Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Alvaro Garcia

Distributed Denial of Service Attack and OpenFlow:

Improving defense mechanism of DDoS with OpenFlow

Master's Thesis
Espoo, March 15, 2014

DRAFT! — February 14, 2014 — DRAFT!

Supervisors:	Aapo Kalliola M.Sc. (Tech.)
Advisor:	Aapo Kalliola M.Sc. (Tech.)

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author:	Alvaro Garcia		
Title:			
Distributed Denial of Service Attack and OpenFlow: Improving defense mechanism of DDoS with OpenFlow			
Date:	March 15, 2014	**Pages:**	20
Major:	Computer Science and Engineering	**Code:**	T
Supervisors:	Aapo Kalliola M.Sc. (Tech.)		
Advisor:	Aapo Kalliola M.Sc. (Tech.)		

Since DoS attacks are becoming more commons and emerging new technologies to separate the control plane and the data plane from the network devices (SDN), throughout this survey we will investigate how OpenFlow can help to prevent and locate these kinds of attacks. We will study the different DoS attacks and the current mitigation techniques. We will discuss as well, which of these techniques could be improved with OpenFlow and how develop them. In the end, we will implement and test some identified mitigation techniques and we will study their behaviour.

Keywords:	DoS, DDoS, SDN, OpenFlow
Language:	English

Acknowledgements

Espoo, March 15, 2014

Alvaro Garcia

Abbreviations and Acronyms

IP	Internet Protocol
SDN	Software Defined Networking

Contents

List of Figures

Chapter 1

Introduction

The original aim of the Internet was to provide an open and scalable network among research and educational communities, where billions of users are served through a global system of interconnected computer networks.

Unfortunately, with the rapid growth of the Internet over the last two decades, the number of attacks on the Internet has also increased rapidly. One of this attacks consists in disrupt the service provided by a network or server, either crashing the system sending some packets that exploit a software vulnerability or sending a large number of useless traffic to collapse the resources of the service. This kind of attack is known as Denial of Service (DoS) attack, or Distributed Denial of Service attack if it is launched by multiple hosts.

There are some design principles of the Internet that facility these kinds of attacks [5]:

Resource sharing: in IP networks, due to the packet-switched service, users share all the resources, and one user's service can be disturbed by other user's behaviour, so bandwidth attacks can disrupt services for legitimate users.

Simple Core and Complex Edge: One of the principles of the Internet is that the core network should be simple and pushes all the complexity into the end hosts. That means that the core of the networks is not able to integrate complex applications, such as authentication, security. Due to this simplification, when an attacker sends packets into the network and the victim receives them, it is almost impossible to recognize the real owner of the packets.

Fast Core Networks and Slow Edge Networks: The core networks needs to have high capacity due to the heavy traffic that has to support from many sources to many destinations. In contrast, an edge network needs less capacity because it only needs to support its end users. A disadvantage is that traffic from high-capacity core can crush the slow-capacity edge.

Taking advantage of these principles and their vulnerabilities, have been arising a large number of different DoS and DDoS attacks and, as a result, a parallel growing of defense mechanisms to avoid these attacks. We might consider it like a constant battle between both sides, in which technological improvements are taking an important role on it.

In the current network architecture, the network devices (particularly routers) are bundle with a specialized control plane and various features. This vertical integration essentially binds you to whatever software and features are shipped with those particular devices. Software Defined Networking (SDN) effectively breaks these pieces apart.

SDN is a type of network architecture that separates the network data plane (network devices that forwarding traffic) from the control plane (software logic that controls ultimately how traffic is flowing through the network). OpenFlow [4] is a standard interface defined between the control and forwarding layers of an SDN structure.

One of the reasons to separate the control plane and the data plane is that the software control of the network can evolve independently of the hardware.

A second reason is it allows the network to be controlled from a single high-level software program. This software, used to control the network (in our case, POX), even though taking in count that works with a high-level programming language, has a lower layer abstraction and the difficulty for the Network Programmers is increased. Due to this inconvenient, it is time to speak about *Frenetic* [2].*Frenetic* is a Network Programming language which gives a high-level abstraction from POX, allowing them direct control over the network. *Pyretic (Python + Frenetic)* is one of the Frenetic family programming languages which provide a domain specific sub-language for specifying dataplane packet processing.

The aim of this survey is how might SDN help us to improve current DDoS defense mechanism. Throughout this project, we will review the main DDoS defense and attack mechanisms and further we will go through some algorithms already developed and then, we will explain how could be improve them with OpenFlow. We will test these algorithms on virtual scenarios-through Mininet.

