# 50.020 Network Security Lab 3: Remote DNS Attack (Kaminsky Attack)

## Task 1: Configure the User VM

IP Addresses

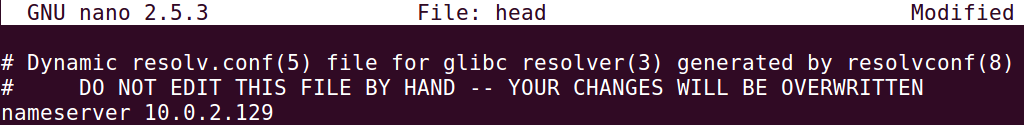
User VM 10.0.2.128

Local DNS Server 10.0.2.129

Attacker VM 10.0.2.130

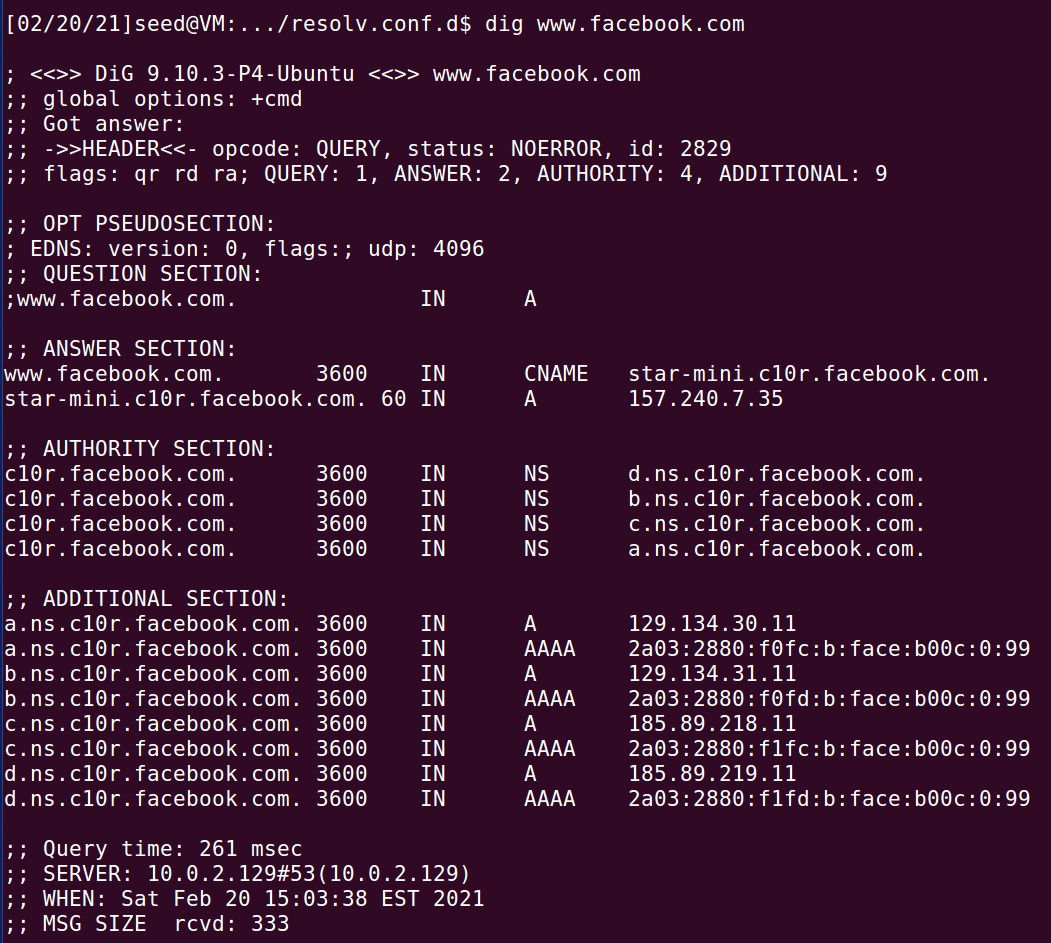
Setup: User Machine

Following the lab’s instructions, the /etc/resolvconf/resolv.conf.d/head file is edited to include the line “nameserver 10.0.2.129” to specify 10.0.2.129 as the local DNS server:



The command “sudo resolvconf -u” is then run for the changes to take effect.

A simple dig command is used to resolve the hostname of a random website:



The IP address of the DNS server queried is stated at the bottom of the server, which is 10.0.2.129. This shows that the setup is successful as the response is indeed from 10.0.2.129.

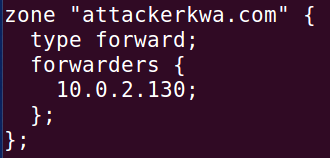
## Task 2: Configure the Local DNS Server (the Server VM)

Step 1: Remove the example.com zone

There is no need for this step because we did not do the “Local DNS Attack Lab”. The /etc/bind/named.conf file does not contain an zone entry for example.com.

