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

2010SE183 榊原広樹　2010SE198 下方章裕
指導教員：後藤邦夫

1　はじめに

近年，情報通信ネットワーク及びPCの普及が著しい．さらにインターネットの普及に伴い，不正アクセスによる被害が増加傾向にある．特に，送信元アドレスを偽装した上で，パケットを大量に送信し通信を処理している回線やサーバの処理能力を占有することで，システムを使用困難にしたりダウンさせたりするDenial of Service attack(以下DoS攻撃とする)や攻撃内容はDoS攻撃と同じだが，踏み台となる多数のコンピュータを利用して一斉にターゲット・サーバに攻撃をするDistributed Denial of Service attack(以下DDoS攻撃とする)は，システムを停止に追いやることもあり，政府系サイトにまで被害も出ている．

攻撃ツールが容易に入手でき，誰でも攻撃に参加することができることや，何も前触れがなく攻撃されることから，DoS攻撃，DDoS攻撃の対策は困難である．特にDDoS攻撃は攻撃手段が多く，送信元のアドレスを偽装するため，通常のDoS攻撃よりも防御が困難であるため，攻撃による被害はDoS攻撃よりも大きくなると考えられる．

先行研究[5]では，DDoS攻撃のシミュレータの構成と検出方法を示しており，結果は示していない．そこで，簡単なネットワークを構築し，攻撃ソフトウェアを使用する．ネットワーク負荷等を計測し，そこから対策を練る．簡易ネットワークでの実験を終えたあとは，規模を大きくし，複雑なネットワークでの実験を行い再度計測する．

本研究では，エミュレータを用い，擬似攻撃を行う．まず，DoS攻撃のエミュレーションを行い，さらに検知を行う．その後にパケットフィルタリングを行い，パケット受信量を減らす．次に，DDoS攻撃をエミュレータ上でDDoS攻撃ツールを動作させることにより不正アクセス状況を再現し，DoS攻撃と同様，検知とパケットフィルタリングを行い不正アクセス状況の詳細の把握，対処方法を検討する．

本研究の成果として検出ツールで攻撃先のIPアドレスを検出し，そのIPアドレスをフィルタリングすることでDDoS攻撃を軽減できることが確認できる．ネットワーク構築と文書作成は共同で行った．下方は自作プログラムの作成，iptablesの設定，snortの設定を担当し，榊原はシミュレーション，数値の計算を担当した．

2　DoS/DDoS攻撃の種類

DoS/DDoS攻撃の中で，以下の4つを説明する[7]．選んだ理由としては，典型的な4つだからである．

過大なHTTP要求

Webサーバにコンテンツを応答させる処理を大量に実行する負荷計測ツールのapache benchを用いる．多数のホストからこれを使用することでDDoS攻撃となる．

TCP SYN Flood攻撃

3ウェイハンドシェイクの最初のSYNパケットを送る段階で，クライアント自身のIPアドレスを送るところ，偽ったIPアドレスをサーバーに送り，サーバーは偽ったIPアドレスからのACKを待ち続けることを利用した攻撃．

UDP Flood攻撃

原理はTCP SYN Flood攻撃と同じであるが，コネクションレスの通信であるため，一方的にパケットを送り付けることができる．また，UDPでは，最大1500バイトのパケットを正常の通信と偽って送ることができるので，膨大なパケットを送信することができる．

Smurf攻撃

ICMPエコー要求パケットを受信した多数のPCは，一斉に詐称された発信元IPアドレスに対してICMPエコー応答パケットを返送する．詐称された発信元のネットワークでは大量のICMPエコー応答パケット到着によりトラフィックが増大し，サービス不能になる．

TCP SYN Flood攻撃，UDP Flood攻撃，Smurf攻撃は既存のツールがなかったため，自作プログラムを作成し使用することにより再現し，シミュレーションをする．

