Lab 2: Local DNS Attack Lab

DNS (Domain Name System) is the Internet’s phone book; it translates hostnames to IP addresses (and vice versa). Here we are focusing on the local DNS attacks. We are doing the DNS cache poisoning in the local server and therefore we are creating our own DNS server environment for the attack.

Using the docker commands dcbuild and dcup we can start the servers and here we can see both the attacker and local DNS servers are up and ready.

Text

Description automatically generated with low confidence

Text

Description automatically generated

Now we can get the list of users in the docker using the command “dockps” as shown below.

Text

Description automatically generated

From the above screenshot we can see that here we have 4 components i) user -10.9.0.5 ii) the seed router iii attacker- ns-10.9.0.53 and iv) local DNS server 10.9.0.53

Text

Description automatically generated

Text

Description automatically generated with low confidence

Now when we run the dig command “ dig ns.attacker32.com” the local DNS server will send the request to the attacker and got the following result.

Text

Description automatically generated

Now I am going to run the command “dig [www.example.com](http://www.example.com)” I got the original ip address as shown in the figure.

Text

Description automatically generated

Obviously, nobody is going to ask ns.attacker32.com for the IP address of www.example.com; they will always ask the example.com domain’s official nameserver for answers. The objective of the DNS cache poisoning attack is to get the victims to ask ns.attacker32.com for the IP address of www.example.com. Namely, if our attack is successful, if we just run the first dig command, the one without the @ option, we should get the fake result from the attacker, instead of getting the authentic one from the domain’s legitimate nameserver.

Task 1

The main objective of DNS attacks on a user is to redirect the user to another machine B when the user tries to get to machine A using A’s host name. When a user types the name of a web site (a host name, such as www.example.com) in a web browser, the user’s computer will send a DNS request to the local DNS server to resolve the IP address of the host name. Attackers can sniff the DNS request message, they can then immediately create a fake DNS response, and send back to the user machine. If the fake reply arrives earlier than the real reply, it will be accepted by the user machine. Therefore out task is to create a spoofed response to the DNS response .

I made the necessary changes to the code.

Text

Description automatically generated

With the help of the command “docker network ls” we can get the interface value.

Text

Description automatically generated with medium confidence

Text

Description automatically generated

Now ran the code with the spoofed address as 9.8.7.6 and when I ran the command “ dig [www.example.com](http://www.example.com)” from the user side I got the spoofed ip address which is 9.8.7.6 , therefore

We can say the attack was successful .

Text

Description automatically generated

Text

Description automatically generated

We should make sure to clear the cache during an attack . If the cache has the answer, the reply from the local DNS server will be faster than the one you spoofed, and your attack will not be able to succeed. Here we can see that now when we run again we can see the original ip address of the host.

Text

Description automatically generated

The local DNS servers always saves the cache cookie and gives us the same output until we clears it or expires.

Text

Description automatically generated

Task 2:

Here we are going to perform DNS cache poisoning. In order to achieve long-lasting effect, every time the user’s machine sends out a DNS query for www.example.com the attacker’s machine must send out a spoofed DNS response. This might not be so efficient; there is a much better way to conduct attacks by targeting the DNS server, instead of the user’s machine. When a local DNS server receives a query, it first looks for the answer from its own cache; if the answer is there, the DNS server will simply reply with the information from its cache. If the answer is not in the cache, the DNS server will try to get the answer from other DNS servers. When it gets the answer, it will store the answer in the cache, so next time, there is no need to ask other DNS servers.

Here I made changes to the code. The fake ip address is 9.8.7.7. This is the ip address in the cache and we are supposed to get this instead of the original address when the user tries to reach the site “www.example.com”.

Graphical user interface, text, application

Description automatically generated

Now when we try to ping [www.example.com](http://www.example.com) the server has to return the ip address 9.8.7.7 as shown in the figure.

Text

Description automatically generated

We can confirm this by checking the local DNS server dump as shown in the figure below

A picture containing text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Task 3

In the previous task, our DNS cache poisoning attack only affects one hostname, i.e., www.example.com. If users try to get the IP address of another hostname, such as mail.example.com, we need to launch the attack again. It will be more efficient if we launch one attack that can affect the entire example.com domain. The idea is to use the Authority section in DNS replies. Basically, when we spoofed a reply, in addition to spoofing the answer (in the Answer section), we add the following in the Authority section. When this entry is cached by the local DNS server, ns.attacker32.com will be used as the nameserver for future queries of any hostname in the example.com domain. Since ns.attacker32.com is controlled by attackers, it can provide a forged answer for any query.

I have added the NS record in our code. I gave the ip address 2.5.3.5 in the Anssec and 10.9.0.53 as the host address as shown below.

Text

Description automatically generated

Now I run the command “dig abc.example.com”. Here the server returns the ip address 1.2.3.6 which has already been given in ns32.attacker.com.

Text, letter

Description automatically generated

Now the cache is controlled by the attacker when I checked the cache dump.

A picture containing text

Description automatically generated

Text

Description automatically generated