ARP Cache Poisoning Mitigation and Forensics Investigation

Heman Awang Mangut
ACE, UEL, United Kingdom
Email: u1345346@uel.ac.uk

Ameer Al-Nemrat
ACE, UEL, United Kingdom
Email: ameer@uel.ac.uk

Chafika Benzaïd
Dept. of Computer Science
USTHB, Algérie
Email: cbenzaid@usthb.dz

Abdel-Rahman H. Tawil
ACE, UEL, United Kingdom
Email: atawil@uel.ac.uk

***Abstract*—Address Resolution Protocol (ARP) cache spoofing or poisoning is an OSI layer 2 attack that exploits the statelessness vulnerability of the protocol to make network hosts susceptible to issues such as Man in the Middle attack, host impersonation, Denial of Service (DoS) and session hijacking.**

In this paper, a quantitative research approach is used to propose forensic tools for capturing evidences and mitigating ARP cache poisoning. The baseline approach is adopted to validate the proposed tools. The evidences captured before attack are compared against evidences captured when the network is under attack in order to ascertain the validity of the proposed tools in capturing ARP cache spoofing evidences.

To mitigate the ARP poisoning attack, the security features DHCP Snooping and Dynamic ARP Inspection (DAI) are enabled and configured on a Cisco switch. The experimentation results showed the effectiveness of the proposed mitigation technique.

***Keywords*-ARP cache poisoning; Forensic investigation; Attack mitigation.**

I. Introduction

Computer networks provide means through which users communicate, and make purchases through the Internet. Providing security to computer networks becomes imperative as the use of Internet in e-commerce and communication increases, proliferation of computer network threats is also spreading fast [1]. There are a variety of computer network threats operating at different layers of computer network [2]. These threats include DoS, DDoS, ARP cache spoofing, DNS poisoning, viruses etc. [3].

Controls needed for protecting computers from threats or attacks are dependent on decision variables that the management of an organization makes [4]. When a particular threat becomes a concern, controls needed for mitigating such such attacks should be be convergent [5]. Hence, to prevent ARP cache spoofing threats on the network, decision tools such as baseline approach or risk assessment approach need to be adopted to ascertain that all controls needed to mitigate such attacks as part of organization's Business Impact Analysis (BIA) are considered [6].

Some organizations make Business Impact Analysis (BIA) for controls to mitigate attacks on their information assets. Usually, they focus on external attacks coming from outside their network, while neglecting internal attacks launched from inside the network. In some other settings, an organization could have controls in place, however it may neglect the fact that a computer network is considered secure only if the vulnerability in it has not yet been discovered. Hence, this type of organizations may fail to put forensics tools in place against unknown attacks. It is important that they are able to study those attacks which could help them forestall feature occurrence and possibly identify source of attacks whether internal to the organization or external [7]. ARP cache spoofing is the commonest attack that can be launched from within a network and could have a very high destruction profile to an organization.

The purpose of this research work is to use baseline approach to validate the proposed forensics and mitigation approach against; (1) ARP cache spoofing, (2) Man in the Middle attack, (3) Host impersonation, (4) Denial of Service attack (DoS), (5) Session hijacking.

The rest of the paper is organized as follows. Section II gives an overview on attacks based on ARP cache poisoning. Section III summarizes related work in the literature. Section IV describes the investigation and mitigation methodology adopted. Section V presents the forensic investigation approach, followed by a detailed analysis of validation results. The mitigation approach is presented in Section VI. Finally, Section VII concludes the paper.

II. Overview of Internal Attacks based on ARP Cache Poisoning

A. Man in the Middle Attack

Man in the Middle (MitM) attack is a hacking methodology whereby an attacker poisoned the ARP caches of two communicating hosts to intercept their communication with the aim of causing host exploitation such as session hijacking, theft of sensitive data, port stealing and impersonation of login credential [8]. To launch the attack, the attacker first collect the MAC addresses of its victims by broadcasting an ARP request to the victims' entire network. After that, the attacker sends an ARP reply to the victim hosts in order to associate their IP addresses to its MAC address.

B. Denial of Service (DoS) Attack

According to [9], DoS is an exploitation of ARP cache poisoning by an attacker to identify itself as the default gateway to the victim host. Thus, all traffic sent to the gateway will be redirected to the attacker which will choose to drop. The attacker can also give the victim host a fake default gateway which does not exist on the network. Hence, the victim host will lose connection to the network.

C. Host Impersonation

An adversary can exploit ARP cache poisoning to impersonate another host and gain access to sensitive information sent to this host [10].

D. Session Hijacking

Session hijacking consists in taking over an active communication session of a legitimate user (i.e., victim) once it has authenticated to a server [11].

III. Related Work

Gouda and Huang [12] proposed a mitigation technique for ARP cache poisoning using invite-accept protocol and request-reply protocol such that host computers on the network could have their IP-MAC address binding stored on a database of a centralized server on the network. A host computer uses the protocols to get any host it needs to communicate with on the network. However, [13] argued that the implementation of this technique requires the alteration of ARP protocol of every. Moreover, the server could be a single point of failure and proposed mitigation technique is prune to DoS attack.

A Spoofing Prevention Method (SPM) is proposed in [14], where a distinctive key is associated with the source and destination domains. The key is appended into a packet at the source end and verified at the destination end.

[15] proposed the use of Hash Message Authentication Code (HMAC) to check the integrity and authenticity of user cookies. HMAC is computed on four concatenated variables: *username*, *expiration time*, *data*, and *session key*. The cookies are stored on the server side which complicates the possibility of session hijacking to an attacker. However, the proposed protocol is not scalable due to its reliance on SSL. In a related development, [16] proposed SessionLock; a mitigation technique against session hijacking. To this end, SessionLock relies on a session secret to generate unique authentication tokens and uses HMAC. Also, [17] proposed SessionShield to prevent session hijacking. SessionShield is a proxy outside of the browser that inspects all outgoing requests and incoming responses to identify session cookies, and then blocks script access to these session cookies. Nevertheless, the use of a proxy technique poses a scalability and compatibility issues [18].

Although efforts have been made to provide mitigation techniques to ARP cache spoofing, each proposition has its weaknesses which makes the quest for an ARP cache spoofing mitigation lingering. Besides, none of the propositions presented above have addressed the attack issue from forensics point of view. Whenever there is an attack on a network, there is always a need to provide evidence which is conform to forensics investigation standard. This allow to infer that the attack has really happened and to be able to identify the source of the attack so that it could be used against the suspect in the law court [19].

IV. The Proposed Investigation and Mitigation Methodology

In this work, a qualitative approach is used to collect data. A baseline approach is adopted to verify whether the network is under attack or not using several forensic tools, namely: TCPdump [20], Wireshark [21], and Linux commands TCPstat and Ntop. A baseline is a "value or profile of a performance metric against which changes in the performance metric can be usefully compared" [6]. In our case, the evidences captured before attack are compared against evidences captured when the network is under attack.

To mitigate the ARP cache poisoning attack, DHCP snooping and dynamic ARP inspection (DAI), two port security features, are enabled and configured on the switch. Those security features are available on the latest Cisco Catalyst switches, including 6500, 4500 and 2960 Series. The Cisco Catalyst 2960 Series switch is a reasonable choice for Small and Medium Enterprises (SMEs) due to its affordable price. To validate the proposed mitigation technique, an experiment network is setup as shown in Figure 1.

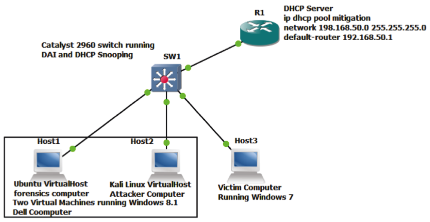


Figure 1. Topology diagram of the network

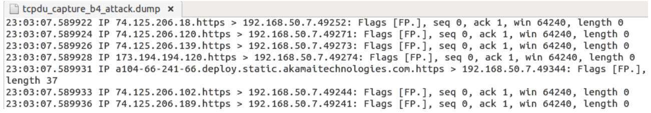
The network comprises a Cisco router 1841, a Cisco Catalyst 2960 switch, a victim machine running Windows 7, and a laptop containing two virtual machines (VMs) running Linux family Operating Systems (OS) on Windows 8.1 professional. The VM running Kali Linux serves as the attacker host. Two Ethernet-to-USB adapters are used to connect each of the VMs running on the laptop to the external DHCP in order to ensure that all hosts are connected to the same network. The network 192.168.50.0/24 is used with the default gateway 192.168.50.1 configured on Fast Ethernet 0. DHCP is enabled on the router. VLAN 50 is configured on the switch. Fast Ethernet 2, 3, 5, 7 and Gigabit Ethernet 1 interfaces are assigned to VLAN 50. All devices are put under VLAN 50 to avoid possibility of interference on the result as a requirement for a valid research design.

To validate the proposed forensics investigation of ARP cache spoofing, comparison will be made between the forensics evidences collected when there is no attack on the network against those collected when the network is under attack. Ettercap is used to launch the ARP cache poisoning and the corresponding attacks: MitM, DoS, session hijacking, and host impersonation. To validate the proposed mitigation technique, the evidences collected when the network is under attack with DHCP snooping and DAI enabled will be compared to those when the two security features are disabled.

V. Forensics Investigation of ARP Cache Poisoning

A. *Normal Traffic Flow on the Network*

The first evidences were captured when the network was in normal operation (i.e., no attack) using TCPdump, Wireshark, TCPstat and Ntop. The extracted evidences will be used as a baseline to ascertain whether the network is under attack or not.

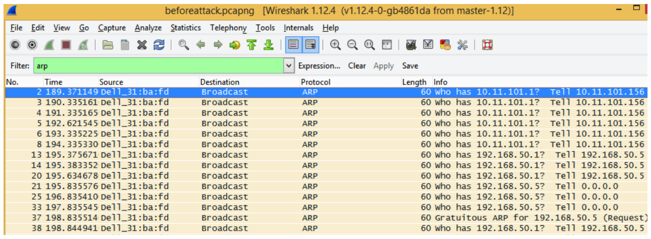


```
tcpdu_capture_b4_attack.dump
23:03:07.589922 IP 74.125.206.18.https > 192.168.50.7.49252: Flags [FP.], seq 0, ack 1, win 64240, length 0
23:03:07.589924 IP 74.125.206.120.https > 192.168.50.7.49271: Flags [FP.], seq 0, ack 1, win 64240, length 0
23:03:07.589926 IP 74.125.206.139.https > 192.168.50.7.49273: Flags [FP.], seq 0, ack 1, win 64240, length 0
23:03:07.589928 IP 173.194.194.120.https > 192.168.50.7.49274: Flags [FP.], seq 0, ack 1, win 64240, length 0
23:03:07.589931 IP a104-66-241-66.deploy.static.akamaitechnologies.com.https > 192.168.50.7.49344: Flags [FP.],
length 37
23:03:07.589933 IP 74.125.206.102.https > 192.168.50.7.49244: Flags [FP.], seq 0, ack 1, win 64240, length 0
23:03:07.589936 IP 74.125.206.189.https > 192.168.50.7.49241: Flags [FP.], seq 0, ack 1, win 64240, length 0
```

Figure 2. TCPdump output when the network traffic was normal

Figure 2 shows the packets collected with TCPdump. The TCPdump output comprises eleven fields that represent, respectively: the timestamp, the protocol, the source IP address and port, the destination IP address and port, the TCP flags, the relative sequence number, the relative acknowledgement number, the window size, and the packet length. For instance, the first line says that an IP packet was sent at timestamp $23:03:07$ from source port $https$ on the source host 74.125.206.18 to destination port 49252 on the destination host (in our case, the victim machine) 192.168.50.7. The https denotes that response from source host was over a Secured Socket Layer (SSL/TLS). The TCP flags F and P are set in the packet. The FIN flag, F, indicates the sender's intention to terminate the connection to the receiving host. The PUSH flag, P, signals the immediate push of data from the source host to the destination host. The packet's relative sequence number was 0 (seq 0) and it contained no data (length 0). A sequence number of zero indicates that this is the first packet sent by the source in this TCP session. A packet length of zero means that the packet contains only the header without payload. The available receive window was 64240 $bytes$.

