Name: Kunal Goyal

Reg No.: 22BCY10053

Department: CSE (Spl. in Cyber Security & Digital Forensics)

Course: Cyber Security Analyst

Project Title: Who's Watching? (Simulating Man-in-the-Middle Attacks Over Public Wi-Fi)

Executive Summary:

This project aimed to demonstrate the vulnerability of data transmitted over unsecured networks by simulating a Man-in-the-Middle (MITM) attack. Using tools like Kali Linux, Ettercap, arpspoof, and Wireshark, the attacker intercepted and analyzed HTTP traffic between a victim and a server on a compromised public Wi-Fi network. The simulation highlighted how easily sensitive information, such as login credentials, can be captured without encryption. Key results showed that using HTTPS and VPNs effectively prevented data theft, while ARP spoofing and packet sniffing were successful in exposing cleartext information. The project emphasized the importance of encryption, network security, and user awareness in safeguarding against such attacks.

Project Overview:

Problem Statement:

Public and unsecured Wi-Fi networks pose a significant security risk as attackers can intercept sensitive information, such as login credentials, through Man-in-the-Middle (MITM) attacks. Users often connect to these networks without considering the security implications, leading to data breaches. This project aimed to simulate such an attack to highlight vulnerabilities in unsecured communication channels and raise awareness of effective security practices.

Objectives:

- **Objective 1**: Demonstrate how an attacker can intercept sensitive data, such as usernames and passwords, on an unsecured Wi-Fi network using MITM techniques.

- **Objective 2**: Show the impact of using insecure communication protocols (like HTTP) and emphasize the importance of encrypting traffic (HTTPS).
- **Objective 3**: Propose practical security measures and defense strategies, such as using VPNs and HTTPS, to mitigate the risks associated with MITM attacks.

Scope of Work:

- **Included**:
  - Simulating an MITM attack using tools like Ettercap, arpspoof, and Wireshark.
  - Capturing unencrypted traffic from a victim machine over HTTP.
  - Analyzing the captured data for sensitive information such as login credentials.
  - Demonstrating the effectiveness of HTTPS and VPNs in securing traffic.
  - Providing security recommendations based on findings.
- **Excluded**:
  - Full-scale network security implementation (outside the scope of the simulation).
  - Encryption or SSL stripping techniques (focused on interception and analysis only).
  - Long-term monitoring or detection of MITM attacks in large-scale environments.

Tools & Lab Setup:

Primary Tools Used:

- **Kali Linux: The attacker machine, used for launching MITM attacks and running sniffing tools.**
- **Wireshark: A network packet analyzer used to capture and analyze the intercepted traffic.**
- **Ettercap: A tool used to perform ARP poisoning and launch the MITM attack.**
- **arpspoof: A tool used for ARP spoofing to redirect the traffic through the attacker's machine.**

- **Python HTTP Server: Used to simulate an unsecured HTTP login page for the victim to interact with.**

Environment Details:

- **Virtual Machine Setup:**
    - **VM1: Kali Linux (Attacker machine)**
    - **VM2: (Victim machine)**
    - **Hosted on VMware Workstation.**
- **Network Mode: Bridged Adapter / Host-only (ensuring both machines are on the same network for successful traffic interception)**

Tool Configuration & Commands:

● Wireshark

- **Purpose: To capture and analyze network traffic.**
- **Configuration:**
    - **Interface selected: eth0 or ens33 (depending on VM)**
    - **Filter used to view login attempts:**

```
http.request.method == "POST"
```

● Ettercap (GUI Mode)

- **Purpose: To perform ARP poisoning and sniff traffic between victim and gateway.**
- **Commands:**
    - **Launch Ettercap:**

```
sudo ettercap -G
```

- **Ettercap GUI Steps:**
    - **Sniff > Unified Sniffing > eth0**
    - **Hosts > Scan for Hosts**
    - **Add victim to Target 1**
    - **Add gateway to Target 2**
    - **Mitm > ARP poisoning > Sniff remote connections**
    - **Start > Start sniffing**

● arpspoof (Optional - CLI Alternative)

- **Purpose: Command-line tool for ARP spoofing.**
- **Commands:**

```
sudo arpspoof -i eth0 -t <Victim_IP> <Gateway_IP>
sudo arpspoof -i eth0 -t <Gateway_IP> <Victim_IP>
```

