

Name - HARSHIL AMIT BUCH  
Class - TY CSF-1  
Roll Number - 14

## Experiment No. 4

# Detecting ARP Spoofing & Understanding MITM Attacks

---

### Aim:

Understand ARP, detect ARP spoofing, and simulate defensive steps against Man-in-the-Middle (MITM) attacks in a safe, controlled environment.

### Requirements:

1. Kali Linux (Virtual Machine or physical machine)
2. Tools: Wireshark, tcpdump, arpwatch, nmap
3. Isolated lab network (VirtualBox/VMware internal network)
4. Two or more virtual machines (attacker, victim, gateway simulation)

### Procedure:

#### Step 1: Install Required Tools

Update system and install tools:

```
sudo apt update  
sudo apt install -y wireshark tcpdump arpwatch arping nmap iproute2 net-tools
```

#### Step 2: Check Network Interface

Find your active interface:



The screenshot shows a terminal window titled "zsh — Konsole". The command "ip -br a" is run, displaying the following network interface status:

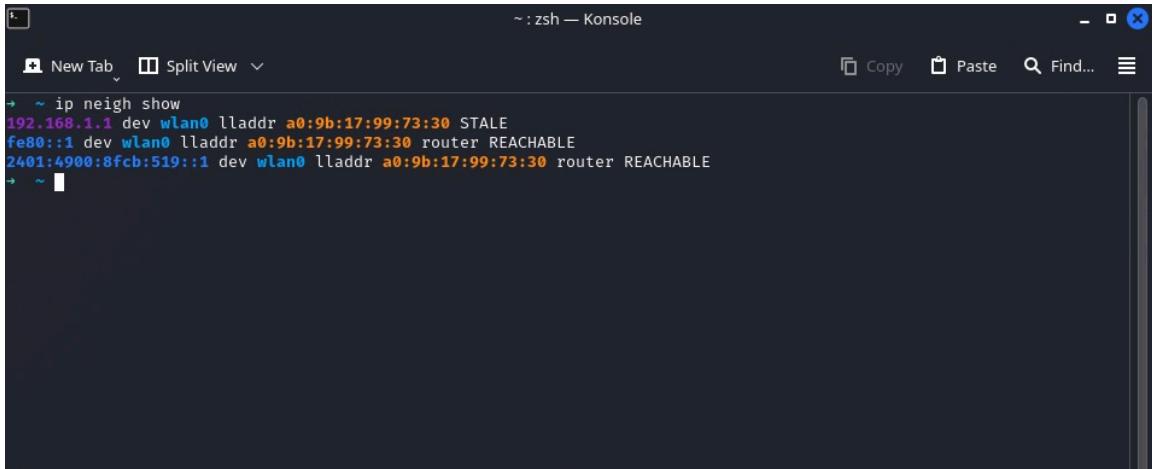
Interface	Status	IP Address	MAC Address
lo	UNKNOWN	127.0.0.1/8	:1/128
eth0	DOWN		
wlan0	UP	192.168.1.84/24	2401:4900:8fc:519:358c:7dd8:7395:9311/64
4ff:fe:ff:babe:64	fe80::b27d:64ff:fe:ff:babe/64		
docker0	DOWN	172.17.0.1/16	

```
ip -br a  
(Example: eth0, ens33, wlan0)
```

