

Name: Kunal Goyal

Reg No.: 22BCY10053

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

Tools Used: Ettercap, arpspoof, Wireshark, Python HTTP Server / Apache, Victim Machine (Windows/Linux) & Kali Linux.

Project Title

Who's Watching? Simulating Man-in-the-Middle Attacks Over Public Wi-Fi

Lab Environment Setup:

- Host Machine: **Windows 11**
- VM1 (Kali Linux): **Attacker machine**
- VM2 (Windows 10): **Victim machine**
- Network Mode: **Bridged Adapter - Host-only**
- Attacker IP: **192.168.232.128**
- Victim IP: **192.168.232.129**
- Gateway IP: **192.168.232.2**
- Key Tools Installed: **Ettercap, arpspoof, Wireshark, Python3 HTTP Server**

Step-by-Step Implementation:

Step 1: Initial Setup:

1. Setting Up VM Machines:

1.a. Attacker Machine(Kali Linux):

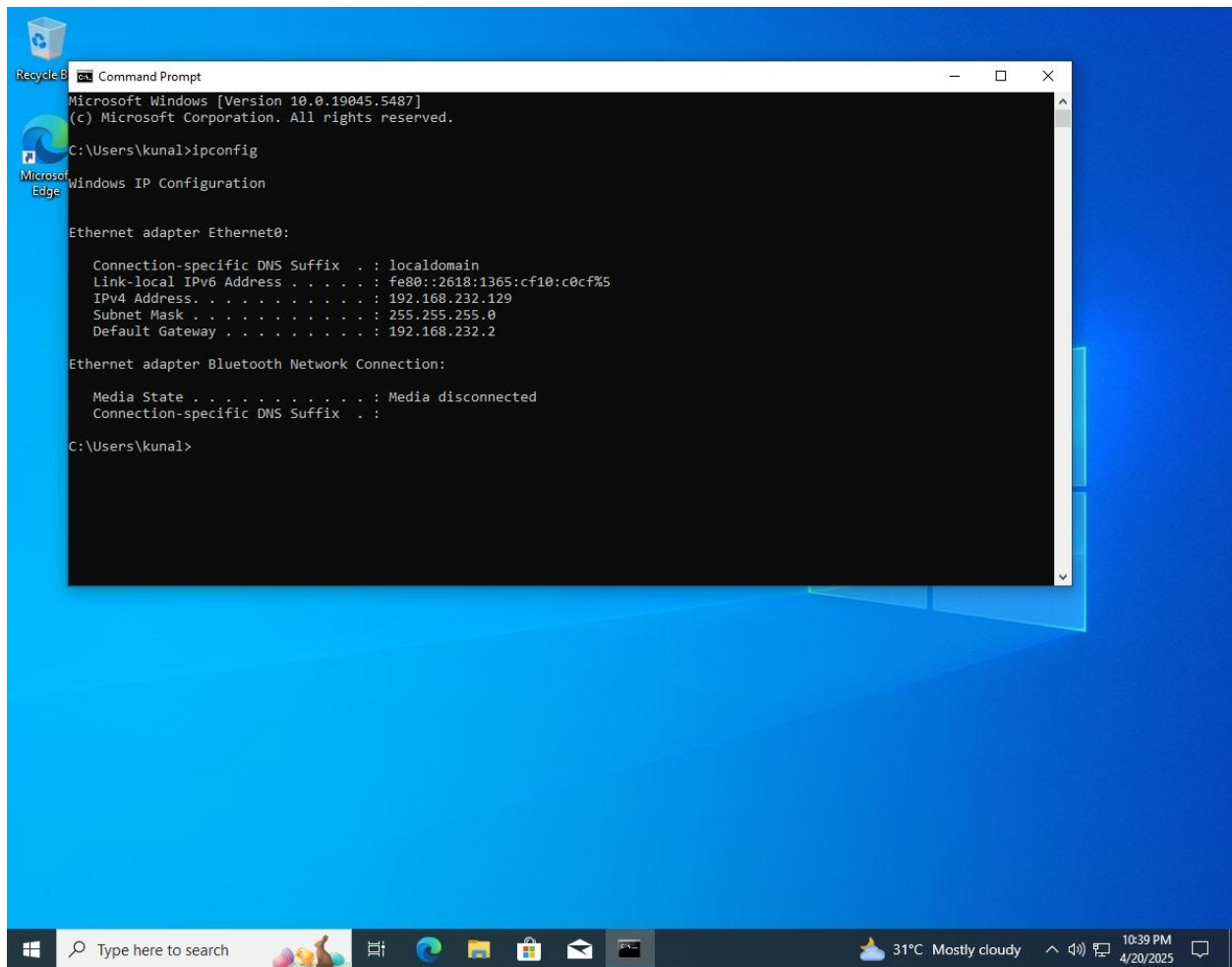
The screenshot shows a Kali Linux desktop environment with a terminal window open. The terminal displays the output of the `ifconfig` command. The output shows details for the `eth0` interface, including its flags, MTU, IP address (192.168.232.128), netmask (255.255.255.0), broadcast address (192.168.232.255), MAC address (00:0c:29:84:ee:fb), and statistics for RX and TX packets and errors. It also shows details for the `lo` (loopback) interface, including its flags, MTU, IP address (127.0.0.1), netmask (255.0.0.0), and statistics.

