Luke Evans

February 20th, 2021

Lab 4 – DNS

User: 10.0.2.4

DNS Server: 10.0.2.5

Attacker: 10.0.2.15

*Part I: Setting Up a Local DNS Server*

*1.1: Configure the User Machine*

On the user machine, we edit the head file to change our DNS server to 10.0.2.5. The head file is located at /etc/resolvconf/resolv.conf.d/head:

*A screenshot of a computer

Description automatically generated with low confidence*

In order for our change to take affect we must refresh our DNS setting. To do this we use the command sudo resolvconf -u. Then we ping google.com and check we can resolve the IP and reach the site:

Text

Description automatically generated

If we sniff the packets with wireshark, we can see that our user machine uses 10.0.2.5 as its local DNS server:

Graphical user interface, application

Description automatically generated

Observation: In this task, we set up our local server VM to act as our DNS server. In order to make the user VM use our local DNS server we had to edit the /etc/resolvconf/resolv.conf.d/head file and specify our nameserver as our DNS server IP. After we refreshed our DNS settings we tested our configuration by pinging [www.google.com](http://www.google.com). Using wireshare, we were able to sniff the packets and see that the DNS queries were in fact being sent to IP 10.0.2.5 which is our local DNS server.

Explanation: Changing the DNS server on Linux can be done by chaning the nameserver location by editing the heads file located in /etc/resolvconf/resolv.conf.d/ directory. This will allow us to run attacks on our local DNS server.

*1.2: Set up a Local DNS Server*

Options for our bind DNS server are configured in /etc/bind/named.conf.options:

Text

Description automatically generated

When we ping [www.google.com](http://www.google.com) we can see that the server VM is used by our User VM as the local DNS server:

Graphical user interface, application

Description automatically generated

Observation: In this task we setup our DNS server on our server VM. We had to configure disabling dnssec and changed the name of our cache dump file to dump.db. Bind 9 was already configured in our prebuilt image so I only had to check that the configuration settings were correct. Just as in the previous step, when pinging google.com, our server VM is used by the User VM as our local DNS server. This is shown from the screenshot from wireshark.

Explanation: We setup our server VM to serve as our local DNS server for our user machine. This will allow us to simulate DNS attacks on computers on the same LAN.

*1.3: Host a Zone in the Local DNS Server*

First, we create our zones in /etc/bind/named.conf file. These two zone specify forward and reverse lookup for example.com:

Text

Description automatically generated

Next, we need to setup our forward lookup zone file that we specified above as example.com.db:

Text

Description automatically generated

Lastly, we need to setup the reverse lookup file:

Text

Description automatically generated

To test our configuration, we restart out bind server and then run the dig example.com command from our user VM:

Text, letter

Description automatically generated

Observation: For this task we configured hosting a zone on our DNS server. The domain that we are hosting is [www.example.com](http://www.example.com). In the above screenshots, we show the files that we created for our DNS records and routing information. In order for our changes to take affect, we had to restart our DNS server. To test our setup, we ran dig [www.example.com](http://www.example.com) from the user VM to see that it routes to the local IP.

Explanation: Bind setting up the appropriate bind configurations, we can direct web addresses to use our local IP address. They can then be queried on our DNS server by using the dig command from a machine that is configured to use our server as the local DNS server.

*Part 2: Attacks on DNS*

*2.1: Modifying the Hosts File*

We modify the /etc/hosts file to point [www.bank32.com](http://www.bank32.com) to google.com IP address.

Text

Description automatically generated

We run the dig command and see that it ignores our local hosts file and returns the real IP address for [www.bank32.com](http://www.bank32.com):

Text

Description automatically generated

Next we ping [www.bank32.com](http://www.bank32.com):

Text

Description automatically generated

Finally we try to navigate to [www.bank32.com](http://www.bank32.com) in our browser on the VM that we changed the hosts file in:

Graphical user interface, text

Description automatically generated

It sends us to the IP we specified which cannot be resolved by google’s servers.