Step 2: Set up a forward zone

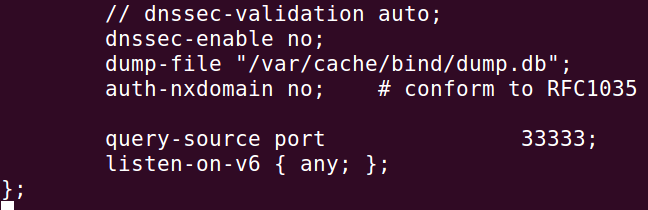
Since the ns.attacker32.com server does not belong to us, we cannot configure the DNS server running on ns.attacker32.com, so if we query this attack server at the end of the DNS query process, we cannot carry out the forgery attacks. To counter this, we need to reflect the IP address of our Attack VM as the nameserver for the attacker32.com domain so that our Attack VM can act as the malicious nameserver. Thus, there is a need to edit the /etc/bind/named.conf configuration file of the BIND9 server to set up a forward zone:



This is done to forward any queries to attackerkwa.com nameserver to the Attacker VM (10.0.2.130).

Step 3: Configure a few options

The /etc/bind/named.conf.options file is edited to configure where to dump the DNS cache, to turn off the DNSSEC protection mechanism (which protects against spoofing attacks), and to specify 33333 as the source port for spoofing DNS queries later:



Step 4: Restart DNS Server

The “sudo service bind9 restart” command is run to restart the BIND9 DNS server.

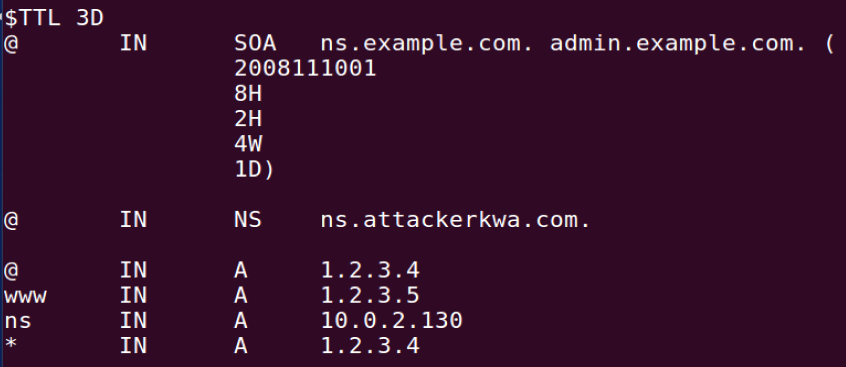
## Task 3: Configure the Attacker VM

Step 1: Download the attackerkwa.com.zone and example.com.zone files from the lab’s website.

Done.

Step 2: Modify these files accordingly based on students’ actual network setup (e.g., some IP addresses need to be changed).

The example.com.zone file is edited so that the nameserver of example.com will point to the Attack VM’s IP address (10.0.2.130).



The same is done for the attackerkwa.com.zone file so the nameserver of ns.attackerkwa.com will point to the Attack VM’s IP address (10.0.2.130):

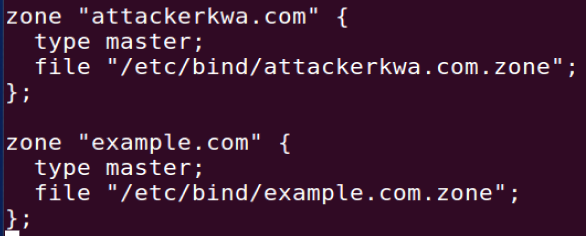


Step 3: Copy these two files to the /etc/bind folder.

Done.

Step 4: Add the following entries to /etc/bind/named.conf

The following zone entries are added to the /etc/bind/named.conf file:



