router(内側): 2001:1::1/64
Router(外側): 2001:3::2/64
```

3.3　Node.js

Node.js とは，Google が Google Chrome 用に開発した高速な JavaScript エンジンを利用したサーバ側で動作する JavaScript 環境である．HTTP サーバを構築する際，Apache では 1 つのクライアントからの要求に対して 1 つのスレッドしか作成されないが，Node.js は 1 つのスレッドで複数の WEB ブラウザからの要求を処理できる．1 つのスレッドで効率的に並列処理するために，特定のイベントが発生するとそれに対応する処理を実行する，イベント駆動モデルとネットワーク I/O やデータベースへのアクセスを非同期で実行するノンブロッキング I/O を採用している．そのため，WEB サーバのリソースを効率的に使い，高速で大量のデータ処理を実現できる．

本研究では，Target 側の 80 番ポートで待機させる HTTP サーバを構築するために用いる．

4　実現

本節では，試作した DoS 攻撃プログラムの概要について述べる．またページの都合上，TCP SYN Flood 攻撃

についてのみ説明する．

4.1 DoS 攻撃プログラム

本研究では，異常トラフィックを発生させる手段として DoS 攻撃プログラムを試作した．そして，複数の Attacker が同時に Target へ実行することによって DDoS 攻撃を実現する．本プログラムでは，Raw Socket を用いることで，通常では自動で割り当てられる情報を任意で設定できるため，事実上無制約なパケットを生成することが可能になる．しかし，Raw Socket プログラミングにより自由度が向上する半面，普段意識することのないプロトコルの詳細まで設定しなければならない．そのため，Raw Socket プログラミングに慣れていない人にとっては実装が容易ではない．

本プログラムでは，実行時にユーザが DstIP アドレス，Attacker 側のルータアドレス，interval(msec)，データ長 (octets) を入力する．作成するソケットの内容は，sockaddr_in6 構造体，チェックサム計算のための擬似ヘッダ，IPv6 や TCP など各プロトコルのヘッダの値である．Raw Socket による実現のため IP ヘッダとそれに続くヘッダに適切な値を埋めていく．表 1 に本研究で使用した DoS 攻撃プログラムの IP ヘッダ，TCP ヘッダに代入したパラメータ一覧を示す．

表 1 各ヘッダのパラメータ情報

IP ヘッダ	ip6_vfc	0x60
	ip6_plen	IP ヘッダ (40 octes) +TCP ヘッダ (20 octes) +ペイロード長
	ip6_nxt	IPPROTO_TCP(6)
	ip6_hlim	任意 例:225
	ip6_src	任意 例:inet_pton(AF_INET6,2001:db8::1,&src)
	ip6_dst	2001:1::10(Target のアドレス)
TCP ヘッダ	source	任意 例:htons(80)
	dst	任意 (攻撃対象ポート) 例:htons(80)
	seq	任意 例:htonl(12345)
	doff	5
	syn	1
	window	任意 例:htons(65535)
	check	擬似ヘッダ[1]+TCP ヘッダ +ペイロード長

SrcIP アドレスの生成は，drand48() を使い 0 から $2^{16}-1$ までの整数の一様乱数を実現する．これを 16 進数表記で出力することで，0000 から FFFF までの SrcIP アドレスを実現する．また，IP ヘッダの SrcIP アドレスは，inet_pton 関数を用い文字列形式からバイナリ形式に変換して実装する．そして，一連の流れをループさせることにより，DDoS 攻撃の際に詐称した送信元 IP アドレスを大量に生成する．このようにして，異常なトラフィックを発生させ DoS 攻撃を行う．

SrcIP アドレス詐称箇所のソースコード

```
srand48(12345);
  while(1){
   unsigned int x;
   char srcstr[100];

   x = (unsigned int) (65536*drand48());

   sprintf(srcstr,"2001:db8:100::%x",x);
   printf("Src: %s\n", srcstr);
   inet_pton(AF_INET6,srcstr,&src);
   ip6->ip6_src = src;
```

データの送信には，sendto() 関数を用い，使用ソケット (raw socket)，データ (IPv6 ヘッダ，TCP ヘッダ，ペイロード長)，データサイズ (ペイロード長が 0 の場合，60 octets)，フラグ (0)，接続先のアドレス (2001::1)，接続先のアドレスサイズ (0x1c) の情報を格納している．

データ送信箇所のソースコード

```
n = sendto(sock, sendbuff,
    sizeof(struct ip6_hdr) + ntohs(ip6->ip6_plen),
    0, (const struct sockaddr*)&server,
    sizeof(server));
```

チェックサム計算 [1] には，checksumadjust 関数を用いる．計算は擬似ヘッダ，TCP ヘッダ+ペイロード長の 2 度に分けて計算する．

チェックサム計算箇所のソースコード