### Step 3: View ARP Table

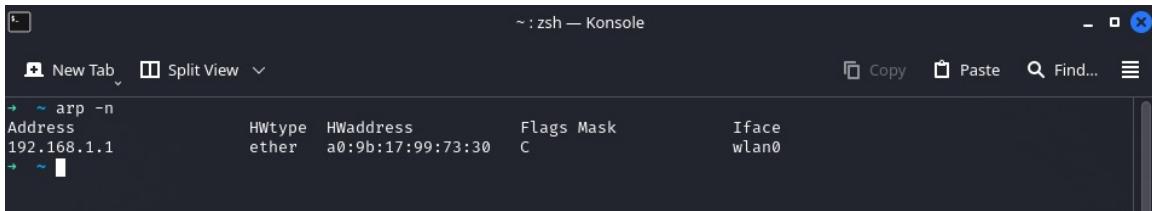
Display ARP cache:

```
ip neigh show
```



```
~ : zsh — Konsole
New Tab Split View ~
Copy Paste Find... ⌂
+ ip neigh show
192.168.1.1 dev wlan0 lladdr a0:9b:17:99:73:30 STALE
fe80::1 dev wlan0 lladdr a0:9b:17:99:73:30 router REACHABLE
2401:4900:8fc:519::1 dev wlan0 lladdr a0:9b:17:99:73:30 router REACHABLE
~ ~
```

```
arp -n
```



```
~ : zsh — Konsole
New Tab Split View ~
Copy Paste Find... ⌂
+ arp -n
Address      HWtype  HWaddress          Flags Mask   Iface
192.168.1.1  ether    a0:9b:17:99:73:30  C      wlan0
~ ~
```

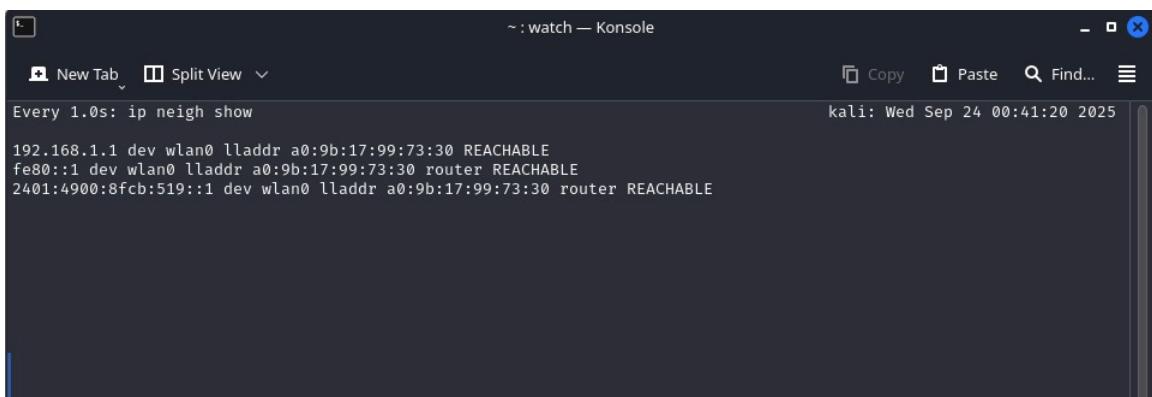
Observe IP and MAC mappings.

### Step 4: Monitor ARP Table in Real Time

Run:

```
watch -n 1 "ip neigh show"
```

Check for changes in MAC addresses for same IP.



```
~ : watch — Konsole
New Tab Split View ~
Copy Paste Find... ⌂
Every 1.0s: ip neigh show
kali: Wed Sep 24 00:41:20 2025
192.168.1.1 dev wlan0 lladdr a0:9b:17:99:73:30 REACHABLE
fe80::1 dev wlan0 lladdr a0:9b:17:99:73:30 router REACHABLE
2401:4900:8fc:519::1 dev wlan0 lladdr a0:9b:17:99:73:30 router REACHABLE
```