```
(kunal@kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.232.128 netmask 255.255.255.0 broadcast 192.168.232.255
    inet6 fe80::20c:29ff:fe84:ee:fb prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:84:ee:fb txqueuelen 1000 (Ethernet)
    RX packets 483768 bytes 724276165 (690.7 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 60623 bytes 3908169 (3.7 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

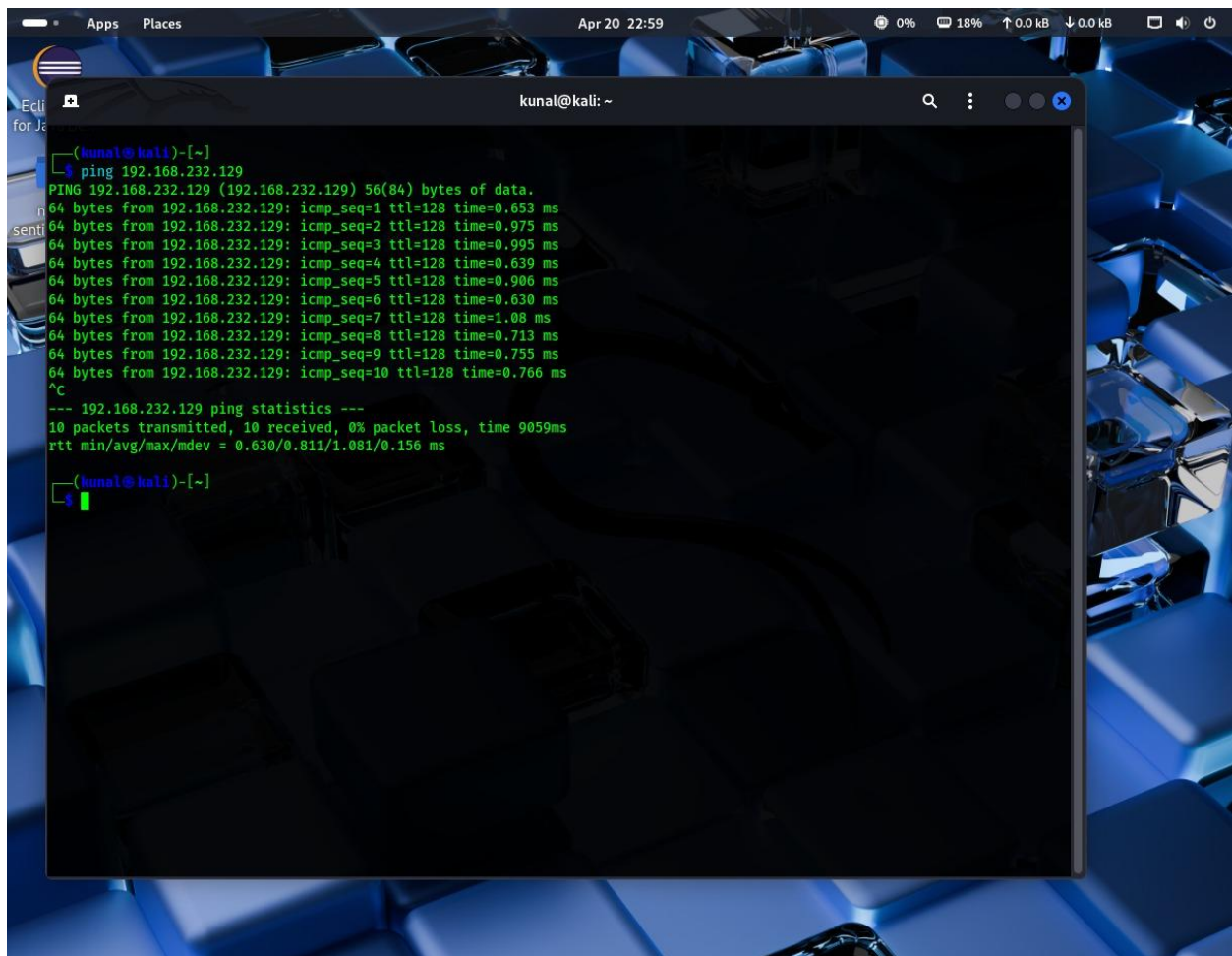
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 24 bytes 1440 (1.4 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 24 bytes 1440 (1.4 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(kunal@kali)-[~]
└─$
```

1.b. And Target Machine(Windows 10):



2. Check Connectivity:

A screenshot of a Kali Linux desktop environment. The background is a blue-tinted image of a computer keyboard. A terminal window is open, displaying the results of a ping command. The terminal title is 'kunal@kali: ~'. The output shows 10 successful ping packets to 192.168.232.129 with varying response times. The window's top bar shows system status: Apr 20 22:59, 0% CPU, 18% memory, and network activity.