This thesis is structure as follows: The next chapter we will explain the background related with this survey. We talk about the current situation of DDoS attacks and defenses and how OpenFlow works and its structure. In the chapter 3,

Chapter 2

Background

2.1 DDoS attack and defense mechanisms

A denial-of-service attack is characterized by an explicit attempt by attackers to prevent the legitimate users of a service from using that service [1] provided by a network or server. There are two manner to launch this kind of attack. The first approach is overwhelm the network and occupy all the resources of a service sending massive volumes of useless traffic

2.1.1 Protocol Attacks

2.1.2 Bandwidth Attacks

2.1.3 Logic Attacks

2.2 OpenFlow

The explosion of mobile devices, server virtualization, security problems and advent of cloud service are among the reasons because the networking industry is beginning to question the traditional network architecture. OpenFlow is intended to solve the problem of assigning resources to users in a easy-way giving them the control plane of the network without disturbing the traffic flows.

In traditional routers and switches, both control plane (high level routing decisions) and data plane (packet forwarding) are embedded in the same device. An OpenFlow Switch separates these two functions (Figure 2.1). The data plane function still resides on the switch, while the control plane is moved to a separate device called Controller (see 2.2.2) that manages the switch and communicates to each other over the Secure Channel (see 2.2.1.2) via the OpenFlow protocol (see 2.2.1.3).

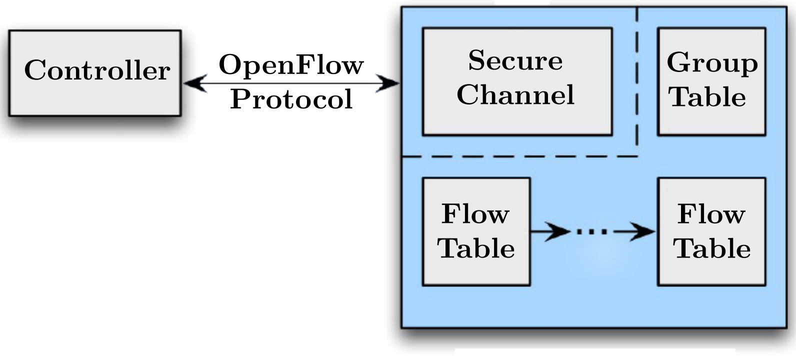


Figure 2.1: OpenFlow Switch components

The switch contains *flow tables* 2.2.1.1, which are updated through OpenFlow protocol adding, updating and deleting their *flow entries*. When the flow traffic arrives to the switch, it checks if the arrived packets match in the flow table, if so, the action defined in the flow entry is executed. Otherwise, the packet is either sent it to the Controller or dropped.

Throughout this section, we will explain in detail the main parts of the OpenFlow Switch, as well as the Controller and how they work together. The first version of the OpenFlow (1.1) protocol was released on 2011, one year later, in February 2012, the ONF approved and published the version 1.2. Nowadays, the current version of the protocol and the one that will be used in this project is the 1.4 [6].

2.2.1 Switch Components

2.2.1.1 Flow Table

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie

Table 2.1: Main components of a flow entry

Ingress Port	Ether source	Ether dst	Ether type	VLAN id	VLAN priority	IP src	IP dst	IP proto	IP ToS bits	src port	dst port