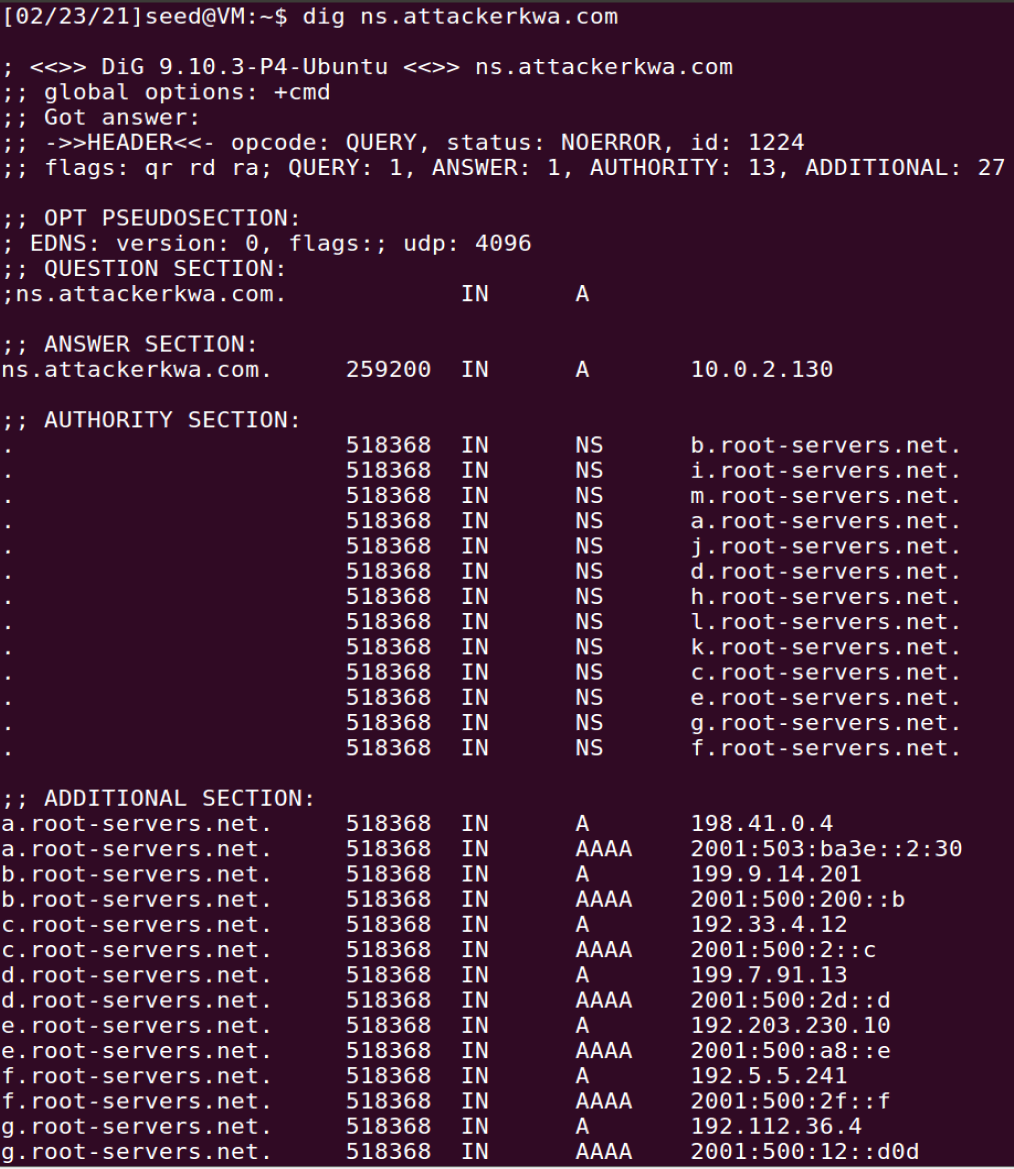
Step 5: Restart the DNS server

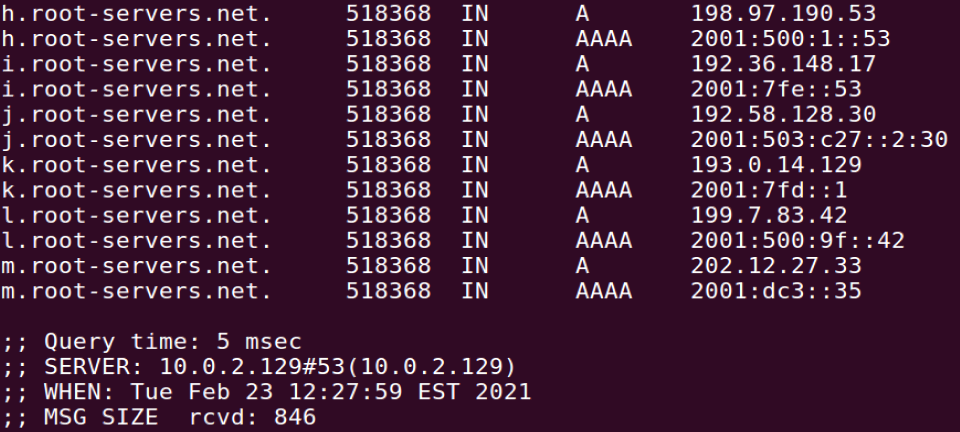
The command “sudo service bind9 restart” is run to restart the BIND9 DNS server.

## Task 4a: Testing the Setup

Get the IP address of ns.attackerkwa.com

Sending the query to our local DNS server, we will get the Attacker VM’s IP address of 10.0.2.130:

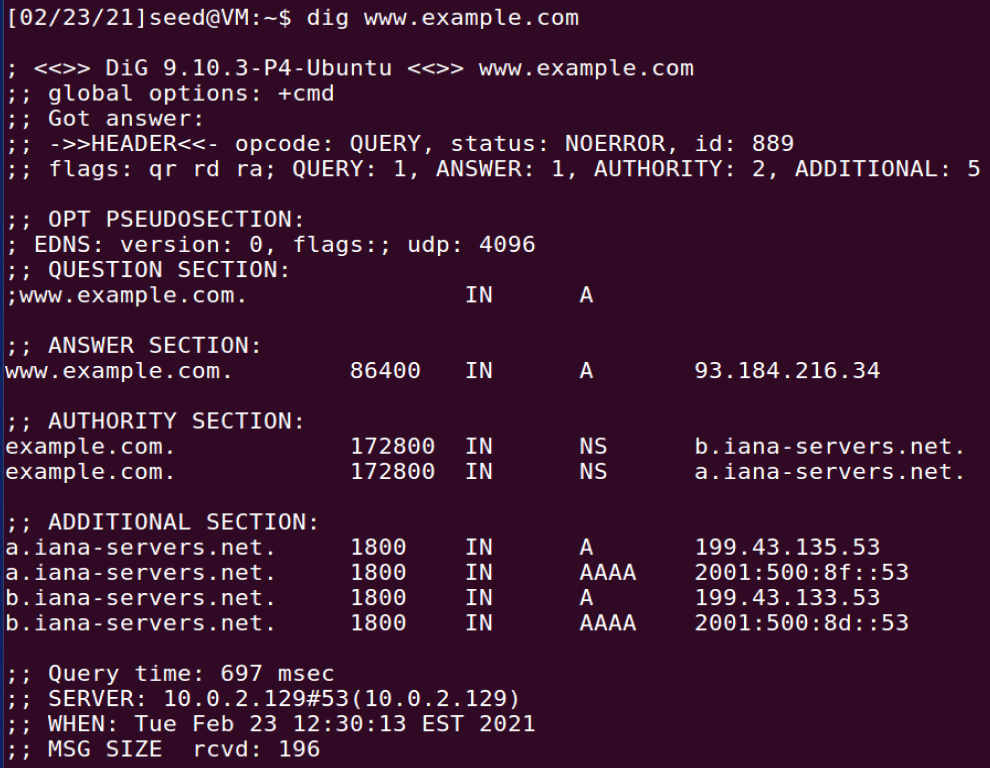




After the DNS server forwards the request to the Attacker VM due to the forward zone entry, the attacker will respond with the IP address of the nameserver stated in the attackerkwa.com.zone file. In this case, this IP address is 10.0.2.130.

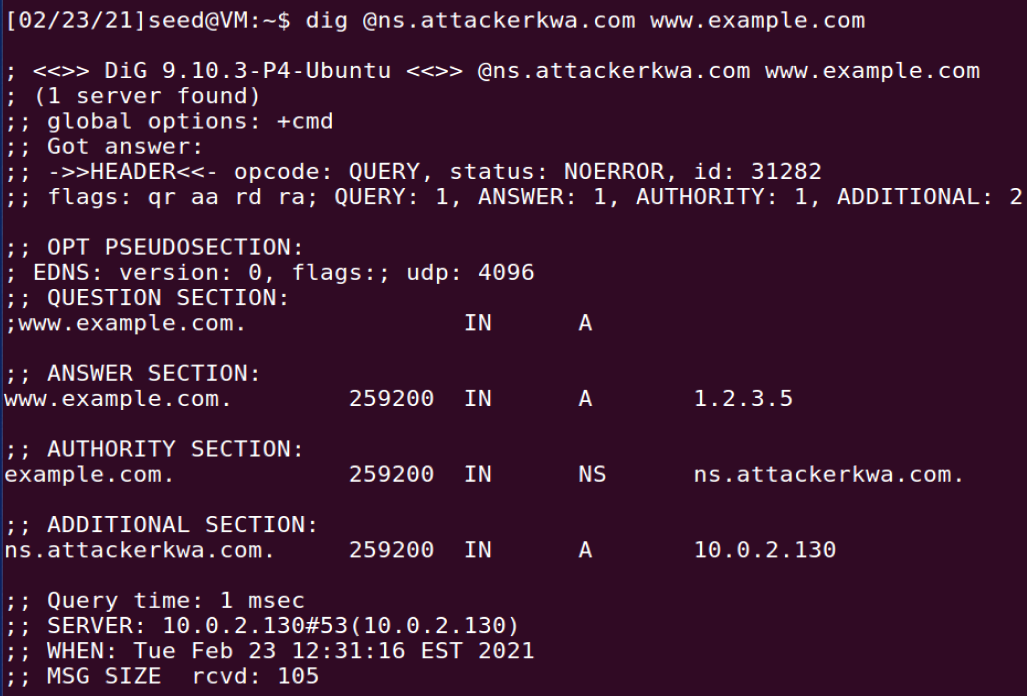
Get the address of www.example.com

Sending the query to our local DNS server, we will get www.example.com’s actual IP address of 93.184.216.34:



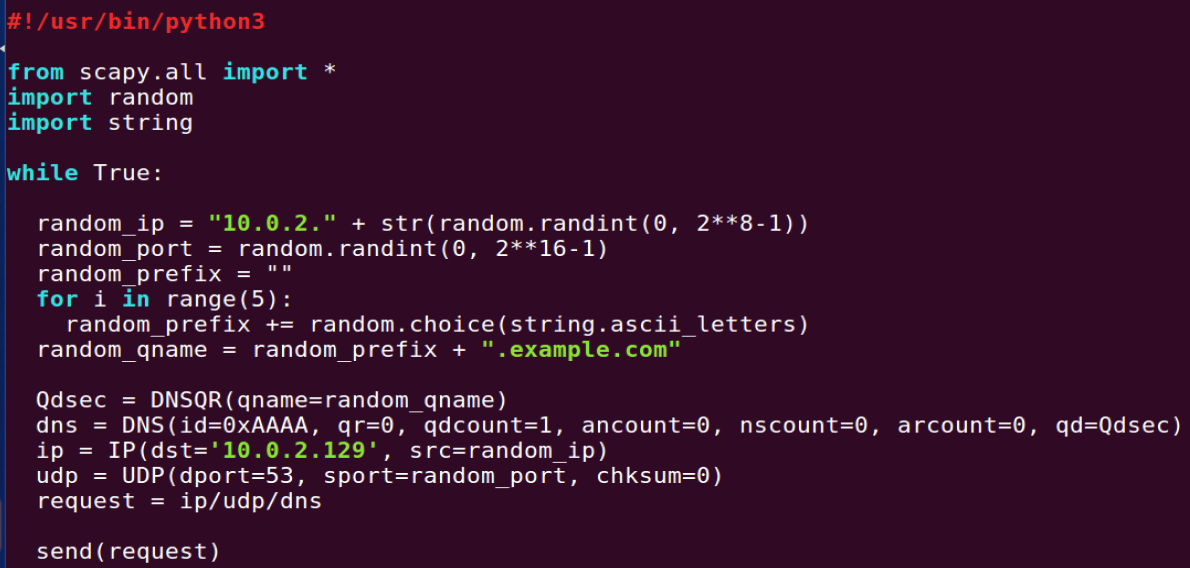
This is because the local DNS server (10.0.2.129) queried example.com’s official nameserver.

