Foundations of Computer Networks

PROJECT REPORT - ARP SPOOFING DETECTION ALGORITHM USING ICMP PROTOCOL - GROUP 07

ANIKET GIRIYALKAR | TANAY DUSANE | ASHISH PARALKAR

CSCI 651 | 04/20/2018

Table of Contents

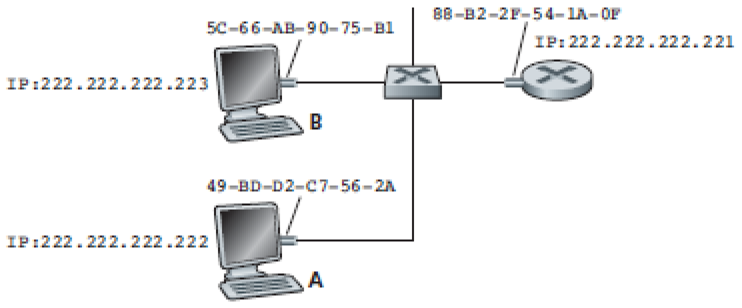
ABSTRACT

ARP spoofing is one of the easiest spoofing man-in-the-middle attacks in the local area network. In this project we implement the algorithm to detect ARP spoofing as proposed by Gao Jinhua and Xia Kejian in the paper "ARP Spoofing Detection Algorithm Using ICMP Protocol" which was published in the International Conference on Computer Communication and Informatics (ICCCI - 2013). In the technique implemented, we make use of the ICMP protocol and set up a detective host to detect malicious MITM hosts. We collect and analyze ARP response and request packets. Then we inject ICMP packets to find out if a suspicious host is actually malicious with IP forwarding enabled, malicious without IP forwarding enabled, or actually legitimate.

INTRODUCTION

What is an ARP?
ARP stands for address resolution protocol. Quoting from Computer Networks : A Top Down Approach: All hosts and routers that constitute an internal network make use of both link layer addresses and network layer addresses, for the purpose of having an identity inside that network. In practice it is the adapters of the routers and hosts that have the link layer and network layer addresses associated to them. The link layer address is also called the MAC address or the media access control access, or the Ethernet address. This is a 48-bit address, and an example MAC address looks like this: 1A:23:F9:CD:06:9B. No two adapters have the same MAC address.

Since in a LAN, both network addresses (which is IP addresses) and link layer addresses (which are MAC addresses) are used at each port of a host or router, there arises a need to keep a track of their mappings for every such host.



In the LAN shown above, each host has one IP and one MAC address. When host A (222.222.222.222) wants to send a packet to host B (222.222.222.223), A needs to have MAC address of B to send that packet. A sends a broadcast request packet asking for MAC address of B and includes IP of B in this request packet. All hosts in this network will receive it but upon finding mismatch of their IP and intended recipient's IP, they will all drop it, and only B will act upon receiving this. Thus, A gets back an ARP response packet which contains MAC of B. It uses this to send link layer packet with its own MAC as source and B's MAC as destination.

What is ICMP?

Hosts and routers in a LAN use ICMP to communicate network layer information with each other. ICMP is typically used in the network layer for communicating error messages. ICMP messages are encapsulated inside IP datagrams, similar to TCP or UDP. There are two fields in ICMP, type and code field. It also includes first 8 bytes of the IP packet that triggered the error in the first place so as to let the sender know which packet is the problematic one.

Following table shows some of the ICMP packet types.

ICMP Type	Code	Description
0	0	echo reply (to ping)
3	0	destination network unreachable
3	1	destination host unreachable
3	2	destination protocol unreachable
3	3	destination port unreachable
3	6	destination network unknown
3	7	destination host unknown
4	0	source quench (congestion control)
8	0	echo request
9	0	router advertisement
10	0	router discovery
11	0	TTL expired
12	0	IP header bad

BACKGROUND

What is ARP poisoning?
As we discussed above, when a host wants to send a packet to another host, it has to first find out its physical address. Only the intended recipient host sends a unicast message containing its MAC address to the original sender. The ARP module is the module that allows this process to occur, and its loopholes have been exploited for malicious purposes by simulating Man in the middle attacks. With an MITM attack, a malicious host pretends to be someone that he is not and sniffs all the packets by being an unintended recipient. In other words, he can sniff the traffic flowing between two legitimate hosts. This is a fairly easy to exploit loophole and we implement an idea to circumvent this loophole and keep the LAN free from these attacks.

