ARP Poisoning Attack with Man In the Middle Attack

Tusher Bhomik (2005046) Sheikh Rahat Mahmud (2005048)

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)

Network Topology

The attack was conducted in a VirtualBox Host-Only Network environment with the following configuration:

Machine	**IP Address**	**Role**
Client	192.168.56.200	Victim machine
Server	192.168.56.250	Target server
Attacker	192.168.56.150	Attack machine
Gateway	192.168.56.1	Network gateway

Table 1: Network Configuration

Network Topology

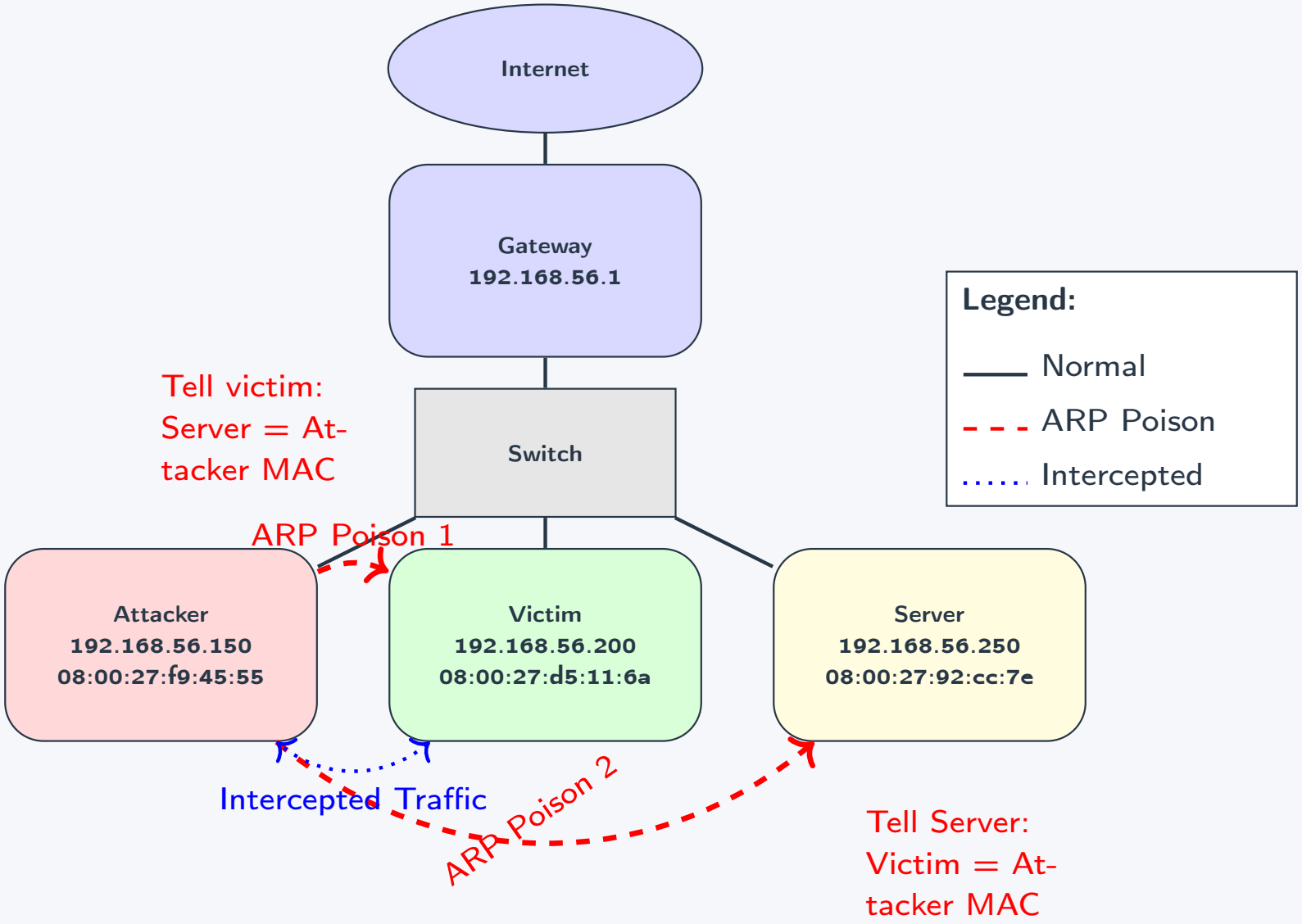

Figure 1: ARP Cache Poisoning Attack Network Topology

Client IP: 192.168.56.200, Server IP: 192.168.56.250

Our Interface: enp0s3

```
client@client-VirtualBox:~/Desktop$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFA
    link/ether 08:00:27:d5:11:6a brd ff:ff:ff:ff:ff:ff
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFA
    link/ether 08:00:27:1f:27:31 brd ff:ff:ff:ff:ff:ff
```

