OLUWAFIKAYO OYEWOLE

IDEAS/24/95560

Week 1: Reconnaissance, Information Gathering, and Scanning

INT302: Kali Linux Tools and System Security – Lab 4: Basic Port Scanning

Lab Overview

In this lab, you will perform basic and advanced port scanning techniques using nmap and nikto. You will gather the IP addresses of your OWASP Broken Web Applications Project VM and utilize these IPs for scanning to identify open ports, services running on those ports, potential vulnerabilities, and the operating system of the target.

Lab Objectives

By the end of this lab, you will:

1. Conduct a basic port scan using nmap.
2. Perform an aggressive scan to determine service versions and the operating system.
3. Utilize nmap for vulnerability scanning.
4. Use nikto to perform web server vulnerability scans.

Tools Used

- **Kali Linux**: A Linux distribution tailored for penetration testing.
- **nmap**: A versatile network scanning tool for discovering hosts and services on a computer network.
- **nikto**: A web server scanner that tests for dangerous files/programs, outdated server software, and other vulnerabilities.

Prerequisites

- Basic knowledge of Kali Linux and command-line operations.

- Access to the OWASP Broken Web Applications Project VM.
- nmap and nikto installed in your Kali Linux environment (they usually come pre-installed).

Lab Steps

Step 1: Gather the IP Address of Your OWASP VM

Instructions:

1. Start your OWASP Broken Web Applications Project VM.
2. Open a terminal and run the following command to find the IP address:

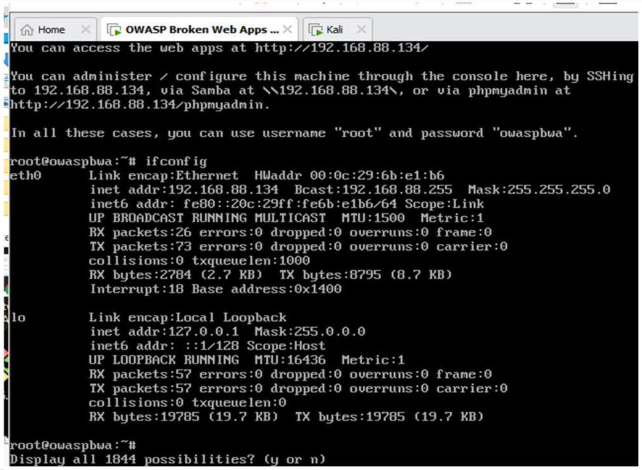
Command Syntax:

```
ifconfig
```

3. Look for the inet address under your active network interface (usually eth0 or ens33).

Record the IP Address:

- **OW**



```
You can access the web apps at http://192.168.88.134/

You can administer / configure this machine through the console here, by SSHing
to 192.168.88.134, via Samba at \\192.168.88.134\, or via phpmyadmin at
http://192.168.88.134/phpmyadmin.

In all these cases, you can use username "root" and password "owaspbwa".

root@owaspbwa:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:6b:e1:b6
          inet addr:192.168.88.134  Bcast:192.168.88.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe6b:e1b6/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:26 errors:0 dropped:0 overruns:0 frame:0
          TX packets:73 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2784 (2.7 KB)  TX bytes:8795 (8.7 KB)
          Interrupt:18 Base address:0x1400

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:57 errors:0 dropped:0 overruns:0 frame:0
          TX packets:57 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:19785 (19.7 KB)  TX bytes:19785 (19.7 KB)

root@owaspbwa:~#
Display all 1844 possibilities? (y or n)
```

ASP VM IP Address: ___________

Step 2: Basic Port Scanning with nmap

Now that you have the IP address of your OWASP VM, use nmap to discover open ports.

