



COMP 8505

Assignment 4

Testing Doc

Alex Zielinski – A00803488

Table of Contents

Testing Procedure Explained	3
Test Case #1 – Usage Message.....	4
Test Case #2 – MAC Address Retrieval	5
Test Case #3 – ARP Poison (ARP Cache Poison)	7
Test Case #4 – ARP Poison (Capturing DNS Traffic)	8
Test Case #5 – DNS Spoof	9

Testing Procedure Explained

Testing for this software was done in lab-323 SE12 at BCIT Burnaby Campus. Two computers were used for the testing procedure. The machine that acted as the attacker and ran the ARP poisoning and DNS spoofing software was **192.168.0.18** and the target machine that acted as the victim was **192.168.0.19**. The command below was to run the dns spoofer software throughout test cases. Test cases will refer to this command when instructing to start the dns spoofer program.

```
14:58:46(-)root@datacomm-192-168-0-18:code$ ./dns_spoof.py 192.168.0.18 192.168.0.19 192.168.0.100
./dns_spoof.py 192.168.0.18 192.168.0.19 192.168.0.100
```

Within the dns spoofer program there is a 2d array that specifies a number of whitelisted sites. The format of the 2d array is as follows: **domain -> ip**. This means that for any DNS query that the attacker sniffs from the target that contains a **domain** from the whitelist, send a DNS response with the **IP** that goes with that domain. This way the target will be redirected to the IP you specify whenever they try to visit a whitelisted domain. The contents of the 2d array for this test is as follows.

Domain: milliways.bcit.ca -> **IP:** 192.168.0.18

Domain: sd43.bc.ca -> **IP:** 192.168.0.18

Domain: bcit.ca -> **IP:** 192.168.00.18

Domain: sfu.ca -> **IP:** 192.168.0.18

Domain: ubc.ca -> **IP:** 192.168.0.18

Domain: cbc.ca -> **IP:** 192.168.0.18

In this case all whitelisted domains get redirected to 192.168.0.18. This machine ran an apache web server, so that the target machine would be redirected to this web server whenever they tried to access a whitelisted domain. The webpage simply displays text saying "Gotch Ya!".

Please refer to the user guide within the design document for a detailed explanation of the program usage.


(Test Cases start on next page)

Test Case #1 – Usage Message

Description

The purpose of this test is to ensure the user can get a 'Usage' message describing how to run the program when they run the DNS spoofer program with no arguments.

Test

Steps	Expected	Screenshot	Result
Step 1 On the attacker machine start the DNS spoofer program	A usage message should pop up describing how to run the DNS spoofer along with the description of program arguments	 <pre>14:54:44(-)root@datacomm-192-168-0-18:code\$./dns_spoof.py Error: invalid arguments Usage: ./dns_spoof.py <ATTACKER IP> <TARGET IP> <ROUTER IP> 14:54:49(-)root@datacomm-192-168-0-18:code\$</pre>	PASS

(Test Case #2 on next page)

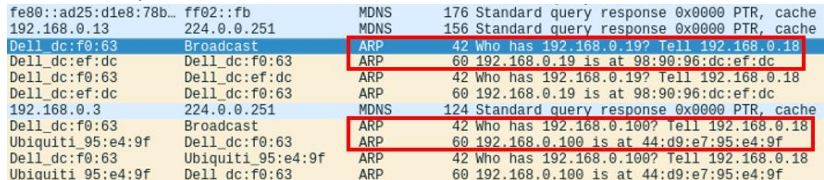
Test Case #2 – MAC Address Retrieval

Description

The purpose of this test is to ensure the DNS spoofer program is able to retrieve the MAC address of the attacking machine, target machine and the router. These MAC addresses must be used later to forge ARP packets for ARP poisoning.