As mentioned, every host maintains an ARP cache. The cache at each host is maintained by their own OS. New entries are created when new hosts are discovered and the need for communication between two hosts previously unknown to each other arises. ARP response packets are not authenticated, and hosts will cache ARP entries as they come. Entries remain in a host's cache till it reboots or resets. This is the reason it is easy to carry out ARP spoofing. Thus, in ARP spoofing, a host sends out a request or response packet with fake <ARP, MAC> pair.

If X and Y are two legitimate hosts with MAC addresses x and y respectively, and M is the malicious host, M could pretend to be a host with MAC x and intercept all messages from

Y that were intended for X. The attacker M could either drop the packets or further send them to X by once again forging Y's MAC address y.

RELATED WORKS

Apart from the method proposed by the authors which we implemented, few other techniques to mitigate the problem of ARP spoofing exist:

Secure ARP also known as S-ARP is a solution that makes use of end to end encryption for authenticating origin of ARP packets. It makes use of the Digital Signature Algorithm, which is an overhead on the LAN's resources.

Some high-end switches manufactured by Cisco include mitigation software as part of the Cisco IOS which prevent ARP spoofing.

Apart from this static ARP caches are a basic way to prevent ARP poisoning attacks. However, this alternative does not scale well.

THE PROPOSED ALGORITHM FOR DETECTING SPOOFING

In the following sections we describe the details of the algorithm proposed in the paper. Firstly, we construct an experiment simulation network, and we practice the proposed method based on this network.

EXPERIMENT SIMULATION NETWORK

This experiment requires the following components for simulating the experiment network:

- VMware WorkStation Ver 14.
- Installed three Ubuntu-16.04.4 on VMware.
- Live Packet Generated by each of the VMware machine.
- Program used to capture ARP and ICMP packets, analyze these packets using Wireshark (2.4.6) and send trap ICMP ping packets.
- PyCharm for the coding part
- Scapy package for the python program.

The figure 1 given below shows the topology of the experiment network. The suspicious host (IP: 192.186.83.182) sends ARP packets to victim host (IP: 192.186.83.180), the detection host (IP: 192.186.83.2) will capture the packets and send trap ICMP ping packets to the suspicious host and make a judgement on the suspicious host according to the responding packets.