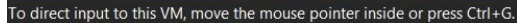
```
(kunal@kali)-[~]
└─$ ping 192.168.232.129
PING 192.168.232.129 (192.168.232.129) 56(84) bytes of data.
64 bytes from 192.168.232.129: icmp_seq=1 ttl=128 time=0.653 ms
64 bytes from 192.168.232.129: icmp_seq=2 ttl=128 time=0.975 ms
64 bytes from 192.168.232.129: icmp_seq=3 ttl=128 time=0.995 ms
64 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)-[~]
└─$
```

Step 2: Tool Configuration:

1. Update Your System

```
kunal@kali: ~  
(kunal@kali)~  
$ sudo apt update && sudo apt upgrade -y  
[sudo] password for kunal:  
Get:1 http://kali.download/kali kali-rolling InRelease [41.5 kB]  
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [21.0 MB]  
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [51.6 MB]  
Get:4 http://kali.download/kali kali-rolling/contrib amd64 Packages [121 kB]  
Get:5 http://kali.download/kali kali-rolling/contrib amd64 Contents (deb) [328 kB]  
Get:6 http://kali.download/kali kali-rolling/non-free amd64 Packages [204 kB]  
Get:7 http://kali.download/kali kali-rolling/non-free-firmware amd64 Packages [10.6 kB]  
Fetched 73.3 MB in 1min 33s (784 kB/s)  
1584 packages can be upgraded. Run 'apt list --upgradable' to see them.  
The following packages were automatically installed and are no longer required:  
crackmapexec libmbdtd14t64  
firebird3.0-common libmbdx509-1t64  
firebird3.0-common-doc libmsgraph-0-1  
fonts-inter-variable libnetcdf19t64  
icu-devtools libokular5core11  
kde-baseapps libopencv-core406t64  
kde-plasma-desktop libopencv-imgproc406t64  
kdoctools5 liboxygenstyle5-5  
kwrite liboxygenstyleconfig5-5  
libbbfio1 libpaper1  
libc++1-19 libpoppler140  
libc++abi1-19 libpoppler145  
libcapstone4 libpython3.12-dev  
libconfig++9v5 libpython3.12-minimal  
libconfig9 libpython3.12-stdlib  
libdirectfb-1.7-7t64 libpython3.12t64  
libegl-dev libqt5quickshapes5  
libflac12t64 libqt5webkit5  
libfmt9 libqt5webview5  
libfwupd2 libspdlog1.12  
libgdal35 libsuperlu6  
libgeos3.13.0 libtag1v5  
libgl1-mesa-dev libtag1v5-vanilla  
libglapi-mesa libtagc0  
libgles-dev libunwind-19  
libgles1 libvncclient1  
libglvnd-core-dev libwebRTC-audio-processing1  
libglvnd-dev libx265-209
```