The evidences captured by Wireshark are described in Figure 3. Only ARP traffic is displayed. The results show that the victim machine with MAC address $Dell\_31:ba:fd$ and IP address 192.168.50.5 made a broadcast asking for the MAC address of the default gateway with the IP address 192.168.50.1. It went further to making *gratuitous* ARP request of the existing default gateway. When a host's IP address changes, a gratuitous ARP packet is transmitted on the network to force any device that receives it to update its cache with the new IP-to-MAC address mapping.

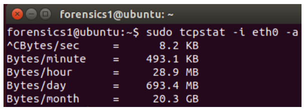


beforeattack.pcapng [Wireshark 1.12.4 (v1.12.4-0-gb4861da from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save

Filter: arp

```
No.   Time       Source         Destination   Protocol  Length Info
    2 189.371149 Dell_31:ba:fd  Broadcast     ARP       60 Who has 10.11.101.1?  Tell 10.11.101.156
    3 190.335161 Dell_31:ba:fd  Broadcast     ARP       60 Who has 10.11.101.1?  Tell 10.11.101.156
    4 191.335165 Dell_31:ba:fd  Broadcast     ARP       60 Who has 10.11.101.1?  Tell 10.11.101.156
    5 192.621545 Dell_31:ba:fd  Broadcast     ARP       60 Who has 10.11.101.1?  Tell 10.11.101.156
    6 193.335225 Dell_31:ba:fd  Broadcast     ARP       60 Who has 10.11.101.1?  Tell 10.11.101.156
    8 194.335330 Dell_31:ba:fd  Broadcast     ARP       60 Who has 10.11.101.1?  Tell 10.11.101.156
   13 195.375671 Dell_31:ba:fd  Broadcast     ARP       60 Who has 192.168.50.1?  Tell 192.168.50.5
   14 195.383352 Dell_31:ba:fd  Broadcast     ARP       60 Who has 192.168.50.1?  Tell 192.168.50.5
   20 195.634678 Dell_31:ba:fd  Broadcast     ARP       60 Who has 192.168.50.1?  Tell 192.168.50.5
   21 195.835576 Dell_31:ba:fd  Broadcast     ARP       60 Who has 192.168.50.5?  Tell 0.0.0.0
   25 196.835410 Dell_31:ba:fd  Broadcast     ARP       60 Who has 192.168.50.5?  Tell 0.0.0.0
   33 197.835545 Dell_31:ba:fd  Broadcast     ARP       60 Who has 192.168.50.5?  Tell 0.0.0.0
   37 198.835514 Dell_31:ba:fd  Broadcast     ARP       60 Gratuitous ARP for 192.168.50.5 (Request)
   38 198.844941 Dell_31:ba:fd  Broadcast     ARP       60 Who has 192.168.50.1?  Tell 192.168.50.5
```

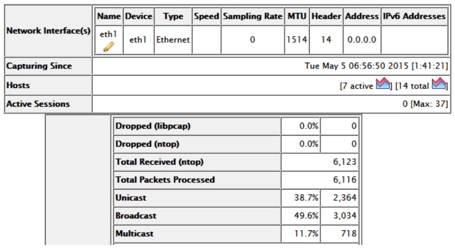
Figure 3. Wireshark evidence output when traffic was normal



```
forensics1@ubuntu: ~
forensics1@ubuntu:~$ sudo tcpstat -i eth0 -a
^CBytes/sec      =       8.2 KB
Bytes/minute     =     493.1 KB
Bytes/hour       =      28.9 MB
Bytes/day        =     693.4 MB
Bytes/month      =      20.3 GB
```

Figure 4. Output of evidence captured with TCPstate

TCPstat is a network probing application which functions as an IDS. The traffic statistics depicted in Figure 4 report the estimated number of bytes transmitted per second, minute, hour, day, and month.



	Name	Device	Type	Speed	Sampling Rate	MTU	Header	Address	IPv6 Addresses
Network Interface(s)	eth1	eth1	Ethernet		0	1514	14	0.0.0.0	
Capturing Since	Tue May 5 06:56:50 2015 [1:41:21]								
Hosts	[7 active] [14 total]								
Active Sessions	0 [Max: 37]								

Dropped (libpcap)	0.0%	0
Dropped (ntop)	0.0%	0
Total Received (ntop)	6,123	
Total Packets Processed	6,116	
Unicast	38.7%	2,364
Broadcast	49.6%	3,034
Multicast	11.7%	718

Figure 5. Percentage of Unicast, Broadcast and Multicast packets captured by Ntop on the network when there was no attack

Figure 5 shows the captured evidences using Ntop. It is observed that Ntop was activated on Ethernet port 1 (eth1) operating on a normal Maximum Transmission Unit (MTU) of 1514. The evidences were captured in 1 hour, 41 minutes, and 21 seconds. Neither libcap nor ntop have dropped any packet. Ntop received $6,123$ packets that were processed. The total number (resp. percentage) of unicast, broadcast, and multicast packets was $2,364$ (39.7%), $3,034$ (49.6%), and 718 (11.7%), respectively.

