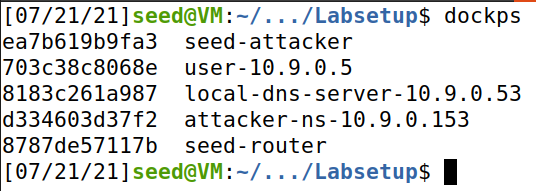
**Local DNS Attack Lab**

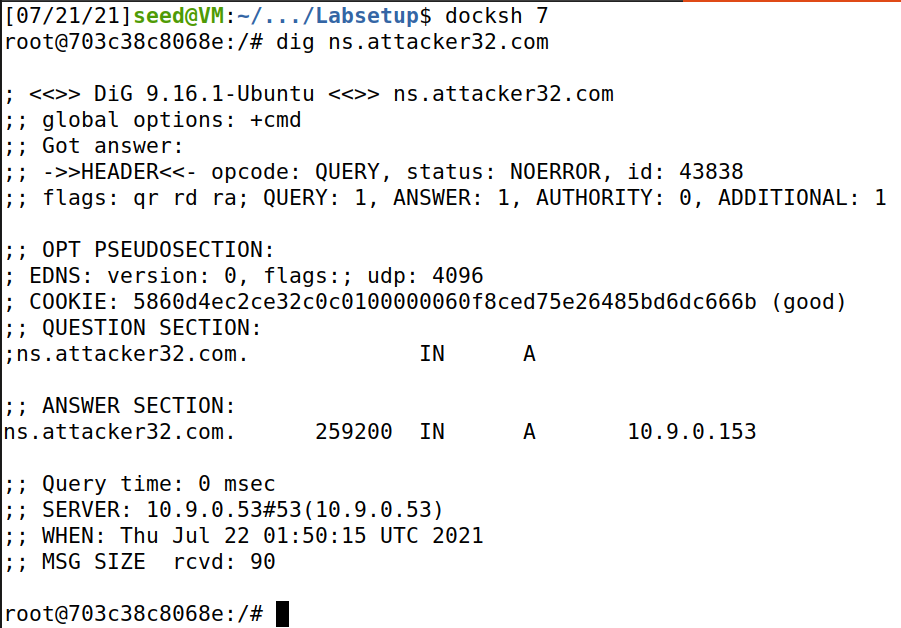
**57118113 蔡义涵**

首先查看本次实验各主机ip地址

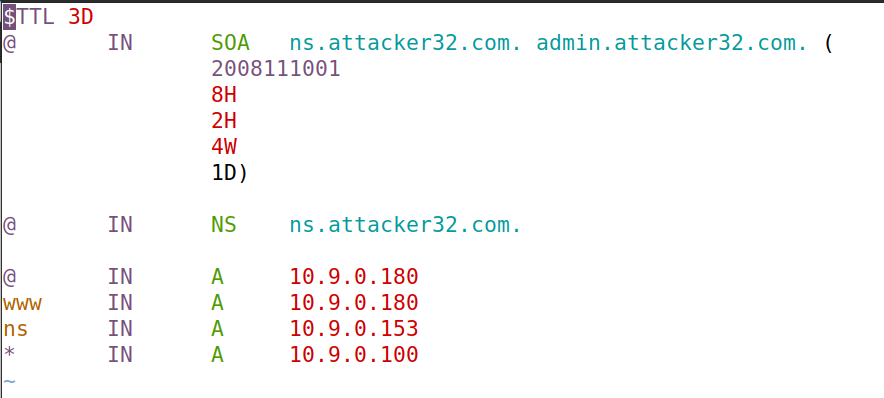


**Task0: Testing the DNS Setup**

首先在10.9.0.5上执行dig ns.attack32.com，结果如下：

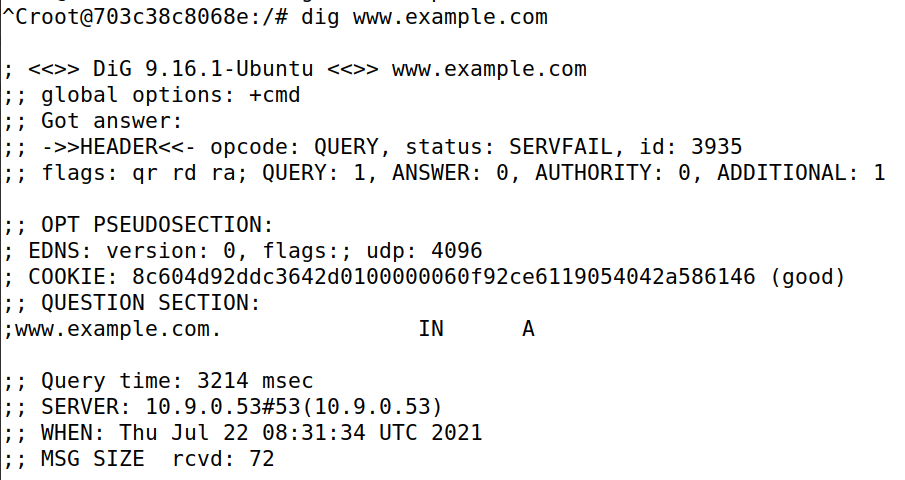


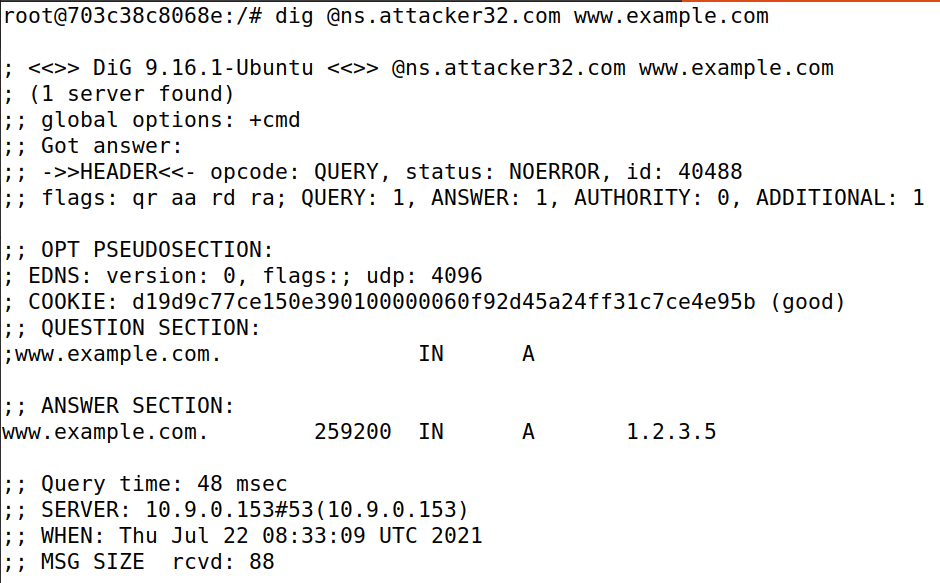