Instructions:

1. Open your **Terminal** in Kali Linux.
2. Use the following command to scan for open ports on your OWASP VM.

Command Syntax:

nmap <IP address>

Example: nmap 192.168.56.101 # Replace with the actual IP of your

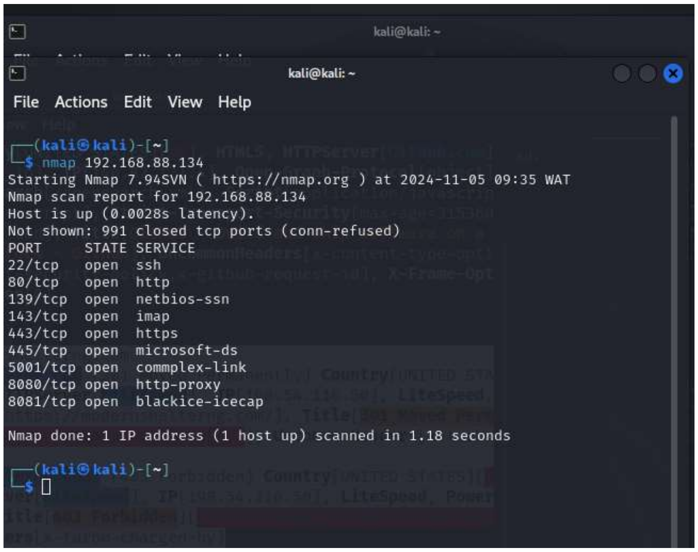
OWASP VM

Expected Output:
The output will display a list of open ports on the specified IP address.

Exercise 1:

Perform a basic port scan on your OWASP VM IP address and record your findings:

- **Open Ports**:



```
┌──(kali㉿kali)-[~]
└─$ nmap 192.168.88.134
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-05 09:35 WAT
Nmap scan report for 192.168.88.134
Host is up (0.0028s latency).
Not shown: 991 closed tcp ports (conn-refused)
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
139/tcp  open  netbios-ssn
143/tcp  open  imap
443/tcp  open  https
445/tcp  open  microsoft-ds
5001/tcp open  commplex-link
8080/tcp open  http-proxy
8081/tcp open  blackice-icecap

Nmap done: 1 IP address (1 host up) scanned in 1.18 seconds

┌──(kali㉿kali)-[~]
└─$ 
```

○

Step 3: Aggressive Scanning with nmap

Aggressive scanning with nmap can reveal service versions and the operating system running on open ports.

Instructions:

1. Use the following command to perform an aggressive scan.

Command Syntax:

nmap -sV -O <IP address>

Example: nmap -sV -O 192.168.56.101 # Replace with the actual IP of your

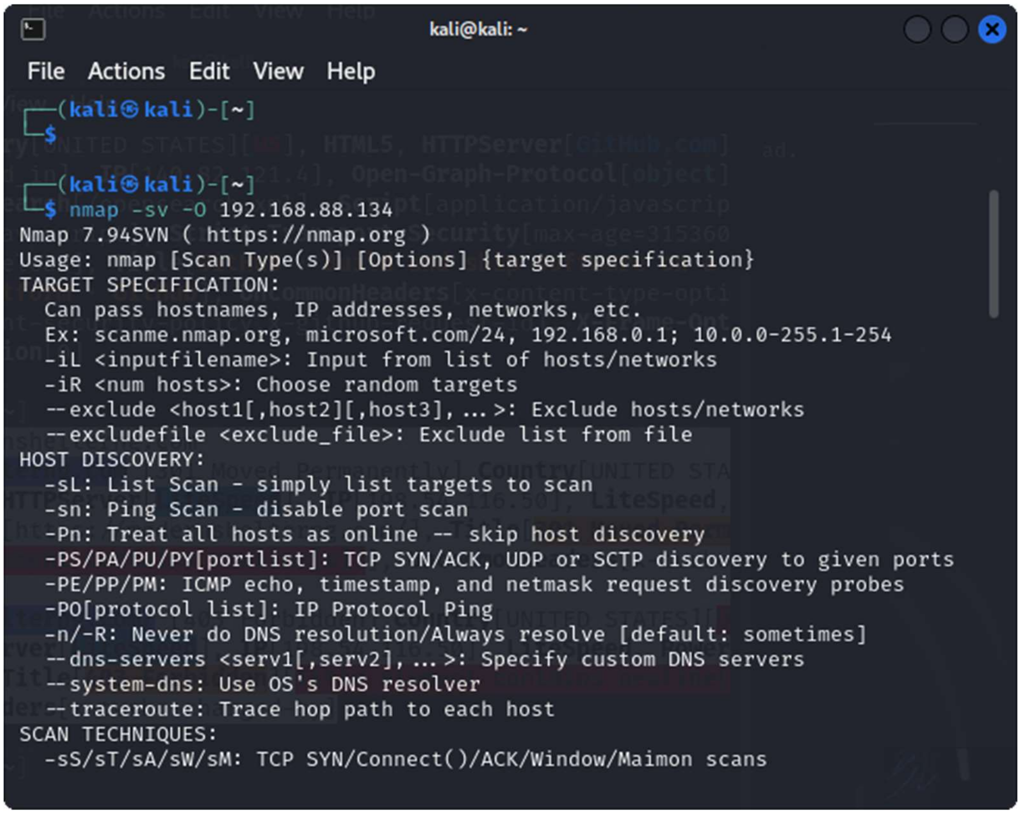
OWASP VM

Expected Output:
The output will display open ports, service versions, and operating system details.

Exercise 2:

Perform an aggressive scan on your OWASP VM IP address and record your findings:

- **Service Versions**:



```
┌──(kali㉿kali)-[~]
└─$ 

┌──(kali㉿kali)-[~]
└─$ nmap -sv -O 192.168.88.134
Nmap 7.94SVN ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <serv1[,serv2],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver
  --traceroute: Trace hop path to each host
SCAN TECHNIQUES:
  -sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
```

  -

- **Operating System**:
  -

Step 4: Vulnerability Scanning with nmap

nmap allows you to run vulnerability scans against the target

system.

Instructions:

1. Use the following command to perform a vulnerability scan.

Command Syntax:

nmap --script vuln <IP address>

Example: nmap --script vuln 192.168.56.101 # Replace with the actual IP of your

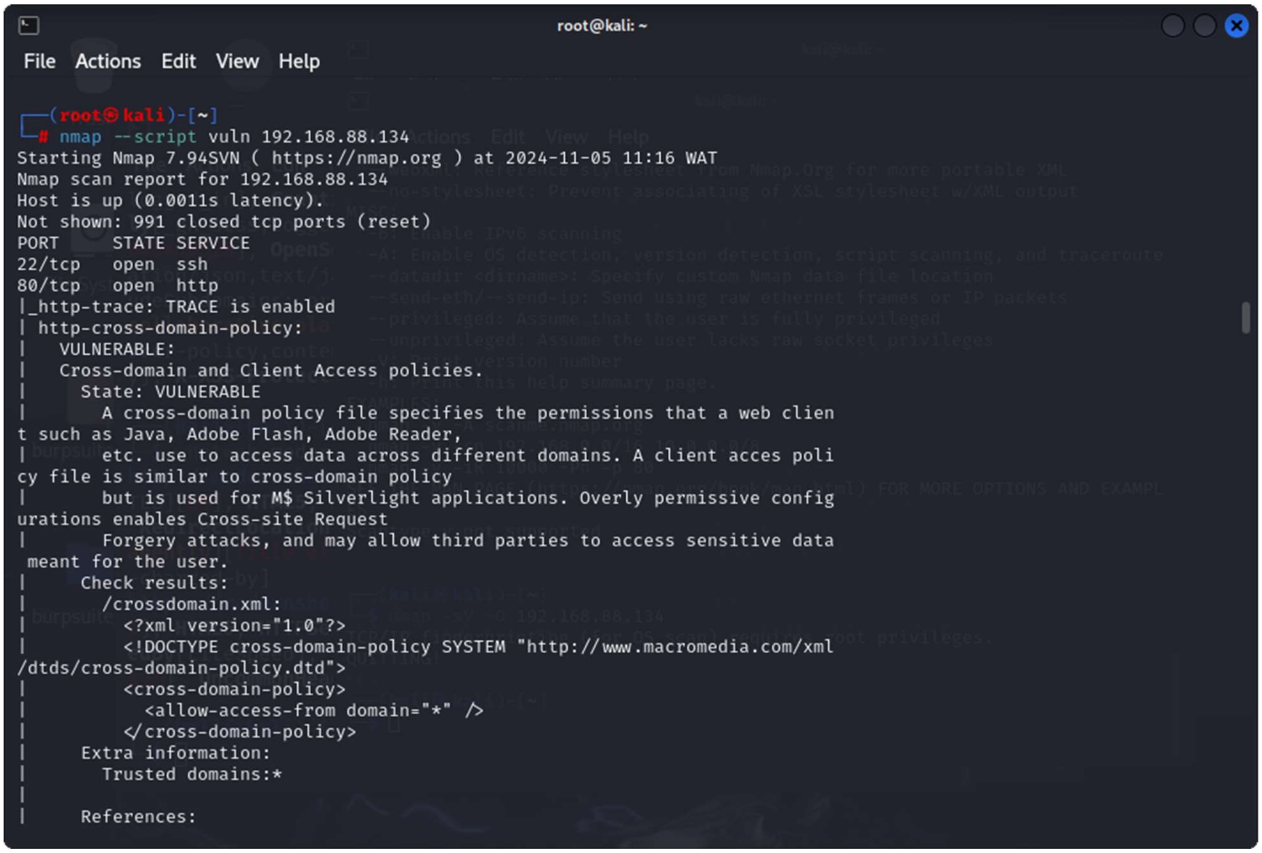
OWASP VM

Expected Output:
The output will display any vulnerabilities found on the target system.

Exercise 3:

Conduct a vulnerability scan on your OWASP VM IP address and record your findings:

- **Vulnerabilities**:



```
┌──(root㉿kali)-[~]
└─# nmap --script vuln 192.168.88.134
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-05 11:16 WAT
Nmap scan report for 192.168.88.134
Host is up (0.0011s latency).
Not shown: 991 closed tcp ports (reset)
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
|_http-trace: TRACE is enabled
| http-cross-domain-policy:
|   VULNERABLE:
|   Cross-domain and Client Access policies.
|     State: VULNERABLE
|       A cross-domain policy file specifies the permissions that a web clien
t such as Java, Adobe Flash, Adobe Reader,
|       etc. use to access data across different domains. A client acces poli
cy file is similar to cross-domain policy
|       but is used for M$ Silverlight applications. Overly permissive config
urations enables Cross-site Request
|       Forgery attacks, and may allow third parties to access sensitive data
 meant for the user.
|     Check results:
|       /crossdomain.xml:
|         <?xml version="1.0"?>
|         <!DOCTYPE cross-domain-policy SYSTEM "http://www.macromedia.com/xml
/dtds/cross-domain-policy.dtd">
|         <cross-domain-policy>
|           <allow-access-from domain="*" />
|         </cross-domain-policy>
|     Extra information:
|       Trusted domains:*
|
|     References:
```

○

Step 5: Web Vulnerability Scanning with nikto

nikto is a comprehensive web server scanner that checks for various

vulnerabilities.

Instructions:

1. Use the following command to perform a web server vulnerability scan.

Command Syntax:

nikto -h <target URL>

Example: nikto -h http://192.168.56.101 # Replace with the actual URL of your

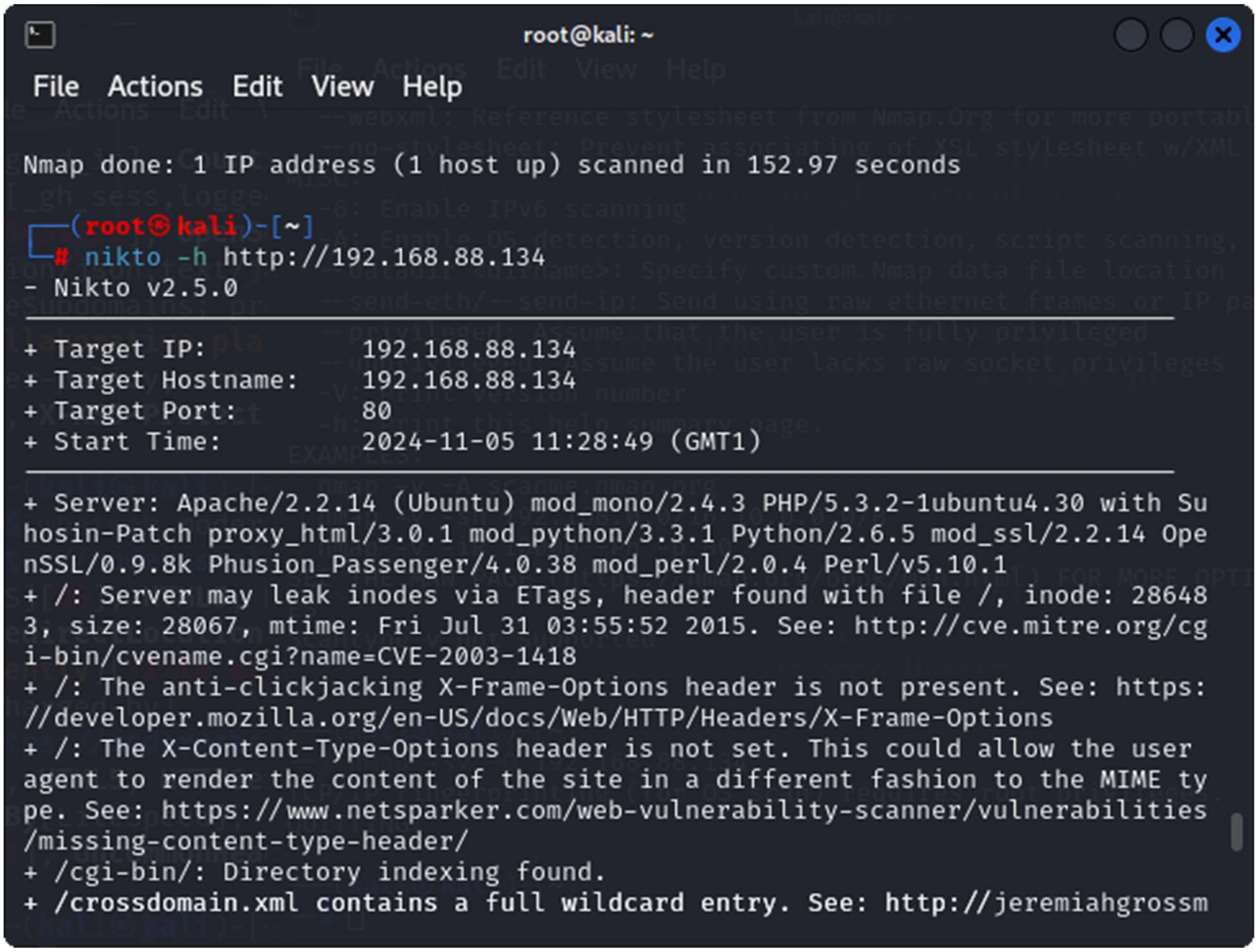
OWASP VM

Expected Output:
The output will display any vulnerabilities found on the web server.

Exercise 4:

Perform a vulnerability scan on your OWASP VM and record your findings:

- **Vulnerabilities Found**:



```
Nmap done: 1 IP address (1 host up) scanned in 152.97 seconds

┌──(root㉿kali)-[~]
└─# nikto -h http://192.168.88.134
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          192.168.88.134
+ Target Hostname:    192.168.88.134
+ Target Port:        80
+ Start Time:         2024-11-05 11:28:49 (GMT1)
---------------------------------------------------------------------------
+ Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Su
hosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 Ope
nSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
+ /: Server may leak inodes via ETags, header found with file /, inode: 28648
3, size: 28067, mtime: Fri Jul 31 03:55:52 2015. See: http://cve.mitre.org/cg
i-bin/cvename.cgi?name=CVE-2003-1418
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https:
//developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user
agent to render the content of the site in a different fashion to the MIME ty
pe. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities
/missing-content-type-header/
+ /cgi-bin/: Directory indexing found.
+ /crossdomain.xml contains a full wildcard entry. See: http://jeremiahgrossm
```

○

Submission Instructions

Submit your results from all exercises, including:

- Detected open ports from the basic scan.
- Service versions and operating system from the aggressive scan.
- Any vulnerabilities discovered using nmap.
- Vulnerabilities found using nikto.

Conclusion

In this lab, you explored techniques for basic port scanning and vulnerability assessment using nmap and nikto. These skills are essential for identifying potential attack vectors and securing network infrastructures.

INT302: Kali Linux Tools and System Security – Lab 5: Wireshark

Lab Overview

Wireshark is a powerful, open-source network protocol analyzer used for network troubleshooting, analysis, and software development. In this lab, you will learn to capture and analyze network traffic using Wireshark and its command-line tool, tshark. Understanding network packets and their structure is crucial for identifying vulnerabilities and securing networks.

Lab Objectives

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

1. Launch and navigate the Wireshark GUI effectively.
2. Capture live network traffic using both Wireshark and tshark.
3. Apply filters to isolate and analyze specific packets.
4. Understand packet details, including protocols and their flags.
5. Utilize advanced features such as statistics and graphing.
6. Recognize potential security issues in captured traffic.

Tools Used

- **Wireshark**: A graphical network protocol analyzer.
- **tshark**: The terminal-based version of Wireshark.

Prerequisites

- Basic knowledge of networking concepts.
- Installed Wireshark on your Kali Linux environment.

Lab Steps

Step 1: Launching Wireshark

1. Open a terminal in Kali Linux.
2. Launch Wireshark by typing the following command:

Command:

```
wireshark
```

3. Familiarize yourself with the interface, noting the main components:
   - **Capture Interfaces**: Where you can select which network interface to capture from. ○ **Packet List Pane**: Displays a list of captured packets. ○ **Packet Details Pane**: Shows detailed information about the selected packet.
   - **Packet Bytes Pane**: Displays the raw data of the selected packet.
   - **Statistics**: Provides information about protocols, conversations, and endpoints.

Exercise 1:

- Explore the Wireshark GUI. Identify and list the main components you see, including where to find the **Statistics** menu.

Answer:

Menu Bar:

1) Main Toolbar
2) Filter Toolbar
3) Packet List Pane
4) Packet Details Pane

5) Packet Bytes Pane
6) Status Bar

The Statistics menu opens analysis tools for your network data.

Step 2: Capturing Network Traffic

Using the Wireshark GUI:

1. Select an interface to capture traffic (e.g., wlan0 for wireless or eth0 for wired).
2. Click the **Start Capturing Packets** button (the shark fin icon).
3. Allow the capture to run for a few minutes while you browse the internet or perform other network activities.

Using tshark:

1. Open a new terminal window.
2. Use the following command to capture packets on a specific interface:

Command Syntax:

```
tshark -i <interface>
```

Example: tshark -i wlan0 # Replace with your

actual interface

Exercise 2:

- Capture network traffic using both Wireshark and tshark. Compare the two methods and note any differences in the user experience.

Answer:

1) Ease of Use: Wireshark is more accessible for beginners due to its graphical user interface and ability to visualize in real time; tshark is dependent upon understanding how to operate a command line interface.

2) Filtering: Filtering can be done much easier in Wireshark using the visual filter bar, whereas in tshark, one needs to denote filters in commands themselves.
System Resources: Tshark runs lighter and is better for batch processing; Wireshark uses more resources, and the interface is much more intuitive.

Step 3: Analyzing Captured Packets

1. Stop the packet capture in Wireshark by clicking the **Stop Capturing Packets** button (the red square icon).
2. Analyze the captured packets in the Packet List Pane.
3. Apply display filters to isolate specific types of traffic. Common filters include:

    o Filter for HTTP traffic: http o Filter for DNS

    traffic: dns o Filter for specific IP addresses:

    ip.addr == <target IP> o Filter for TCP packets: tcp

Exercise 3:

- Use filters to analyze different types of traffic. Record the following:
    - Number of HTTP packets captured: _____22_____
    - Number of DNS packets captured: _____212_____
    - Specific IP addresses you identified in the traffic: ____192.168.88.2______

Step 4: Understanding Packet Details

1. Click on a packet in the Packet List Pane to view its details in the Packet Details Pane.
2. Expand different protocol layers to understand the encapsulation and the data contained within each packet.

Key Areas to Focus On:

- Source and Destination IP Addresses
- Protocol Types (TCP, UDP, ICMP, etc.)
- TCP Flags (SYN, ACK, FIN, etc.)
- Application Layer Protocols (HTTP, DNS, etc.)

Exercise 4:

- Select a packet and list the following information:
    - Source IP:

      ____192.168.88.137______
    - Destination IP:

      ____34.117.188.166______

- Protocol: ____TCP______
- Any TCP Flags observed:

  ____FIN,ACK______

Step 5: Advanced Packet Analysis Techniques

1. **Follow TCP Stream**:
   - Right-click on any TCP packet and select **Follow** > **TCP Stream** to see the entire conversation.
   - This feature is useful for understanding the context of the traffic.

Exercise 5:

- Follow a TCP stream for a specific session and summarize the data exchanged between the client and server.

2. **Protocol Hierarchy**:
   - Navigate to **Statistics** > **Protocol Hierarchy** to see a breakdown of captured protocols.
   - This will help you identify which protocols are most common in your capture.

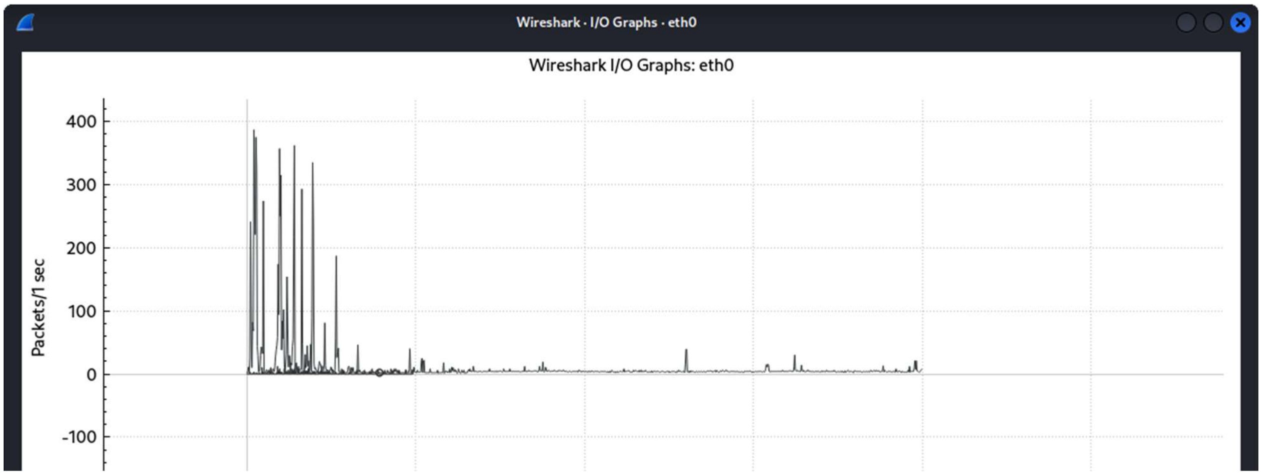
Exercise 6:

- Take a screenshot of the Protocol Hierarchy and analyze the data. Which protocol is most prevalent in your capture? __________

3. **IO Graphs**:
   - Access **Statistics** > **IO Graphs** to visualize traffic over time.
   - This can help identify spikes in traffic, indicating potential issues or security events.

Exercise 7:

- Create an IO Graph showing TCP traffic. Describe any noticeable patterns you observe: ___________



Step 6: Exporting Captured Data

1. Save your captured packets for further analysis or reporting.
   - Go to **File** > **Save As** and choose a file format (e.g., .pcap).

Exercise 8:

- Save your capture file and describe a scenario where you would need to review this data later. What specific findings do you hope to extract?

Step 7: Practical Applications of Wireshark

1. **Detecting Network Issues**:
   - Use Wireshark to analyze a failing network connection or slow performance.
   - Look for excessive retransmissions or packet loss indicators.

Exercise 9:

- Describe a real-world scenario where you would use Wireshark to troubleshoot a network issue. What specific symptoms would you investigate? ___________

Answer:

One realistic scenario where Wireshark can be put into use for troubleshooting a problem in a network involves any case of slow internet performance on a corporate network. Let's say users have been complaining about the slow pace of the Internet or frequent disconnections from the Internet throughout the office.

Exercise 10:

- Identify at least two potential security threats in your captured traffic. What indicators led you to suspect these activities? __________

INT302: Kali Linux Tools and System Security – Lab 6: Advanced Packet Analysis Techniques

Exercise 1:

- Describe the purpose of the SYN and ACK flags in the TCP handshake. How do these flags indicate the status of a connection? __________

Answer:

The SYN and ACK flags in TCP handshakes, in brief, create, synchronize, and acknowledge the readiness of communicating sides. A series of packets including SYN, SYN-ACK, and ACK notify that a connection is established, and data can securely and reliably stream between devices.

Exercise 2:

- Choose an HTTP packet and summarize its request method, status code, and any notable headers. What can you infer about the transaction? __________

Answer:

This is an example of an HTTP packet with a regular page request-a client pulls a webpage or any other similar resource from the server. The GET method with a status of 200 OK would indicate that the resource was located and returned successfully. Headers like User-Agent and Accept indicate that this is likely a web browser client expecting content that it can render, like HTML.

From which, we might have concluded that the transaction had to do with routine browsing behavior: where the client asks for some webpage, and the server returns the expected data without any issues.

Exercise 3:

- Identify a DNS query and its corresponding response. What information does the response provide, and how is it structured? __________

Step 2: Creating Custom Filters

Exercise 4:

- Create a custom filter that captures only TCP traffic from your machine to a specific target IP. Document the filter syntax and the packets captured. __________

Exercise 5:

- Write a filter that captures traffic on a specific port (e.g., HTTP port 80) and analyze the results. What packets were captured? __________

Exercise 6:

- Analyze your capture for any anomalies or indicators of potential vulnerabilities. Document your findings and suggest possible remediation steps. __________

Exercise 7:

- Capture HTTPS traffic and identify the initial handshake packets. What information is exchanged during this handshake, and how does it contribute to security? __________

Step 4: Practical Applications and Reporting

Exercise 8:

- Prepare a brief report summarizing your findings during the assessment. Include potential risks and recommended actions. __________

Exercise 9:

- Create a capture report that includes your objectives, methods, key findings, and any recommendations for improving network security. __________