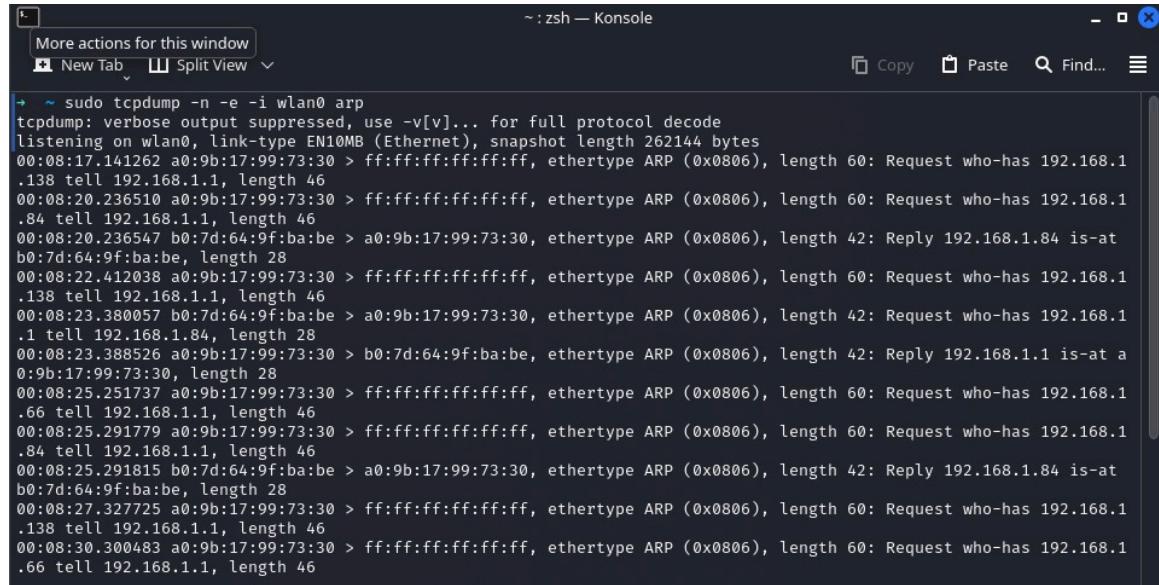
## Step 5: Capture ARP Packets

Use tcpdump:

```
sudo tcpdump -n -e -i <interface> arp
```

Indicators of ARP spoofing:

- Multiple ARP replies without requests
- Gateway IP mapping to unusual MAC addresses



```
~ :zsh — Konsole
More actions for this window
New Tab Split View
Copy Paste Find... ≡

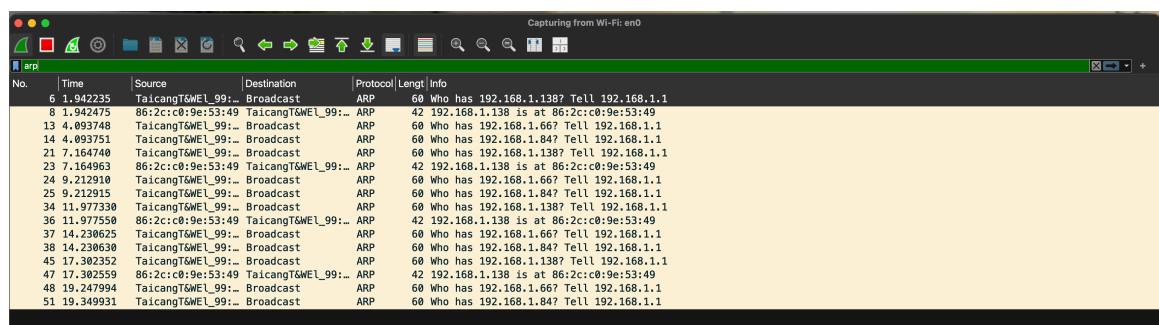
+ ~ sudo tcpdump -n -e -i wlan0 arp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on wlan0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
00:08:17.141262 a0:9b:17:99:73:30 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 60: Request who-has 192.168.1.138 tell 192.168.1.1, length 46
00:08:20.236510 a0:9b:17:99:73:30 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 60: Request who-has 192.168.1.84 tell 192.168.1.1, length 46
00:08:20.236547 b0:7d:64:9f:ba:be > a0:9b:17:99:73:30, ethertype ARP (0x0806), length 42: Reply 192.168.1.84 is-at b0:7d:64:9f:ba:be, length 28
00:08:22.412038 a0:9b:17:99:73:30 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 60: Request who-has 192.168.1.138 tell 192.168.1.1, length 46
00:08:23.380057 b0:7d:64:9f:ba:be > a0:9b:17:99:73:30, ethertype ARP (0x0806), length 42: Request who-has 192.168.1.1 tell 192.168.1.84, length 28
00:08:23.388526 a0:9b:17:99:73:30 > b0:7d:64:9f:ba:be, ethertype ARP (0x0806), length 42: Reply 192.168.1.1 is-at a0:9b:17:99:73:30, length 28
00:08:25.251737 a0:9b:17:99:73:30 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 60: Request who-has 192.168.1.66 tell 192.168.1.1, length 46
00:08:25.291779 a0:9b:17:99:73:30 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 60: Request who-has 192.168.1.84 tell 192.168.1.1, length 46
00:08:25.291815 b0:7d:64:9f:ba:be > a0:9b:17:99:73:30, ethertype ARP (0x0806), length 42: Reply 192.168.1.84 is-at b0:7d:64:9f:ba:be, length 28
00:08:27.327725 a0:9b:17:99:73:30 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 60: Request who-has 192.168.1.138 tell 192.168.1.1, length 46
00:08:30.300483 a0:9b:17:99:73:30 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 60: Request who-has 192.168.1.66 tell 192.168.1.1, length 46
```