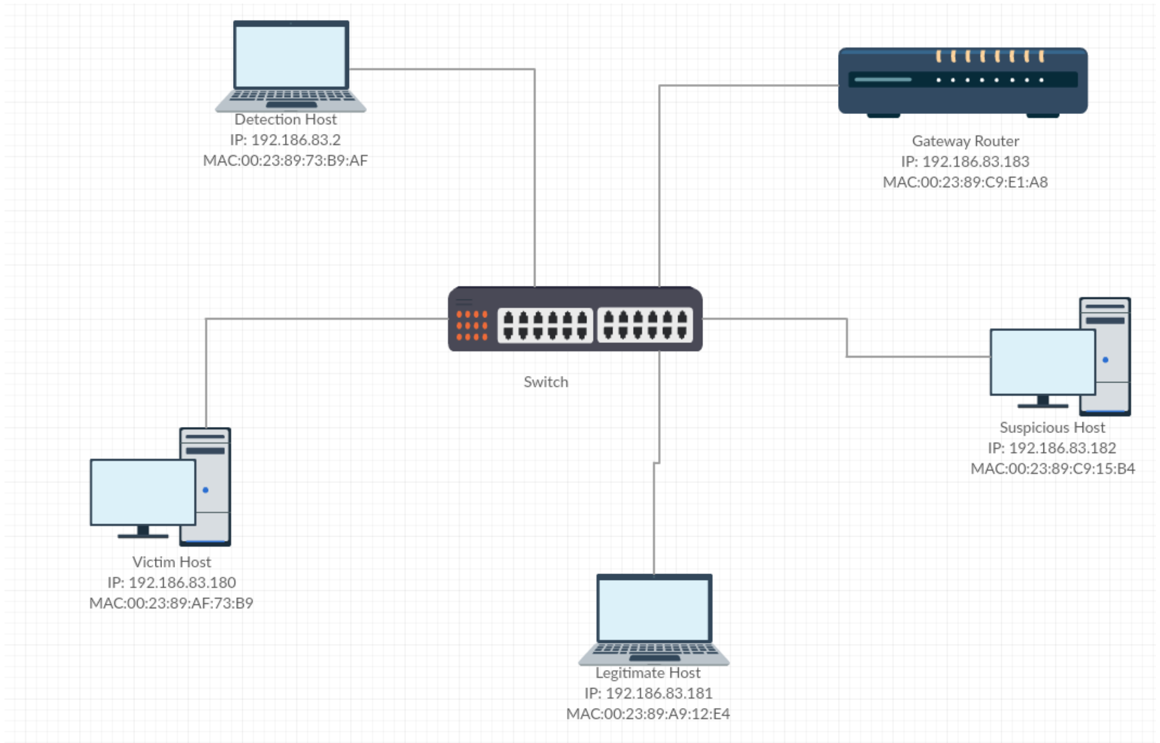


Fig. 1 The Experiment Simulation Network Structure

ARCHITECTURE

We adopt a modularized approach by dividing the ARP spoofing detection into the following modules:

- ARP Packet Sniffer Module: This module sniffs all ARP packets from the Ethernet packet using the command sniff (filter="arp", prn = ARPanalysis, iface="VMware Virtual Ethernet Adapter for VMnet8")
- Invalid Packet Detection Module: This module divides the ARP packets into valid and invalid packets in two steps. If we find any invalid packets, then it is guaranteed that an ARP attack is occurring. All the new IP-MAC mappings are sent to the ARP Spoof Detection Module.
- ARP Spoof Detection Module: This is the main module in we feed new valid packets into it as input.
- IP-MAC Mapping Database - IP-MAC mappings are proved to be valid will be added into our dictionary which contains all other IP-MAC mappings.
- Response Module: This module alerts the network administrator about the ARP spoofing attack taking place who will then decide what to do with the malicious

user (for example it can deny the hosts that are identified as malicious hosts by creating and sending ARP packets with random MAC address to the attacker).

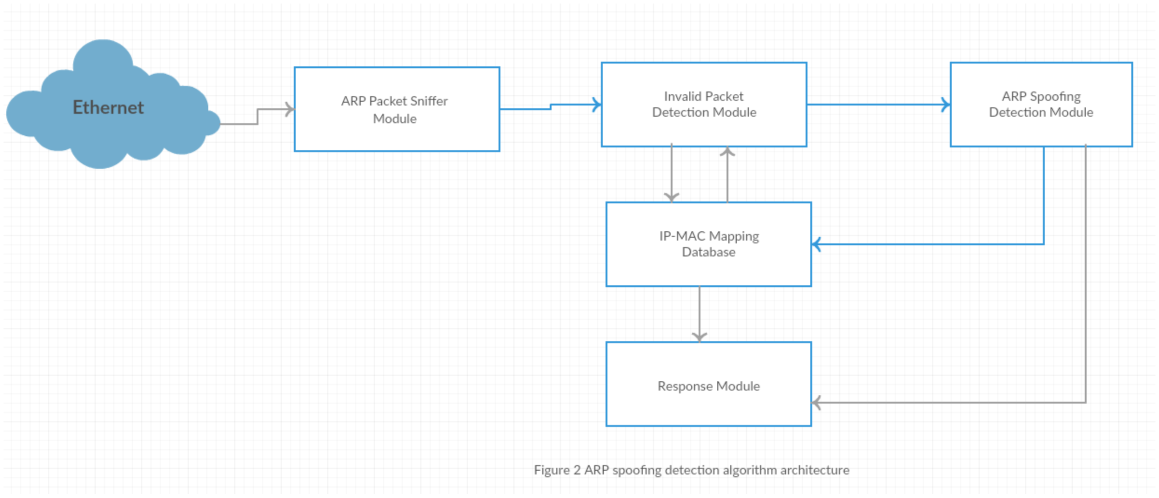

Ethernet
ARP Packet Sniffer Module
Invalid Packet Detection Module
ARP Spoofing Detection Module
IP-MAC Mapping Database
Response Module

Figure 2 ARP spoofing detection algorithm architecture

INVALID PACKET DETECTION MODULE

Packets that are obtained from the ARP Sniffer module will undergo two steps in this module and finally will be classified into two types: valid and invalid packets. Invalid packets with contradictions will be discarded and the program will be directed to the Response Module to alert the Administrator. Valid Packets that remain will then be sent to the ARP Spoofing Detection Module.

