マルチメディア通信と分散処理 111－22
コンピュータセキュリティ 20－22
（２００３．２．２８）

トラフィックジェネレータによるDDoS攻撃の再現

三輪 信介* 滝澤 修* 大野 浩之*

ネットワークセキュリティに関する実装ベースの実験を行う場合には、攻撃の再現は重要である。脆弱性を衝く攻撃に関しては、攻撃の実現手法が分かっていれば容易に再現することができるが、DDoS攻撃のように大量の通信を用いる攻撃の再現は容易ではない。

本稿では、DDoS攻撃のように大量の通信を用いる攻撃をトラフィックジェネレータで再現する試みについて、攻撃を分類し、それぞれの再現に必要な要件から機能を導き、概念設計や開発計画について述べる。

A Concept and Design of DDoS attack generator

Shinsuke Miwa† Osamu Takizawa† Hiroyuki Ohno†

To replay attack is important on the network security experiment. Because DDoS attacks send massive traffic to a target, to replay DDoS attack must generate massive traffic. Therefore replaying DDoS attack is difficult on experiment.

In this paper, we describe a concept and design of DDoS attack generator using a traffic generator system.

1 背景

ネットワークセキュリティに関する実験を行う場合、大きく分けると、モデルベースのアプローチと実装ベースのアプローチがある。

モデルベースのアプローチでは、ネットワークを模式化し、攻撃の影響や対策の有効性などを、シミュレータなどによって検証する。

実装ベースのアプローチでは、実際に使われているサーバやルータ、ファイアウォール、IDSなどを用いて、攻撃の貫通試験を行ったり、実際に対策技術を施した装置などを用いて検証を行う。そのため、実装ベースのアプローチでは、攻撃の再現が重要である。

1.1 攻撃の再現

Buffer Overflow Exploitのような脆弱性を衝くような攻撃については、攻撃を実現するプログラムコードと手順を模擬することによって、比較的容易に再現することが可能である。

しかし、DDoS攻撃[1]など、大量の通信による攻撃については、通信トランザクションを大量に発生させる必要があり、その再現は容易ではない。

1.1.1 実機による再現

攻撃に実際に用いられるPCやサーバなどの実機を大量に用いて再現する方法が考えられるが、整備や運用にかかるコストは攻撃の規模に比例して大きくなるため、現実的とはいえない。

実機を用いる場合には、実際の攻撃を実際の攻撃に利用された実装を用いて再現できるという利点はある[2]が、攻撃側の挙動を計測する必要がないのであれば、実機による再現は過剰である。

さらに、実機を用いる場合には、攻撃の規模が実

*独立行政法人 通信総合研究所 情報通信部門 非常時通信グループ
†Emergency Communications Group, Information and Network Systems Division, Communications Research Laboratory

機の台数によって制約を受けるため、攻撃の規模の変化に柔軟に対応することはできない。

1.1.2 トラフィックジェネレータによる再現

大量の通信による攻撃を再現するために、トラフィックジェネレータによって大量の通信を発生させる方法が考えられる。

トラフィックジェネレータを用いれば、単純に大量の通信パケットを必要とする攻撃の再現は可能である。

しかし、近年の攻撃の再現においては、TCP セッションや各種のプロトコルのトランザクションを扱う必要がある。また、異常なパケット、セッションやトランザクションの再現も重要である。

現在の多くのトラフィックジェネレータは、TCP セッションを扱うことはできるが、各種のプロトコルのトランザクションを扱うことはできない。[1] また、正常な通信を模擬することを目的とするため、異常な通信の模擬機能は不足している。

このように、現状のトラフィックジェネレータでは十分とはいえない。

そこで、本稿では、トラフィックジェネレータによって DoS/DDoS 攻撃を中心として、攻撃を再現するために必要な機能拡張などに関して議論し、トラフィックジェネレータを用いた DDoS 攻撃再現の試みに関して述べる。

2 DoS/DDoS 攻撃の分類

まず、再現の対象とする攻撃を分類し、その特徴を明らかにする。

DoS 攻撃とは、何らかの攻撃手法によって、インターネット上のサービスを利用できない状態にする攻撃である。攻撃の対象となるのは、サービスを提供しているホスト自身か、そのホストがサービスを提供する上で必要となるネットワーク自身およびルータなどのネットワーク機器である。

攻撃によって、インターネット上で何らかのサービスを不能にする方法は、大きく分けて、

1. ホスト自身もしくはルータなどを機能不全の状態にする
2. 必要な資源を強制浪費し資源不足にする

の 2 種類である。[2]

ここでは、1 を脆弱性攻撃型、2 をその代表的な手法から大量パケット送りつけ型として分類し、さらにこれらの複合的な手法としてワームによる DoS 攻撃を取り上げる。

2.1 脆弱性攻撃型

ホストおよびルータの OS やプロトコルスタック、サービスプログラムに存在しているバグなどの脆弱性を衝く攻撃手法を脆弱性攻撃と呼ぶ。

サービスを提供しているホストやルータを機能不全の状態にするような脆弱性攻撃を、ここでは脆弱性攻撃型の DoS 攻撃と呼ぶことにする。

脆弱性攻撃によって、行われる DoS 攻撃で引き起こされる損害は、主に、

1. システムがダウンする
2. 余分に資源が浪費される

の 2 種類である。

1 には、不正な`ICMP echo request` パケットによって受信したホストをハングアップさせる *Ping Of Death* や、Microsoft Windows を搭載したホストの NetBIOS ポートに対し Out of Band データを送信することによってシステムをハングアップさせる *WinNuke* などがある。

2 には、`finger` サービスに特定のコマンド文字列を問い合わせとして送ることでメモリと CPU 資源を浪費させる *finger attack* や、送信元 IP アドレスと送信先 IP アドレスを同じにしたパケットを送ることで無限ループに陥らせ CPU 資源を浪費する *LanD (Snork) attack* などがある。

LanD attack では、攻撃のために必要なので、送信元 IP アドレスは必ず詐称される。他の攻撃手法でも、ほとんどの場合、攻撃者は特定を避けるために、送信元 IP アドレスを詐称したパケットを用いる。これらの方法では、被害ホストからのパケットを受け取る必要がないため、IP アドレスを詐称して攻撃するのは容易である。

ただし、*finger attack* は TCP 接続を用いる必要があるために、送信元 IP アドレスを詐称する場合には、TCP 接続における ISN（初期シーケンス番号）の予測などの工夫が必要になる。しかし、この攻撃手法でも被害ホストからのパケットを受け取る必要

[1] 特定のプロトコルについて扱うことができる製品は存在する。

[2] メール爆弾なども DoS 攻撃の一種であるが、本稿では取り扱わない。

はないので、ISN 予測などが可能なら攻撃を成立させることができる。

これらの脆弱性攻撃型 DoS には、

- 少量の通信で損害を与えることができる
- 脆弱性を塞ぐパッチなどの修正で対策をとれる

といった特徴がある。

2.2 大量パケット送りつけ型

インターネット上でサービスを行っているホストのサービスを不能にする最も単純な方法は、大量のパケットを送りつけることである。

大量のパケットを送りつけることで、

1. サービスに必要なネットワーク帯域の強制浪費
2. ホストのサービス向け資源の強制浪費

を図る攻撃をここでは大量パケット送りつけ型の DoS 攻撃と呼ぶことにする。

1 の典型的なものは、*ICMP flood* や *UDP flood* だろう。対象のネットワーク帯域を埋め尽くすほどの量の ICMP パケットや UDP パケットを送りつけた場合、対象はネットワーク帯域の欠乏からサービスを継続できなくなる。

2 の典型的なものは、*SYN flood* だろう。対象に対し、TCP のスリーウェイハンドシェイクを不完全な形で大量に行い、TCP のコネクション維持に必要なメモリを浪費させることで、対象はメモリ資源が枯渇し、サービスを継続できなくなる。同様に、*Telenet flood* では、大量の `Ctrl+D` を送ることで、対象の CPU 資源を枯渇させると同時に、ネットワーク経由での操作を妨害する。

2.2.1 増幅器

大量パケット送りつけ型の DoS 攻撃では、何らかの方法で大量にパケットを発生させる必要がある。単一のホストでこれを担うのは、効率的とはいえない。そこで、この種の攻撃では、

1. 増幅器となる何らかの仕組みを利用する
2. 大量のホストから同時に攻撃を行う

などの方法が取られる。

1 は増幅器と呼ばれ、2 は DDoS（Distributed Denial of Service）攻撃と呼ばれる。

増幅器には、

- IP ブロードキャストを利用する方法
- 過剰な応答を利用する方法

などいくつかの方法がある。

IP ブロードキャストを利用する方法では、単純に要求に対し応答が返るようなサービスを利用し、IP ブロードキャストアドレスに対し、要求パケットを送信する。これに対する応答パケットは、IP ブロードキャストを受信したすべてのホストから返ってくることになる。

つまり、たった一つの要求パケットで大量の応答パケットを発生させることができる。この要求の際に、送信元 IP アドレスが対象ホストの IP アドレスに設定されていれば、対象ホストに対して大量の応答パケットが送りつけられることになる。

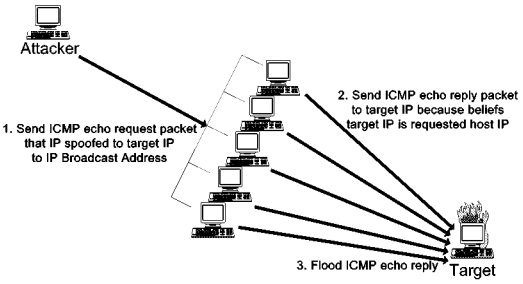

図 1: Smurf 攻撃

Smurf は、送信元 IP アドレスを対象ホストに設定した `ICMP echo request` を IP ブロードキャストアドレスに送ることで、`ICMP echo reply` が対象ホストに大量に送られる攻撃である（図 1 参照）。*Fraggle* は、同様に UDP の echo サービス、daytime サービス、qotd サービス、chargen サービスなどを利用する。

過剰な応答を利用する方法では、一つの要求パケットに対し大きなもしくは大量の応答パケットが返ってくるサービスを利用する。

DoomDNS は、DNS サーバに対し、多数の応答を生成させるような異常な要求を UDP で送信する。この際に、送信元 IP アドレスに対象の IP アドレスを設定しておくことで、対象に対し、DNS サーバから多数の応答が送信される。

echo-chargen **ループ**は、echo サービスを実行しているホスト（仮にホスト A とする）の IP アドレスを送信元とし、送信元ポートを echo サービスのポートに設定した UDP パケットを別の chargen サービスを実行しているホスト（ホスト B とする）に送信する。受信したホスト B では、chargen によって生

成された文字列をホストAのechoサービスポートに送り返すため、ホストAではこれをechoし、ホストBのchargenに送り返すというループが発生する。これによって、対象となっているホストAおよびホストBはCPU資源を浪費するとともに、その周辺ネットワークでは帯域が浪費されることになる。

2.2.2 DDoS攻撃

大量のホストから同時に攻撃を行うためには、大量のホストに何らかの方法で攻撃プログラムをホストに仕込む方法が用いられる。攻撃プログラムを仕込んだホストに対して、攻撃対象を指定し、攻撃を命令することで、大量の攻撃が対象に対して行われることになる。

しかし、大量のホストに仕込んだ攻撃プログラムに対し、手動で攻撃命令を送信するのは効率的ではないために、この種の攻撃では専用のツールを用いる。これをDDoSツールと呼ぶ。

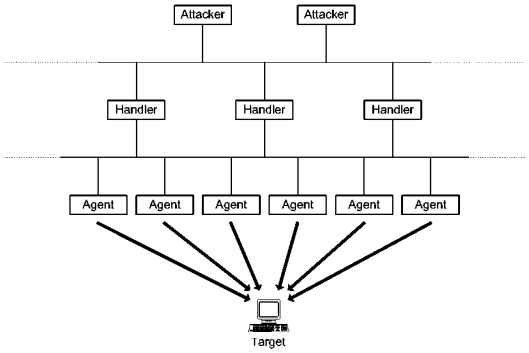

図2: DDoS攻撃における階層 (mstream)

多くのDDoSツールでは、攻撃者およびそのホストの負担をより軽減するために、階層構造をとっている。攻撃ホスト側のClientとClientから直接指示を受けるHandler、そして実際に攻撃を行うAgentの3階層である（図2参照)。一つのClientに対しHandlerは数台から数十台、一つのHandlerに対しAgentは数十台から千台程度である。Clientからの一度の攻撃命令で、場合によっては何万というAgentから攻撃が行われることになる。攻撃内容は、*ICMP flood*や*UDP flood*、*SYN flood*などの単純なものであったとしても、その効果は非常に大きいといえるだろう。

代表的なものに *trinoo*[3] や *Stacheldraht*[4]、*mstream*[5] などがあるが、どれも通信の方法や攻撃内容に若干の違いはあるものの、概要は上述の通りで大きな差はない。

これらのような大量パケット送りつけ型DoSには、

- 大量の通信で資源を浪費させる
- 浪費させられるだけなので対策が難しい

といった特徴がある。

2.3 ワーム

ワームとは自己増殖機能をもった単体動作プログラムのことであり、ワームだからといって必ずしもDoS攻撃を発生させるわけではない。

しかし、近年いくつかのワームはネットワークを通じて伝播増殖する際に、

- 伝播先を見つけるためのネットワークスキャン
- 伝播増殖のための通信

を大量に行うため、結果としてネットワーク帯域を浪費し、DoS攻撃となる。

2001年に流行した *CodeRed*[6] や *Nimda*[7] は、スキャンとHTTPのリクエストを利用した脆弱性攻撃を繰り返しながら増殖していくワームであるが、あまりに大量のスキャンと脆弱性攻撃が行われるため、インターネット全域に影響を及ぼした。

また、*SQL-Slammer*[8] は、Microsoft社のSQLサーバおよびSQLエンジンへの脆弱性攻撃によって増殖するワームであったが、400バイト足らずのUDPパケットを感染しているホストが動作している限り無差別にばら撒き続けたため、インターネット全域に大きな影響を及ぼしたことは記憶に新しい。

このようにワームによるDoS攻撃は、

- 増殖のための通信がDoS攻撃になる
- 脆弱性攻撃を利用するので脆弱性に対する対策で対応できる

などの特徴がある。

なお、ワームは、他のDoS攻撃手法と違い、すでに攻撃を受けた感染ホストから通信が発生するため、送信元IPアドレスなどを詐称する必要はない。

3 要件と設計

トラフィックジェネレータを用いてこれらの攻撃を再現する手法について検討する。

2節に述べたように、DoS/DDoS攻撃には攻撃の手法に応じて、いくつかの類型に分かれる。これらの攻撃を再現する上では、類型ごとに必要な機能に違いがあると考えられる。

3.1 要件

それぞれの特徴から、再現における要件をまとめると以下の通りである。

- 脆弱性攻撃型
  - 少量の通信で良い
  - 特定の通信プロトコルの状態遷移に従う必要がある
  - 送信元IPアドレスなどを変更する必要がある
- 大量パケット送りつけ型
  - 大量の通信が必要
  - 送りつけるだけで状態遷移は特に必要としない
  - 送信元IPアドレスなどを変更する必要がある
- ワーム
  - 大量の通信が必要
  - 特定の通信プロトコルの状態遷移に従う必要がある
  - 送信元IPアドレスなどを変更は必要としない

脆弱性攻撃型については、少量の通信で再現することができるため、トラフィックジェネレータを用いる必要はない。

大量の通信を必要とする大量パケット送りつけ型とワームの再現には、トラフィックジェネレータを用いた再現が必要だと考えられる。

3.2 概念設計

では、どのようにすればトラフィックジェネレータでDDoS攻撃を再現することができるだろうか。

トラフィックジェネレータは、パケットを大量に発生させる装置である。このトラフィックジェネレータでDDoS攻撃に利用されるパケットを大量に発生させることができれば、大量パケット送りつけ型の攻撃を再現することができる。よって、このために必要な機能は、用意したパケットを再生する機能である。

しかし、実際の攻撃では送信元IPアドレスなどを変更しながら大量のパケットが送られる。そこで、送信元IPアドレスやチェックサムなどを計算し、変更する機能も必要となる。

また、実際に再現実験を行う上では、どのようにしてパケットを用意するかが重要である。

一つの方法は、実際にtcpdumpなどのパケットキャプチャソフトウェアを利用して、取得したパケットパターンを利用するという方法が考えられる。この方法の場合には、取得したパターンデータを実験に利用しやすいように編集する必要があるだろう。

もう一つの方法は、何らかの記述言語によって、パケットパターンを記述し、それに従ってパケットを発生するもしくはパケットパターンのデータを生成する方法である。

これらの方法によって、ほとんどの大量パケット送りつけ型の攻撃に関しては再現することが可能となるが、例えば、*Telenet flood*のように特定のアプリケーションプロトコルのトランザクション中に異常な挙動を起こすような攻撃の再現は困難である。また、ワームのように特定のアプリケーションプロトコルを利用する攻撃の再現も困難である。

そのため、何らかの方法でプロトコルのトランザクションを含めた攻撃パターンの記述を行い、それに伴ってパケットを生成するような機能が必要となる。

3.3 開発計画

上記のような概念設計に基づき、トラフィックジェネレータを利用したDDoS攻撃再現のための装置に関して、開発計画を立てた。

追加すべき機能に応じて、その難易度に違いがあることから、計画を三段階に分け、それぞれPhase-I、Phase-II、Phase-IIIとした。

3.3.1 Phase-I

まず、Phase-Iでは、パケットキャプチャソフトウェアで取得したパケットパターンをそのままもしくは

編集し、再生する機能を実現する。再生時には、パケットの生成レートや時間、回数などを指定可能にする。

これにより、状態遷移のまったくないような大量パケット送りつけ型の攻撃を再現することが可能となる。

3.3.2 Phase-II

次に、Phase-II では、パケットパターンを記述言語で記述し、それに基づいてパケットパターンデータを生成する機能を実現する。Phase-I で開発した再生機能によって生成したパケットパターンに基づいてパケットを発生する。

記述言語において、遅延などを記述できるようにすることで、攻撃者側で閉じた状態遷移に関して記述を可能とする。これによって、攻撃者側で閉じた状態遷移を持つ攻撃も再現することができるようになる。

3.3.3 Phase-III

Phase-III では、プロトコルのトランザクションを記述し、パケットの受信とそれに伴う状態遷移を行い、アプリケーションプロトコルを利用した攻撃を実現する。

これにより、ワームなどの特定のプロトコルの状態遷移を必要とする攻撃の再現を可能とする。

4 課題と展望

現在、Phase-I が終了し、Phase-II は検証段階に入ったところである。

今後は、Phase-II の検証を行い、Phase-III の詳細設計や実装を経て、多くの攻撃を再現できるようにする必要がある。

また、現在はすべてトラフィックジェネレータの上で動作しているが、本格的な試験を行う前の予備調査などのために、PC 上で動くような環境を構築することも重要であると考えている。

5 おわりに

本稿では、トラフィックジェネレータによって大量トラフィックを用いる DDoS 攻撃を再現するための試みについて述べた。現状では、検証なども十分ではないが今後、研究開発を進め随時公開していきたい。

謝辞

CawNetworks 社（現スパイレントコミュニケーションズ社）の Roy Chua 氏、Roland Hendel 氏には、技術的な面で助言を頂くとともに、実験装置の実装面で多大な協力を頂いた。奈良先端科学技術大学院大学の山口 英教授には、いくつかの有益な助言を頂いた。記して感謝の意を表したい。

参考文献

[1] Geng, Xianjun, and Andrew B. Whinston, “Defeating Distributed Denial of Service Attacks”, IT Pro, July-August 2000.

[2] 大野 浩之, 武智 洋, 永島 秀己, “インターネットの脅威に対抗しうる脆弱性データベースと検証システムの構築”, 情報処理学会, DSM シンポジウム 2001, Feb. 2001.

[3] D. Dittrich, “The DoS Project’s ”trinoo” distributed denial of service attack tool”, ⟨URL: http://staff.washington.edu/dittrich/misc/trinoo.analysis⟩, Oct. 1999.

[4] D. Dittrich, “The ”stacheldraht” distributed denial of service attack tool”, ⟨URL: http://staff.washington.edu/dittrich/misc/stacheldraht.analysis⟩, Dec. 1999.

[5] D. Dittrich, G. Weaver, S. Dietrich, and N. Long, “The ”mstream” distributed denial of service attack tool”, ⟨URL: http://staff.washington.edu/dittrich/misc/mstream.analysis.txt⟩, May 2000.

[6] IPA, “「CodeRed ワームに関する情報」—新種の CodeRed II に注意を—”, ⟨URL: http://www.ipa.go.jp/security/ciadr/vul/20010727codered.html⟩, Aug. 2001.

[7] SecurityFocus, “ARIS Predictor: Nimda Worm Analysis”, ⟨URL: http://aris.securityfocus.com/alerts/nimda/010919-Analysis-Nimda.pdf⟩, Sep. 2001.

[8] CERT Coordination Center, “MS-SQL Server Worm”, CERT Advisory CA-2003-04, Jan. 2003.