## Step 6: Wireshark Analysis

Start Wireshark and apply filter:

```
arp
```

Look for duplicate IP and unsolicited ARP replies.



No.	Time	Source	Destination	Protocol	Length	Info
6	1.942235	TaicangT&WEI_99:..	Broadcast	ARP	60	Who has 192.168.1.138? Tell 192.168.1.1
8	1.942475	86:2:c:0:9e:53:49	TaicangT&WEI_99:..	ARP	42	192.168.1.138 is at 86:2:c:0:9e:53:49
13	4.493748	TaicangT&WEI_99:..	Broadcast	ARP	60	Who has 192.168.1.667 Tell 192.168.1.1
14	4.493751	TaicangT&WEI_99:..	Broadcast	ARP	60	Who has 192.168.1.847 Tell 192.168.1.1
21	7.164740	TaicangT&WEI_99:..	Broadcast	ARP	60	Who has 192.168.1.138 Tell 192.168.1.1
23	7.164963	86:2:c:0:9e:53:49	TaicangT&WEI_99:..	ARP	42	192.168.1.138 is at 86:2:c:0:9e:53:49
24	9.212916	TaicangT&WEI_99:..	Broadcast	ARP	60	Who has 192.168.1.667 Tell 192.168.1.1
25	9.212919	TaicangT&WEI_99:..	Broadcast	ARP	60	Who has 192.168.1.847 Tell 192.168.1.1
34	11.977338	TaicangT&WEI_99:..	Broadcast	ARP	60	Who has 192.168.1.138 Tell 192.168.1.1
36	11.977558	86:2:c:0:9e:53:49	TaicangT&WEI_99:..	ARP	42	192.168.1.138 is at 86:2:c:0:9e:53:49
37	14.238625	TaicangT&WEI_99:..	Broadcast	ARP	60	Who has 192.168.1.667 Tell 192.168.1.1
38	14.238638	TaicangT&WEI_99:..	Broadcast	ARP	60	Who has 192.168.1.847 Tell 192.168.1.1
45	17.302352	TaicangT&WEI_99:..	Broadcast	ARP	60	Who has 192.168.1.138 Tell 192.168.1.1
47	17.302559	86:2:c:0:9e:53:49	TaicangT&WEI_99:..	ARP	42	192.168.1.138 is at 86:2:c:0:9e:53:49
48	19.247994	TaicangT&WEI_99:..	Broadcast	ARP	60	Who has 192.168.1.667 Tell 192.168.1.1
51	19.349931	TaicangT&WEI_99:..	Broadcast	ARP	60	Who has 192.168.1.847 Tell 192.168.1.1

## **Step 7: Test HTTP vs HTTPS Security**

Compare unencrypted and encrypted traffic:

```
curl -v http://example.com  
curl -v https://example.com
```