Figure 2: Client MAC Address: 08:00:27:d5:11:6a

```
server@server-VirtualBox:~/Desktop$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT gro
up default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode D
EFAULT group default qlen 1000
    link/ether 08:00:27:92:cc:7e brd ff:ff:ff:ff:ff:ff
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode D
EFAULT group default qlen 1000
    link/ether 08:00:27:10:60:d0 brd ff:ff:ff:ff:ff:ff
```

Figure 3: Server MAC Address: 08:00:27:92:cc:7e

Attacker IP: 192.168.56.150

Our Interface: enp0s3

```
client@client-VirtualBox:~/Desktop$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFA
    link/ether 08:00:27:d5:11:6a brd ff:ff:ff:ff:ff:ff
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFA
    link/ether 08:00:27:1f:27:31 brd ff:ff:ff:ff:ff:ff
```

Figure 4: Attacker MAC Address: 08:00:27:f9:45:55

Before Attack

```
client@client-VirtualBox:~/Desktop$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.3.2                 ether   52:55:0a:00:03:02   C                     enp0s8
192.168.56.250           ether   08:00:27:92:cc:7e   C                     enp0s3
192.168.56.1             ether   0a:00:27:00:00:13   C                     enp0s3
10.0.3.3                 ether   52:55:0a:00:03:03   C                     enp0s8
```

Figure 5: ARP Table of Client

```
server@server-VirtualBox:~/Desktop$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.56.200           ether   08:00:27:d5:11:6a   C                     enp0s3
10.0.3.2                 ether   52:55:0a:00:03:02   C                     enp0s8
```

Figure 6: ARP Table of Server

ARP Attack Started

```
attacker@attacker-VirtualBox:~/Downloads$ sudo python3 simple_arp_attack.py
[+] Scapy library detected - using enhanced mode
======================================================================
Enhanced ARP Cache Poisoning Attack - CSE406 Project
Mode: Scapy Enhanced
======================================================================
[+] Starting traffic monitoring...
```

Figure 7: simple_arp_attack.py

```
[+] Sending initial rapid poison burst...
[+] Sending rapid poison burst (10 packets)...
[+] Rapid poison packet 1/10 sent
[+] Rapid poison packet 2/10 sent
[+] Rapid poison packet 3/10 sent
[+] Rapid poison packet 4/10 sent
[+] Rapid poison packet 5/10 sent
[+] Rapid poison packet 6/10 sent
```

Figure 8: Attack Ongoing

Attack Continues: ARP Table Poisoned

```
client@client-VirtualBox:~/Desktop$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.3.2                 ether   52:55:0a:00:03:02   C                     enp0s8
192.168.56.250           ether   08:00:27:f9:45:55   C                     enp0s3
192.168.56.1             ether   0a:00:27:00:00:13   C                     enp0s3
192.168.56.150           ether   08:00:27:f9:45:55   C                     enp0s3
10.0.3.3                 ether   52:55:0a:00:03:03   C                     enp0s8
```

Figure 9: Client's ARP Table Poisoned After ARP Attack

```
server@server-VirtualBox:~/Desktop$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.56.200           ether   08:00:27:f9:45:55   C                     enp0s3
192.168.56.1             ether   0a:00:27:00:00:13   C                     enp0s3
192.168.56.150           ether   08:00:27:f9:45:55   C                     enp0s3
10.0.3.3                 ether   52:55:0a:00:03:03   C                     enp0s8
10.0.3.2                 ether   52:55:0a:00:03:02   C                     enp0s8
```

Figure 10: Server's ARP Table Poisoned After ARP Attack

Attack Effectiveness Metrics

Metric	Result
ARP Poisoning Success Rate	100%
Traffic Interception	Successful
Attack Duration	300 seconds (continuous)
Poison Packets Sent	100+ packets
Detection by Targets	None
Network Disruption	Minimal

Table 2: Attack Effectiveness Summary

Defense: Static ARP Tables

Static ARP entries prevent ARP cache poisoning by locking IP-MAC mappings, ignoring spoofed ARP replies from the attacker.