However, if we send the query directly to ns.attackerkwa.com, we will get the fake IP address of 1.2.3.5 as stated by the Attack VM’s DNS server:



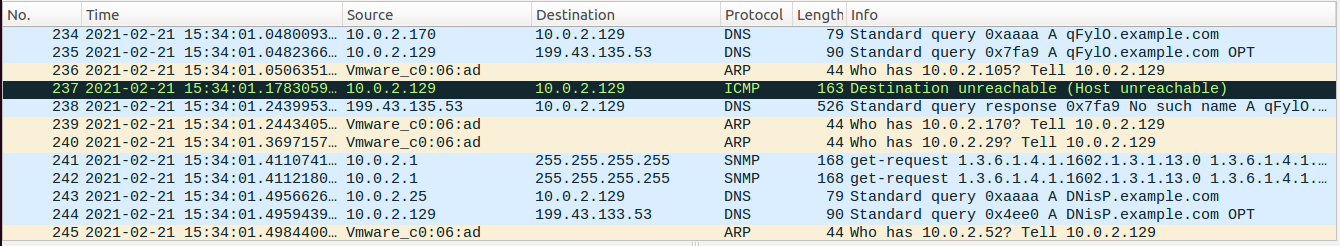
## Task 4b: Construct DNS request

The following code in request.py is run on the Attacker machine to continuously send DNS requests to the victim DNS server:



The prefix of the hostname is randomised for each packet to avoid the caching effect. The source IP address is also randomised for each packet to avoid 10.0.2.130 getting noticed as the Attacker, but this IP address has to be within the range of IP addresses for the LAN subnet in order for the Victim DNS server to respond and continue the query.

The Victim DNS Server responds to every packet by sending a query to the example.com nameserver:

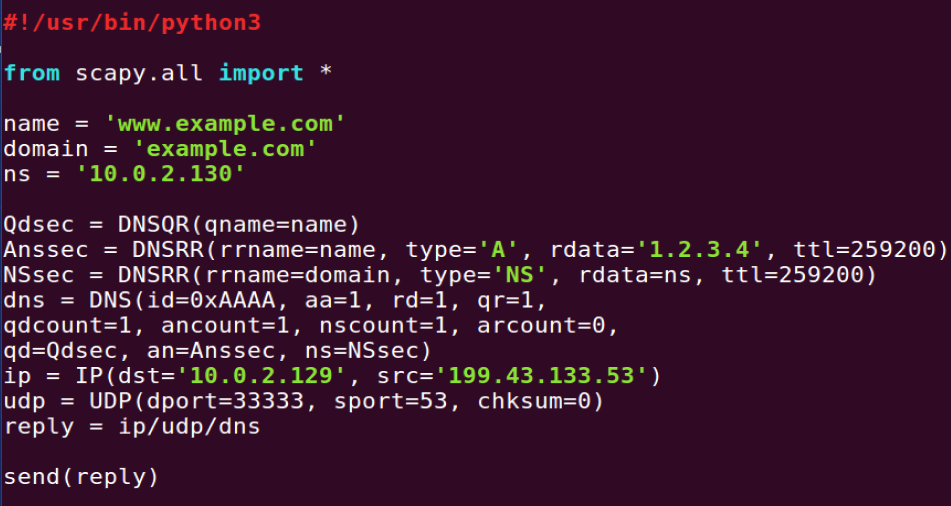


The example.com nameserver (199.43.133.53 and 199.43.135.53) responds to the Victim DNS Server and the Victim DNS Server attempts to send this response back to the machine that send the initial DNS query but is unable to because the source IP address is spoofed.

Since the Attack VM’s queries are shown to trigger the target DNS server to send out corresponding DNS queries, our DNS request is constructed correctly.