Table 2.2: Main components of the Match Field

2.2.1.2 Secure Channel

2.2.1.3 OpenFLow Protocol

2.2.2 Controller

2.2.2.1 POX

2.2.2.2 Frenetic

Chapter 3

Theory

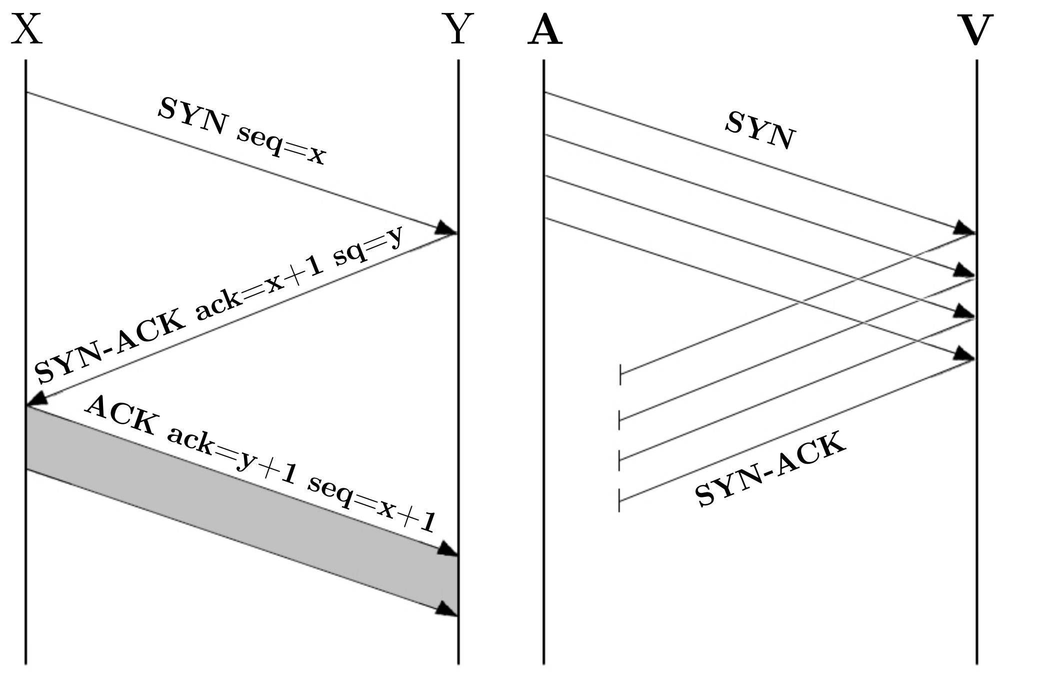
3.1 Protocol Attacks: TCP SYN Flooding

TCP (Transmission Control Protocol) is one of the most common protocols within the transport layer, and one of the core of the Internet Protocol suite (IP). TCP provides reliable, ordered, error-checked of stream of packets between two hosts. In addition to these characteristics, TCP is a connection-oriented protocol, that is, a prior connection between both parties is necessary before starting the exchange of information. This process is known as TCP three-way handshake (Figure 3.1(a)).

Suppose *X* as a client that wants to carry out a friendly TCP connection with the server *Y*. First of all, *X* requests by sending a synchronize *(SYN)* message to *Y*. The server receives the request and responses by sending an acknowledge *(SYN-ACK)* back to the client. Once the client receives the SYN-ACK, it responses with an ACK, an the connection is established.

While the server waits for the *SYN-ACK's* response, it keeps the connection in a half open state and maintains a backlog queue for the information about the connections. Once the server receives the *ACK*, it changes the state to *established* and frees up memory of the queue. Because the size of the backlog queue is not infinite, the half-open connection will remain on it until a time-out is exceeded. In the case that the queue is full, all new incoming connection requests will be dropped.

TCP SYN Flooding attack, as suggested by the name, aims to exhaust the server's backlog queue flooding it with *SYN* messages, but once they receive the corresponding *SYN-ACK* from the server, they will not response with the *acknowledge* message, forcing the server to keep the connection information in the backlog queue until the time-out is exceeded (Figure 3.1(b)). As a result, when a *friendly* client wants to set up a TCP connection with the server, will be denied. The biggest challenge of the attacker is ensure that the source IP address which is used to establish the connection is not reachable by the server. Otherwise, the source will send a *RST* packet to the server that cause server to reset the connection and free up the memory of the queue.



(a) TCP Tree-way handshake (b) SYN Flooding

Figure 3.1: TCP Three-way handshake and SYN Flooding

3.1.1 Methods of Attack

The attack can be categorized depending on how the attacker carries out the attack over the victim: Direct Attack, Spoofed-based Attack and Distributed Attack [3].

3.1.1.1 Direct Attack (Figure 3.2)

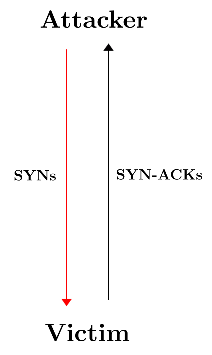


Figure 3.2: Direct Attack

In this case, the attacker is the one that accomplishes the attack direct to the victim. It does not even spoof the source IP address. Instead, it will just ensures that there will not be response after it receives the *SYN-ACK* message.

3.1.1.2 Spoofed-based Attack (Figure 3.3)

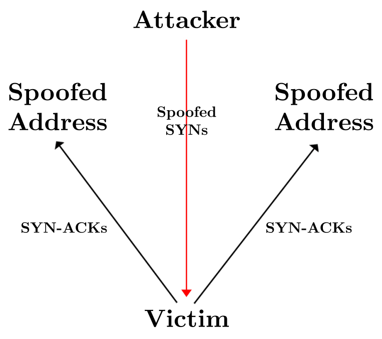


Figure 3.3: Spoofed-based Attack

This version of SYN flooding attack direct to the victim, but it spoofs the source IP address in the *SYN* packet. As we have explained before, a primary consideration is address selection. An attacker can choose spoofed IP address either using a single source which is known that will not response the *SYN-ACK*, or using a list of source address under the assumption that some percentage of them will not response.

3.1.1.3 Distributed Attack (Figure 3.4)

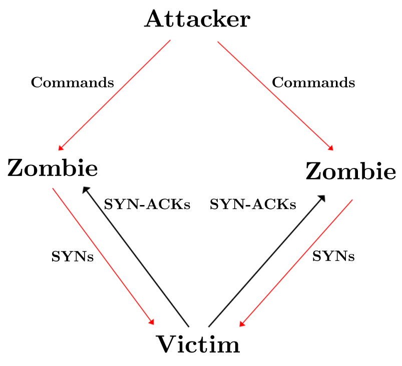


Figure 3.4: Distributed Attack

In this case, the attacker carries out the attack through numerous zombie machines in internet. This attack is much more difficult to counter, due to the attacker is not the one that accomplish the attack. Zombie machines are constantly added and removed from the botnet.

3.1.2 Prevention and Response

Explain host-end defense and network defense. We will use network defense.

3.1.3 TCP FLOODING DEFENSE PROPOSED (DRAFT!!!!)

Hi Aapo, this is an another idea that I have thought using anomaly detection. Basically, I will implement two different kinds of anomaly detection, one to control the amount of traffic and the other filtering the headers of the packets to check if they have something unusual (IP header length, TCP payload, TCP TTL...). Also there will be implemented a Rate-Limiting filter, using a "delay-queue". I will try to explain you the main idea by steps below:

If the amount of traffic is usual, the router will follow the flow traffic as usual to the server but the Controller will store the IPs address that complete the three handshake connection in a "working set"(for future connection if an attack is detected).

Once the anomaly detection system detects that the traffic is bigger than a boundary or is increasing rapidly, it will be launch the defense mechanism, installing a flow match in the flow table that all the TCP SYN packets that arrive to the switch, will be send them to the Controller.

The scope of the controller is as follow:

First, it will match the incoming IP address with the "working set", if so, two flows in either directions between the internal host and the remote host will be set in the flow table. If it does not match, the other anomaly detection system will filter the headers of the packet (There is a survey that I have attached in the email that explain this algorithm, but you can see it in the Figure 3.5). It might happen:

1. If everything is OK, the request connection will be enqueue in the delay-queue (to control the rate of the incoming connections). Every d seconds a new connection will be moved from the queue to the switch to forward it to the server. Whenever a positive connection reply, the IP address will be store in the "working set" and the new flow entries will be set in the flow table.

2. If some anomaly in the packet is detected, the IP address will be store in a "suspect IP addresses table". Here I am not sure what can I do yet, either drop the packet or the switch can send the SYN-ACK back, and if it response, a new flow entry will be set in the flow table.

I know that the packet header filtering will not find all the attacks, but at least a big part of them. The main idea is that the server will receive the traffic in a fixed rate and not whole the traffic that arrive to the network.

Algorithm for Packet Filtering System

```
for each packet arrival do
    check the IP Header Length
       if IP Header Length = 20 then
           choose the protocol = TCP
           check the payload packet
              if payload normal
                goto destination
              else distinguish the packet for analysis
                if TCP SYN Flood  then
                   report to administrator
                else analysis for other threats
               end if
             end if
            other protocol
            goto destination
     end if
end for
```

Figure 3.5: Packet Filtering Algorithm

Bibliography

[1] CC., C. Denial of service attack. `http://www.cert.org/tech_tips/denial_of_service.html`.

[2] Frenetic. Frenetic official webpage. http://frenetic-lang.org/overview.php.

[3] Inc., C. Defenses Against TCP SYN Flooding Attacks. http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_9-4/syn_flooding_attacks.html.

[4] OpenFlow. Open Flow standard. http://archive.openflow.org/.

[5] Peng, T., Leckie, C., and Ramamohanarao, K. Survey of network-based defense mechanisms countering the dos and ddos problems. *ACM Computing Surveys (CSUR) 39*, 1 (2007), 3.

[6] Specification, O. S. Version 1.4.0. *Open Networking Foundation* (2013).

Appendix A

First appendix

This is the first appendix. You could put some test images or verbose data in an appendix, if there is too much data to fit in the actual text nicely.

For now, the Aalto logo variants are shown in Figure A.1.

A! Aalto University
School of Science

(a) In English

A" Aalto-yliopisto
Perustieteiden
korkeakoulu

(b) Suomeksi

A" Aalto-universitetet
Högskolan för
teknikvetenskaper

(c) På svenska

Figure A.1: Aalto logo variants