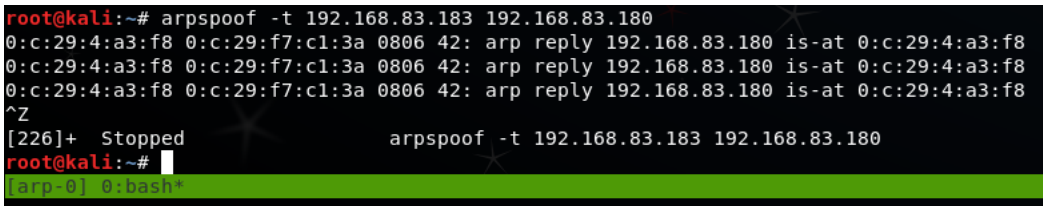
Step 1: Detection of Fake Packets with Conflicting Hardware Addresses

A cross-layer detection is used. If the source and/or destination MAC address do not match with those in the ARP header, then we can Say that the packet is invalid, and the ARP spoofing attacks are taking place.

Step 2: Detection of Fake Packet with Inconsistent IP-MAC Mappings:

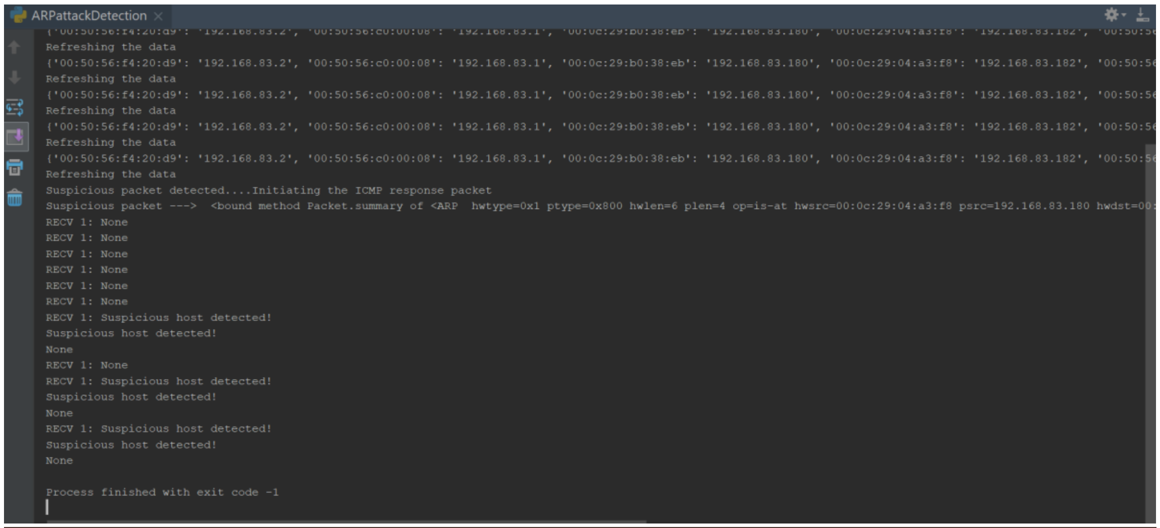
The packet obtained in the previous step is filtered in this step by comparing the IP-MAC mappings of the ARP packet with that in our database which contains the IP-MAC Mappings. All the packets who have identical mappings with the database mappings will be dropped. If any conflicts occur than it is assured that an ARP Spoofing attack is taking place, and this is forwarded to Response Module for help. The newly seen address pairs are passed on to the ARP Spoof Detection Module.

