DUBLIN CITY UNIVERSITY

ELECTRONIC AND COMPUTER ENGINEERING

EEXXX - Module Title

Assignment Subtitle

Author

Michael Lenehan michael.lenehan4@mail.dcu.ie

Student Number: 15410402

XX/XX/20XX

Declaration

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I have read and understood the Assignment Regulations set out in the module documentation. I have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

I have read and understood the DCU Academic Integrity and Plagiarism at `https://www4.dcu.ie/sites/default/files/policy/1%20-%20integrity_and_plagiarism_ovpaa_v3.pdf` and IEEE referencing guidelines found at `https://loop.dcu.ie/mod/url/view.php?id=448779`.

Signed: _____ Date: XX/XX/20XX

Michael Lenehan

Contents

Listings

1 Introduction

The aim of a Distributed Denial of Service attack is to disrupt the connection between a service, such as an application server or DNS server, and that services users.

Denial of Service (DoS) attacks typically exploited vulnerabilities with network or application layer protocols, for example, Syn Floods or HTTP Floods. With these types of attacks, spoofed IP addresses were typically used, both to mask the IP address of the attacker, and to take advantages of vulnerabilities of the protocols being used.

As DoS mitigation techniques improved, and attacks using IP spoofing became easier to defend against, attackers began utilising botnets to perform amplification attacks. Distributed Denial of Service (DDoS) attacks utilise large number of hosts to perform attacks. These types of attacks prove more difficult to defend against than typical Denial of Service attacks.

In order to extensively evaluate the impact of Distributed Denial of Service attacks, and the role which IoT devices play in modern DDoS attacks, simulations can be used. Simulations can show the impact to a network, and the disruption caused by these attacks, without having to use real network devices.

1.1 Simulation Software

There were a number of simulation software packages considered for the tests contained in this document. Each of these packages offer their own set of advantages and disadvantages.

1.1.1 ns-3

The most commonly used network simulation software is "ns-3"; a "discrete-event network simulator for Internet systems". This package is commonly used for research

purposes, and numerous examples of D/DoS simulations implemented using ns-3 can be found.

1.1.2 Cooja

Another simulation software package considered was "Cooja". This package simulates Wireless Sensor Networks built on the Contiki IoT operating system. While the underlying system being simulated is an IoT device, there was little information to be found on implementing the types of testing required, and as such this software package was ruled out.

1.1.3 Mininet

The final software package considered was "Mininet". Mininet is a network simulator which uses Linux containers to emulate network devices. As the network devices are emulated using containers, each host uses real Linux Kernel code.

As the Mininet hosts run Linux Kernel code, Linux applications can be easily run. This is advantageous as common attack tools can be installed and executed. Each host can be assigned a unique IP address allowing the attacking host to address it's target as in a real attack situation. As each host is implemented as a container, a terminal can be open for each host for manual command execution.

Mininet implements Software Defined Networking using the OpenFlow protocol. By default, Mininet uses the OpenFlow reference controller, however, it also allows for the use of remote controllers. The "Floodlight" controller will be used for this purpose, as it has a GUI application which can display the network topology.

Mininet virtualizes the network links between each host or switch in the network. These links can be assigned parameters such as a delay time on the link, or a set bandwidth.

The Python API implemented by Mininet allows for the topology and the terminal commands to be created and executed programmatically. The Python API will be used in order to execute repeatable tests with reproducible results.

Due to the advantages offered by the Mininet simulation software package, this will be the simulator of choice for testing. There are however a number of disadvantages which must be noted.

1. Containers share the file system of the host, i.e. the desktop or VM on which Mininet is being run
2. A network cannot exceed the bandwidth of a single server
3. Non-Linux-compatible OpenFlow switches and applications are not supported

While these limitations must be kept in mind, they will not prove to be a hindrance for the purposes of these simulations.

2 Denial of Service Simulation

For the initial Denial of Service simulation, the network topology is as shown in Figure 1. This network consists of an attacking traffic source (shown on the left), a legitimate user traffic source (shown on the right), and a server (shown on the bottom).