Test

Steps	Expected	Screenshot	Result
<p>Step 1 On the attacker machine start a Wireshark session</p> <p>Step 2 On the attacker machine run <i>ifconfig</i> and notice the MAC address</p> <p>Step 3 On the target machine run <i>ifconfig</i> and notice the MAC address</p> <p>Step 4 On the target machine run <i>arp -a</i> and notice the MAC address of the router</p> <p>Step 5 On the attacker machine start the DNS spoofer program and notice the output</p> <p>Step 6 Stop the Wireshark session and stop the dns spoofer via <i>ctrl + c</i> and notice the ARP queries being sent to target and router</p>	<p>A startup output of the dns spoofer should print out the IP and MAC address of the attacker, target and router. The info displayed there should match the output of <i>ifconfig</i> and <i>arp -a</i> commands run on the respective machines. The Wireshark capture should show the attacker machine sending ARP queries to target and router to get their MAC addresses</p>	<p>Attacker IP 192.168.0.18 with MAC 98:90:96:dc:f0:63</p> <pre>14:55:32(-)root@datacomm-192-168-0-18:code\$ ifconfig eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500 inet 192.168.0.18 netmask 255.255.255.0 broadcast inet6 fe80::ad25:d1e8:78b5:b0f4 prefixlen 64 scope ether 98:90:96:dc:f0:63 txqueuelen 1000 (Ethernet) RX packets 235073 bytes 184947330 (176.3 MiB)</pre> <p>Target IP 192.168.0.19 with MAC 98:90:96:dc:ef:dc</p> <pre>14:56:19(-)root@datacomm-192-168-0-19:~\$ ifconfig eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500 inet 192.168.0.19 netmask 255.255.255.0 broadcast inet6 fe80::ad25:d1e8:78b5:b0f4 prefixlen 64 scope ether 98:90:96:dc:ef:dc txqueuelen 1000 (Ethernet) RX packets 1026991 bytes 512536145 (488.7 MiB)</pre> <p>Router IP 192.168.0.100 with MAC 44:d9:e7:95:e4:9f</p> <pre>14:56:58(-)root@datacomm-192-168-0-18:code\$ arp -a ? (192.168.0.11) at 98:90:96:dc:f2:e9 [ether] on eno1 gateway (192.168.0.100) at 44:d9:e7:95:e4:9f [ether] on eno1 ? (192.168.0.19) at 98:90:96:dc:ef:dc [ether] on eno1</pre> <p>Output of program shows correct IP and MAC pairs for attacker, target and router machines</p> <pre>Target ARP Response: 192.168.0.19 -> 98:90:96:dc:ef:dc Router ARP Response: 192.168.0.100 -> 44:d9:e7:95:e4:9f Attacker: 192.168.0.18 -> 98:90:96:dc:f0:63 Target : 192.168.0.19 -> 98:90:96:dc:ef:dc Router : 192.168.0.100 -> 44:d9:e7:95:e4:9f</pre>	PASS

		<p>Wireshark output on attacker machine shows the ARP query sent from the attacker to the target and router and the respective machines sending ARP responses back to the attacker with their respective MAC addresses</p>  <pre> fe80::ad25:die8:78b... ff02::fb MDNS 176 Standard query response 0x0000 PTR, cache 192.168.0.13 224.0.0.251 MDNS 156 Standard query response 0x0000 PTR, cache Dell_dc:f0:63 Broadcast ARP 42 Who has 192.168.0.19? Tell 192.168.0.18 Dell_dc:ef:dc Dell_dc:f0:63 ARP 60 192.168.0.19 is at 98:90:96:dc:ef:dc Dell_dc:f0:63 Dell_dc:ef:dc ARP 42 Who has 192.168.0.19? Tell 192.168.0.18 Dell_dc:ef:dc Dell_dc:f0:63 ARP 60 192.168.0.19 is at 98:90:96:dc:ef:dc 192.168.0.3 224.0.0.251 MDNS 124 Standard query response 0x0000 PTR, cache Dell_dc:f0:63 Broadcast ARP 42 Who has 192.168.0.100? Tell 192.168.0.18 Ubiquiti_95:e4:9f Dell_dc:f0:63 ARP 60 192.168.0.100 is at 44:d9:e7:95:e4:9f Dell_dc:f0:63 Ubiquiti_95:e4:9f ARP 42 Who has 192.168.0.100? Tell 192.168.0.18 Ubiquiti_95:e4:9f Dell_dc:f0:63 ARP 60 192.168.0.100 is at 44:d9:e7:95:e4:9f </pre>	
--	--	--	--

(Test Case #3 on next page)

Test Case #3 – ARP Poison (ARP Cache Poison)

Description

The purpose of this test is to ensure the target machine's ARP cache is being poisoned by changing the MAC address of the router IP to the MAC address of the attacker machine so that the target machine thinks the router is the attacker machine.