● Enable IP Forwarding

- **Purpose: Allow traffic to pass through attacker machine (Kali).**
- **Command:**

```
echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward
```

● Simple HTTP Server (Simulated Login Page)

- **Purpose: Host a dummy HTTP login form for victim to interact with.**
- **Commands:**

```
mkdir /tmp/web && cd /tmp/web

echo '<form method="POST"><input name="username"><input name="password"
type="password"><input type="submit"></form>' > index.html

sudo python3 -m http.server 80
```

Implementation & Execution Summary:

Key Steps Performed:

1. Lab Setup and Configuration

- **Created two virtual machines: Kali Linux (attacker) and Windows 10 (victim).**
- **Configured both VMs in the same network (Bridged/Host-only) and verified connectivity.**

2. Tool Installation and Configuration

- **Installed essential tools: Wireshark, Ettercap, arpspoof, and Python HTTP server.**
- **Enabled IP forwarding on Kali to allow traffic routing.**

3. Simulated HTTP Login Page

- **Hosted a simple login form using Python's HTTP server on Kali Linux.**
- **Victim (Windows 10) accessed the page to simulate login over HTTP.**

4. Started Packet Capture

- **Launched Wireshark on Kali to monitor and capture network packets.**
- **Applied appropriate filters to focus on HTTP POST traffic.**

5. Captured Credentials

- **Victim submitted credentials via the HTTP form.**
- **Captured cleartext username and password in Wireshark packet capture.**

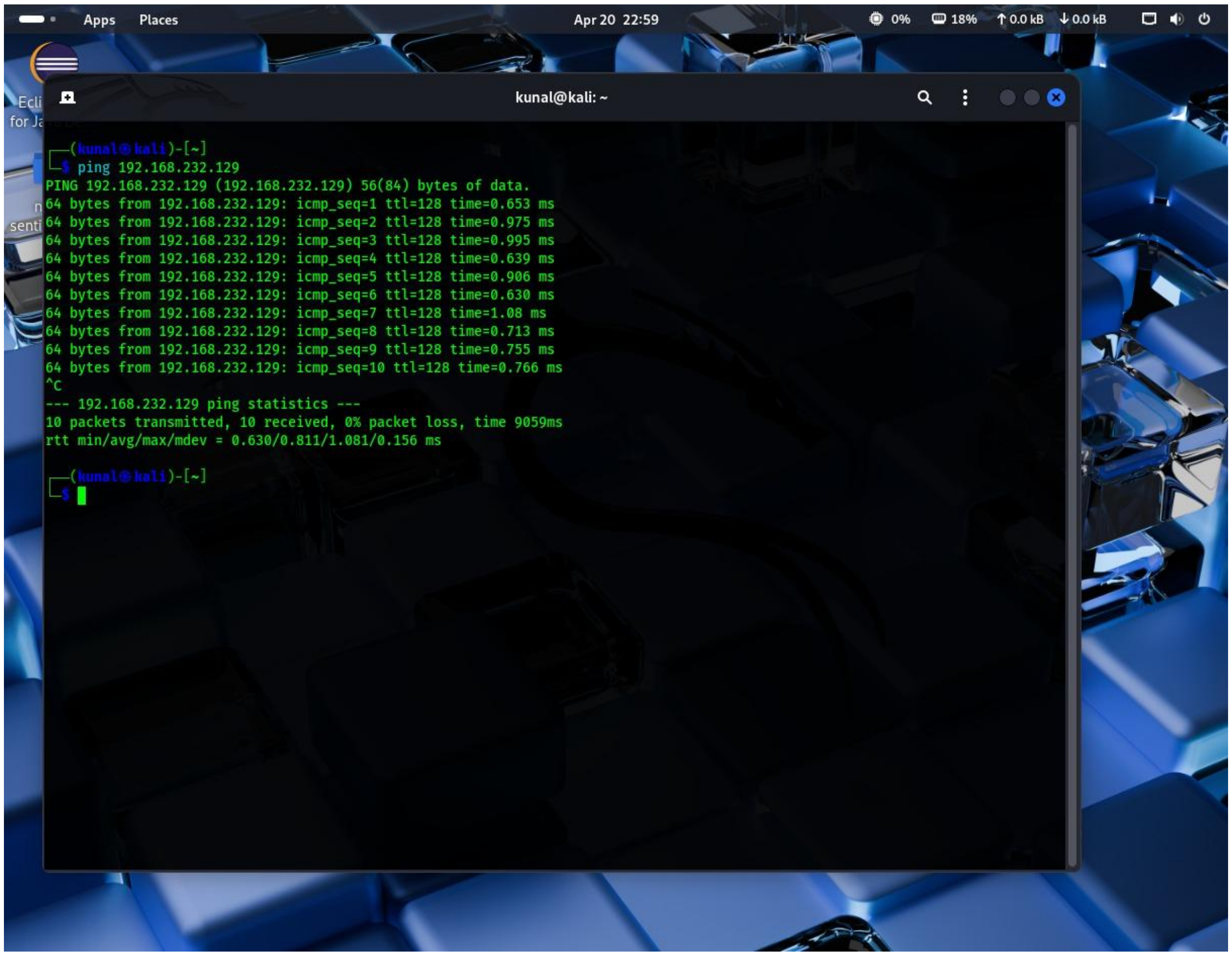
6. Analyzed and Saved Results

- **Inspected packets in Wireshark to extract sensitive data.**

- **Saved .pcap file and relevant screenshots for documentation.**

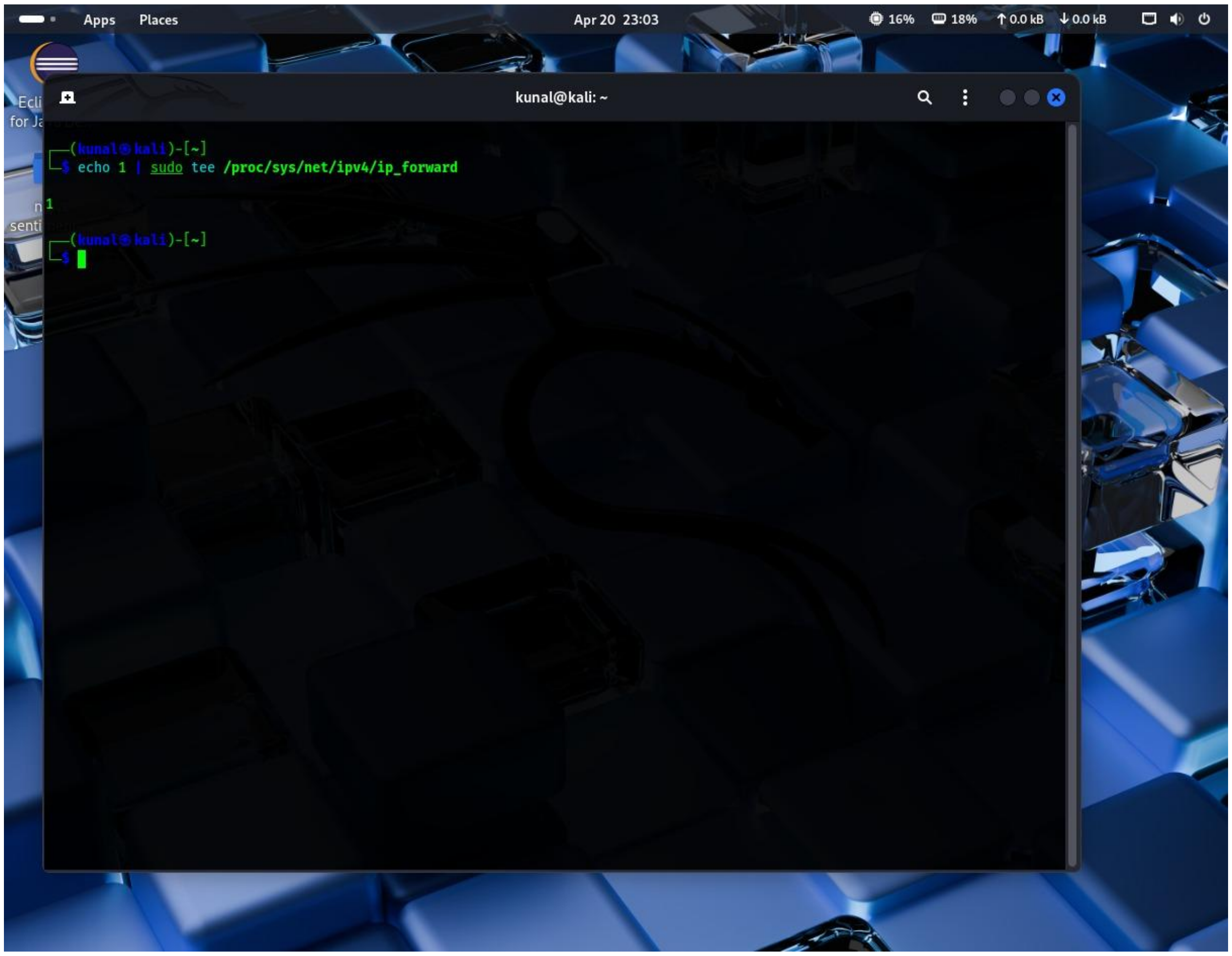
Screenshots / Evidence:

ScreenShot-1: Checking Connectivity between Machines:



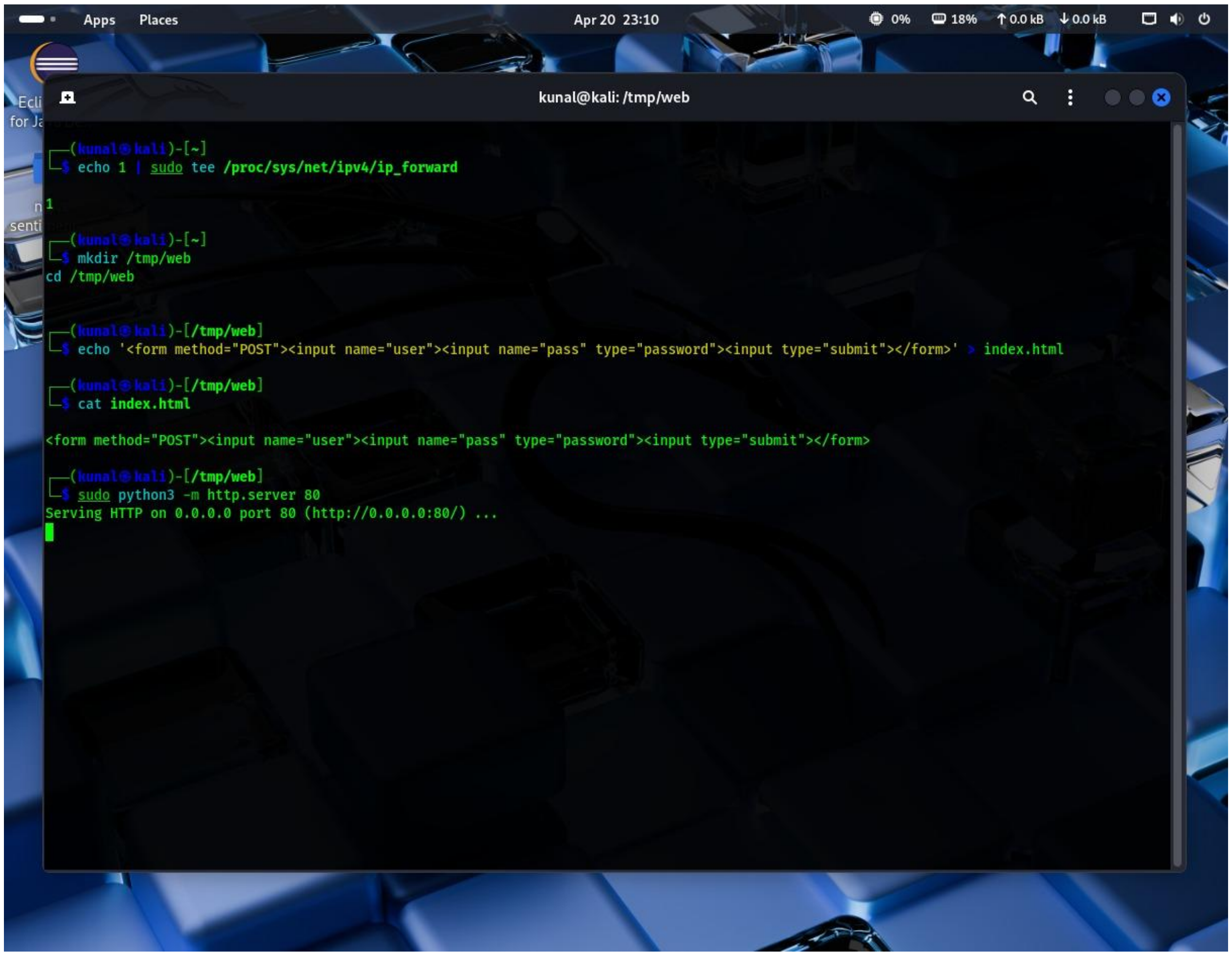
```
┌──(kunal㉿kali)-[~]
└─$ ping 192.168.232.129
PING 192.168.232.129 (192.168.232.129) 56(84) bytes of data.
64 bytes from 192.168.232.129: icmp_seq=1 ttl=128 time=0.653 ms
64 bytes from 192.168.232.129: icmp_seq=2 ttl=128 time=0.975 ms
64 bytes from 192.168.232.129: icmp_seq=3 ttl=128 time=0.995 ms
64 bytes from 192.168.232.129: icmp_seq=4 ttl=128 time=0.639 ms
64 bytes from 192.168.232.129: icmp_seq=5 ttl=128 time=0.906 ms
64 bytes from 192.168.232.129: icmp_seq=6 ttl=128 time=0.630 ms
64 bytes from 192.168.232.129: icmp_seq=7 ttl=128 time=1.08 ms
64 bytes from 192.168.232.129: icmp_seq=8 ttl=128 time=0.713 ms
64 bytes from 192.168.232.129: icmp_seq=9 ttl=128 time=0.755 ms
64 bytes from 192.168.232.129: icmp_seq=10 ttl=128 time=0.766 ms
^C
--- 192.168.232.129 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9059ms
rtt min/avg/max/mdev = 0.630/0.811/1.081/0.156 ms

┌──(kunal㉿kali)-[~]
└─$ 
```

ScreenShot-2: Enable IP Forwarding:



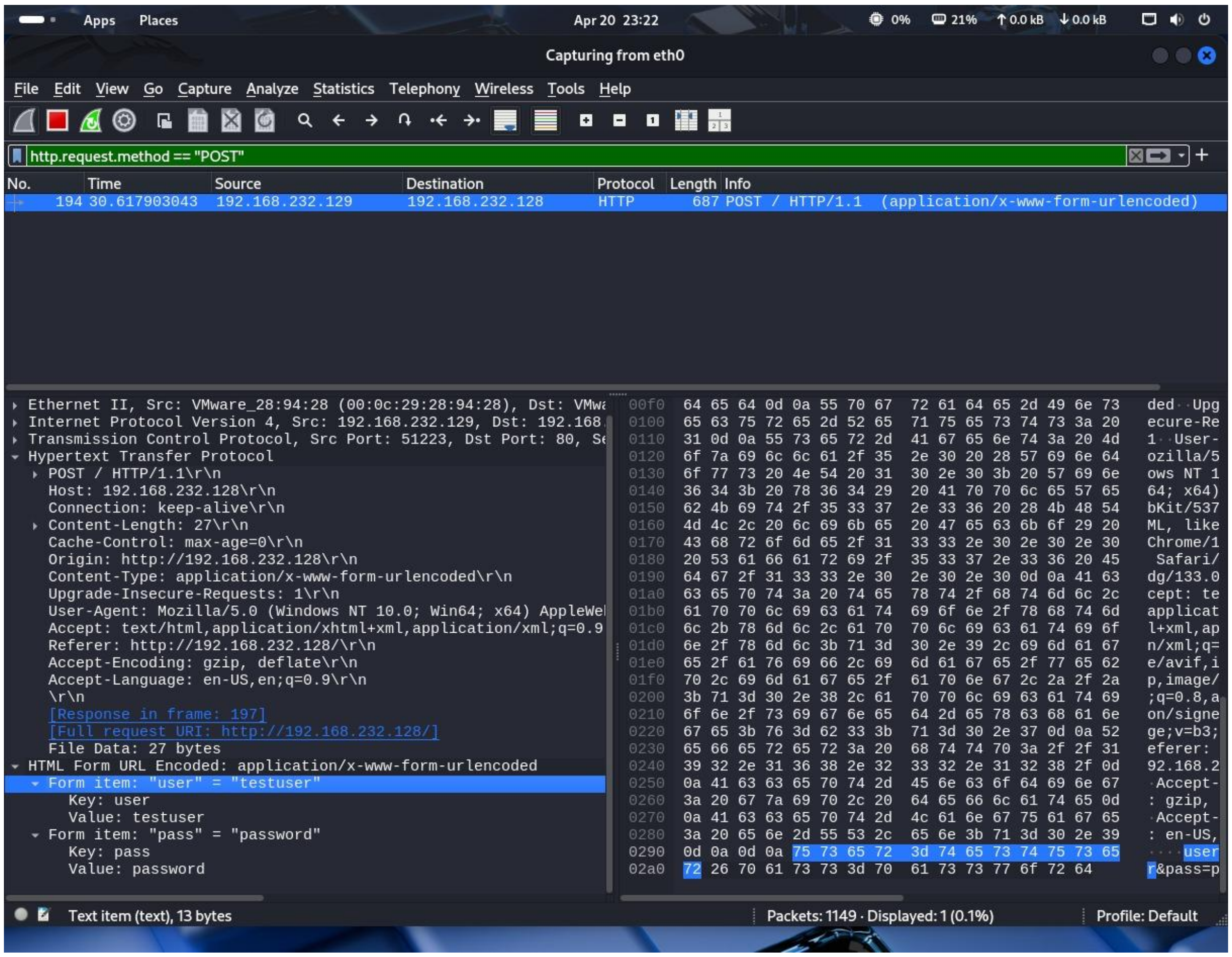
```
┌──(kunal㉿kali)-[~]
└─$ echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward

1

┌──(kunal㉿kali)-[~]
└─$ 
```

ScreenShot-3: Creating HTTP login page:



```
┌──(kunal㉿kali)-[~]
└─$ echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward

1

┌──(kunal㉿kali)-[~]
└─$ mkdir /tmp/web
cd /tmp/web


┌──(kunal㉿kali)-[/tmp/web]
└─$ echo '<form method="POST"><input name="user"><input name="pass" type="password"><input type="submit"></form>' > index.html

┌──(kunal㉿kali)-[/tmp/web]
└─$ cat index.html

<form method="POST"><input name="user"><input name="pass" type="password"><input type="submit"></form>

┌──(kunal㉿kali)-[/tmp/web]
└─$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

ScreenShot-4: Capturing Packet via WireShark:


Apps
Places
Apr 20 23:22
0%
21%
0.0 kB
0.0 kB
Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

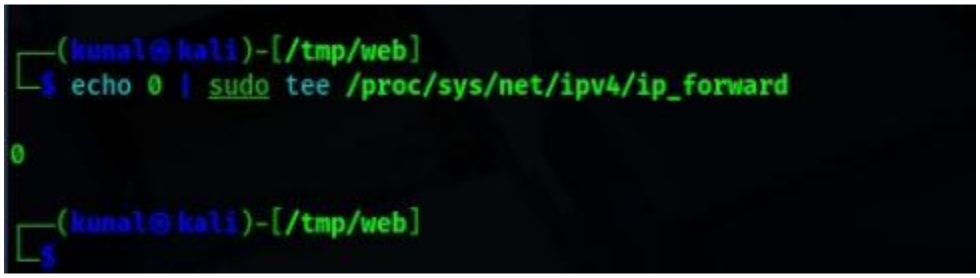
```
http.request.method == "POST"

No.   Time          Source           Destination      Protocol Length Info
  194 30.617903043  192.168.232.129  192.168.232.128  HTTP        687 POST / HTTP/1.1  (application/x-www-form-urlencoded)
```

```
› Ethernet II, Src: VMware_28:94:28 (00:0c:29:28:94:28), Dst: VMwa
› Internet Protocol Version 4, Src: 192.168.232.129, Dst: 192.168
› Transmission Control Protocol, Src Port: 51223, Dst Port: 80, Se
⌄ Hypertext Transfer Protocol
  › POST / HTTP/1.1\r\n
    Host: 192.168.232.128\r\n
    Connection: keep-alive\r\n
  › Content-Length: 27\r\n
    Cache-Control: max-age=0\r\n
    Origin: http://192.168.232.128\r\n
    Content-Type: application/x-www-form-urlencoded\r\n
    Upgrade-Insecure-Requests: 1\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWel
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9
    Referer: http://192.168.232.128/\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: en-US,en;q=0.9\r\n
    \r\n
    [Response in frame: 197]
    [Full request URI: http://192.168.232.128/]
    File Data: 27 bytes
⌄ HTML Form URL Encoded: application/x-www-form-urlencoded
  ⌄ Form item: "user" = "testuser"
      Key: user
      Value: testuser
  ⌄ Form item: "pass" = "password"
      Key: pass
      Value: password
```

```
00f0  64 65 64 0d 0a 55 70 67  72 61 64 65 2d 49 6e 73   ded··Upg
0100  65 63 75 72 65 2d 52 65  71 75 65 73 74 73 3a 20   ecure-Re
0110  31 0d 0a 55 73 65 72 2d  41 67 65 6e 74 3a 20 4d   1··User-
0120  6f 7a 69 6c 6c 61 2f 35  2e 30 20 28 57 69 6e 64   ozilla/5
0130  6f 77 73 20 4e 54 20 31  30 2e 30 3b 20 57 69 6e   ows NT 1
0140  36 34 3b 20 78 36 34 29  20 41 70 70 6c 65 57 65   64; x64)
0150  62 4b 69 74 2f 35 33 37  2e 33 36 20 28 4b 48 54   bKit/537
0160  4d 4c 2c 20 6c 69 6b 65  20 47 65 63 6b 6f 29 20   ML, like
0170  43 68 72 6f 6d 65 2f 31  33 33 2e 30 2e 30 2e 30   Chrome/1
0180  20 53 61 66 61 72 69 2f  35 33 37 2e 33 36 20 45    Safari/
0190  64 67 2f 31 33 33 2e 30  2e 30 2e 30 0d 0a 41 63   dg/133.0
01a0  63 65 70 74 3a 20 74 65  78 74 2f 68 74 6d 6c 2c   cept: te
01b0  61 70 70 6c 69 63 61 74  69 6f 6e 2f 78 68 74 6d   applicat
01c0  6c 2b 78 6d 6c 2c 61 70  70 6c 69 63 61 74 69 6f   l+xml,ap
01d0  6e 2f 78 6d 6c 3b 71 3d  30 2e 39 2c 69 6d 61 67   n/xml;q=
01e0  65 2f 61 76 69 66 2c 69  6d 61 67 65 2f 77 65 62   e/avif,i
01f0  70 2c 69 6d 61 67 65 2f  61 70 6e 67 2c 2a 2f 2a   p,image/
0200  3b 71 3d 30 2e 38 2c 61  70 70 6c 69 63 61 74 69   ;q=0.8,a
0210  6f 6e 2f 73 69 67 6e 65  64 2d 65 78 63 68 61 6e   on/signe
0220  67 65 3b 76 3d 62 33 3b  71 3d 30 2e 37 0d 0a 52   ge;v=b3;
0230  65 66 65 72 65 72 3a 20  68 74 74 70 3a 2f 2f 31   eferer: 
0240  39 32 2e 31 36 38 2e 32  33 32 2e 31 32 38 2f 0d   92.168.2
0250  0a 41 63 63 65 70 74 2d  45 6e 63 6f 64 69 6e 67   ·Accept-
0260  3a 20 67 7a 69 70 2c 20  64 65 66 6c 61 74 65 0d   : gzip, 
0270  0a 41 63 63 65 70 74 2d  4c 61 6e 67 75 61 67 65   ·Accept-
0280  3a 20 65 6e 2d 55 53 2c  65 6e 3b 71 3d 30 2e 39   : en-US,
0290  0d 0a 0d 0a 75 73 65 72  3d 74 65 73 74 75 73 65   ····user
02a0  72 26 70 61 73 73 3d 70  61 73 73 77 6f 72 64      r&pass=p
```

Text item (text), 13 bytes   Packets: 1149 · Displayed: 1 (0.1%)   Profile: Default

ScreenShot-5: Stopping IP Forwarding:



```
┌──(kunal㉿kali)-[/tmp/web]
└─$ echo 0 | sudo tee /proc/sys/net/ipv4/ip_forward

0

┌──(kunal㉿kali)-[/tmp/web]
└─$
```

Findings & Analysis:

● **Unencrypted Traffic is Easily Readable**

- **During the simulation, HTTP credentials submitted via a login form were successfully intercepted and clearly visible in the packet payload using Wireshark.**

● **ARP Spoofing Effectively Redirects Traffic**

- **Tools like arpspoof and ettercap allowed successful ARP poisoning, enabling the attacker to position themselves between the victim and the gateway.**

● **Sensitive Data Leaked Without User Awareness**

- **The victim was unaware of the interception, emphasizing how MITM attacks can go unnoticed without proper endpoint protection or encrypted communication.**

● **HTTPS Prevented Credential Exposure**

- **Attempts to capture credentials over HTTPS were unsuccessful, proving the effectiveness of encryption in protecting user data.**

● **Public Networks Are Inherently Risky**

- **The simulation demonstrated that devices on open or public Wi-Fi are especially vulnerable to MITM attacks when secure practices aren't followed.**

● **Wireshark Filtering Made Analysis Precise**

- **Applying filters like http.request.method == "POST" helped isolate login attempts and extract relevant information efficiently from the packet capture.**

Security Recommendations:

● Avoid using HTTP for login pages; enforce HTTPS across all web applications
● Use VPNs when accessing public or untrusted Wi-Fi networks to encrypt traffic
● Implement ARP spoofing detection tools or intrusion detection systems (IDS) in networks
● Use strong SSL/TLS certificates and enable HSTS (HTTP Strict Transport Security)
● Educate users to check for HTTPS and valid certificates before entering credentials
● Segment networks using VLANs to limit broadcast domains and reduce ARP attack surface
● Disable unused services and ports to minimize attack vectors
● Regularly update and patch systems to fix known vulnerabilities in network protocols

Learning Outcomes:

- **Understanding of Network Vulnerabilities**
    - **Gained practical knowledge of how unsecured networks can be exploited using MITM attacks.**
    - **Identified weaknesses in protocols like HTTP and ARP that can be leveraged by attackers.**
- **Hands-on Experience with Cybersecurity Tools**
    - **Learned to use real-world tools like Ettercap, arpspoof, and Wireshark for traffic interception and analysis.**
    - **Practiced capturing and filtering network packets to extract sensitive information.**
- **Importance of Secure Protocols**
    - **Observed how easily credentials can be exposed over HTTP, and how HTTPS encrypts data and prevents interception.**
    - **Understood the role of encryption, VPNs, and secure browsing habits in mitigating such risks.**
- **Simulated Real-World Attack Scenarios**
    - **Replicated attacker behavior in a controlled lab environment to better understand offensive tactics.**
    - **Enhanced ability to think like an attacker to improve defensive strategies.**
- **Improved Analytical and Reporting Skills**
    - **Documented each step methodically and analyzed captured data to draw meaningful security insights.**
    - **Learned to communicate technical findings in a structured report format.**

Future Scope:

- **Incorporating SSL Stripping Techniques**
    - **Future simulations can include HTTPS interception using tools like *sslstrip* or *bettercap* to demonstrate more advanced MITM attacks on encrypted traffic.**
- **Implementing Detection Mechanisms**
    - **Expand the project to include network-based intrusion detection systems (NIDS) like Snort or Suricata to detect ARP spoofing or unusual traffic patterns.**
- **Simulating Attacks in Larger Networks**
    - **Test the attack in a more complex network environment with multiple devices to study its scalability and impact.**
- **Automating the Attack Workflow**
    - **Develop scripts or use frameworks to automate ARP spoofing and packet capture, simulating how real attackers might streamline their attacks.**
- **Integration with Security Awareness Training**
    - **Use the simulation in workshops or training sessions to educate non-technical users about the risks of public Wi-Fi and the importance of secure browsing.**
- **Comparing Defensive Tools**
    - **Evaluate and compare the effectiveness of different security tools (e.g., VPNs, browser extensions, IDS) in preventing or detecting MITM attacks.**