ARP Spoofing Detection Module



```
root@kali:~# arpspoof -t 192.168.83.183 192.168.83.180
0:c:29:4:a3:f8 0:c:29:f7:c1:3a 0806 42: arp reply 192.168.83.180 is-at 0:c:29:4:a3:f8
0:c:29:4:a3:f8 0:c:29:f7:c1:3a 0806 42: arp reply 192.168.83.180 is-at 0:c:29:4:a3:f8
0:c:29:4:a3:f8 0:c:29:f7:c1:3a 0806 42: arp reply 192.168.83.180 is-at 0:c:29:4:a3:f8
^Z
[226]+  Stopped                 arpspoof -t 192.168.83.183 192.168.83.180
root@kali:~# 
[arp-0] 0:bash*
```

Figure: Initiating the attack using ARP spoof tool.



```
{'00:50:56:f4:20:d9': '192.168.83.2', '00:50:56:c0:00:08': '192.168.83.1', '00:0c:29:b0:38:eb': '192.168.83.180', '00:0c:29:04:a3:f8': '192.168.83.182', '00:50:56
Refreshing the data
{'00:50:56:f4:20:d9': '192.168.83.2', '00:50:56:c0:00:08': '192.168.83.1', '00:0c:29:b0:38:eb': '192.168.83.180', '00:0c:29:04:a3:f8': '192.168.83.182', '00:50:56
Refreshing the data
{'00:50:56:f4:20:d9': '192.168.83.2', '00:50:56:c0:00:08': '192.168.83.1', '00:0c:29:b0:38:eb': '192.168.83.180', '00:0c:29:04:a3:f8': '192.168.83.182', '00:50:56
Refreshing the data
{'00:50:56:f4:20:d9': '192.168.83.2', '00:50:56:c0:00:08': '192.168.83.1', '00:0c:29:b0:38:eb': '192.168.83.180', '00:0c:29:04:a3:f8': '192.168.83.182', '00:50:56
Refreshing the data
{'00:50:56:f4:20:d9': '192.168.83.2', '00:50:56:c0:00:08': '192.168.83.1', '00:0c:29:b0:38:eb': '192.168.83.180', '00:0c:29:04:a3:f8': '192.168.83.182', '00:50:56
Refreshing the data
Suspicious packet detected....Initiating the ICMP response packet
Suspicious packet --->  <bound method Packet.summary of <ARP  hwtype=0x1 ptype=0x800 hwlen=6 plen=4 op=is-at hwsrc=00:0c:29:04:a3:f8 psrc=192.168.83.180 hwdst=00:
RECV 1: None
RECV 1: None
RECV 1: None
RECV 1: None
RECV 1: None
RECV 1: None
RECV 1: Suspicious host detected!
Suspicious host detected!
None
RECV 1: None
RECV 1: Suspicious host detected!
Suspicious host detected!
None
RECV 1: Suspicious host detected!
Suspicious host detected!
None

Process finished with exit code -1
```

Figure: Detection of suspicious host

This module is the heart of the whole architecture and it applies the detection algorithm to detect ARP Spoofing. This is based on the following two rules:

Rule 1: NIC of host only accepts packets when with its own hardware address(MAC), broadcast address, and subscribed multicast address. The network layer can only accept IP packets addressed to its IP address and it will drop the other packers silently.

Rule 2: Hosts with enabled IP packet routing will forward the packet to the destination host. All legitimate hosts present in the network who do not enable IP packet routing will respond after they receive an ICMP echo request packet.

We can classify the ARP packets as real or fake based on the above two rules. This can be divided in the following two steps:

Step 1: Generation of trap ICMP echo request packet

Ping packet is an ICMP packet usually used to detect the connectivity between two hosts. If a host A wants to ping another host B in a network, it will an ICMP echo and wait for a reply from B.
When the detection host gets and ARP packet with MAC address X and source IP address Y which does not have an entry in the mapping database, we will regard the source addresses <MAC address = Mac-X, IP address == IP-Y> as the address of the suspicious host. It will then construct a trap ICMP ping packet with them as the destination addresses.