2. Install Ettercap:



3. Install arpspoof:

```
Apr 20 23:02 0% 18% ↑ 0.0 kB ↓ 0.0 kB

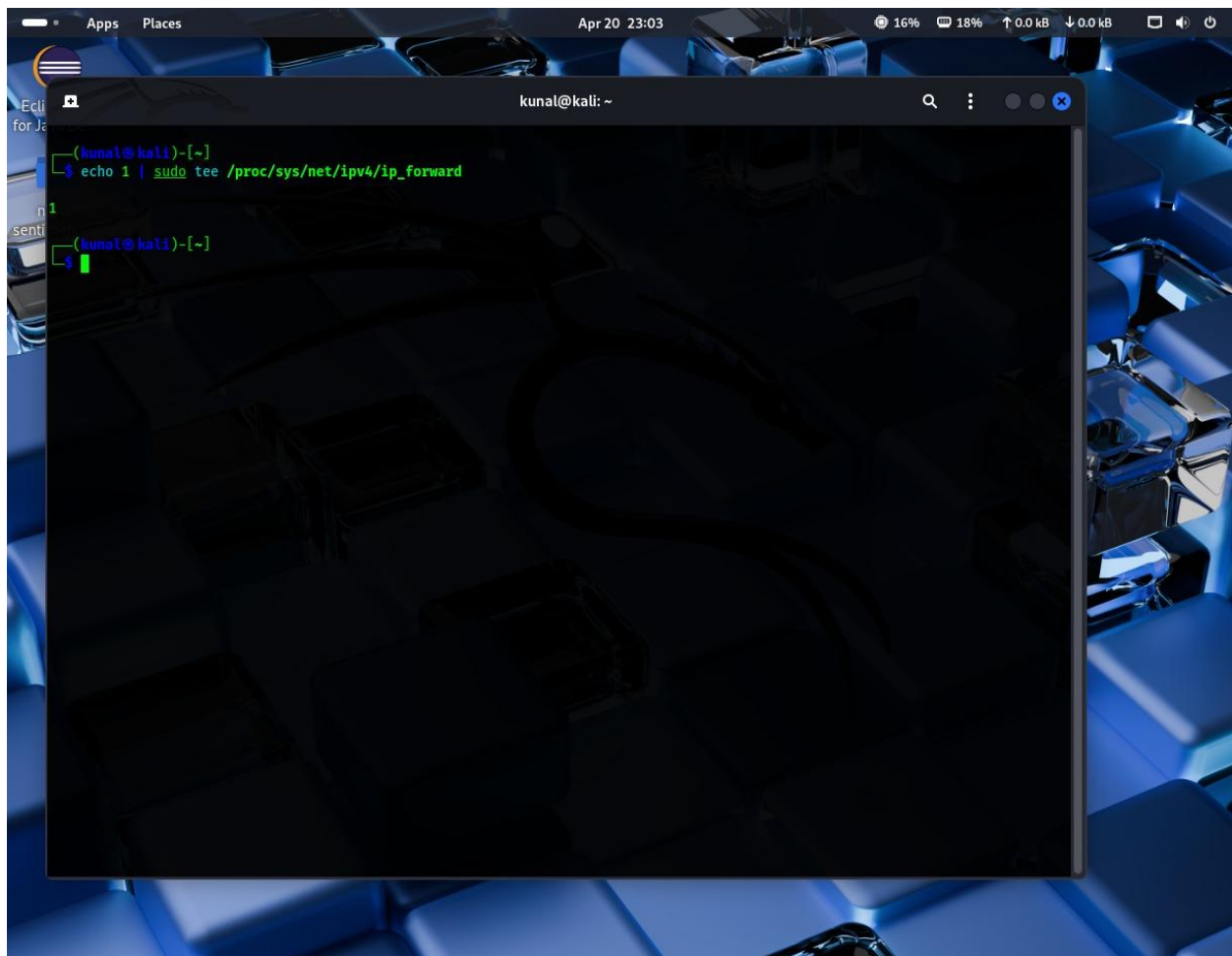
kunal@kali: ~
(kunal@kali)-[~]
└─$ sudo apt install dsniff

dsniff is already the newest version (2.4b1+debian-34).
dsniff set to manually installed.
The following packages were automatically installed and are no longer required:
firebird3.0-common libkdssoap1 libtag1v5
firebird3.0-common-doc libkf5dnssd-data libtag1v5-vanilla
kde-baseapps libkf5dnssd5 libtagc0
kde-plasma-desktop libkf5kdcraw5 libunwind-19
kwrite libkf5pty-data libvncclient1
libbfio1 libkf5pty5 libwebRTC-audio-processing1
libc++1-19 libkf5purpose-bin libx265-209
libc++abi1-19 libkf5purpose5 openjdk-23-jre
libcapstone4 libkf5su-bin openjdk-23-jre-headless
libconfig++9v5 libkf5su-data python3-appdirs
libconfig9 libkf5su5 python3.12
libdirectfb-1.7-7t64 libkf6config-bin python3.12-dev
libegl-dev libkimageannotator-qt5-0 python3.12-minimal
libfmt9 libllhttp9.1 python3.12-venv
libfwupd2 libmbdcrypto7t64 qml-module-org-kde-kcmutils
libgdal35 libmbdrtls14t64 qml-module-org-kde-purpose
libgl1-mesa-dev libmbdrtls14t64 qml-module-org-kde-syntaxhighlighting
libgles-dev libmsgraph-0-1 qml-module-qtquick-shapes
libgles1 libnetcdf19t64 qml-module-org-kde-kquickcontrolsaddons
libglvnd-core-dev libokular5core11 vlc
libglvnd-dev liboxyenstyle5-5 vlc-bin
libgtksourceview-3.0-1 liboxyenstyleconfig5-5 vlc-l10n
libgtksourceview-3.0-common libpaper1 vlc-plugin-access-extra
libgtksourceviewmm-3.0-0v5 libpoppler140 vlc-plugin-notify
libgumbo2 libpython3.12-dev vlc-plugin-qt
libhdf5-103-1t64 libqt5quickshapes5 vlc-plugin-samba
libhdf5-hl-100t64 libqt5webengine5 vlc-plugin-skins2
libjxl0.9 libqt5webview5 vlc-plugin-video-splitter
libkcolorpicker-qt5-0 libspdlog1.12 vlc-plugin-visualization
libkdecorations3private1 libsuperlu6 xsettingsd

Use 'sudo apt autoremove' to remove them.

Summary:
Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 1584
```

4. Enable IP Forwarding (Required for MITM):



Step 3: Simulate HTTP Login Page (Victim Machine):

1. Create a HTTP login page in the attacker machine for the target to put details into and start the server on port 80:

The screenshot shows a Kali Linux desktop environment with a terminal window open. The terminal window title is 'kunal@kali: /tmp/web'. The user is performing the following steps:

- Enabling IP forwarding: `echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward`
- Creating a web directory: `mkdir /tmp/web` and `cd /tmp/web`
- Creating an HTML form: `echo '<form method="POST"><input name="user"><input name="pass" type="password"><input type="submit"></form>' > index.html`
- Verifying the file: `cat index.html`
- Starting a web server: `sudo python3 -m http.server 80`

The terminal output shows the form content and the server starting on port 80.

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

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

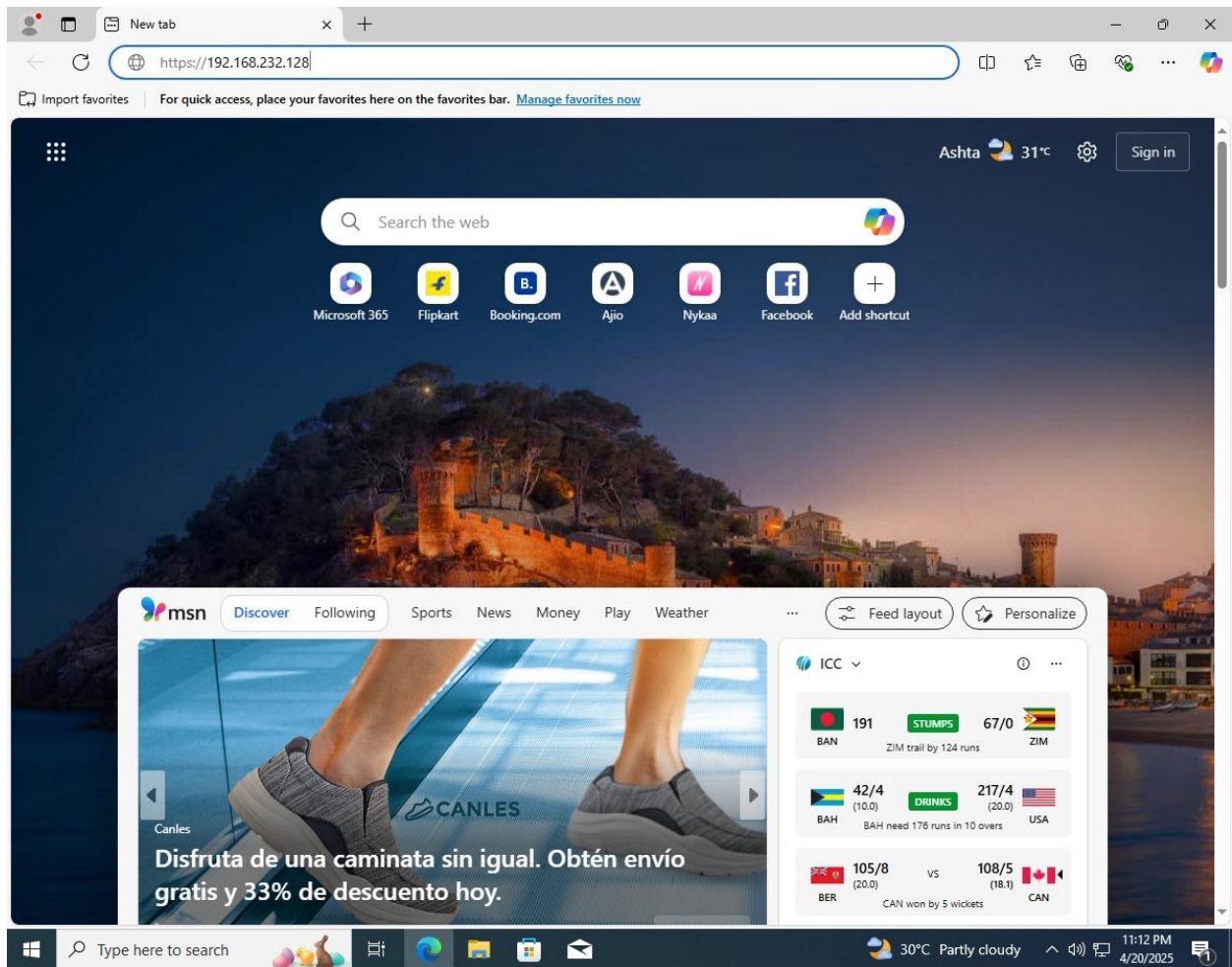
(kunal@kali)-[/tmp/web]
└─$ echo '<form method="POST"><input name="user"><input name="pass" type="password"><input type="submit"></form>' > index.html

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

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

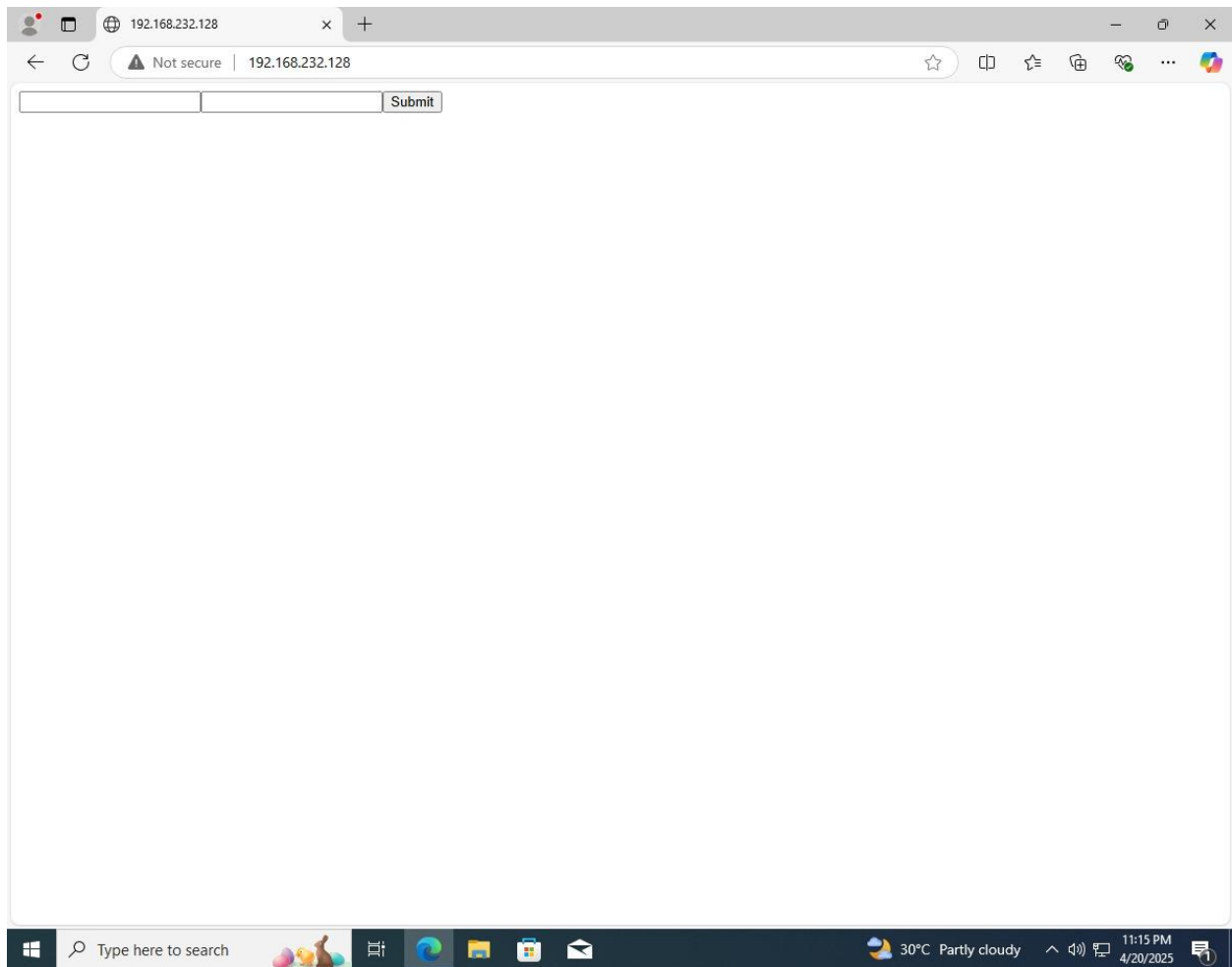
(kunal@kali)-[/tmp/web]
└─$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

