Experiment No. 9
Detect ARP spoofing using nmap and/or open source tool ARPWATCH and wireshark.
Date of Performance:
Date of Submission:

Experiment No. 9

Detect ARP spoofing using nmap and/or open source tool ARPWATCH and wireshark

Course Outcome [CSL602.6]: Apply security basics for different attacks on network.

Aim: Detect ARP spoofing using nmap and/or open source tool ARPWATCH and wireshark

Objectives:

• To understand ARP spoofing.

• To understand ARPWATCH and use it to detect ARP spoofing.

Theory:

1. Nmap (Network Mapper):

While Nmap isn't specifically designed for ARP spoofing detection, it can be used indirectly. Nmap can perform a quick network scan to identify active devices and their MAC addresses. You can then compare this information with the ARP table on your machine (using arp -a on Linux/macOS) to identify any discrepancies.

For example, if Nmap identifies a device with a specific IP address but the ARP table shows a different MAC address associated with that IP, it might indicate ARP spoofing. However, this method can be unreliable as legitimate network configurations can also cause MAC address changes.

2. Arpwatch:

Arpwatch is a dedicated tool for monitoring ARP activity on your network. It keeps track of learned MAC addresses for IPs and monitors for any changes. Here's how it helps detect ARP spoofing:

Database: Arpwatch maintains a database of learned IP/MAC mappings.

Monitoring: It continuously monitors ARP packets on the network.

Alerting: If Arpwatch detects an unsolicited ARP reply (attacker trying to modify the ARP table) or a change in the MAC address associated with a known IP, it raises an alert in the system logs.

3. Wireshark:

Wireshark is a powerful network packet analyzer. While not solely for ARP spoofing detection, it can be used for in-depth analysis of network traffic. Here's how it helps:

Packet Capture: Wireshark can capture live network traffic.

Filtering: You can filter captured packets to focus specifically on ARP traffic.

Analysis: By examining ARP packets, you can identify inconsistencies. For instance, if you see multiple ARP replies for the same IP address with different MAC addresses, it might indicate ARP spoofing.

Implementation :

```
ubuntu@ubuntu:~$ nmap -sn 192.168.1.0/24
Starting Nmap 7.94 ( https://nmap.org ) at 2024-04-19 00:28 UTC
Nmap scan report for www.routerlogin.com (192.168.1.1)
Host is up (0.0045s latency).
Nmap scan report for ubuntu (192.168.1.9)
Host is up (0.000095s latency).
Nmap done: 256 IP addresses (2 hosts up) scanned in 7.43 seconds
```

```
ubuntu@ubuntu:~$ nmap -p 22,53,80 scanme.nmap.org
Starting Nmap 7.94 ( https://nmap.org ) at 2024-04-19 00:26 UTC
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.39s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2
f

PORT   STATE  SERVICE
22/tcp open   ssh
53/tcp closed domain
80/tcp open   http
```

Conclusion :

The aim of detecting ARP spoofing using nmap, ARPWATCH, and Wireshark was achieved through comprehensive understanding and practical application of ARP spoofing detection

techniques. By meeting the objectives of understanding ARP spoofing and utilizing ARPWATCH, participants gained valuable insights into identifying and mitigating this common network attack. Nmap provided network scanning capabilities to detect unusual ARP traffic patterns, while ARPWATCH served as a dedicated tool for monitoring ARP activity and detecting spoofed ARP packets. Wireshark complemented these tools by enabling detailed packet analysis, facilitating the identification of ARP spoofing attacks through abnormal network behavior.