Rajat Sethi - CPSC 8580

**Local DNS Lab – Report**

**Task 0 - DNS Setup Test:**

Running dig sends a “question” to the ns.attacker32.com domain. The DNS server looks at its file and finds that ns.attacker32.com corresponds with the IP Address 10.9.0.153, as expected.

Text

Description automatically generated

[www.example.com](http://www.example.com) is a real website for testing purposes. Entering the domain name in a standard browser yields the following page.

Graphical user interface, text, application, email

Description automatically generated

Running “dig [www.example.com](http://www.example.com)” yields the true IP address of the web page.

Text

Description automatically generated

On the other hand, asking ns.attacker32.com for the IP Address of [www.example.com](http://www.example.com) yields a garbage result, 1.2.3.5.

Text

Description automatically generated

This “garbage” entry corresponds with the entry in zone\_attacker32.com, as shown in the below snapshot.

Text

Description automatically generated

**Task 1 – Direct Spoofing**

While the packets are not being spoofed, [www.example.com](http://www.example.com) corresponds to the correct 93.184.216.34 IP address. This means that the DNS server is sending the correct response and the packet is not being spoofed.

Text

Description automatically generated

Using this python script, I was able to create a program that sniffs and spoofs packets before they reach the DNS server, then send a fake response back to the victim. In this script, the destination and source IP addresses are flipped. This fools the victim into thinking that the packet “came” from the DNS server, when in reality it is a malicious packet coming from the attacker.

The next change I make to the packet is in the answer section. The IP Address is kept in the “rdata” variable of the Answer section. In this malicious packet, I change the IP address to “2.4.6.8” instead of the usual 93.184.216.34 address. In a real attack, the IP address would be a fake website where I can siphon the data from the user. The packet is then reassembled with the new data and sent back to the victim.

Text

Description automatically generated

By running the spoofing script, I can now change the response that the victim gets when they try to dig for [www.example.com](http://www.example.com). As shown below, the original IP address was spoofed to my fake “2.4.6.8” address.

Text

Description automatically generated

**Task 2 – DNS Cache Poisoning**

In this task, I changed the destination of the spoofed packet. Instead of going back to the victim, the packet goes to the Local DNS server container. Also, I edited the “fake” IP address to 4.8.12.16, as to differentiate the result from Task 1. In reality, this would be point to the fake, malicious website that siphons data.

Text

Description automatically generated

As shown below, the Local DNS cache has been poisoned with the fake IP address.

Text

Description automatically generated

When the user digs for [www.example.com](http://www.example.com), they get the poisoned result instead.

Text

Description automatically generated

**Task 3 – Spoofed NS Records**

In this task, the packet is spoofed in the same way as Task 2, with one change. This time, I added the NSsec Authority section to the packet, which directs the victim to ns.attacker32.com as the nameserver for all “example.com” related requests. To differentiate from other tasks, I changed the fake IP address again to “6.12.18.24,” but since ns.attacker32.com already has a wildcard, this IP address will not show up (1.2.3.6 will appear instead).

Text

Description automatically generated

As shown in this cache dump, the nameserver has changed to ns.attacker32.com, [www.example.com](http://www.example.com) corresponds to the entry in ns.attacker32.com (1.2.3.5), and anything else goes to 1.2.3.6 (also in the fake nameserver).

Text

Description automatically generated

As shown below, abc.example.com corresponds with the cache dump.

Text

Description automatically generated

[www.example.com](http://www.example.com)Text

Description automatically generated

dig NS example.com (shows the Additional Section)

Text

Description automatically generated