Local DNS Attack Lab

Lab Environment Setup 2
Verification of the DNS setup 2
Attacks on DNS 3
Task 1: Directly Spoofing Response to User 4
Task 2: DNS Cache Poisoning Attack – Spoofing Answers 6
Task 3: Spoofing NS Records 7
Task 4: Spoofing NS Records for Another Domain 9
Task 5: Spoofing Records in the Additional Section 11
Submission 13

Lab Environment Setup

Please download the Labsetup.zip file from the link given below :
https://seedsecuritylabs.org/Labs_20.04/Networking/DNS/DNS_Local/
Follow the instructions in the lab setup document to set up the lab environment.

The main target for this lab is a local DNS server. Obviously, it is illegal to attack a real server, so we need to set up our own DNS server to conduct the attack experiments. The lab environment needs four separate machines:
one for the victim, one for the local DNS server, and two for the attacker.
The lab environment setup is illustrated in Figure 1. This lab focuses on the local attack, so we put all these machines on the same LAN.

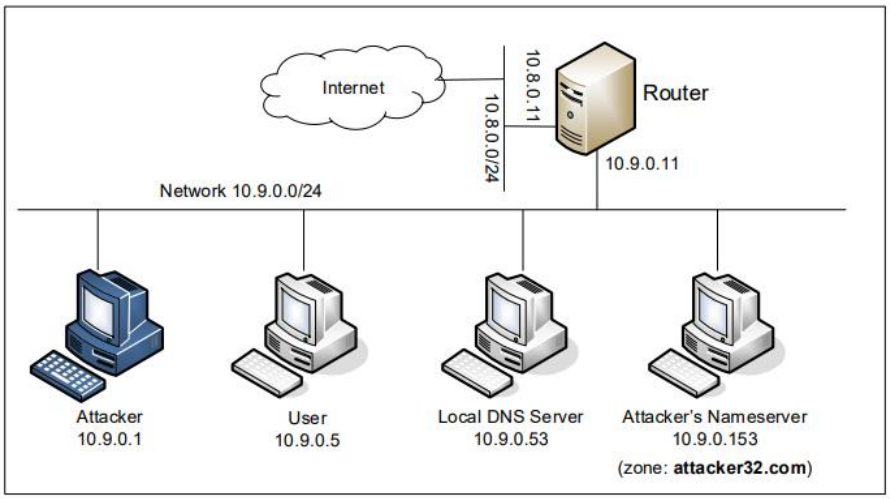

Figure 1 : Lab Environment setup

Verification of the DNS setup

From the **User container**, we will run a series of commands to ensure that our lab setup is correct. In your lab report, please document your testing results.

Get the IP address of ns.attacker32.com

When we run the following dig command, the local DNS server will forward the request to the Attacker name server due to the forward zone entry added to the local DNS server's configuration file. Therefore, the answer should come from the zone file (attacker32.com.zone) that we set up on the Attacker nameserver. If this is not what you get, your setup has issues.

On the victim terminal run the command:

 # dig ns.attacker32.com

Get the IP address of www.example.com

Two nameservers are now hosting the example.com domain, one is the domain's official nameserver, and the other is the Attacker container. We will query these two nameservers and see what response we will get. Please run the following two commands (from the User machine), and describe your observation.

On the victim terminal run the commands:

 # dig www.example.com

 # dig @ns.attacker32.com www.example.com

Attacks on DNS

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. For example, when the user tries to access online banking, if the adversaries can redirect the user to a malicious web site that looks very much like the main web site of the bank, the user might be fooled and give away the password of his/her online banking account.

Task 1: Directly Spoofing Response to User

In this task, when the client sends the DNS request to the local DNS server it accepts a response back, but if the attacker sends a spoofed DNS response to the user before the legitimate attack from the local DNS server then the attack is successful.

First show the legitimate response from the example.com domain's authoritative nameserver as well as the requests as seen in wireshark.

Please remember to clear the cache on the local DNS server first.

On the local DNS server's terminal run the command:

 # rndc flush

The victim machine sends out a DNS query to the local DNS server, which will eventually send out a DNS query to the authoritative nameserver of the example.com domain. This is done using the dig command. Before running the command keep wireshark open to view the packets being sent.