Ethernet Header	
Destination MAC Address =	00:23:89:C9:15:B4
Source MAC Address =	00:23:89:73:B9:AF
Ethernet Type =	0x0800
IP Header	
Destination IP Address =	192.186.83.182
Source IP Address =	192.186.83.2
ICMP Header	
Type =	8(echo request)
Code =	0
Identifier =	-
Sequence Number =	-

Table: Trap ICMP Ping Packet

Step 2: Identification of the suspicious hosts

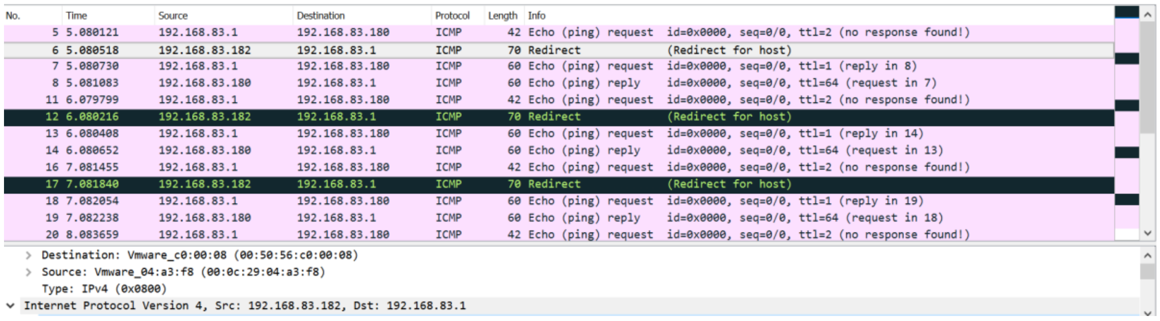
A suspicious host can be classified into 3 types:

1) Malicious host without enabled IP forwarding:

When the detection hosts find an <ARP, MAC> pair not already existing in the cache, it sends an ICMP packet to further check the legitimacy of this host. The ICMP packet has the destination MAC and destination IP set to whatever values were discovered for the new host in the previous step. The host will respond if and only if both MAC and IP match, else it will do nothing. This will prompt the detection host to timeout and hence realize that this is indeed a malicious host.

2) Malicious host with enabled IP forwarding.

As in the previous case, detection host sends out its ICMP packet with the <ARP, MAC> pair set in its Ethernet part and IP part respectively. A malicious host that has IP forwarding enabled will forward this packet to the victim host who IP is mentioned on the ICMP packet. And this victim host will in turn respond to detection host with an ICMP response packet using its own MAC. And when this ICMP response finally reaches the detection host, the host will realize that the destination MAC it had set and the MAC of the packet it received are different, and therefore detect the malicious host again.



```
No.   Time        Source           Destination      Protocol  Length  Info
    5 5.080121    192.168.83.1     192.168.83.180   ICMP        42 Echo (ping) request  id=0x0000, seq=0/0, ttl=2 (no response found!)
    6 5.080518    192.168.83.182   192.168.83.1     ICMP        70 Redirect             (Redirect for host)
    7 5.080730    192.168.83.1     192.168.83.180   ICMP        60 Echo (ping) request  id=0x0000, seq=0/0, ttl=1 (reply in 8)
    8 5.081083    192.168.83.180   192.168.83.1     ICMP        60 Echo (ping) reply    id=0x0000, seq=0/0, ttl=64 (request in 7)
   11 6.079799    192.168.83.1     192.168.83.180   ICMP        42 Echo (ping) request  id=0x0000, seq=0/0, ttl=2 (no response found!)
   12 6.080216    192.168.83.182   192.168.83.1     ICMP        70 Redirect             (Redirect for host)
   13 6.080408    192.168.83.1     192.168.83.180   ICMP        60 Echo (ping) request  id=0x0000, seq=0/0, ttl=1 (reply in 14)
   14 6.080652    192.168.83.180   192.168.83.1     ICMP        60 Echo (ping) reply    id=0x0000, seq=0/0, ttl=64 (request in 13)
   16 7.081455    192.168.83.1     192.168.83.180   ICMP        42 Echo (ping) request  id=0x0000, seq=0/0, ttl=2 (no response found!)
   17 7.081840    192.168.83.182   192.168.83.1     ICMP        70 Redirect             (Redirect for host)
   18 7.082054    192.168.83.1     192.168.83.180   ICMP        60 Echo (ping) request  id=0x0000, seq=0/0, ttl=1 (reply in 19)
   19 7.082238    192.168.83.180   192.168.83.1     ICMP        60 Echo (ping) reply    id=0x0000, seq=0/0, ttl=64 (request in 18)
   20 8.083659    192.168.83.1     192.168.83.180   ICMP        42 Echo (ping) request  id=0x0000, seq=0/0, ttl=2 (no response found!)

  > Destination: Vmware_c0:00:08 (00:50:56:c0:00:08)
  > Source: Vmware_04:a3:f8 (00:0c:29:04:a3:f8)
    Type: IPv4 (0x0800)
v Internet Protocol Version 4, Src: 192.168.83.182, Dst: 192.168.83.1
```

Figure: Packet Analysis of the malicious host with enabled IP forwarding using Wireshark.