Discuss how HTTPS prevents MITM attacks.

```
[*] New Tab [x] Split View Copy Paste Find
* curl -v http://httpforever.com/
* Host httpforever.com:80 was resolved.
* IPv6: 2004:a880:4:1d0::1f1:2000
* IP4: 16.198.62.39:80
* Trying [2004:a880:4:1d0::1f1:2000]:80...
* Trying 16.198.62.39:80...
* Connected to httpforever.com (2004:a880:4:1d0::1f1:2000) port 80
* HTTP/1.1 200 OK
* Date: Mon, 22 Mar 2023 14:54:48 GMT
* Server: nginx/1.18.0 (Ubuntu)
* Content-Type: text/html
* Content-Length: 512
* Connection: keep-alive
* ETAG: "641b1b0b-146"
* X-Content-Options: nosniff
* Feature-Policy: accelerometer 'none'; camera 'none'; geolocation 'none'; gyroscope 'none'; magnetometer 'none'; microphone 'none'; payment 'none'; usb 'none'
* Content-Security-Policy: default-src 'script' 'src' cdnjs.cloudflare.com 'self' 'report-sha256'; style-src cdnjs.cloudflare.com 'self' fonts.googleapis.com 'unsafe-inline'; font-src fonts.googleapis.com fonts.gstatic.com cdnjs.cloudflare.com frame-ancestors 'none'; report-uri https://scotthelme-report-ua.com/c/ua/enforce
* Accept-Ranges: bytes
<!DOCTYPE HTML>
<html>
<head>
<title>HTTP Forever</title>
<meta rel="equiv" content="text/html; charset=utf-8" />
<meta name="description" content="A site that will always be available over HTTP!" />
<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0" />
<!-- If lte IE 8--><script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js" integrity="sha256-4QzZ+E/3SDUHzsCkUJLWqyqjwvXmBqOoqquweddw==" crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/skel/2.2.1/skel.min.js" integrity="sha256-3e-N0gDp/vc/y1zawqyksUre51CLUmqC2iZ74r==" crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/skel/2.2.1/skel.min.js" integrity="sha256-0xgf/CphKcIAu0A1WmpFy9VzCzxsGfcw==" crossorigin="anonymous"></script>
<script src="js/init.min.js"></script>
<script src="js/style.min.js"></script>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/skel/2.2.1/skel.css" integrity="sha256-HoTb0jxAGIeMqAD2qN6adFwC0wiPnr7mC7qBw==" crossorigin="anonymous" />
<link rel="stylesheet" href="css/style.min.css" />
<link rel="stylesheet" href="css/style-wide.min.css" />
</noscript>
<!-- If lte IE 8--><link rel="stylesheet" href="css/vb.min.css" /><![endif-->
</head>
<body class="landing">
<section id="banner">
<h2>HTTP FOREVER</h2>
<p>not really insecure connection</p>
</section>
<div class="wrap">
<div class="container">
<header class="major">
<h2>Why does this site exist?</h2>
<!-- This domain started out as my personal 'captive portal buster' but I wanted to publicise it for anyone to use. If you're on a train, in a hotel or bar, on a flight or anywhere that you have to login for WiFi, this site could help you!</p>