2. On Victim Machine open browser and visit:



3. This will open a webpage asking for userId and Password for login.

I created a simple site but to fool the target we can add many details so that it looks like a real website.



Step 4: Launch the MITM Attack:

Here we have 3 options by which we can sniff user details:

Those are :

- **Wireshark**
- **Arpspoof**
- **Ettercap**

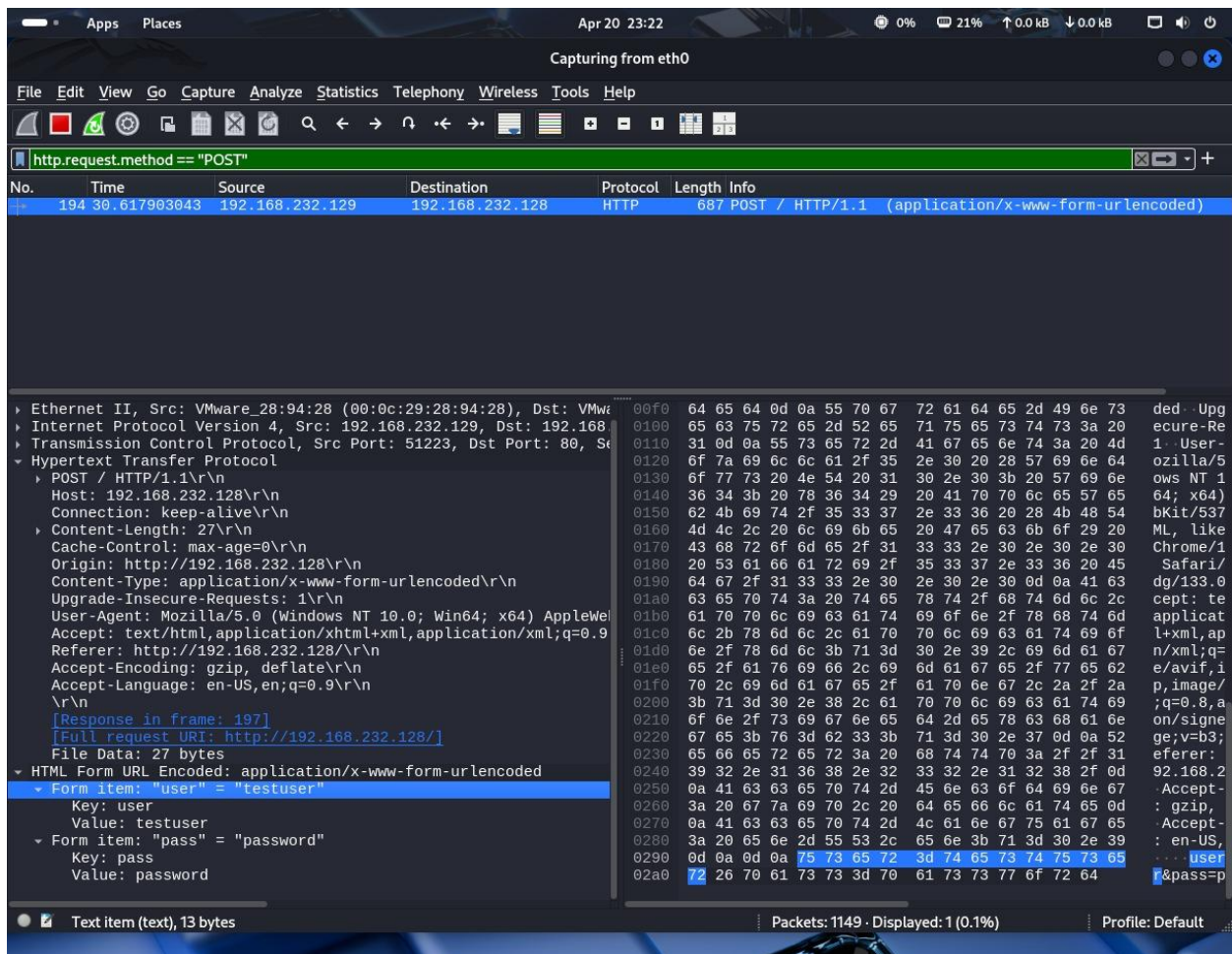
I will be using Wireshark.