3　システムの概要

本節では，システム全体の流れと本研究に使用したシステムの概要について説明する．

3.1　実験ネットワークの構成

本研究では図1に示す擬似ネットワーク上で実験を行う．ネットワークの構築はCommon Open Research Emulator[1](以下COREとする)を使用した．記入されているIPアドレスはCOREの自動設定機能によるものである．攻撃対象はn16とし，DoS攻撃の攻撃者はn7，DDoS攻撃の攻撃者はn7 n12として同時に攻撃をする．

また，効率的で大規模なものまで実現できること，GUIのため使いやすいこと，現実的なソフトウェアの再現ができることを理由にCOREを使用した．

CORE

COREは，1つ以上のマシンでネットワークをエミュレートするためのツールである．COREは，描画のためのGUI軽量仮想マシン，およびスクリプティングネットワークエミュレーションのためのPythonモジュールの

トポロジで構成されている．それぞれのノードから端末を起動することができ，その端末上で攻撃ツールや検出ツールを動かすことができる．
CORE を使用した理由は，効率的で大規模なものまで実現できること，GUI のため使いやすいこと，リアルなソフトウェアの再現ができること等のことから使用した．

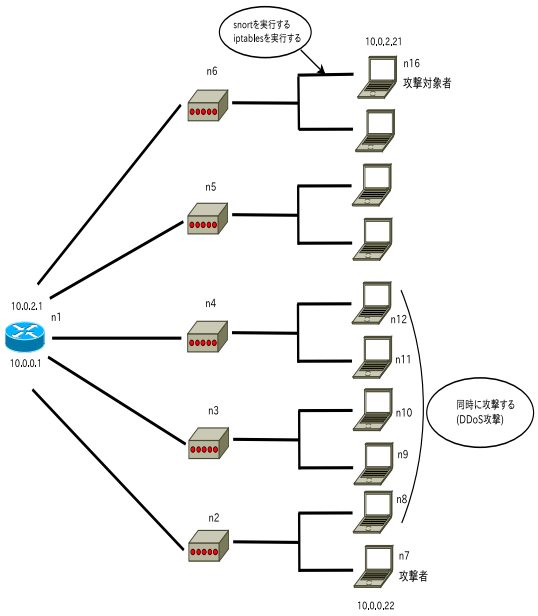

図 1　DoS 攻撃ネットワークモデル

3.2　攻撃ツール

CORE で作成したネットワークで以下の 2 つの攻撃ツールを使用する．それぞれ共通して DoS 攻撃は n7，DDoS 攻撃は n7 n12 を攻撃者とし，n16 を攻撃対象とする．

slowhttptest[3]

slowhttptest は攻撃パターンの多彩なツールであるので本研究で使用する．slowhttptest とは、アプリケーション層のサービス不能攻撃の高機能なオプションを持ったシミュレーションツールのことである．メジャーな Linux や OS X，Cygwin で動作する．サーバの読み込みに多大な時間をかけさせる攻撃である．攻撃手段ではないが apache bench と同じようなツールとして扱う．このツールの初期値は表 1 のようになっており，この初期値を基準として各項目の値を変更し実験を行う．

表 1　パラメータの初期値

Test type	SLOW HEADERS
Number of connections	50
URL	http://localhost
Verb	GET
Interval between follow up date	10 seconds
Connections per second	50
Test duration	240 seconds
Probe connection timeout	5 seconds
Max length of followup date field	32 bytes

3.3　検出ツール

本研究では，DoS/DDoS 攻撃を検出するため以下のシステムを使用する．

snort

Snort[2] とは，コンピュータやネットワークへの不正な侵入や，攻撃の検出を行うシステム IDS(不正侵入検知システム) のひとつ．Snort は監視しているネットワークに流れるパケット常時観測し，不正な侵入や攻撃をその手法である「パターンデータ」と照らし合わせ，マッチングしているものを検出し，アラートを通知しログとして記録する．監視対象となるネットワークを記述する．var HOME_NET any ではすべてのネットワークを監視することになるので本研究では，HOME_NET で図 1，2 の 10.0.2.1/24 を監視するための設定と内部からのアクセスを不正アクセスとして扱わない設定を以下の通り変更する．この設定を図 1 の攻撃対象となる n16 の端末上で行う．

var HOME_NET の設定

10.0.2.1/24 を監視する
var HOME_NET any
↓
var HOME_NET 10.0.2.1/24

内部からのアクセスを不正アクセスとして扱わない
var EXTERNAL_NET any
↓
var EXTERNAL_NET !$HOME_NET

iptables

iptables は，本来ホストで送受信する IPv4 のパケットのフィルタリングや，NAT（network address translation）を管理するためのコマンドである．DoS/DDoS 攻撃対策のためパケット制限を行う．