## Task 5: Spoof DNS Replies

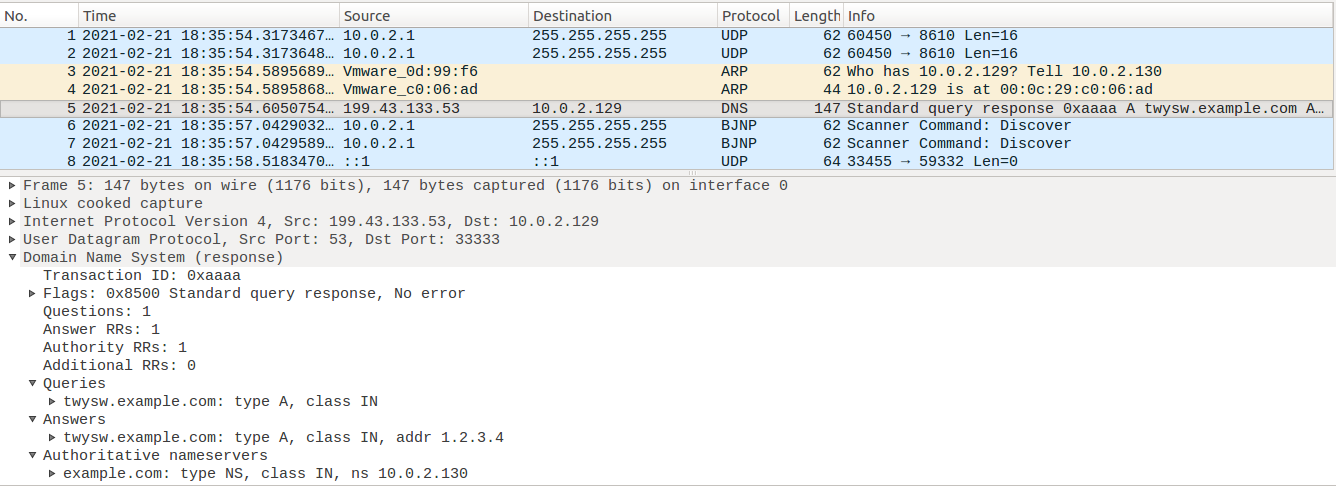
The following code in reply.py is run on the Attacker VM so as to spoof DNS replies:



The name is set up be ‘www.example.com’ temporarily but this should be set as the hostname to be resolved in the DNS request packet sent out by the Attacker VM.

The domain is set to be ‘example.com’ and ns is set to be 10.0.2.130 because we want to include an entry of type NS in the Additional section that indicates ns.attacker32.com (10.0.2.130) as the nameserver for the example.com domain. The destination IP address is set to be 10.0.2.129 because we are spoofing DNS reply packet on the way back to the Victim DNS Server. The source IP address is set to be 199.43.133.53 because this is the IP address of one of the nameservers of the example.com domain. The destination port is set to be 33333 because we configured the source port for all DNS queries on the Victim Nameserver to be this number. The source port is set to be 53 because the Victim DNS Server would query port 53 on the example.com nameserver for DNS queries.

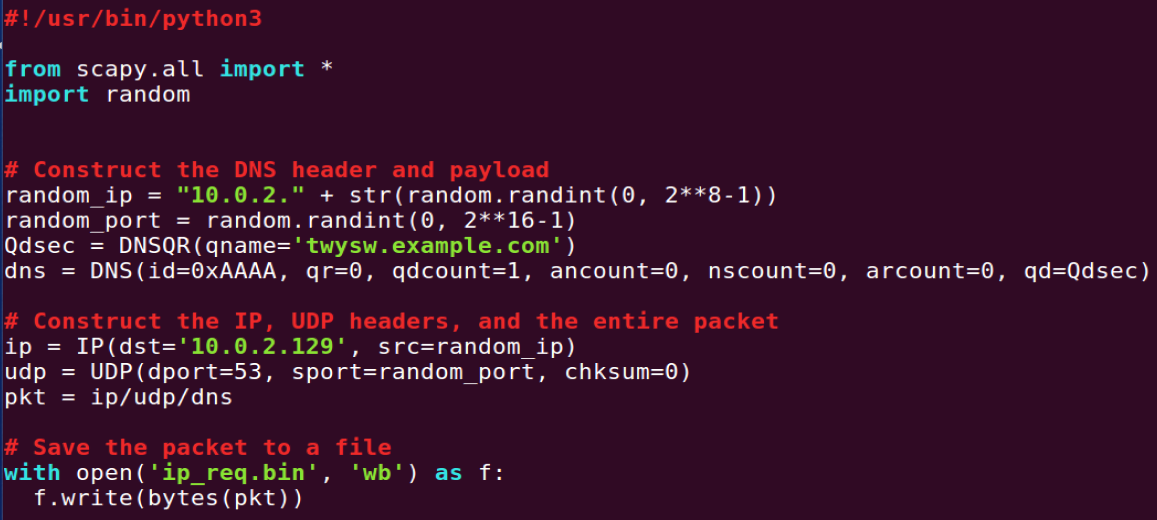
The following screenshot shows the packet capture of the spoofed DNS response packet on the Victim DNS Server:



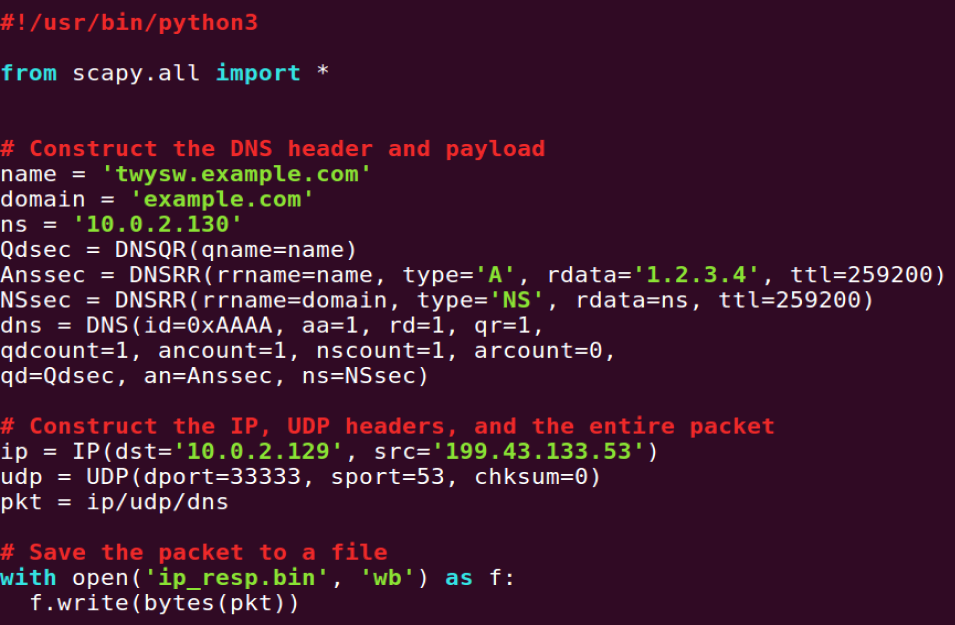
The fields of the DNS response packet are filled in nicely, according to how they should be spoofed.