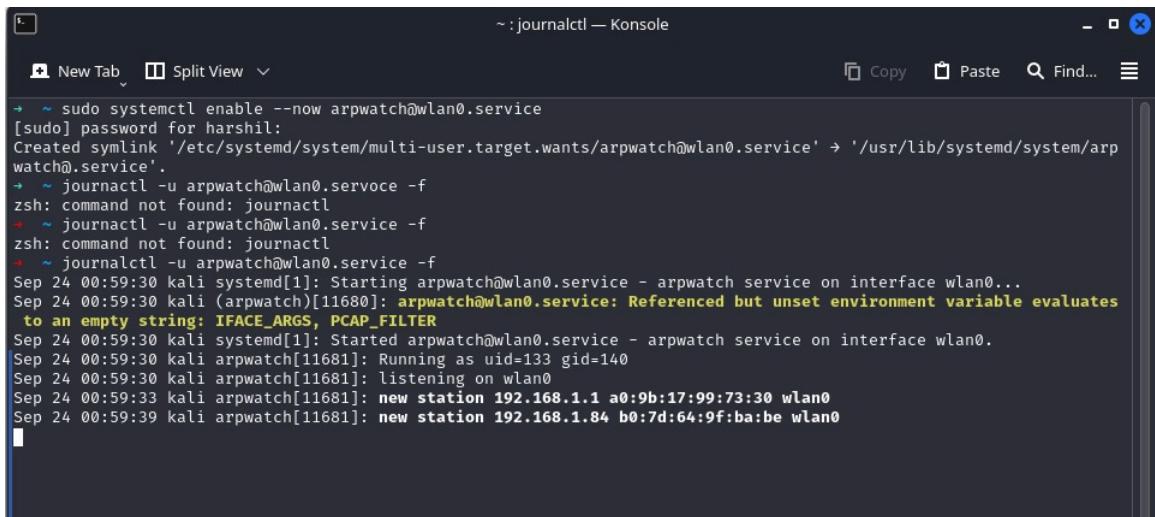
```

```
* curl -v https://niriksh.com
* Host niriksh.com:443 was resolved.
* IPv6: 2a0e:da18:b3d:ec2::258, 2a0e:da18:b3d:ec2::259
* TCP Conn ID: 23343333333333333333333333333333
* Trying [2a0e:da18:b3d:ec2::258]:443...
* ALPN: curl offers H2,Http/1.1
* ALPN: Selected: H2
< Client hello (1):
  <file: /etc/ssl/certs/ca-certificates.crt
> Capath: /etc/ssl/certs
> TLSv1.3 (IN) TLS handshake, Server hello (2):
  < TLSv1.3 (IN) TLS handshake, Change cipher spec (1):
  < TLSv1.3 (IN) TLS handshake, Encrypted Extensions (8):
  < TLSv1.3 (IN) TLS handshake, Certificate (1):
    <file: /etc/ssl/certs/ca-certificates.crt
  < TLSv1.3 (IN) TLS handshake, Finished (20)
* TLSv1.3 (OUT) TLS change cipher, Change cipher spec (1):
* TLSv1.3 (OUT) TLS handshake, Finished (20)
< SSL connection using TLSv1.3 / TLS_AES_128_GCM_SHA256 / x25519 / id-eCPubKey
* ALPN: server accepted h2
* Server certificate:
*     subject: CN=niriksh.com
*     start date: Jul 31 12:51:44 2016 GMT
*     exp. date: Aug 31 12:51:40 2026 GMT
*     subjectAltName: host "niriksh.com" matched cert's "niriksh.com"
*     issuer: C=US O=Lets Encrypt; CN=letsencrypt.org
*     certificate level 0: Public key type EC/prime256v1 (256/128 Bits/secBits), signed using ecdsa-with-SHA384
*     certificate level 1: Public key type EC/x25519 (384/192 Bits/secBits), signed using sha256WithRSAEncryption
*     certificate level 2: Public key type EC/prime256v1 (256/128 Bits/secBits), signed using sha256WithRSAEncryption
* Connected to niriksh.com [2a0e:da18:b3d:ec2::258] port 443
* Using HTTP/2
< [HTTP/2] OPEN_STREAM for https://niriksh.com/
* [HTTP/2] [1] [:method: GET]
* [HTTP/2] [1] [:scheme: https]
* [HTTP/2] [1] [:authority: niriksh.com]
* [HTTP/2] [1] [:path: /]
* [HTTP/2] [1] [:user-agent: curl/8.14.1]
* [HTTP/2] [1] [:accept: */*]
* [HTTP/2] [1] [:version: HTTP/2]
* Host: niriksh.com
> User-Agent: curl/8.14.1
> Accept: */*
< TLSv1.3 (IN) TLS handshake, NewSession Ticket (4):
* Message completely sent off
< HTTP/2 200
< accept-ranges: bytes
< etag: "f75a370d9f638084a2a2d07bec861-ss1"
< cache-control: public,max-age=0,must-revalidate
< cache-status: "Netflix Edge"; hit
< content-type: application/javascript; charset=UTF-8
< date: Tue, 23 Sep 2025 16:46:53 GMT
< etag: "f75a370d9f638084a2a2d07bec861-ss1"
< strict-transport-security: max-age=31536000
< x-nf-request-id: 01KSVYCVI3QMWZ1R840BT8KX6CD
< content-length: 454
<
<Content-Type: application/javascript>
```

## Step 8: Continuous Monitoring with arpwatch

Enable arpwatch service:

```
sudo systemctl enable --now arpwatch@<interface>.service  
journalctl -u arpwatch@<interface>.service -f
```



```
~ :journalctl — Konsole  
New Tab Split View ~ :journalctl — Konsole  
Copy Paste Find...  
+ ~ sudo systemctl enable --now arpwatch@wlan0.service  
[sudo] password for harshil:  
Created symlink '/etc/systemd/system/multi-user.target.wants/arpwatch@wlan0.service' → '/usr/lib/systemd/system/arpwatch@.service'.  
+ ~ journalctl -u arpwatch@wlan0.servoce -f  
zsh: command not found: journalactl  
+ ~ journalctl -u arpwatch@wlan0.service -f  
zsh: command not found: journalactl  
+ ~ journalactl -u arpwatch@wlan0.service -f  
Sep 24 00:59:30 kali systemd[1]: Starting arpwatch@wlan0.service - arpwatch service on interface wlan0...  
Sep 24 00:59:30 kali (arpwatch)[11680]: arpwatch@wlan0.service: Referenced but unset environment variable evaluates  
to an empty string: IFACE_ARGS, PCAP_FILTER  
Sep 24 00:59:30 kali systemd[1]: Started arpwatch@wlan0.service - arpwatch service on interface wlan0.  
Sep 24 00:59:30 kali arpwatch[11681]: Running as uid=133 gid=140  
Sep 24 00:59:30 kali arpwatch[11681]: listening on wlan0  
Sep 24 00:59:33 kali arpwatch[11681]: new station 192.168.1.1 a0:9b:17:99:73:30 wlan0  
Sep 24 00:59:39 kali arpwatch[11681]: new station 192.168.1.84 b0:7d:64:9f:ba:be wlan0
```

Observe new MAC and IP changes.

## Step 9: Network Scanning

Run a ping sweep:

```
nmap -sn 192.168.0.0/24
```



```
~ :zsh — Konsole  
New Tab Split View ~ :zsh — Konsole  
Copy Paste Find...  
+ ~ nmap -sn 192.168.1.84  
Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-24 00:21 IST  
Nmap scan report for 192.168.1.84  
Host is up.  
Nmap done: 1 IP address (1 host up) scanned in 0.02 seconds
```

Compare results with your expected network map.

## **Step 10: Static ARP Entry (Defense)**

Get gateway IP and MAC:

```
ip route | grep default  
ip neigh show | grep <gateway_ip>
```

Set static entry:

```
sudo ip neigh replace <gateway_ip> lladdr <gateway_mac> nud permanent dev <interface>
```

Remove static entry:

```
sudo ip neigh del <gateway_ip> dev <interface>
```

## **Expected Outcomes:**

1. Students will learn how to check ARP tables and monitor changes.
2. Detect unusual ARP behavior using tcpdump and Wireshark.
3. Understand the difference between HTTP and HTTPS security.
4. Use arpwatch for ARP spoof detection.
5. Apply static ARP entries for security.

## **Viva/Review Questions:**

1. What is ARP and why is it vulnerable?
2. How does ARP spoofing enable MITM attacks?
3. Why does HTTPS reduce MITM risk?
4. What is the purpose of static ARP entries?
5. Which network configurations prevent ARP spoofing?