B. *Launching an ARP Poisoning Attack*

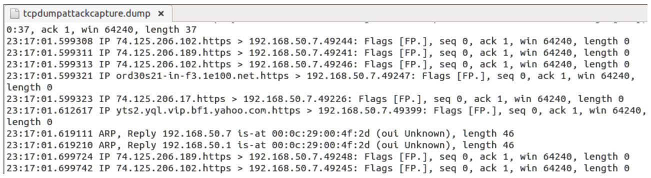
The purpose of launching attack is to find the performance profile of the attack which will be used to measure the mitigation capability of DAI and DHCP Snooping on ARP cache snooping.

Ettercap is used to launch the attack. A unified sniffing is started on the Ethernet interface connecting the attacker machine to the network. The interface is placed into promiscuous mode. The default gateway and the victim machine are selected as the first and the second target machine, respectively. A MitM ARP poisoning attack is then initiated to spoof the ARP tables of the target machines.

From this point, the plugins ”$repoison\_arp$” and ”$remote\_browser$” are selected alternately to launch a session hijacking attack.

C. Evidence Collected under the Attack

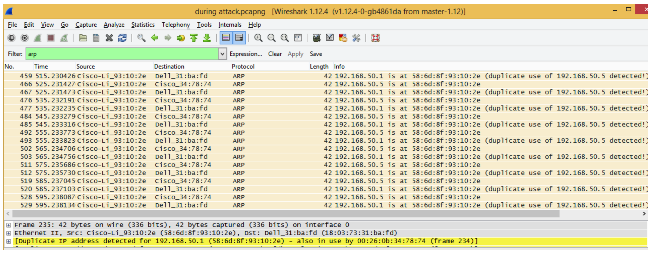
From Figure 6, it could be seen that the time stamp was 23 hours, 17 minutes, and 1 second of the day. The IP source was from the subnet 74.125.206.0 transmitting over SSL, and the destination IP address was 192.168.50.7. The sequence number was 0, the acknowledgement was 1, length mostly 0, and the TCP flag was [FP] meaning “finish push”. Although, there was ARP Reply in two places with the IP address of the default gateway and that of the victim machine. The length of ARP reply packet is 46 $bytes$.



```
tcpdumpattackcapture.dump ×
0:37, ack 1, win 64240, length 37
23:17:01.599308 IP 74.125.206.102.https > 192.168.50.7.49244: Flags [FP.], seq 0, ack 1, win 64240, length 0
23:17:01.599311 IP 74.125.206.189.https > 192.168.50.7.49241: Flags [FP.], seq 0, ack 1, win 64240, length 0
23:17:01.599313 IP 74.125.206.102.https > 192.168.50.7.49246: Flags [FP.], seq 0, ack 1, win 64240, length 0
23:17:01.599321 IP ord30s21-in-f3.1e100.net.https > 192.168.50.7.49247: Flags [FP.], seq 0, ack 1, win 64240,
length 0
23:17:01.599323 IP 74.125.206.17.https > 192.168.50.7.49226: Flags [FP.], seq 0, ack 1, win 64240, length 0
23:17:01.612617 IP yts2.yql.vip.bf1.yahoo.com.https > 192.168.50.7.49399: Flags [FP.], seq 0, ack 1, win 64240,
length 0
23:17:01.619111 ARP, Reply 192.168.50.7 is-at 00:0c:29:00:4f:2d (oui Unknown), length 46
23:17:01.619210 ARP, Reply 192.168.50.1 is-at 00:0c:29:00:4f:2d (oui Unknown), length 46
23:17:01.699724 IP 74.125.206.189.https > 192.168.50.7.49248: Flags [FP.], seq 0, ack 1, win 64240, length 0
23:17:01.699742 IP 74.125.206.102.https > 192.168.50.7.49245: Flags [FP.], seq 0, ack 1, win 64240, length 0
```

Figure 6. Evidence collected with TCPdumps when victim computer was under attack

In Figure 7, the evidence was collected using Wireshark. Note that there was a duplicate use of 192.168.50.5 which was the victim's IP address. Obviously, the source IP address was from a cisco device and the destination IP addresses comprise Dell and cisco devices.


during attack.pcapng [Wireshark 1.12.4 (v1.12.4-0-gb4861da from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save