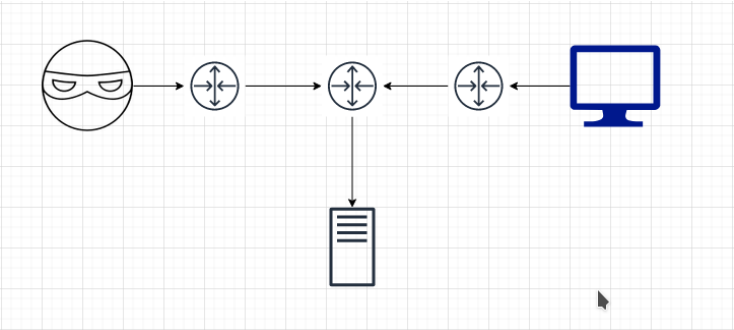

Figure 1: Denial of Service Network Topology

As an initial attack, the "hping" tool is used. Hping is a tool for creating and transmitting TCP, UDP or ICMP packets. This tool has multiple functions, such as port scanning, and firewall mapping, but can also be utilised as a Denial of Service tool. By utilising the tools "rand-source" and "flood" flags, the tool can execute a SYN flood, sending TCP SYN packets to the target address via spoofed IP addresses.

3 Distributed Denial of Service Simulation

For the Distributed Denial of Service simulation, an expanded version of the topology used in Figure 1, consisting of a significantly larger number of attacking sources is used. This reflects the increase in attackers, which is common in DDoS attacks involving IoT botnets.

4 Results

5 Resources

The following list outlines both the hardware and software resources which were utilised in the execution of these simulations:

5.1 Hardware:

As these simulations utilised Linux containers, only a single PC was required to execute the simulations. The specifications of the laptop PC are as follows:

- HP Envy 17
  - Intel Core i7-6500U (Dual Core)
  - 12GB RAM

5.2 Software:

For these simulations, the Linux operating system was used for it's ease of installation of software packages, and also for the availability of containers. Note that a Linux virtual machine would be a suitable replacement.

5.2.1 Operating System:

- Manjaro Linux
  - Kernel Version 5.4.52-1-MANJARO
  - OS Type: 64-bit

5.2.2 Simulation Software:

- Mininet
  - Version: 2.3.0d6
- Floodlight Controller
  - Version: 1.2

6 Reproduction

The following steps must be performed in order to reproduce the achieved simulations. As the Manjaro Linux distribution was used for these simulations, all package manager instructions will be given for Manjaro.

6.1 Mininet

Mininet must first be installed. This can be done according to the Mininet documentation **??**. For Debian based systems, Mininet can be installed via the "apt" package manager, using the command:

```
$ apt-get install mininet
```

Listing 1: Debian-based Distro Mininet Install

For Arch based distributions, Mininet may be installed using the Arch User Repository using the following command:

```
$ yay -S mininet-git
```

Listing 2: Arch-based Distro Mininet Install

To test the installation, execute the command:

```
$ sudo mn --test pingall
```

Listing 3: Mininet installation test

If the error "ovs-vsctl: unix:/run/openvswitch/db.sock database connection failed" occurs, the Open vSwitch must be started using the command:

```
$ sudo /usr/share/openvswitch/scripts/ovs-ctl
    start
```

Listing 4: Open vSwitch service start command

6.2 Floodlight OpenFlow Controller

The Floodlight OpenFlow Controller can be installed via GitHub**??**. For version 1.2 (as used for these simulation), the Java 8 development kit must be installed:

```
$ sudo pacman -S openjdk8-src
```

Listing 5: openjdk8 installation

Floodlight has a number of dependencies which can be installed via the following command:

```
sudo pacman -S git ant maven python-dev
```

Listing 6: Floodlight Dependencies

To download and build Floodlight, execute the following commands:

```
$ git clone git://github.com/floodlight/
    floodlight.git
$ cd floodlight
$ git submodule init
```

```
$ git submodule update
$ ant
# If the ant build fails, Floodlight can be built
    using Maven
$ sudo mvn package
```

Listing 7: Floodlight installation commands

6.3 Source Code

The source code for the simulations can be found on GitHub, and can be downloaded using the following command:

```
$ git clone git://github.com/
```

Listing 8: Source code download

7 Conclusion

A Appendix