3) Legitimate host

The process will proceed with no hiccups. ICMP packet will be sent by detection host with destination ARP and MAC, and it will get a response from the legitimate host itself, there will be a proper match, and legitimacy will be established.

THE DETECTION ALGORITHM FOR ARP SPOOFING

Here is the Python style pseudocode for this method

```
if(ntohs(ether_protocol.ether_type)==0x0806)
        #if the packet is an ARP packet
        ArpPacketAnalysis()
        if (arp_dst_mac!= dmac ||arp_src_mac != smac)
                Droppacket()
                ResponseModule()
        QueryMapping DB()
```

```
if (have an entry)
    if(IP-MAC pair of the packet is coherent with the DB entry)
        refresh the entry()
    Elif (IP-MAC pair of the packet is not coherent with the DB entry)
        Droppacket()
        Response Module()
elif (don’t have an entry)
    SendICMPPacket()
    if(receives ICMP reply packet with identical identifier and sequence number)
    if(arp_src_mac==icmp_src_mac and arp_src_ip==icmp_src_ip)
        SetNewEntry()
    elif(arp_src_mac!=icmp_src_mac and arp_src_ip==icmp_src_ip)
        Get address pair of the true host;
        Droppacket()
        ResponseModule()
elif(no ICMP reply packet with identical identifier and sequence number)
    Droppacket()
    ResponseModule()
```

CONCLUSION

Our findings somewhat agree with those of the authors. Their 3 cases of detecting a suspicious host almost tally with our 3 cases. A slight difference was encountered in the third case. Their way of detecting an anomaly or a malicious host in the case where the malicious host had enabled IP forwarding made use of checking ICMP sequence numbers and ICMP identification number. When we ran this case in our environment of Virtual Machines and Python instances with Scapy, we found the ICMP field of sequence numbers and identification to not show any discrepancies the way the authors found. Instead we have relied on the attributes of the final ICMP response packet sent by the victim host to the detection host to detect the presence of a malicious host.

Also, this whole model will not work in the scenario where the <ARP,MAC> dictionary we are learning at the detection host is reset due to some occurrence, since resetting will render the dictionary blank, and when the malicious host starts with malicious activity of sending an ARP packet with purposely incorrect MAC/IP pair, the detection host will learn this wrong info, and will never doubt this info again, and in fact will end up trusting the malicious host. Also, it can end up with entries that have repeated IP entries (think of it!).

FUTURE SCOPE

ARP poisoning is one of the most easily exploitable attack in network. There are no security mechanisms in ARP. Constant checking of ARP traffic by firewall is necessary to maintain proper <IP,MAC pairs>.

ARP poisoning is mostly used to intercept traffic between two hosts. The basic intention to perform ARP traffic is to read the communications between the two hosts. To prevent the attacker to from reading the traffic, security measures like encrypting the data should be enforced.

The most popular reason to intercept traffic is snoop the web(HTTP) traffic of the victim. Standards like HTTPS protocols or enforcing HTTPS strict transport security can be used to prevent readability of the traffic.

REFERENCES

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6466290 - Paper Publication
http://scapy.readthedocs.io/en/latest/usage.html - Scapy package usage
https://www.monkey.org/~dugsong/dsniff/ - dsniff is a collection of tools for network auditing and penetration testing.
https://thepacketgeek.com/scapy-p-04-looking-at-packets/-Scapy p.04 – Looking at Packets
https://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml#icmp-parameters-codes-0
https://www.petri.com/csc_arp_cache - ARP Cache
http://scapy.readthedocs.io/en/latest/usage.html?highlight=srploop - Usage of srploop from Scapy
https://linuxconfig.org/how-to-turn-on-off-ip-forwarding-in-linux - How to Disable/Enable IP forwarding in Linux
Computer Networking - A Top-Down Approach (Sixth Edition) - James F. Kurose, Keith W. Ross