Test

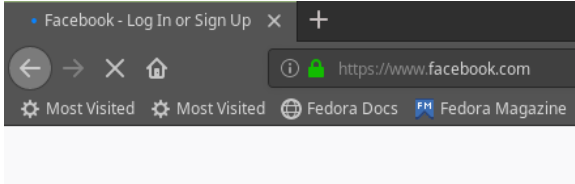
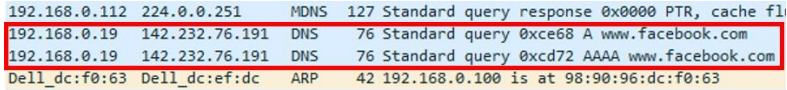
Steps	Expected	Screenshot	Result
<p>Step 1 On the target machine run arp -a and notice the arp cache</p> <p>Step 2 On the attacker machine start a Wireshark session</p> <p>Step 3 On the attacker machine start the DNS spoofer program and notice the output</p> <p>Step 4 On the target machine run arp -a and notice the arp cache, the MAC address of router should be that of the target machine</p> <p>Step 5 Stop the Wireshark session and stop the dns spoofer via ctrl + c and notice the ARP responses being sent to target and router</p>	<p>Within the targets ARP cache the router should now have the attackers MAC address. The Wireshark output should show the attacker machine sending out ARP responses to the router and target</p>	<p>Before ARP poisoning the router IP has the proper MAC</p> <pre>15:03:09(-)root@datacomm-192-168-0-19:~\$ arp -a ? (192.168.0.11) at 98:90:96:dc:f2:e9 [ether] on eno1 ? (192.168.0.244) at b8:ca:3a:7f:22:37 [ether] on eno1 ? (192.168.0.17) at 98:90:96:dc:ee:d6 [ether] on eno1 gateway (192.168.0.100) at 44:d9:e7:95:e4:9f [ether] on eno1 ? (192.168.0.18) at 98:90:96:dc:f0:63 [ether] on eno1 ? (192.168.0.112) at 98:90:96:c6:e5:75 [ether] on eno1 15:03:11(-)root@datacomm-192-168-0-19:~\$</pre> <p>After ARP poisoning the router IP has the attackers MAC</p> <pre>15:03:11(-)root@datacomm-192-168-0-19:~\$ arp -a ? (192.168.0.11) at 98:90:96:dc:f2:e9 [ether] on eno1 ? (192.168.0.244) at b8:ca:3a:7f:22:37 [ether] on eno1 ? (192.168.0.17) at 98:90:96:dc:ee:d6 [ether] on eno1 gateway (192.168.0.100) at 98:90:96:dc:f0:63 [ether] on eno1 ? (192.168.0.18) at 98:90:96:dc:f0:63 [ether] on eno1 ? (192.168.0.112) at 98:90:96:c6:e5:75 [ether] on eno1 15:04:41(-)root@datacomm-192-168-0-19:~\$</pre> <p>Wireshark output of attacker shows attacker sending spoofed ARP responses as to ARP poison target and router</p> <pre>192.168.0.3 224.0.0.251 MDNS 124 Standard query response 0x0000 PTR, ca Dell_dc:f0:63 Dell_dc:f0:63 ARP 42 192.168.0.100 is at 98:90:96:dc:f0:63 Dell_dc:f0:63 Ubiquiti_95:e4:9f ARP 42 192.168.0.19 is at 98:90:96:dc:f0:63 192.168.0.19 224.0.0.251 MDNS 126 Standard query response 0x0000 PTR, ca 192.168.0.112 224.0.0.251 MDNS 127 Standard query response 0x0000 PTR, ca 192.168.0.6 224.0.0.251 MDNS 123 Standard query response 0x0000 PTR, ca 192.168.0.3 224.0.0.251 MDNS 124 Standard query response 0x0000 PTR, ca 192.168.0.19 224.0.0.251 MDNS 126 Standard query response 0x0000 PTR, ca 192.168.0.112 224.0.0.251 MDNS 127 Standard query response 0x0000 PTR, ca Ubiquiti 1f:3... Spanning-tree(f... STP 60 RST. Root = 32768/0/b4:fb:e4:1f:28:08 Dell_dc:f0:63 Dell_dc:f0:63 ARP 42 192.168.0.100 is at 98:90:96:dc:f0:63 Dell_dc:f0:63 Ubiquiti_95:e4:9f ARP 42 192.168.0.19 is at 98:90:96:dc:f0:63 fe80::ad25:d1... ff02::fb MDNS 215 Standard query 0x0000 ANY 4.f.0.b.5.b.</pre>	PASS

Test Case #4 – ARP Poison (Capturing DNS Traffic)

Description

The purpose of this test is to ensure ARP poisoning is working so that the attacker is getting network traffic from the target machine to the router (in specific we are looking for DNS packets from target to the router).

Test

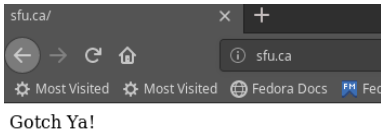
Steps	Expected	Screenshot	Result
<p>Step 1 On the attacker machine start a Wireshark session</p> <p>Step 2 On the attacker machine start the DNS spoofer program</p> <p>Step 3 On the target machine open up a browser in the URL enter www.facebook.com</p> <p>Step 4 Stop the Wireshark session and stop the dns spoofer via ctrl + c and notice the ARP responses being sent to target and router</p>	<p>Within the Wireshark output of the attacker machine there should be visible DNS queries containing the site Facebook going from the target to the router</p>	<p>The target going to www.facebook.com</p>  <p>Wireshark output of attacker shows Facebook DNS queries from target</p> 	<p>PASS</p>

Test Case #5 – DNS Spoof

Description

The purpose of this test is to ensure that DNS spoofing works as expected. Whenever the target machine tries to access one of the whitelisted domains (as defined at the beginning of this document) then they would be redirected to an apache webserver run by the attacker.

Test

Steps	Expected	Screenshot	Result
<p>Step 1 On the attacker machine start a Wireshark session</p> <p>Step 2 On the target machine start a Wireshark session</p> <p>Step 3 On the attacker machine start the DNS spoofer program</p> <p>Step 4 On the target machine open a browser and, in the URL, enter sfu.ca and notice the webpage that pops up</p> <p>Step 5 On the attacker machine notice the program output</p> <p>Step 6 Stop the Wireshark sessions and stop the dns spoofer via ctrl + c and notice the ARP queries being received and the responses being sent</p>	<p>On the target machine the browser should be redirected to an apache web server running on 192.168.0.18 when trying to access sfu.ca. The dns spoofer output should show it received the sfu.ca dns query and sent a response with 192.168.0.18. The Wireshark output of the attacker should show the sfu.ca DNS query from the target to the router as well as the DNS response containing 192.168.0.18 the attacker sends (looking like it's from the router to the target)</p>	<p>Target redirected to apache web server when trying to access sfu.ca</p>  <p>Dns Spoofer output shows it received a dns query with whitelisted domain and sent a response with IP of apache web server</p> <pre> Target ARP Response: 192.168.0.19 -> 98:90:96:dc:ef:dc Router ARP Response: 192.168.0.100 -> 44:d9:e7:95:e4:9f Attacker: 192.168.0.18 -> 98:90:96:dc:f0:63 Target : 192.168.0.19 -> 98:90:96:dc:ef:dc Router : 192.168.0.100 -> 44:d9:e7:95:e4:9f sfu.ca. -> 192.168.0.18 sfu.ca. -> 192.168.0.18 </pre> <p>Wireshark output of attacker shows the sfu.ca dns queries and the forged dns response the attacker sends back (looking like it's from the router to the target)</p> <pre> Dell_dc:f0:63 Ubiquiti_95:e... ARP 42 192.168.0.19 is at 98:90:96:dc:f0:63 (duplicate use of 19: 192.168.0.19 142.232.76.191 DNS 66 Standard query 0x1843 A sfu.ca 192.168.0.19 142.232.76.191 DNS 66 Standard query 0xa9c3 AAAA sfu.ca 142.232.76.191 192.168.0.19 DNS 88 Standard query response 0x1843 A sfu.ca A 192.168.0.18 142.232.76.191 192.168.0.19 DNS 88 Standard query response 0xa9c3 AAAA sfu.ca A 192.168.0.18 192.168.0.19 192.168.0.18 TCP 74 44916 -> 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM= </pre>	PASS