パケットをカウントし回数で制限するのに、iptables には 2 種類のモジュールが用意されている．1 つは,「limit」モジュールである．パケットをカウントし，回数に応じて制限を実施する．ただし送信元の IP アドレスで回数制限を加えるには，対象の IP アドレスを都度指定し iptables を実行する必要がある．そのため送信元を特定できない DDoS 攻撃には役立たない．もう 1 つの「hashlimit」モジュールなら，クライアントの IP アドレスごとにパケット数をカウントし，iptables 実行時にクライアントごとに制限を実施することができる．そのさい，個々の IP アドレスを指定する必要もない．

limit で ssh のサーバポートへのアクセスの受け入れ制限を設定した場合，攻撃的パケットもそうでない真っ当なパケットも一緒に管理されるので，攻撃が行われている間は，権限を持った人もアクセスが制限されてしまう．一方 hashlimit を使えば，アクセス元毎に管理できるので，攻撃が行われている間でも，権限のある人は正常にアクセスすることができる．

最大応答回数を 10 回とする．10 回までは無条件に応答

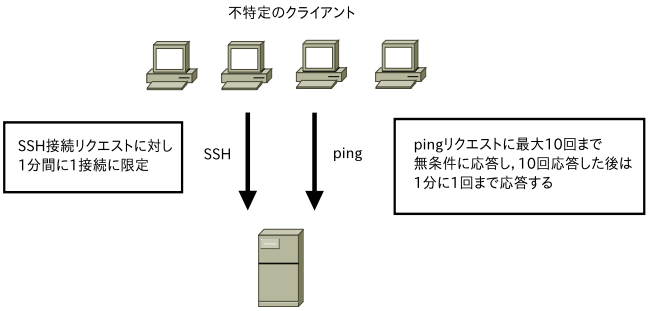

図 2 limit モジュール

し，11 回目の応答は破棄される．パケットの規制は 1 分間に 1 回まで ssh 接続が可能とする．パスワードを総当たりで解析するブルートフォース攻撃に有効である．

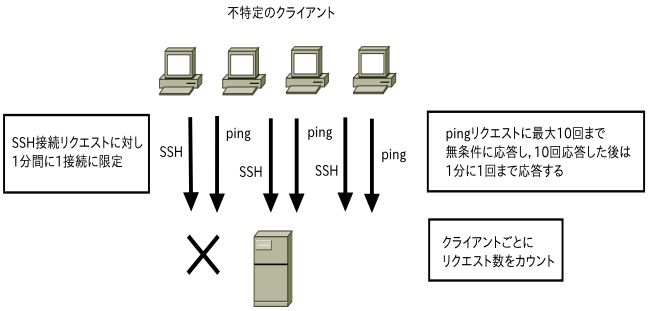

図 3 hashlimit モジュール

同一ホストからの最大接続回数を 1 回，その後は 1 分間に 1 回まで接続を許可する．こうした接続回数は，管理ファイルによりクライアントごとにカウントされる．limit の制限では，正常なリクエストも不正アクセスに紛れてしまうが，あるクライアントを不正アクセスで規制している間も，他のクライアントから接続することが可能となる．

以下のコマンドを入力することで IP アドレス 10.0.0.20 からくるパケットをすべて破棄することができる．iptables -F，iptables -X のコマンドを入力することで初期化を行うことができる．

web は 1000 回までは許可、以降は 1 分間に 120 回までに制限する (limit)

ssh は 5 回まで新規接続許可それ以降は 1 分に 1 回に制限、6 分間無接続なら制限解除．web は 700 回まで新規接続許可それ以降は 1 秒に 1 回に制限、6 分無接続なら制限解除 (hash limit)

以上の設定を図 1 の攻撃対象となる n16 の端末上で行う．

iftop

ネットワークトラフィック情報を表示するツール [4] である．IP アドレスとポート番号をベースに，それぞれの通信にトラフィックの送信量，トラフィックの受信料，直前 2 秒，10 秒，40 秒のトラフィック量の平均値を視覚的に表示する．ターミナル全体に情報を表示し，定期的に更新する．DoS 攻撃，DDoS 攻撃のシミュレーションで観測したトラフィック量をまとめ表にする．

4 疑似 DDoS 攻撃プログラムの実現

TCP SYN Flood 攻撃，UDP Flood 攻撃，Smurf 攻撃のツールがないため，TCP，UDP，ICMP のプロトコルを持ち発信元 IP アドレスを偽装する 3 種類のプログラムを作成した．これらのプログラムは，IPv4 のネットワークに対して，発信元 IP アドレスのネットワーク部を 192.168 と固定し，ホスト部を 0 から 255 までの数字を任意に摘出し，送信するプログラムである．なお，プロトコルが TCP，UDP の場合，ポート番号は 100 に固定し，プロトコルが ICMP の場合，Type と Code はそれぞれ 8，0 に固定した．IP ヘッダや UDP/TCP データグラムに対し，通信時に起こったエラーを検出するために checksum を用意する．checksum[6] とは，データを送受信する際の誤り検出方法の一つである．送信前にデータを分割し，それぞれのブロック内のデータを数値とみなして合計を取ったものである．求めた checksum はデータと一緒に送信される．受信側では送られてきたデータ列から同様に checksum を計算し，送信側から送られてきた checksum と一致するかどうかを確認する．送信側と受信側の値が異なれば，通信系路上でデータに誤りが生じていることになる．checksum 関数はプログラム上で次のように使用した．

checksum 計算部分と使用部分

```
void checksumadjust(unsigned char *chksum,
            unsigned char *optr, int olen,
            unsigned char *nptr, int nlen)
   {
                        :
                        :
                        :
//擬似ヘッダ分だけ checksum 計算
//icmpfake4 は擬似ヘッダを使わないので不要
u_int16t tmp = 0;
ipph.src = ip->saddr;
tmp=checksumadjust((unsigned char *)&tcph->check,
          0,0,
          (unsigned char *)&ipph,sizeof(ipph));

//TCP ヘッダ + データ分を追加した checksum 計算
tmp=checksumadjust((unsigned char *)&tcph->check,
          0,0,
          (unsigned char *)tcph, tcph->doff*4);
          tcph->checksum=sum;
```

IP アドレスを偽装するため，rand 関数を利用しホスト部を任意な数字とするプログラムの一部を次に記載する．

IP アドレス偽装部分

```
unsigned x,y;
char srcstr[100];
x=(unsigned)(255*(random()/(1.0*RAND_MAX))),
y=(unsigned)(255*(random()/(1.0*RAND_MAX))),
sprintf(srcstr,"192.168.%u.%u",x,y);
printf("Src: %s\n", srcstr);
inet_pton(AF_INET,srcstr,&src);
ip->saddr = src.s_addr;
```

パケットを送信する部分を次に記載する．

パケット送信部分

```
n=sendto(sock,sendbuff,sizeof(struct iphdr)+8, 0,
(const struct sockaddr*) &server,sizeof(server));
```

5 実験と結果

本節では，本研究で提案したシステムの動作実験をし，その結果を述べる．

5.1 実験環境

本研究ではUbuntu Linux12.04(32bit OS) をインストールしたPCを用意しCOREを起動する．このPCで図1のネットワークを構築する．
COREで構築したネットワークは以下の，攻撃を行う攻撃者 (図1でいうn7)，攻撃者からの攻撃を受ける攻撃対象 (図1でいうn16) で構成される．

5.2 実験の手順

まず，ネットワーク全体が通信可能であるかをpingコマンドで確認する．通信が確認できたらslowhttptestや自作プログラムでDoS攻撃のシミュレーションをする．そのさい，iftopとsnortで通信状況を確認する．大量のパケットが送信されている (攻撃されている) ことが確認できたら，iptablesでパケットフィルタリングの設定を行い，再度DoS攻撃のシミュレーションと通信状況の確認をする．パケットの送受信量にはばらつきが生じるため，攻撃を5回行いその平均を取ったものを結果とする．その結果，攻撃先からの受信パケットの量が減っていればフィルタリングができているということなので実験成功とする．また，複数のノードから攻撃をすることでDDoS攻撃とし同じ手順で実験をする．

5.3 実験結果

まず，slowhttptestでDoS攻撃のシミュレーションをした．実験で得られた結果の比較を以下に示す．以下の表から分かるように，iptablesでフィルタリングをする前後でパケットの送受信量が減っていることを確認した．同様にDDoS攻撃のシミュレーションをしたところ，iptablesで，フィルタリングをする前後でパケットの送受信量が減っていることを確認した．よって，攻撃先からのIPアドレスが分かれば，有効にフィルタリングできることを確認した．

表2 接続回数50でのDoS攻撃の場合

接続回数		フィルタリング前	フィルタリング後
-c 50	TX(送信量)	46.3kB	0kB
	RX(受信量)	34.0kB	12.2kB

自作プログラムではフィルタリング前後で送受信量が変わらなかった．理由として，プログラムが間違っている可能性があることが考えられるが解決することができなかった．

表3 接続回数50でのDDoS攻撃の場合

接続回数		フィルタリング前	フィルタリング後
-c 50	TX(送信量)	275.0kB	0kB
	RX(受信量)	211.4kB	72.9kB

6 おわりに

本研究ではDoS，DDoS攻撃のシミュレーション，および対策の実験を試みた．実験環境が整わなかったため，snortで検知することが出来なかったが，ログを見て攻撃元や攻撃手法を判断することが可能であることを示した．また判断できた攻撃先をiptablesで設定し，有効にフィルタリングをすることも確認できた．また，自作プログラムは送信することは可能だが，応答が返ってこず納得する結果を得ることができなかったが，応答が返ってこればslowhttptestで成功したようにフィルタリングが可能であると考えられる．

今後の課題としては，自作プログラムを改善し，攻撃を再現可能にすることである．それから，再度実験をする必要がある．また，snortでの設定にDoS攻撃のパターンデータを登録し，検出させる必要がある．さらに本実験のiptablesの設定は必要最低限なものだったが，TCP SYN Flood攻撃対策，smurf攻撃対策など細かい使い分けができる設定が可能であるので，さらに細かい軽減が可能であろう．今後の課題を以下に述べる．

1. 自作プログラムの改善と実験.
2. snortでパターンデータとマッチしているものを検出，通知し，ログとして記録する機能の追加.
3. 各攻撃ごとにiptablesの設定を作り替える．

参考文献

[1] Ahrenholz, J.: Comparison of CORE Network Emulation Platforms, *Proc. of IEEE MILCOM Conference*, IEEE, pp. 864–869 (2010).

[2] ITmedia Inc.: snort (accessed Jan. 2014). http://www.itmedia.co.jp/help/howto/security/ids/01/.

[3] Shekyan, S.: slowhttptest (accessed Jan. 2014). http://code.google.com/p/slowhttptest/.

[4] Warren, P.: iftop (accessed Jan. 2014). http://www.ahref.org/tech/server/server-tips/801.html.

[5] 永島秀己, 大野浩之：DDoS Attackシミュレータ, 横河技報, Vol. 45, No. 4, 横河電機, pp. 201–204 (2001).

[6] 山本和彦：checksum (accessed Jan. 2014). http://www.mew.org/ kazu/doc/bsdmag/cksum.html.

[7] 寺田真敏：DoS/DDoS攻撃とは，情報処理， Vol. 54, No. 578, pp. 428–435 (2013).