Observations: In this task, we modified the hosts file on the users machine. We change the IP for [www.bank32.com](http://www.bank32.com) to an IP that is hosted by google.com. We then experimented by running the dig command and verifying that the dig command does not reference the hosts file. However when using ping and navigating from a browser we confirmed that our change took affect.

Explanation: By modifying the the hosts file on a users computer, we can override the dns server and redirect domains. This file is often used when needing to setup local servers for development environments. The dig command does not reference the hosts file but ping command will.

*2.2: Directly Spoofing Response to User*

First on the users machine, we dig [www.priceless.com](http://www.priceless.com) and resolve the IP address and authority:

Text

Description automatically generated

Next we clear the cache on both our server and user VM’s then run the netwox 105 program on our attacker VM to spoof the DNS query for [www.priceless.com](http://www.priceless.com) to 1.2.3.4:

Text

Description automatically generated

Then we run the dig [www.priceless.com](http://www.priceless.com) again from our user VM:

Text, letter

Description automatically generated

On the attackers machine we see the following output:

Text, letter

Description automatically generated

Observation: In this task we used the nextwox 105 tool that allowed us to sniff and spoof DNS udp packets. We first used the dig command to resolve the actual dns records for [www.priceless.com](http://www.priceless.com). Then we flushed the cache so that a new dns query would be triggered for www.priceless.com , setup the netwox tool and reran the dig command. This time we see the spoofed response as output from our dig command.  
  
Explanation: By sniffing and spoofing packets, we can manipulate the DNS and redirect DNS queries. Since we are on the same network, we are able to sniff all necessary information to spoof the packet. By doing this we are able to cause a user to resolve a websites domain to a IP address of our choice. Since we are sending our reply directly to the user, this will only affect the single user we target.

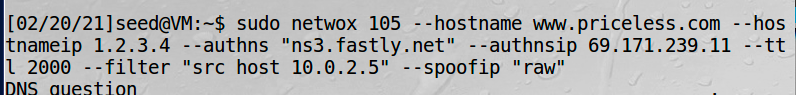
*2.3: DNS Cache Poisoning Attack*

We flush our cache and then check that [www.priceless.com](http://www.priceless.com) resolves to the correct DNS records:

Text, letter

Description automatically generated

Then we clear the cache and poison the DNS cache using the netwox 105 tool to resolve [www.priceless.com](http://www.priceless.com) to the IP address of 1.2.3.4:



Text, letter

Description automatically generated

We can see the query on our Attacking VM:

Text, letter

Description automatically generated

Looking at the dump file on our server machine we can see our DNS has cached the [www.priceless.com](http://www.priceless.com) IP address as 1.2.3.4:

Text

Description automatically generated

Observation: In this lab, we used the netwox 105 tool to sniff and spoof DNS queries. This is like the previous attack except instead of targeting the user directly, we target the DNS server upstream so that we can poison the cache. We first use the dig command to resolve [www.priceless.com](http://www.priceless.com). After confirming we are seeing the real DNS records, we flush the cache so our next request will trigger another DNS query. Next we ran the netwox tool on the attacker VM with the goal of chaning the returned IP address for [www.priceless.com](http://www.priceless.com) to 1.2.3.4. Once the netwox tool was running, we ran the dig command again from the user VM. This time we see the spoofed response in our dig output. When checking the cache on our local DNS server, we can confirm that we have successfully poisoned the cache. If we had a different VM that was also using the same DNS server, all DNS queries for [www.priceless.com](http://www.priceless.com) would resolve to the IP address of 1.2.3.4 until the cache was flushed or the cache time expired.

Explanation: By using sniffing and spoofing attack on DNS queries we can spoof DNS udp packets and poison a DNS servers cache. In this attack, we were able to spoof the DNS query response and trick the DNS server into thinking that [www.priceless.com](http://www.priceless.com) should resolve to 1.2.3.4. Due to the cache being poisoned any user that queries this DNS server for the [www.priceless.com](http://www.priceless.com) IP address will receive the incorrect IP of 1.2.3.4 until the cache is flushed or is expired.