```
uint16_t tmp= 0;
   tcph->check = 0;

   checksumadjust((unsigned char *) &tmp,
                  0,0,
                  (unsigned char *) &ip6ph,
                  sizeof(ip6ph));
   checksumadjust((unsigned char *) &tmp,
                  0,0,
                  (unsigned char *)tcph,
                  sizeof(struct tcphdr)+length);

   tcph->check = tmp;
```

5 実験

本節では，CORE で構築したネットワーク上で DoS 攻撃プログラムを実行した際の実験手順と実験結果について述べる．

5.1 実験環境

実験で使用する PC のスペックを以下に示す．

OS：Linux Ubuntu12.04LTS(32bit)

CPU：Intel(R) Core(TM) i5 CPU M560 2.67GHz

kernel：3.2.0-57-generic-pae

以上の実験環境で TCP，UDP，ICMPv6 の各プロトコルで TCP SYN Flood，UDP Flood，Ping6 Flood，TCP Connect のための IP Spoofing，RA Flood，NA Flood 攻撃プログラムを実行する．

[1]SrcIP アドレス，DstIP アドレス，ゼロ，ネクストヘッダ，パケット長が入る

5.2 実験手順

1. CORE で構築したネットワーク上で Attacker から Target に各種攻撃プログラムを使用する
2. tcpdump コマンドで異常トラフィックが Target でキャプチャできているか否か確認する
   - パケットが正常に Target でキャプチャできた場合は，Target 側の挙動など攻撃の有効性を確認する
   - パケットが正常に Target でキャプチャできなかった場合は，traceroute コマンド等を使い障害が発生している箇所を特定して修正し，1 から再度やり直す
3. トランスポート・プロトコルを切り替えて再度 1 から実行する

5.3 実験結果

CORE で構築したネットワーク上で Attacker が Target に向けて前述した各種攻撃プログラムを実行し，いくつか結果が得られた．ページの都合上，TCP SYN Flood のみの tcpdump でのキャプチャ結果を示す．また NS，NA，RS，RA の情報も実験結果から省略する．

TCP SYN Flood のキャプチャ内容

```
tcpdump -i eth0 -vvv

IP6(hlim62,next-headerTCP(6) payload length:1460)
2001:db8:100::39ae.http>2001:1::10.http: Flags[S],
cksum 0x4d94 (correct),seq 12345:13785,
win 0, length 1440

IP6(hlim 64,next-header TCP(6) payload length:24)
2001:1::10.http>2001:db8:100::39ae.http:Flags[S.],
cksum 0x8897(incorrect -> 0xeb13),seq 1585274118,
ack 12346, win 14400, options [mss 1440],length 0

IP6(hlim 64,next-header TCP(6) payload length:24)
2001:1::10.http>2001:db8:100::39ae.http:Flags[S.],
cksum 0x8897(incorrect -> 0xeb13),seq 1585274118,
ack 12346, win 14400, options [mss 1440],length 0

IP6(hlim62,next-headerTCP(6) payload length:1460)
2001:db8:100::eb4e.http>2001:1::10.http: Flags[S],
cksum 0x9bf3 (correct), seq 12345:13785,
win 0, length 1440

IP6(hlim 64,next-header TCP(6) payload length:24)
2001:1::10.http>2001:db8:100::eb4e.http:Flags[S.],
cksum 0x3a38(incorrect -> 0x0c88), seq 488688462,
ack 12346, win 14400, options[mss 1440], length 0

IP6(hlim 64,next-header TCP(6) payload length:24)
2001:1::10.http>2001:db8:100::eb4e.http:Flags[S.],
cksum 0x3a38(incorrect -> 0x0c88), seq 488688462,
ack 12346, win 14400, options[mss 1440], length 0

                    :
                    :
```

以上の結果から，Target 側にホップ・リミット，ネクスト・ヘッダ，ペイロード長，チェックサムの計算結果の値が正しく表示されているため，これらのパケットが正常に Attacker から Target へ届いていることが確認できた．さらに，Target は SYN,ACK を代入したパケットを Attacker が偽装した SrcIP アドレスに再送信を繰り返していることが確認できた．そのため，TCP SYN Flood 攻撃は成功したと言える．

6 おわりに

CORE で構築したネットワークの静的ルーティング設定と TCP SYN Flood，UDP Flood，ICMPv6 Flood 攻撃プログラムの試作は完成した．また，実験結果からもその有効性を確認することができた．行数に換算すると，各種 include とコメントを除いて合計 617 行であった．

しかし，TCP Connect のための IP Spoofing 攻撃プログラムを実行したとき，Target から誤ったチェックサムのデータをキャプチャしてしまい，本研究ではコネクションを確立するに至れなかった．この原因の一つは，IPv6 固有の tcpdump におけるバグだと考える．

以下は，本研究でやり残した今後の課題である

- TCP Connect のための IP Spoofing 攻撃プログラムの完成
- さらに IPv6 に関係する異常トラフィックを用いた脅威を再現・試作する
  - 詐称した RA を用いた通信妨害及び盗聴
  - 不正な RA による通信妨害
  - 詐称した NS，NA を用いた通信妨害
- 擬似攻撃ツールなども存在すれば，先輩方の卒業論文と混同しない範囲で利用する
- 大規模なネットワークを想定した環境での実験 (CORE で実現)

参考文献

[1] Braden, R., Borman, D. and Partridge, C.: Computing the Internet Checksum, *RFC1072* (1998).

[2] Node.js Developers: Node.js, *http://nodejs.org/* (accessed Jan. 2015).

[3] 榊原広樹，下方章裕 : DDoS 疑似攻撃プログラムの試作と既存 DoS 攻撃プログラムを用いた DoS 攻撃の実験，南山大学情報理工学部システム創成工学科 2013 年度卒業論文 (2014).

[4] The Arbor Networks: The Arbor Networks 7th Annual Worldwide Infrastructure Security Report, *http://www.arbornetworks.com/news-and-events/press-releases/2012-press-releases/4497-the-arbor-networks-7th-annual-worldwide-infrastructure-security-report* (accessed Jan. 2015).

[5] U.S. Naval Research Laboratory Networks and Communication Systems Branch: Common Open Research Emulator, *http://www.nrl.navy.mil/itd/ncs/products/core* (accessed Jan. 2015).