1. Start the Wireshark app on Linux and start capturing packets.

```
kunal@kali: /tmp/web

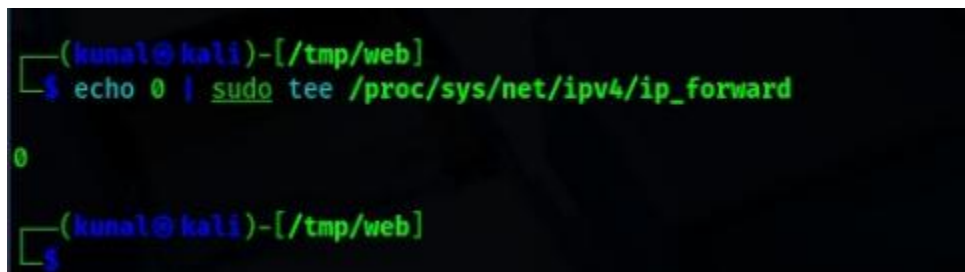
(kunal@kali)-[/tmp/web]
└─$ wireshark
** (Wireshark:4852) 23:18:39.912900 [GUI ECHO] -- virtual const QPalette* Qt6CTPlatformTheme::palette(QPlatformTheme::Palette) const QPlatformTheme::SystemPalette
** (Wireshark:4852) 23:18:39.917612 [GUI ECHO] -- virtual const QPalette* Qt6CTPlatformTheme::palette(QPlatformTheme::Palette) const QPlatformTheme::ToolButtonPalette
** (Wireshark:4852) 23:18:39.917836 [GUI ECHO] -- virtual const QPalette* Qt6CTPlatformTheme::palette(QPlatformTheme::Palette) const QPlatformTheme::ButtonPalette
** (Wireshark:4852) 23:18:39.917945 [GUI ECHO] -- virtual const QPalette* Qt6CTPlatformTheme::palette(QPlatformTheme::Palette) const QPlatformTheme::CheckBoxPalette
** (Wireshark:4852) 23:18:39.918010 [GUI ECHO] -- virtual const QPalette* Qt6CTPlatformTheme::palette(QPlatformTheme::Palette) const QPlatformTheme::RadioButtonPalette
** (Wireshark:4852) 23:18:39.918049 [GUI ECHO] -- virtual const QPalette* Qt6CTPlatformTheme::palette(QPlatformTheme::Palette) const QPlatformTheme::HeaderPalette
** (Wireshark:4852) 23:18:39.918347 [GUI ECHO] -- virtual const QPalette* Qt6CTPlatformTheme::palette(QPlatformTheme::Palette) const QPlatformTheme::ItemViewPalette
** (Wireshark:4852) 23:18:39.918423 [GUI ECHO] -- virtual const QPalette* Qt6CTPlatformTheme::palette(QPlatformTheme::Palette) const QPlatformTheme::MessageBoxLabelPalette
** (Wireshark:4852) 23:18:39.918423 [GUI ECHO] -- virtual const QPalette* Qt6CTPlatformTheme::palette(QPlatformTheme::Palette) const QPlatformTheme::TabBarPalette
** (Wireshark:4852) 23:18:39.918514 [GUI ECHO] -- virtual const QPalette* Qt6CTPlatformTheme::palette(QPlatformTheme::Palette) const QPlatformTheme::LabelPalette
** (Wireshark:4852) 23:18:39.918562 [GUI ECHO] -- virtual const QPalette* Qt6CTPlatformTheme::palette(QPlatformTheme::Palette) const QPlatformTheme::GroupBoxPalette
** (Wireshark:4852) 23:18:39.918585 [GUI ECHO] -- virtual const QPalette* Qt6CTPlatformTheme::palette(QPlatformTheme::Palette) const QPlatformTheme::MenuPalette
** (Wireshark:4852) 23:18:39.918607 [GUI ECHO] -- virtual const QPalette* Qt6CTPlatformTheme::palette(QPlatformTheme::Palette) const QPlatformTheme::MenuBarPalette
** (Wireshark:4852) 23:18:39.918747 [GUI ECHO] -- virtual const QPalette* Qt6CTPlatformTheme::palette(QPlatformTheme::Palette) const QPlatformTheme::TextEditPalette
** (Wireshark:4852) 23:18:39.918915 [GUI ECHO] -- virtual const QPalette* Qt6CTPlatformTheme::palette(QPlatformTheme::Palette) const QPlatformTheme::TextEditPalette
** (Wireshark:4852) 23:18:39.919032 [GUI ECHO] -- virtual const QPalette* Qt6CTPlatformTheme::palette(QPlatformTheme::Palette) const QPlatformTheme::TextLineEditPalette
** (Wireshark:4852) 23:18:39.919076 [GUI ECHO] -- virtual const QPalette* Qt6CTPlatformTheme::palette(QPlatformTheme::Palette) const QPlatformTheme::ToolTipPalette
** (Wireshark:4852) 23:18:39.919323 [GUI ECHO] -- virtual QVariant Qt6CTPlatformTheme::themeHint(QPlatformTheme::ThemeHint) const
** (Wireshark:4852) 23:18:39.921558 [GUI ECHO] -- virtual const QPalette* Qt6CTPlatformTheme::palette(QPlatformTheme::Palette) const QPlatformTheme::SystemPalette
** (Wireshark:4852) 23:18:40.436423 [GUI ECHO] -- virtual QVariant Qt6CTPlatformTheme::themeHint(QPlatformTheme::ThemeHint) const
** (Wireshark:4852) 23:18:40.547266 [GUI ECHO] -- virtual const QPalette* Qt6CTPlatformTheme::palette(QPlatformTheme::Palette) const QPlatformTheme::SystemPalette
```