Implementation Script:

```bash
#!/bin/bash
# arp_defense.sh - Static ARP defense for CSE406
echo "[*] Starting Static ARP Defense - CSE406 Project"
echo "========================================"
CLIENT_IP="192.168.56.200"
SERVER_IP="192.168.56.250"
get_mac() {
    local ip=$1
    echo "[*] Discovering MAC for $ip..."
    if ! ping -c 3 -W 1 "$ip" >/dev/null 2>&1; then
        echo "[-] Cannot ping $ip. Ensure target is up."
        return 1
    fi
    mac=$(arp -n | grep "^$ip\s" | awk '{print $3}' | head -n 1)
    if [ -z "$mac" ]; then
        echo "[-] Failed to discover MAC for $ip. Retrying..."
        sleep 1
        ping -c 2 "$ip" >/dev/null 2>&1
        mac=$(arp -n | grep "^$ip\s" | awk '{print $3}' | head -n 1)
        if [ -z "$mac" ]; then
            echo "[-] MAC discovery failed for $ip."
            return 1
        fi
    fi
    echo "[+] MAC for $ip: $mac"
    echo "$mac"
    return 0
}
```

Defense Script (Continued)

```
set_static_arp() {
    local ip=$1
    local mac=$2
    echo "[*] Setting static ARP for $ip..."
    sudo arp -d "$ip" 2>/dev/null || true
    if sudo arp -s "$ip" "$mac" >/dev/null 2>&1; then
        echo "[+] Static ARP set: $ip -> $mac"
        return 0
    else
        echo "[-] Failed to set static ARP for $ip."
        return 1
    fi
}
verify_protection() {
    local ip=$1
    local mac=$2
    echo "[*] Verifying protection for $ip..."
    arp_entry=$(arp -n | grep "^$ip\s" | awk '{print $3, $5}')
    if [[ "$arp_entry" =~ $mac.*static ]]; then
        echo "[+] Verified: Static ARP entry preserved for $ip"
        arp -a | grep "$ip"
        return 0
    else
        echo "[-] Verification failed: No static entry for $ip."
        return 1
    fi
}

```

Defense Script (Final)

```
main() {
    if [ "$EUID" -ne 0 ]; then
        echo "[-] This script requires root privileges."
        exit 1
    fi
    if [ -z "$1" ]; then
        echo "Usage: sudo bash arp_defense.sh [client|server]"
        exit 1
    fi
    role=$1
    if [ "$role" = "client" ]; then
        target_ip=$CLIENT_IP
    elif [ "$role" = "server" ]; then
        target_ip=$SERVER_IP
    else
        echo "[-] Invalid role: Use 'client' or 'server'"
        exit 1
    fi
    if ! get_mac "$target_ip"; then
        echo "[-] Exiting due to MAC discovery failure."
        exit 1
    fi
    target_mac=$mac
    if ! set_static_arp "$target_ip" "$target_mac"; then
        echo "[-] Exiting due to static ARP failure."
        exit 1
    fi
    if ! verify_protection "$target_ip" "$target_mac"; then
        echo "[-] Exiting due to verification failure."
        exit 1
    fi
    echo "[+] Static ARP defense completed!"
    echo "[+] Current ARP table:"
    arp -a
}
```

Defense Effectiveness

```
client@client-VirtualBox:~/Desktop$ arp -a
? (192.168.56.250) at 08:00:27:92:cc:7e [ether] PERM on enp0s3
? (10.0.3.3) at 52:55:0a:00:03:03 [ether] on enp0s8
_gateway (192.168.56.1) at 0a:00:27:00:00:13 [ether] on enp0s3
_gateway (10.0.3.2) at 52:55:0a:00:03:02 [ether] on enp0s8
client@client-VirtualBox:~/Desktop$ arp -a
? (192.168.56.250) at 08:00:27:92:cc:7e [ether] PERM on enp0s3
? (10.0.3.3) at 52:55:0a:00:03:03 [ether] on enp0s8
_gateway (192.168.56.1) at 0a:00:27:00:00:13 [ether] on enp0s3
_gateway (10.0.3.2) at 52:55:0a:00:03:02 [ether] on enp0s8
? (192.168.56.150) at 08:00:27:f9:45:55 [ether] on enp0s3
client@client-VirtualBox:~/Desktop$
```

Figure 11: client ARP after defense

```
server@server-VirtualBox:~/Desktop$ arp -a
? (192.168.56.150) at 08:00:27:f9:45:55 [ether] on enp0s3
? (10.0.3.3) at 52:55:0a:00:03:03 [ether] on enp0s8
? (192.168.56.200) at 08:00:27:d5:11:6a [ether] PERM on enp0s3
_gateway (192.168.56.1) at 0a:00:27:00:00:13 [ether] on enp0s3
_gateway (10.0.3.2) at 52:55:0a:00:03:02 [ether] on enp0s8
server@server-VirtualBox:~/Desktop$
```

Defense Effectiveness Summary

Metric	**Result**
ARP Poisoning Prevention	100%
Traffic Interception Blocked	Successful
Setup Time	$<$ 10 seconds
MAC Discovery Errors	None
Scalability	Limited (Lab-only)
Network Overhead	Minimal

Table 3: Defense Effectiveness Summary