On the victim terminal run the command:

 # dig www.example.com

Provide a screenshot of your observation.

Provide a Wireshark screenshot of your observation as well.

Before launching the attack, make sure that the cache in the local DNS server is cleaned. 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. The following command is used on the local DNS server to clear its cache.

On the local DNS server's terminal run the command:

 # rndc flush

Now run the program in the attacker machine and show your spoofed information in the reply. Compare your results obtained before and after the attack. Also show the **spoofed packet captured on wireshark** and the cache of the local DNS server and explain your results.

Fill in the appropriate interface name in the code for task 1. More detailed instructions on finding the interface of the attacker machine can be found in the lab setup instructions document. Modify the tasks code and launch the attack.

On the attacker terminal run the command:

 # python3 task1.py

On the victim terminal run the command:

 # dig www.example.com

Provide a screenshot of your observation.
Provide a Wireshark screenshot of your observation as well.

The Wireshark on the attacker machine shows the spoofed response which is sent to the victim. The IP address mapped to www.example.com is 1.1.1.1 which is seen in the above image. We can see that the spoofed response comes before the legitimate response and hence is displayed as such in the victim machine.

To view the cache on the local DNS server we can use the rndc command to dump the cache and this dump is stored in **/var/cache/bind/dump.db** in our case.

On the local DNS server's terminal run the commands:

 # rndc dumpdb -cache

 # cat /var/cache/bind/dump.db | grep example

Describe your results and explain as to why you got the results you did.

A potential issue

When we do this lab using containers, sometimes we see a very strange situation. The sniffing and spoofing inside containers is very slow, and our spoofed packets even arrive later than the legitimate one from the Internet, even though we are local. In the past, when we used VMs for this lab, we have never had this issue. We have not figured out the cause of this performance issue yet (if you have any insight on this issue, please let us know).
If you do encounter this strange situation, we can get around it. We intentionally slow down the traffic going to the outside, so the authentic replies will not come that fast. This can be done using

the following tc command on the router to add some delay to the outgoing network traffic.
The router has two interfaces, eth0 and eth1, make sure to use the one connected to the external network 10.8.0.0/24.

// Delay the network traffic by 100ms
 # tc qdisc add dev eth0 root netem delay 100ms
// Delete the tc entry
 # tc qdisc del dev eth0 root netem
// Show all the tc entries
 # tc qdisc show dev eth0
You can keep the tc entry on for this entire lab, because all the tasks will face a similar situation

Task 2: DNS Cache Poisoning Attack – Spoofing Answers

The above attack targets the user's machine. 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 another DNS server.

Also fill in the appropriate interface name in the code for task 2 as done in previous tasks.

Modify the tasks code and launch the attack. Before doing the attack, please remember to clear the cache on the local DNS server first.

On the local DNS server's terminal run the command:

 # rndc flush

Now run the program **in the attacker terminal** and show your spoofed information in the reply. The victim machine sends out a DNS query to the local DNS server using the dig command. Also show the spoofed packet captured on wireshark and the cache of the local DNS server and explain your results.

On the attacker terminal run the command:

 # python3 task2.py

On the victim terminal run the command:

 # dig www.example.com

Provide a screenshot of your observation.
Provide a Wireshark screenshot of your observation as well.

To view the cache on the local DNS server we can use the rndc command to dump the cache.

On the local DNS server's terminal run the commands:

 # rndc dumpdb -cache

 # cat /var/cache/bind/dump.db | grep example

Describe your results and explain as to why you got the results you did.

Task 3: Spoofing NS Records

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.

```
;; AUTHORITY SECTION:
example.com.            259200  IN      NS      ns.attacker32.com.
```

Fill in the appropriate interface name in the code for task 3 as done in previous tasks.

Before launching the attack, please remember to clear the cache on the local DNS server first.

On the local DNS server's terminal run the command:

 # rndc flush

Now run the program **in the attacker terminal** and show your spoofed information in the reply. The victim machine sends out a DNS query to the local DNS server using the dig command. Also show the spoofed packet captured on wireshark and the cache of the local DNS server and explain your results.

On the attacker terminal run the command:

 # python3 task3.py

On the victim terminal run the command:

 # dig www.example.com

Provide a screenshot of your observation.