This will capture any data inputted by the target.



Here all the details will be available.

3. Now input this to stop ip forwarding:



This concludes this attack.

Observations & Findings:

1. Successful Credential Interception

- When the victim accessed an HTTP login page, the attacker was able to capture the username and password in plain text using Wireshark.

2. ARP Spoofing Effectiveness

- ARP spoofing allowed the attacker to position themselves between the victim and the gateway without being detected by the victim.

3. HTTPS Prevented Credential Leakage

- When the victim accessed a similar login page over HTTPS, credentials were not visible in the captured packets.

4. VPN Usage Blocked MITM Attack

- Testing with a VPN on the victim machine encrypted all traffic, making it unreadable to the attacker, even during ARP spoofing.

5. Tool Integration Worked Smoothly

- Tools like arpspoof, Ettercap, and Wireshark worked effectively together for simulating and analyzing the MITM attack.

6. User Awareness is Critical

- Most users are unaware when ARP spoofing is taking place, emphasizing the need for secure browsing habits and encrypted connections.

Challenges Faced:

- Initial network configuration issues between attacker and victim VMs (had to switch from NAT to Bridged mode)
- Victim machine's firewall blocked ARP spoofing initially; required temporary firewall adjustments
- Ettercap occasionally crashed or failed to start properly; had to troubleshoot with command-line options
- Difficulty filtering relevant HTTP POST traffic in Wireshark due to background noise in network packets
- HTTPS traffic could not be analyzed without proper SSL stripping, which wasn't in project scope
- VPN testing required external setup and extra configuration time for accurate simulation

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

Final Deliverables:

- Captured Traffic File (.pcap)
 - A Wireshark capture file containing HTTP traffic intercepted during the MITM simulation.
 - Includes the full packet trace from the victim machine, showing how credentials were exposed during login.

- **Screenshots of Key Steps**

- Visual documentation of each stage of the attack, such as:
 - ARP spoofing using arpspoof and ettercap
 - Wireshark interface with HTTP POST data
 - Python HTTP server setup on attacker machine
 - Victim interacting with the fake login page
- Annotated if necessary for clarity.

- **Project Documentation (PDF/Word)**

- A complete write-up of the project, covering:
 - Problem statement, objectives, methodology, tools used
 - Step-by-step implementation with commands
 - Observations, findings, challenges, and recommendations
 - Conclusion and summary of learning

- **Network Diagram**

- A simple visual showing how the MITM attack was set up in the lab environment
- Includes VM details, IP addresses, network mode, and traffic flow paths

- **Demo Video (Optional)**

- A screen recording walking through the attack from start to finish
- Shows real-time output of tools like Wireshark and Ettercap during the interception
- Useful for viva, presentation, or submission if permitted

- **Presentation Slides**

- A summary of the project in a slide deck format
- Useful for classroom or final evaluation presentations
- Covers background, setup, key steps, findings, and recommendations

Conclusion:

This project successfully simulated a Man-in-the-Middle (MITM) attack to demonstrate how attackers can intercept sensitive information, such as login credentials, over unsecured networks. By using tools like Ettercap, arpspoof, and Wireshark, the project recreated a real-world scenario where a victim unknowingly connected to a compromised public Wi-Fi network and submitted data over an unencrypted HTTP connection.

The simulation clearly showed that without encryption, critical user information can be easily captured and exploited. It also highlighted the dangers of ARP spoofing and the effectiveness of packet sniffing tools in exposing vulnerabilities in network communication.

Beyond executing the attack, the project placed equal emphasis on analyzing the captured data and understanding the consequences of poor security practices. It reinforced the importance of secure protocols (HTTPS), encrypted tunnels (VPNs), and proper network segmentation as key defenses against MITM attacks.

Overall, this hands-on experience deepened the understanding of both offensive and defensive aspects of cybersecurity. It provided valuable insights into how attackers think and operate, while also underlining the critical need for strong security policies, user education, and constant vigilance in today's connected environments.