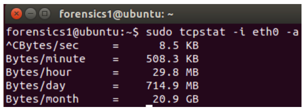
Filter: arp

```
No.   Time       Source             Destination     Protocol  Length Info
  459 515.230426 Cisco-Li_93:10:2e  Dell_31:ba:fd   ARP           42 192.168.50.1 is at 58:6d:8f:93:10:2e (duplicate use of 192.168.50.5 detected!)
  466 525.231427 Cisco-Li_93:10:2e  Cisco_34:78:74  ARP           42 192.168.50.5 is at 58:6d:8f:93:10:2e
  467 525.231473 Cisco-Li_93:10:2e  Dell_31:ba:fd   ARP           42 192.168.50.1 is at 58:6d:8f:93:10:2e (duplicate use of 192.168.50.5 detected!)
  476 535.232191 Cisco-Li_93:10:2e  Cisco_34:78:74  ARP           42 192.168.50.5 is at 58:6d:8f:93:10:2e
  477 535.232235 Cisco-Li_93:10:2e  Dell_31:ba:fd   ARP           42 192.168.50.1 is at 58:6d:8f:93:10:2e (duplicate use of 192.168.50.5 detected!)
  484 545.233279 Cisco-Li_93:10:2e  Cisco_34:78:74  ARP           42 192.168.50.5 is at 58:6d:8f:93:10:2e
  485 545.233316 Cisco-Li_93:10:2e  Dell_31:ba:fd   ARP           42 192.168.50.1 is at 58:6d:8f:93:10:2e (duplicate use of 192.168.50.5 detected!)
  492 555.233773 Cisco-Li_93:10:2e  Cisco_34:78:74  ARP           42 192.168.50.5 is at 58:6d:8f:93:10:2e
  493 555.233823 Cisco-Li_93:10:2e  Dell_31:ba:fd   ARP           42 192.168.50.1 is at 58:6d:8f:93:10:2e (duplicate use of 192.168.50.5 detected!)
  502 565.234706 Cisco-Li_93:10:2e  Cisco_34:78:74  ARP           42 192.168.50.5 is at 58:6d:8f:93:10:2e
  503 565.234756 Cisco-Li_93:10:2e  Dell_31:ba:fd   ARP           42 192.168.50.1 is at 58:6d:8f:93:10:2e (duplicate use of 192.168.50.5 detected!)
  511 575.235686 Cisco-Li_93:10:2e  Cisco_34:78:74  ARP           42 192.168.50.5 is at 58:6d:8f:93:10:2e
  512 575.235730 Cisco-Li_93:10:2e  Dell_31:ba:fd   ARP           42 192.168.50.1 is at 58:6d:8f:93:10:2e (duplicate use of 192.168.50.5 detected!)
  519 585.237045 Cisco-Li_93:10:2e  Cisco_34:78:74  ARP           42 192.168.50.5 is at 58:6d:8f:93:10:2e
  520 585.237103 Cisco-Li_93:10:2e  Dell_31:ba:fd   ARP           42 192.168.50.1 is at 58:6d:8f:93:10:2e (duplicate use of 192.168.50.5 detected!)
  528 595.238087 Cisco-Li_93:10:2e  Cisco_34:78:74  ARP           42 192.168.50.5 is at 58:6d:8f:93:10:2e
  529 595.238134 Cisco-Li_93:10:2e  Dell_31:ba:fd   ARP           42 192.168.50.1 is at 58:6d:8f:93:10:2e (duplicate use of 192.168.50.5 detected!)

⊞ Frame 235: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
⊞ Ethernet II, Src: Cisco-Li_93:10:2e (58:6d:8f:93:10:2e), Dst: Dell_31:ba:fd (18:03:73:31:ba:fd)
⊞ [Duplicate IP address detected for 192.168.50.1 (58:6d:8f:93:10:2e) - also in use by 00:26:0b:34:78:74 (frame 234)]
```

Figure 7. Evidence collected with Wireshark when victim machine was under attack

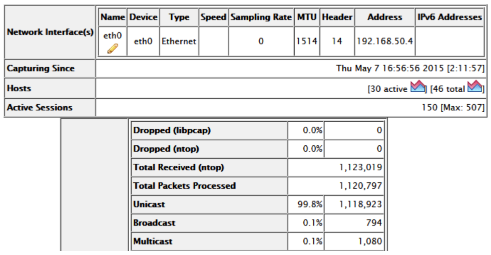
From the statistics generated by TCPstat in Figure 8, the flow rate of data on the network was $8.5KB/Sec$.



```
forensics1@ubuntu:~$ sudo tcpstat -i eth0 -a
^CBytes/sec      =      8.5 KB
Bytes/minute     =    508.3 KB
Bytes/hour       =     29.8 MB
Bytes/day        =    714.9 MB
Bytes/month      =     20.9 GB
```

Figure 8. Evidence collected with TCPstat when victim machine was under attack

From the Ntop statistics shown in Figure 9, it could be seen that $eth0$ was activated for data capture and it was operating on normal MTU of 1541. The IP address was 192.168.50.4. The duration of data capture was 2 hours, 11 minutes, and 57 seconds. There was no packet dropped neither by libpcap nor by ntop. However, there was $1,123,019$ packets received by Ntop and only $1,120,797$ packets were processed. The total number (resp. percentage) of unicast, broadcast, and multicast packets was $1,118,923$ (99.8%), 794 (0.1%), and $1,080$ (0.1%), respectively.



	Name	Device	Type	Speed	Sampling Rate	MTU	Header	Address	IPv6 Addresses
Network Interface(s)	eth0	eth0	Ethernet		0	1514	14	192.168.50.4	
Capturing Since	Thu May 7 16:56:56 2015 [2:11:57]								
Hosts	[30 active] [46 total]								
Active Sessions	150 [Max: 507]								

Dropped (libpcap)	0.0%	0
Dropped (ntop)	0.0%	0
Total Received (ntop)	1,123,019	
Total Packets Processed	1,120,797	
Unicast	99.8%	1,118,923
Broadcast	0.1%	794
Multicast	0.1%	1,080

Figure 9. Evidence collected with Ntop when victim computer was under attack

D. Discussion

1) Analysis of Evidence Collected with Tcpdump:

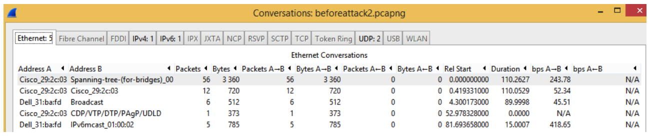
After comparing the timestamps of the captured evidences with TCPdump when no attack was launched (timestamp was $23h3mn7s$) and when the network was under attack (timestamp was $23h17mn1s$), it could be seen that the evidence of a normal traffic flow was captured before launching the attack on the victim machine.

The IP address used by the source machine transmitted from subnet 74.125.206.0 over the Internet was on HTTPS in both cases. The source machine does not seem to maintain a unique IP address; its IP address appeared to be changing in fraction of a second. The victim computer's IP address the suffixed port number remained the same. It is also observed that the TCP flags, the packet sequence and acknowledgement numbers, the packets' length, and the window size remained the same in both cases. However, line 7 of evidence collected when the attack was launched shows that there was an ARP reply claiming to have originated from the second target (i.e., victim machine) with IP address 192.168.50.7 but with MAC address of $00 : 0C : 29 : 00 : 4f : 2d$ which was the MAC address of the attacker machine. In the same way, line 8 shows another ARP reply claiming to have originated from the first target (i.e., default gateway) with IP address 192.168.50.1 but also with the attacker's MAC address $00 : 0C : 29 : 00 : 4f : 2d$. Hence, something suspicious must be going on here. This demonstrate that the MitM by ARP poisoning attack triggered by ettercap was successfully launched causing the ARP cache poisoning for both targets.

2) Analysis of Evidences Collected with Wireshark:

From the conversation depicted in Figure 10, it could be seen that the victim machine was making ARP broadcast asking for the default gateway. It is also observed that the victim machine was a dell product from the address column of the statistics table with shorten form of MAC address 31:ba:fd and the cisco switch too having shorten form MAC address of 29:2c:03.

Recall that the victim machine has made a *gratuitous* ARP request of the existing default gateway. The Figure 11 shows

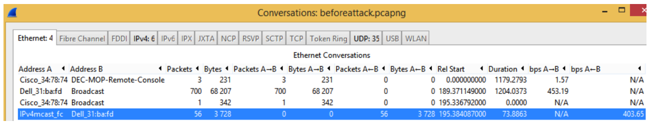


Ethernet Conversations

Address A	Address B	Packets	Bytes	Packets A→B	Bytes A→B	Packets A←B	Bytes A←B	Rel Start	Duration	bps A→B	bps A←B
Cisco_29:2c:03	Spanning-tree-(for-bridges)_00	56	3 360	56	3 360	0	0	0.000000000	110.2627	243.78	N/A
Cisco_29:2c:03	Cisco_29:2c:03	12	720	12	720	0	0	0.419331000	110.0529	52.34	N/A
Dell_31:ba:fd	Broadcast	6	512	6	512	0	0	4.300173000	89.9998	45.51	N/A
Cisco_29:2c:03	CDP/VTP/DTP/PAgP/UDLD	1	373	1	373	0	0	52.978328000	0.0000	N/A	N/A
Dell_31:ba:fd	IPv6mcast_01:00:02	5	785	5	785	0	0	81.693658000	15.0007	418.65	N/A

Figure 10. Conversation between the switch and the victim machine

conversation between the switch and the victim machine at that time. The conversation infers that the switch had acknowledged the existence of the victim machine, added the MAC-IP address binding to its ARP cache, and made a broadcast for the default gateway too.

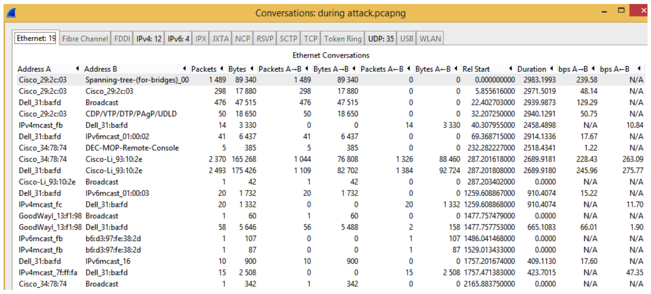


Ethernet Conversations

Address A	Address B	Packets	Bytes	Packets A→B	Bytes A→B	Packets A←B	Bytes A←B	Rel Start	Duration	bps A→B	bps A←B
Cisco_34:78:74	DEC-MOP-Remote-Console	3	231	3	231	0	0	0.000000000	1179.2793	1.57	N/A
Dell_31:ba:fd	Broadcast	700	68 207	700	68 207	0	0	189.371149000	1204.0373	453.19	N/A
Cisco_34:78:74	Broadcast	1	342	1	342	0	0	195.336792000	0.0000	N/A	N/A
IPv4mcast_fc	Dell_31:ba:fd	56	3 728	0	0	56	3 728	195.384087000	73.8863	N/A	403.65

Figure 11. Switch joined broadcast for the default gateway

From the evidence collected with Wireshark when the victim machine was under attack, we observed that from the "info" column it was reported "duplicate use of 192.168.50.5 detected". The Figure 12 shows statistics of conversation at the time. It could be seen that another MAC address (34 : 78 : 74) was captured by Wireshark and a duplication of IP address was reported as shown in Figure 13. It could be observed that the IP address 192.168.50.1 which was earlier known to be the IP address of the default gateway has two MAC address bindings: $58 : 6d : 8f : 93 : 10 : 2e$ which is the initial MAC address of default gateway and $00 : 26 : 0b : 34 : 78 : 74$ which is a fake MAC address of a host that does not exist on the network which was the outcome of the repoison plugins from ettercap. This effect should consequently result in DoS attack preventing access to the default gateway.



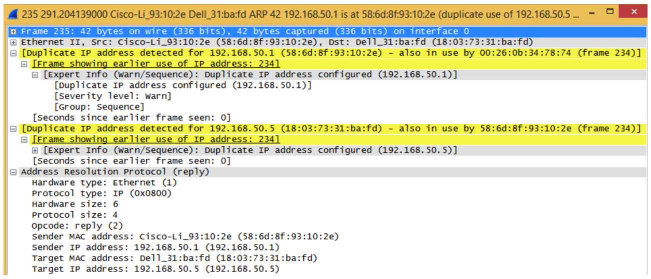
Ethernet Conversations

Address A	Address B	Packets	Bytes	Packets A→B	Bytes A→B	Packets A←B	Bytes A←B	Rel Start	Duration	bps A→B	bps A←B
Cisco_29:2c:03	Spanning-tree-(for-bridges)_00	1 489	89 340	1 489	89 340	0	0	0.000000000	2983.1993	239.58	N/A
Cisco_29:2c:03	Cisco_29:2c:03	298	17 880	298	17 880	0	0	5.855616000	2971.5019	48.14	N/A
Dell_31:ba:fd	Broadcast	476	47 515	476	47 515	0	0	22.402703000	2939.9873	129.29	N/A
Cisco_29:2c:03	CDP/VTP/DTP/PAgP/UDLD	50	18 650	50	18 650	0	0	32.207250000	2940.1291	50.75	N/A
IPv4mcast_fb	Dell_31:ba:fd	14	3 330	0	0	14	3 330	40.307955000	2458.4898	N/A	10.84
Dell_31:ba:fd	IPv6mcast_01:00:02	41	6 437	41	6 437	0	0	69.368715000	2914.1336	17.67	N/A
Cisco_34:78:74	DEC-MOP-Remote-Console	5	385	5	385	0	0	232.282227000	2518.4341	1.22	N/A
Cisco_34:78:74	Cisco-Li_93:10:2e	2 370	165 268	1 044	76 808	1 326	88 460	287.201618000	2689.9181	228.43	263.09
Dell_31:ba:fd	Cisco-Li_93:10:2e	2 493	175 426	1 109	82 702	1 384	92 724	287.201808000	2689.9180	245.96	275.77
Cisco-Li_93:10:2e	Broadcast	1	42	1	42	0	0	287.203402000	0.0000	N/A	N/A
Dell_31:ba:fd	IPv6mcast_01:00:03	20	1 732	20	1 732	0	0	1259.608867000	910.4074	15.22	N/A
IPv4mcast_fc	Dell_31:ba:fd	20	1 332	0	0	20	1 332	1259.608868000	910.4074	N/A	11.70
GoodWayl_13:f1:98	Broadcast	1	60	1	60	0	0	1477.757479000	0.0000	N/A	N/A
GoodWayl_13:f1:98	Dell_31:ba:fd	58	5 646	56	5 488	2	158	1477.757753000	665.1083	66.01	1.90
IPv6mcast_fb	b6:d3:97:fe:38:2d	1	107	0	0	1	107	1486.041468000	0.0000	N/A	N/A
IPv4mcast_fb	b6:d3:97:fe:38:2d	1	87	0	0	1	87	1529.013433000	0.0000	N/A	N/A
Dell_31:ba:fd	IPv6mcast_16	10	900	10	900	0	0	1757.201674000	409.1130	17.60	N/A
IPv4mcast_7f:ff:fa	Dell_31:ba:fd	15	2 508	0	0	15	2 508	1757.471383000	423.7015	N/A	47.35
Cisco_34:78:74	Broadcast	1	342	1	342	0	0	2165.883750000	0.0000	N/A	N/A

Figure 12. Conversation when victim computer was under attack

3) Analysis of TCPstat Collected Evidences: From the TCPstat statistics it could be obviously seen that the estimated traffic rate has increased when the network was under attack. This increase could be due to increase in ARP replies the repoison plugin of the ethercap was enabled.

4) Analysis of Evidences Captured with Ntop: It is obvious to see that the number of unicast during attack increased abruptly. This could be due to spoofed ARP reply from the attacker machine to the victim machine.


235 291.204139000 Cisco-Li_93:10:2e Dell_31:ba:fd ARP 42 192.168.50.1 is at 58:6d:8f:93:10:2e (duplicate use of 192.168.50.5 ...

```
⊞ Frame 235: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
⊞ Ethernet II, Src: Cisco-Li_93:10:2e (58:6d:8f:93:10:2e), Dst: Dell_31:ba:fd (18:03:73:31:ba:fd)
⊟ [Duplicate IP address detected for 192.168.50.1 (58:6d:8f:93:10:2e) - also in use by 00:26:0b:34:78:74 (frame 234)]
  ⊟ [Frame showing earlier use of IP address: 234]
    ⊟ [Expert Info (Warn/Sequence): Duplicate IP address configured (192.168.50.1)]
        [Duplicate IP address configured (192.168.50.1)]
        [Severity level: Warn]
        [Group: Sequence]
    [Seconds since earlier frame seen: 0]
⊟ [Duplicate IP address detected for 192.168.50.5 (18:03:73:31:ba:fd) - also in use by 58:6d:8f:93:10:2e (frame 234)]
  ⊟ [Frame showing earlier use of IP address: 234]
    ⊞ [Expert Info (Warn/Sequence): Duplicate IP address configured (192.168.50.5)]
    [Seconds since earlier frame seen: 0]
⊟ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: Cisco-Li_93:10:2e (58:6d:8f:93:10:2e)
    Sender IP address: 192.168.50.1 (192.168.50.1)
    Target MAC address: Dell_31:ba:fd (18:03:73:31:ba:fd)
    Target IP address: 192.168.50.5 (192.168.50.5)
```

Figure 13. Duplicate of IP address in MAC-IP address binding

VI. Mitigating ARP cache Spoofing

In order to mitigate the ARP cache spoofing, the security features DHCP Snooping and Dynamic ARP Inspection (DAI) were enabled on the switch. The DHCP Snooping protects the network by allowing the switch to accept DHCP response message only from the authorized servers connected to the trusted interfaces in the switch. DAI helps preventing ARP poisoning and other ARP-based attacks by intercepting and verifying the authenticity of any ARP or ICMP request/reply, and dropping any ARP spoofing that is beyond the rate-limit configured on untrusted ports (in our case, this is the port on which the attacker machine is connected).

In our case, the two features were enabled on VLAN 50 and the Gigabit Ethernet port 1 (gi1/1) of the switch was configured as a trusted port. The DAI logging buffer size is set to 1024.

A. Launching Attack Again

At this time both DHCP Snooping and DAI security features were activated and configured on the switch. Then the ARP cache spoofing was launched again with ettercap as described in subsection V-B.

B. Analysis of Mitigation of ARP Cache Spoofing

Obtained results showed that the ARP cache spoofing was prevented when DHCP snooping and DAI were enabled and configured on the switch. Although both $reposoning\_arp$ and $remote\_browser$ plugins were selected, session hijacking was not possible as when there was no mitigation. As the DHCP Snooping and DAI were configured to trust only ARP request/reply from the port connected to the DHCP router, the fake ARP request/reply may certainly be dropped by DAI. Note that we deliberately didn't configure the $rate\ limit$ of incoming ARP packets on untrusted ports in order to avoid putting the port in an $error\_disable$ state which could interfere with the size of data to be collected in the course of forensics investigation, if there could be any.

VII. Conclusion

The primary motivation of this research was to discuss the likelihood and to establish a baseline approach may used to mitigate and investigate ARP cache spoofing, Man in the Middle, host impersonation, Denial of Service, and session hijacking attacks. The result obtained in this research showed that mitigation of ARP cache spoofing using DHCP Snooping and DAI was achievable. This research also introduced TCPdump, wireshark, TCPstat, and Ntop as a useful tools and techniques to be used in forensics investigation of ARP cache spoofing attack.

References

[1] H.-W. Hsiao, C. Lin, and S.-Y. Chang, “Constructing an arp attack detection system with snmp traffic data mining,” *In Proc. of the 11th ACM Int. Conf. on Electronic Commerce (ICEC’09)*, pp. 341–345, 2009.

[2] D. Hill and J. Lynn, “Adaptive system and method for responding to computer network security attacks,” Jul. 11 2000, uS Patent 6,088,804. [Online]. Available: http://www.google.com/patents/US6088804

[3] N. Wattanapongsakorn, P. Sangkatsanee, S. Srakaew, and C. Charnsripinyo, “Classifying network attack types with machine learning approach,” *In Proc. of 7th IEEE Int. Conf. on Networked Computing (INC’11)*, pp. 98–102, Sept. 2011.

[4] H. Tipton, *Official (ISC)2 Guide to the CISSP CBK (2nd ed.)*. Auerbach Publications, 2009.

[5] N. Tripathi and B. Mehtre, “An icmp based secondary cache approach for the detection and prevention of arp poisoning,” *In Proc. of the 2013 IEEE Int. Conf. on Computational Intelligence and Computing Research (ICCIC’13)*, pp. 1–6, Dec. 2013.

[6] M. Whitman and H. Mattord, *Principles of Information Security (4th ed.)*. Course Technology Press, 2011.

[7] S. Gantz and D. Philpott, *FISMA and the Risk Management Framework: The New Practice of Federal Cyber Security (1st ed.)*. Syngress Publishing, 2012.

[8] G. Nath Nayak and S. Samaddar, “Different flavours of man-in-the-middle attack, consequences and feasible solutions,” *In Proc. of the 3rd IEEE Int. Conf. on Comput. Science and Inform. Technology (ICCSIT’10)*, vol. 5, pp. 491–495, Dec. 2010.

[9] M. Oh, Y.-G. Kim, S. Hong, and S. Cha, “Asa: Agent-based secure arp cache management,” *IET Commun*, vol. 6, no. 7, pp. 685–693, 2012.

[10] C. Abad and R. I. Bonilla, “An analysis on the schemes for detecting and preventing arp cache poisoning attacks,” *In Proc. of the 27th IEEE Int. Conf. on Distributed Computing Systems Workshops (ICDCSW ’07)*, pp. 60–, 2007.

[11] N. Nishanth, J. Zareena, and S. Babu, “Pseudo random alteration of sequence numbers (pras): A novel method for defending sessiion hijacking attack in mobile adhoc network,” *In Proc. of the 15th IEEE Int. Conf. on Communication Technology (ICCT’13)*, pp. 20–25, Nov. 2013.

[12] M. Gouda and C.-T. Huang, “A secure address resolution protocol,” *Comput. Netw.*, vol. 41, no. 1, pp. 57–71, Jan. 2003.

[13] G. Jinhua and X. Kejian, “Arp spoofing detection algorithm using icmp protocol,” *In Proc. of the IEEE Int. Conf. on Comput. Commun and Informatics (ICCCI’13)*, pp. 1–6, Jan. 2013.

[14] A. Bremler-Barr and H. Levy, “Spoofing prevention method,” *In Proc. of the 24th Annual Joint Conf. of the IEEE Comput. and Commun Societies (INFOCOM’05)*, vol. 1, pp. 536–547, Jan. 2005.

[15] A. Liu, J. Kovacs, C.-T. Huang, and M. Gouda, “A secure cookie protocol,” *In Proc. of the 14th IEEE Int. Conf. on Comput. Commun and Networks (ICCCN’05)*, pp. 333–338, Oct. 2005.

[16] B. Adida, “Sessionlock: securing web sessions against eavesdropping,” *In Proc. of the 17th ACM Int. Conf. on World Wide Web (WWW’08)*, pp. 517–524, 2008.

[17] N. Nikiforakis, W. Meert, Y. Younan, M. Johns, and W. Joosen, “Sessionshield: lightweight protection against session hijacking,” *In Proc. of the 3rd Int. Conf. on Engineering Secure Software and Systems (ESSoS’11)*, pp. 87–100, 2011.

[18] A. Alabrah and M. Bassiouni, “Preventing session hijacking in collaborative applications with hybrid cache-supported one-way hash chains,” *In Proc. of the Int. Conf. on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom’14)*, pp. 27–34, 2014.

[19] S. Rekhis and N. Boudriga, “A system for formal digital forensic investigation aware of anti-forensic attacks,” *IEEE Trans. on Inform. Forensics and Security*, vol. 7, no. 2, pp. 635–650, 2012.

[20] “Tcpdump,” http://www.tcpdump.org.

[21] “Wireshark,” https://www.wireshark.org.