Provide a Wireshark screenshot of your observation as well.

If your attack is successful, when you run the dig command on the user machine for any hostname in the example.com domain, you will get the fake IP address provided by ns.attacker32.com.

On the victim terminal run the command:

 # dig www.example.com

 # dig ftp.example.com

Please also check the cache on the local DNS server and see whether the spoofed NS record is in the cache or not.To view the cache on the local DNS server we can use the rndc command to dump the cache.

On the local DNS server's terminal run the commands:

 # rndc dumpdb -cache

 # cat /var/cache/bind/dump.db | grep example

Describe your results and explain as to why you got the results you did.

Task 4: Spoofing NS Records for Another Domain

In the previous attack, we successfully poison the cache of the local DNS server, so ns.attacker32.com
becomes the nameserver for the example.com domain. Inspired by this success, we would like to extend its impact to other domains. Namely, in the spoofed response triggered by a query for www.example.com, we would like to add additional entry in the Authority section (see the following), so ns.attacker32.com is also used as the nameserver for google.com. The goal of this task is to see whether the entries we provide in the authority section are cached on the local DNS server or not and explain your results.

```
;; AUTHORITY SECTION:
example.com.            259200  IN      NS      ns.attacker32.com.
google.com.             259200  IN      NS      ns.attacker32.com.
```

Also fill in the appropriate interface name in the code for task 4 as done in previous tasks.

Before doing the attack, please remember to clear the cache on the local DNS server first.

On the local DNS server's terminal run the command:

 # rndc flush

Now run the program in the attacker machine and show your spoofed information in the reply. Also show the **spoofed packet captured on wireshark** and the cache of the local DNS server and explain your results.

On the attacker terminal run the command:

 # python3 task4.py

On the victim terminal run the command:

 # dig www.example.com

Provide a screenshot of your observation.

Provide a Wireshark screenshot of your observation as well.

Please also check the cache on the local DNS server and see whether the spoofed NS record is in the cache or not.

To view the cache on the local DNS server we can use the rndc command to dump the cache.

On the local DNS server's terminal run the commands:

 # rndc dumpdb -cache

 # cat /var/cache/bind/dump.db | grep example

Describe your results and explain as to why you got the results you did.

Task 5: Spoofing Records in the Additional Section

In DNS replies, there is a section called Additional Section, which is used to provide additional information. In practice, it is mainly used to provide IP addresses for some hostnames, especially for those appearing in the Authority section. In particular, when responding to the query for www.example.com, we add the following entries in the spoofed reply, in addition to the entries in the Answer section. The goal of this task is to spoof some entries in this section and see whether they will be successfully cached by the target local DNS server.

```
;; AUTHORITY SECTION:
example.com.            259200  IN    NS    ns.attacker32.com.
example.com.            259200  IN    NS    ns.example.com.

;; ADDITIONAL SECTION:
ns.attacker32.com.      259200  IN    A     1.2.3.4    ①
ns.example.net.         259200  IN    A     5.6.7.8    ②
www.facebook.com.       259200  IN    A     3.4.5.6    ③
```

Before doing the attack, please remember to clear the cache on the local DNS server first.

On the local DNS server's terminal run the command:

 # rndc flush

Now run the program in the attacker machine and show your spoofed information in the reply. Also show the **spoofed packet captured on wireshark** and the cache of the local DNS server and explain your results.

The victim machine sends out a DNS query to the local DNS server using the dig command. Before launching the attack, keep wireshark open to capture the response packet.

On the attacker terminal run the command:

 # python3 task5.py

On the victim terminal run the command:

 # dig www.example.com

Provide a screenshot of your observation.

Also show the spoofed packet captured on wireshark.

Please also check the cache on the local DNS server and see whether the spoofed NS record is in the cache or not.

To view the cache on the local DNS server we can use the rndc command to dump the cache.

On the local DNS server's terminal run the commands:

 # rndc dumpdb -cache

 # cat /var/cache/bind/dump.db | grep example

Describe your results and explain as to why you got the results you did.

Submission

You need to submit a detailed lab report to describe what you have done and what you have observed; you also need to provide explanations to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits.