DDoS 模擬攻撃ツールの改良
— 攻撃種類と GUI の追加 —

2013SE193 杉浦 潤平　2013SE252 山崎 澄音
指導教員 後藤 邦夫

1　はじめに

近年 IT 技術の発展により，PC，スマートフォンなど様々な通信機器が普及した．それにともない大変便利になった半面，セキュリティの意識が低い人々がサイバー犯罪に巻き込まれる事件が増加している．本研究では，様々なサイバー犯罪の中で有効な対策が難しいとされる DDoS(Distributed　Denial of Service) 攻撃について研究し，コマンド操作ができない人でも扱えるよう，GUI を用いて実験を行えるツールを作ることを目的とする．

DDoS 攻撃は大きくわけて 2 種類ある．大きなパケットを大量に送るなどして，サーバまでの接続回線容量を埋め尽くすネットワークの回線の帯域を狙う攻撃 (例：UDP-Flood,ICMP-Flood)，攻撃対象のサーバを過負荷に陥れるサーバを狙う攻撃 (例：TCP-SYN-Flood,　TCP-Connection-Flood, HTTP-GET-Flood, DNS-Reflector) がある．

本研究では，2015 年度川上・黒田の卒業研究「DDoS 攻撃模擬ツールの試作」[7] を 3 点において改良する．

1 つめは，GUI で操作可能にする．2 つめは，2015 年度の研究で成功した実験については動作を確認し，未完成の攻撃については，修正・実験する．3 つめは，攻撃の種類を UDP-Flood，ICMP-Flood，TCP-SYN-Flood，TCP-Connection-Flood の 4 種類から IP-Fragment[6][3]，HTTP-GET-Flood を加えた 6 種類にし，より幅広い実験を可能にする．

DDoS 攻撃の対策を研究，テストをするためには模擬攻撃が必要である．外部に被害や迷惑をかけないために閉鎖されたネットワークで実験をする必要があるため動作確認・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] を使用する．また，CORE を使用するにあたり，実験環境には Ubuntu Linux14.04 を採用した．なお，杉浦は攻撃プログラムの試作を，山崎は GUI の作成を担当する．

2　DDoS 攻撃の概要

DDoS 攻撃の種類を説明する．

2.1　2015 年度の攻撃

先行研究での攻撃について説明する．

ICMP-Flood 攻撃

「ICMP-Echo-Request」と呼ばれる制御用パケットを大量に送信し，対象の回線容量やシステムの処理能力を飽和状態にすることで，通常の通信要求に応答できない状態にする．

TCP-SYN-Flood 攻撃

TCP-SYN-Flood 攻撃は，確立しない TCP 接続を大量に試みる攻撃のことである．SrcIP アドレスの嘘つきが可能である．

TCP-Connection-Flood 攻撃

この攻撃では，コネクションを確立させるが，その後は何もしない．コネクションを大量に確立させることによってメモリを占領する．

攻撃の流れを図 1 に示す．

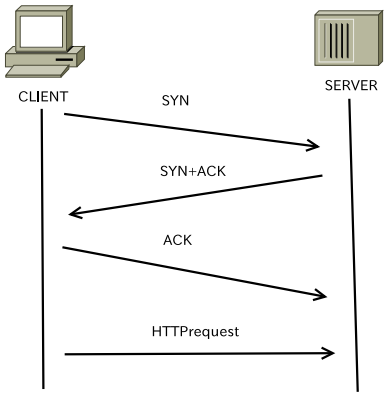

図 1　攻撃の流れ

2.2　追加する攻撃

本研究で追加する攻撃について説明する．

IP-Fragment-Flood 攻撃

IP フラグメンテーションとは，一度に送信することのできない大きな IP パケットを，いくつかに分割して送信する．パケットを受け取ったサーバーは分割されたパケットを元のデータに戻す必要があり，メモリ上にパケットを溜め込んでいく．Fragment 攻撃は分割されたパケットを継続的にサーバーに送り込むことによって，サーバーのリソースを枯渇させることを目的としたもの，またパケットを分割する際にパケットを故意に送信しないなどして待ち時間を発生させることを目的としたものがある．例えばパケットを 3 分割にした場合に，1 つめと 3 つめのパケットを送信するが 2 つめのパケットを送信しない，また 3 つめのパケットを故意に送信しないなどして，待ち時間を発生させることでメモリ上にパケットを溜め込んでいく．IPv4 では，ルータでのフラグメントがあり得るが，IPv6 では，経路上 MTU が最小の部分の MTU をその通信の MTU として使うため，ルータでのフラグメントは起きない．また近年の OS では，TCP で IP パケットを使用する時は，Don't Fragment フラグを 1 にしてフラ

グメントを禁止しているため，本研究ではICMPとUDPで実験を行った．

HTTP-GET-Flood 攻撃

HTTP-GET-Flood 攻撃は，TCP セッションのみを大量に確立させるだけでなく，セッション確立後に大量のHTTP-GET 要求を送信し，サーバーに対して高負荷をかけることでサービス不能状態へと陥れる攻撃である．HTTP プロトコルのバージョンには大きく分けて 2 種類ある．1 つ目は HTTP/1.0，2 つ目は HTTP/1.1 である．2 種類の違いとして 2 つ挙げられる．1 つ目は並行処理できる数が大きく違うことである．同時に出せるリクエストの数が HTTP/1.0 と HTTP/1.1 では違うため，読み込むファイルの数が多ければ多いほど多い程その差が大きく拡がっていくことになる．2 つ目は 1 回のリクエスト，レスポンスの中身が違うことである．HTTP/1.1 ではファイル 1 個を取りに行く時のリクエストとレスポンスの作業の効率化が図られているため，ページ表示速度が大きく変わる．

3　システムの概要

実験環境を作る CORE，作成する攻撃プログラムを説明する．

3.1　攻撃プログラムの使用想定

本研究では，先行研究での攻撃に，IP-Fragment 攻撃，HTTP-GET-Flood 攻撃を追加し，それらを選択して攻撃できる GUI を作成する．本研究に用いる GUI を図 2 に示す．

攻撃のパラメータは IP バージョン，攻撃に使用するプロトコル，攻撃の間隔，Src,DstIP の範囲，Src,Dst ポートの範囲，Payloadlength，一度に送る packetsize，パケットを故意に送らないようにするための FragmentSkip を設定できるようにした．

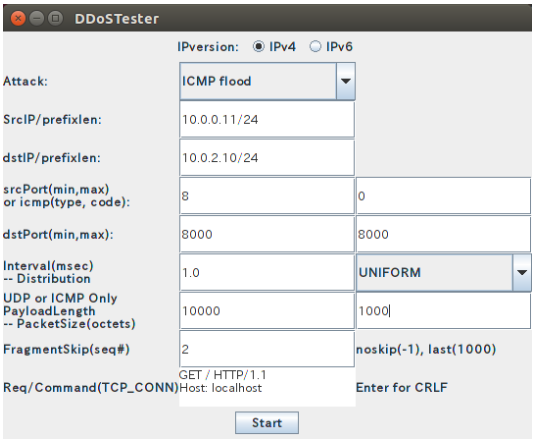

図 2　GUI

3.2　ネットワーク構築

攻撃プログラムは外部で実行があると危険なため，ネットワークエミュレータソフト CORE を用いる．エミュレータを用いることで外部にパケットが漏れることがなく，現物と同じ結果を出すことができる．

本研究の実験に用いるネットワーク構成図を図 3 に示す．図 3 において，n9 に配置したホストの IP アドレスを偽って送信するので，Victim と位置づける．n5 を Attacker，n8 を Target として攻撃を仕掛ける．TCP-Connection-Flood，HTTP-GET-Flood では，http サーバが動いていることを確認するために，netstat コマンドを使用する．また，TCP-Connection-Flood は嘘の SrcIP アドレスであると返事がでないので，default 経路を IPv4 は 10.0.1.1 に，IPv6 は 2001:1::1 に設定する．

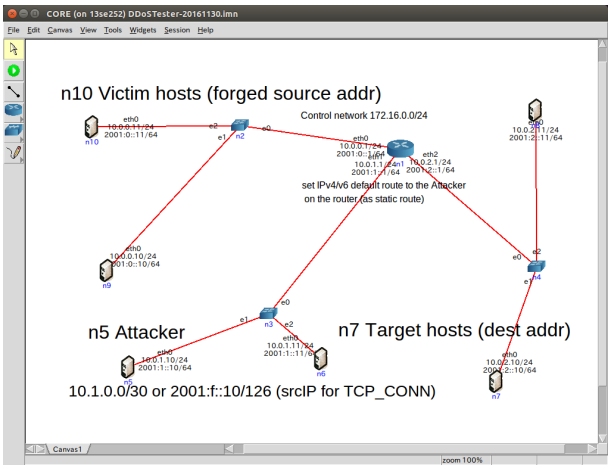

図 3　ネットワーク構成

4　システムの実現

DDoS 模擬攻撃ツールを作成するにあたり，C++のクラス，Java を用いた GUI について説明する．

4.1　HTTP-GET-Flood 攻撃の関数

TCP 接続後に HTTP GET を送信するために setTCPAppPayload を作成する．これは TCP の ACK を送信するためのパケットキャプチャスレッドで，if(tcpAppPayloadlen == 0) continue; のあとで送信するペイロードを設定しておく関数である．C++では

```
void setTCPAppPayload
(const uint8_t * payload, int length);
```

と定義している．

まず IPv4，IPv6 のどちらであるか場合分けをする．次に IP ヘッダ，TCP ヘッダポインタにキャストする．tcp ヘッダが終わった次から memcpy で tcpAppPayload を tcpAppPayloadlen だけコピーする．するとデータの全長が変わるので，totalLen += tcpAppPayloadLen とする．また ip ヘッダ中の totallength も変わるので，

```
sendIph->tot_len = htons(ntohs(sendIph->tot_len)
 + tcpAppPayloadLen);
```

と変更した．

Java のプログラムでは

```
public native void setTCPAppPayload(@cast
("uint8_t*") BytePointer payload, int length);
```

と定義している．

4.2 IP-Fragment 攻撃の関数

fragment 攻撃をするためには，パケットを分割，またパケットを故意に送信しないようにする必要がある．IPv6 ヘッダではフラグメントヘッダが拡張ヘッダに組み込まれている．MTU を超えるパケットは，IP ヘッダを除くデータ部分を 8 の倍数長でフラグメントをする．例として MTU が 1500octets で 2000octets のパケットを 500 ずつ分割する場合，IP ヘッダを除くと 480 となる．480 は 8 の倍数長であるので 480 バイトずつフラグメントする．Java のプログラムでは public native void setFragments(int fragsize, int fragskip) と定義している．

4.3 システム全体の構造

システムを実現するにあたり必用なプログラムと説明を以下に示す．

用意するファイル

DDoSTester-20161130.imn…CORE 図
DDoSTesterApp.java…GUI フレーム
NativeLibrary.java…GUI と C++プログラムのつなぎ
NativeLibrary-code.h…対象ファイルの include
DdoSTester.cpp/DdoSTester.h…攻撃プログラム
Java では書けない処理用の C++クラス
javacpp.jar…依存ライブラリ兼 javacpp 処理実行プログラム
Make f ile…コンパイル手順

4.4 コンパイル方法

コンパイル手順 1 から 4 を MakeFile にまとめる．MakeFile 内でのコンパイル手順を以下に示す．

コンパイル手順

```
1.Wrapper クラスをコンパイル
javac -cp javacpp.jar
NativeLibrary/NativeLibrary.java
2.native コードのコンパイルとリンク
   java -jar javacpp.jar NativeLibrary
                          /NativeLibrary  \
        -Xcompiler -L/usr/lib
                    /x86_64-linux-gnu   \
        -Xcompiler -lccgnu2 \
        (以後略)
3.GUI プログラムのコンパイル
  j avac DDoSTesterApp.java
4. 実行可能 jar ファイル作成
```

4.5 GUI の設計

本研究では，Java を用いて GUI を作成した．Java を使用することによって GUI をより簡単に作成でき，他の言語と共に使用しやすくなる．

4.6 Java からの C++クラスの利用

C++の実験プログラムを Java の GUI で動かす方法として JNI(Java Native Interface)[2][4] がある．JNI とは，Java プラットフォームにおいて Java で記述されたプログラムと，他の言語で書かれた実際の CPU の上で動作するコード（ネイティブコード）とを連携するためのインタフェース仕様であり，Java だけでかけない処理の実行，Java では遅い処理を高速化できる．

また，JNI を自動生成するツールとして，JavaCPP, SWIG, JNA(Java Native Access) が挙げられる．本研究では JavaCPP[1] を使用する．JavaCPP は，一部の C/C++コンパイラアセンブリ言語とやり取りする方法とは異なり，Java 内のネイティブ C++への効率的なアクセスを可能になる．

4.7 GUI の操作

本研究では，ネットワークエミュレータ CORE 上で実験する必要がある．CORE 上の端末からでは，直接 GUI プログラムを表示することができない．ホスト PC からの起動方法を以下に示す．

ホスト PC からの起動方法

1 ホスト OS に sshd を入れる
2 自動起動になるので OS 起動時に sshd がスタートしないように変更する
  sudo vi /etc/init/ssh.conf
  # start on runlevel [2345]
3 core-gui での追加設定を行う
  ・CORE 内 Attacker の SSH を on
  ・CORE 内 network prefix の範囲を指定
4 プログラムを動かしたいノードの端末で IP アドレスを確認
5 ホスト OS から login する
    slogin -X 4 で確認した CORE 内ホストの IP アドレス
    本研究の場合 slogin -X 172.16.0.5
6 ログインできたことを確認して Java プログラムを起動する

5 実験

実験手順及び，実験結果について記述する．実験手順は以下の通りである．
1 CORE 内でネットワークを構築
2 Attacker から Target に向け，攻撃プログラムを実行
3 Target 側で tcpdump を実行し観測
また TCP-SYN-Flood，TCP-Connecion-Flood は図 4 の netstat コマンドを用いる．

5.1 IPv4 ICMP-Fragment

攻撃のパラメータは IP バージョン 4，攻撃に使用するプロトコルは ICMP，攻撃の間隔は 1.0，Src,DstIP の範囲は，10.0.0.11/24，10.0.2.10/24，ICMP(type/code) は

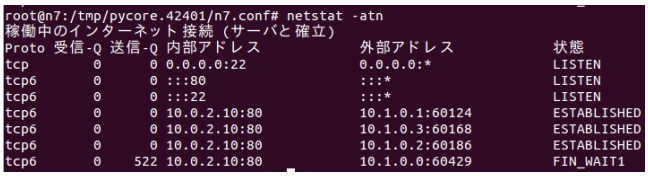

図 4 netstat

8/0，Payloadlength は 1200，一度に送る packetsize は 500，パケットを故意に送らないようにするための FragmentSkip は-1 に設定した．実験結果を以下に示す．

IPv4 ICMP-Fragment

```
13:50:29.259243 IP 10.0.0.11
 > 10.0.2.10: ICMP echo request,
 id 0, seq 0, length 480
13:50:29.259272 IP 10.0.0.11
 > 10.0.2.10: icmp
  // 分割されたパケットが続く
13:50:29.259372 IP 10.0.2.10
 > 10.0.0.11: ICMP echo reply,
 id 0, seq 0, length 1180
```

Attacker が Victim に偽って，Target に向かって分割したパケットを大量に送信している．Target は Victim に対して echo reply を返信していることが分かる．IPv6 も同様に成功した．

5.2 IPv6 UDP-Fragment-Skip

攻撃のパラメータは IP バージョン 6，攻撃に使用するプロトコルは UDP，攻撃の間隔は 1.0，Src,DstIP の範囲は，2001:0::11/64，2001:2::10/128 Src，Dst ポートの範囲は 60000/61000，Payloadlength は 1200，一度に送る packetsize は 500，FragmentSkip は 1 に設定した．実験結果を以下に示す．

IPv6 UDP-Fragment-Skip

```
18:07:22.749503 IP6 2001::218e:eb55:c8af:7c25
> 2001:2::10: frag (0|448) 60301
> 8000: UDP, length 1200
18:07:24.738035 IP6
2001::8182:5074:1d2d:aa53
> 2001:2::10: frag (0|448) 60591
> 8000: UDP, length 1200
18:07:26.463642 IP6 2001:2::10
> 2001::fe0f:3d2c:e8f3:306b:
ICMP6, time exceeded in-transit (reassembly),
length 504
18:07:31.839684 IP6 2001:2::10
> 2001::c9e5:1369:7946:a5c:
ICMP6, time exceeded in-transit (reassembly),
length 504
```

嘘のソースアドレスから target に向かって UDP パケットを送信しているが，2 つ目のパケットを送信していないため，分割が完成していないことが分かる．分割が完成しないことによって時間切れとなり ICMP ip reassembly time exceeded が表示される．よって IPv6 の UDP-Fragment が成功したと言える．同様に IPv4 も成功した．

5.3 その他の実験

上記の実験以外にも，IPv4，IPv6 において UDP-Flood 攻撃，ICMP-Flood 攻撃，TCP-SYN-Flood 攻撃，TCP-Connection-Flood 攻撃の GUI を用いての実験も成功した．

6 おわりに

本研究では，ネットワークエミュレータ CORE 内で，Attacker から Target に向かい様々な種類の DDoS 攻撃を行える GUI ツールを作成した．

攻撃種類については，2015 年度の川上，黒田の卒業論文で作成途中だった UDP-Flood，ICMP-Flood，TCP-SYN-Flood の確認と改良，TCP-Connection-Flood の修正・実験をした．また，IP-Fragment の追加をし 5 種類の攻撃において Java を用いた GUI での動作確認をした．

今後の課題は HTTP-GET-Flood の完成，その他の攻撃種類の追加が挙げられる．また，別のプラットフォーム上でも動作するよう C++プログラムを作成することも挙げられる．その際，ソケット関数はそのままで良いが、データリンク層が違うのでそれぞれに合わせたプログラムを用意する．

参考文献

[1] GitHub: *JavaCPP*, https://github.com/bytedeco/javacpp (Accessed Dec 2016).

[2] ORACLE: *JNI*, http://docs.oracle.com/javase/7/docs/technotes/guides/jni/ (Accessed Dec 2016).

[3] RFC: *IP Fragment*, https://tools.ietf.org/html/rfc791 (Accessed Dec 2016).

[4] Toru, T.: JNI 使用方法, http://www.02.246.ne.jp/ torutk/javahow2/jni.html (Accessed Dec 2016).

[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 Dec 2016).

[6] 佐々木 崇 : アーバネットワークスに学ぶ DDoS 攻撃の実態と対策, http://ascii.jp/elem/000/000/765/765962/ (Accessed Dec 2016).

[7] 川上健人，黒田涼介：DDoS 攻撃模擬ツールの試作，南山大学 情報理工学部システム創成工学科 2015 年度卒業論文 (2016).