## Task 6: launch the Kaminsky Attack

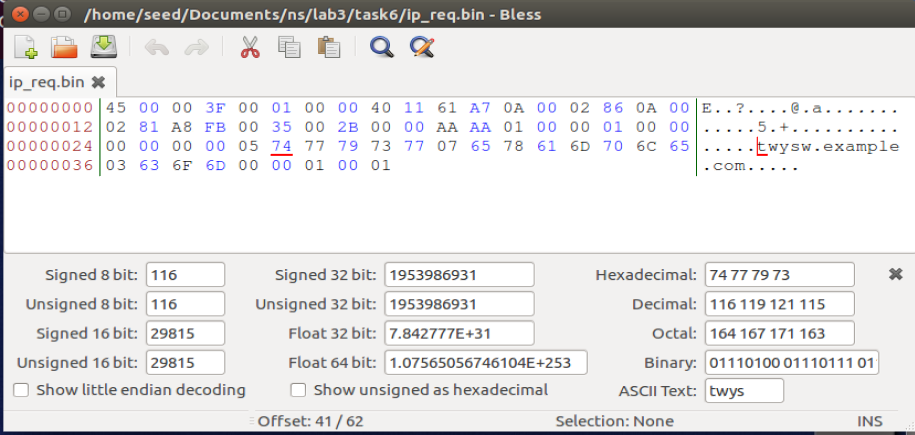
The following code in generate\_dns\_request.py is run on the Attacker VM to save the request packet’s information to a file called ip\_req.bin:

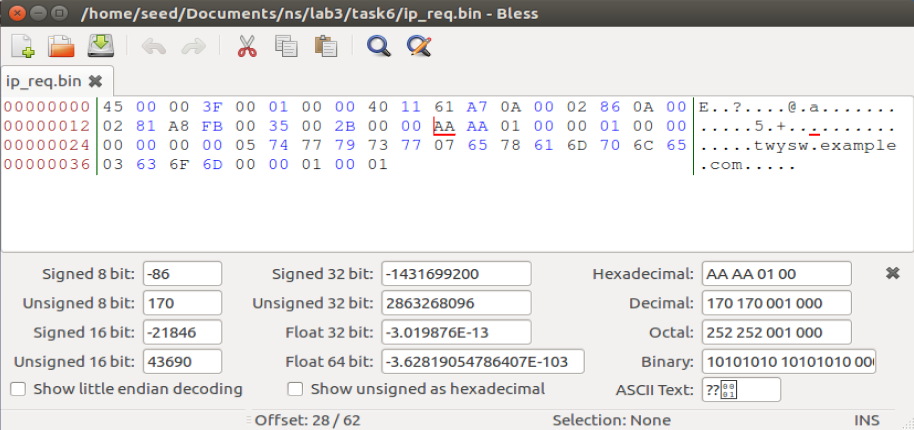


The following code in generate\_dns\_reply.py is run on the Attacker VM to save the spoofed reply packet’s information to a file called ip\_resp.bin:



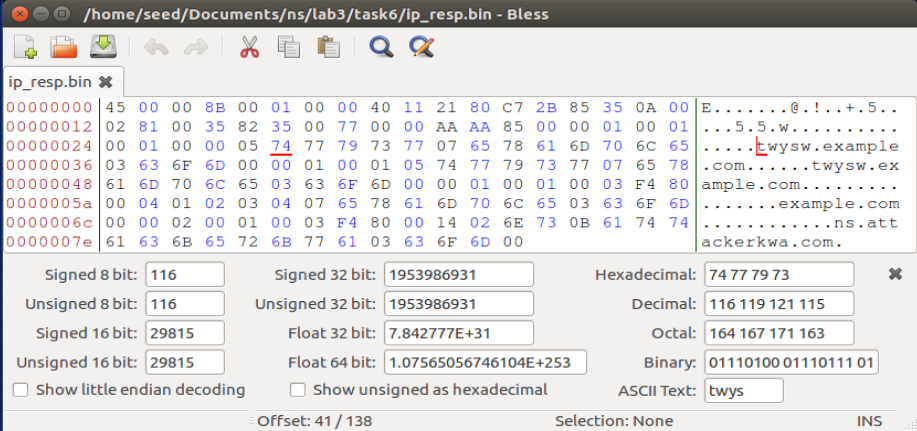
To find out which byte offset in the request packet binary file, the binary editor program *bless* is used to view the binary file ip\_req.bin:

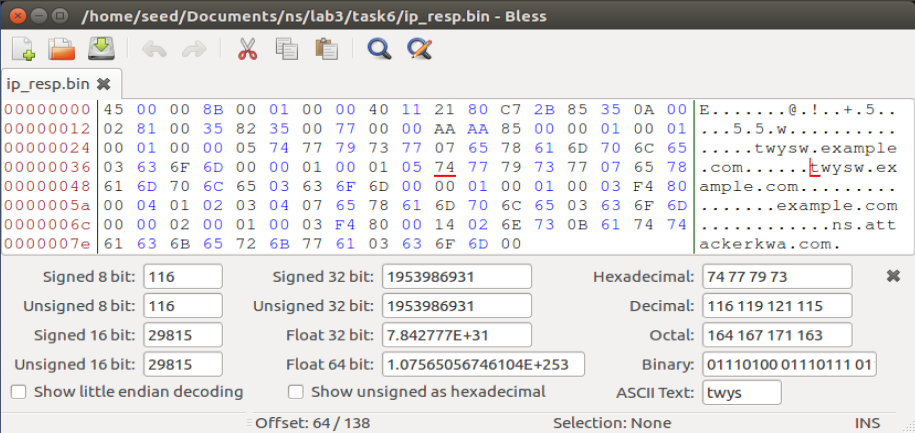


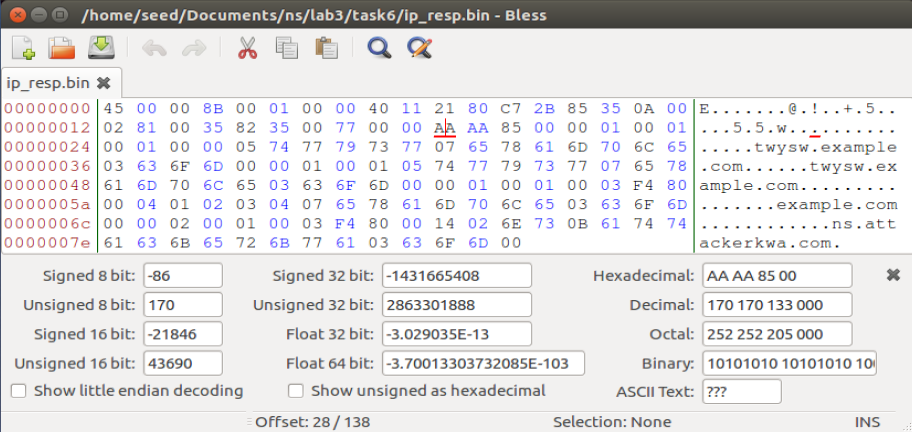


This shows that the hostname prefix offset and the Transaction ID offset is at byte 41 and 28 respectively.

The same is done for the ip\_resp.bin file to find the 2 offsets for the reply packet binary file:







This shows that the hostname prefix offsets are at bytes 41 and 64 and the Transaction ID offset is at byte 28.

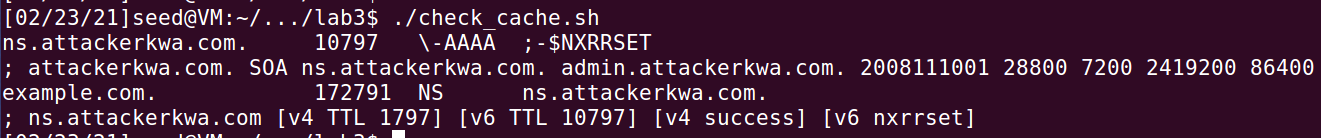
After this, the C code in attack.c is edited to load the binary packets into the C code and then send out the packets continuously. Please refer to the attached source code.

The C code is compiled to produce the binary file attack. This binary program is then run (with sudo because the CAP\_NET\_RAW capability is required) on the Attacker VM to send DNS requests to the Victim DNS Server and spoof the replies from example.com.

The following shell script in check\_cache.sh is run on the Victim DNS Server to check if the attack works and the cache is poisoned:



At the beginning, there is no output, which showed that the attack was not successful yet. After running the script a few times, the following output is finally shown:



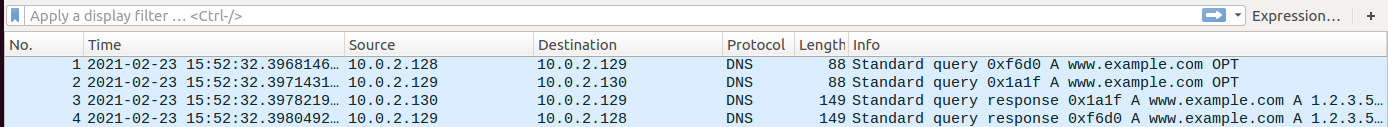
The NS entry in the cache shows that the spoofed DNS response has been successfully accepted by the DNS Server and its cache is successfully poisoned.

## Task 7: Result Verification

Using the User VM to query the Victim DNS Server, we get the fake IP address of [www.example.com](http://www.example.com) as shown:



Taking a closer look at what happens at the Victim DNS Server:



The packet capture shows that after being queried by the User VM, the Victim DNS Server queries the Attacker VM instead of the real nameserver for the example.com domain. This is because its cache has been poisoned by the NS entry.

Using the User VM to query the Attacker VM’s DNS Server, we also get the fake IP address of [www.example.com](http://www.example.com):



The same results prove that the attack is successful.