查看image\_attacker\_ns文件夹中的zone\_attacker32.com文件内容：



可以看到host上解析到的地址与attacker的域名地址相同。

接着分别通过官方域名服务器和容器中的域名服务器分别解析www.example.com。

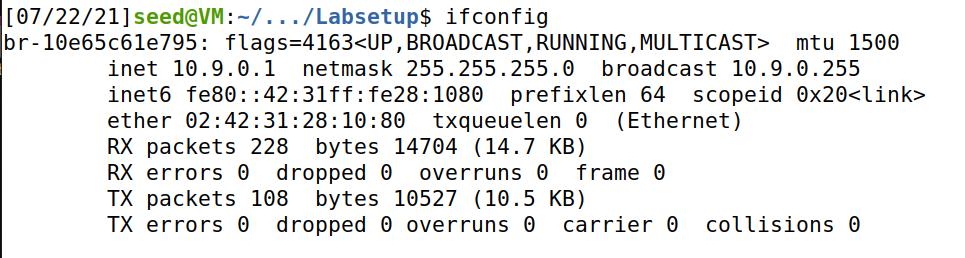




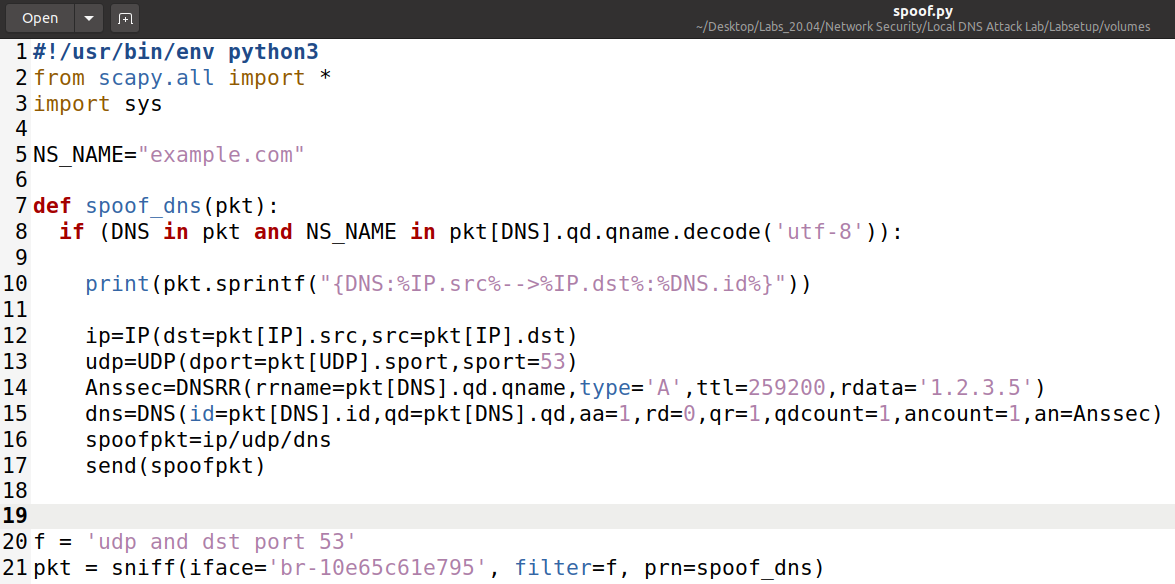
两者结果是不相同的，是因为attacker中也有相关的域名解析信息。

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

查看docker网段的网卡信息：



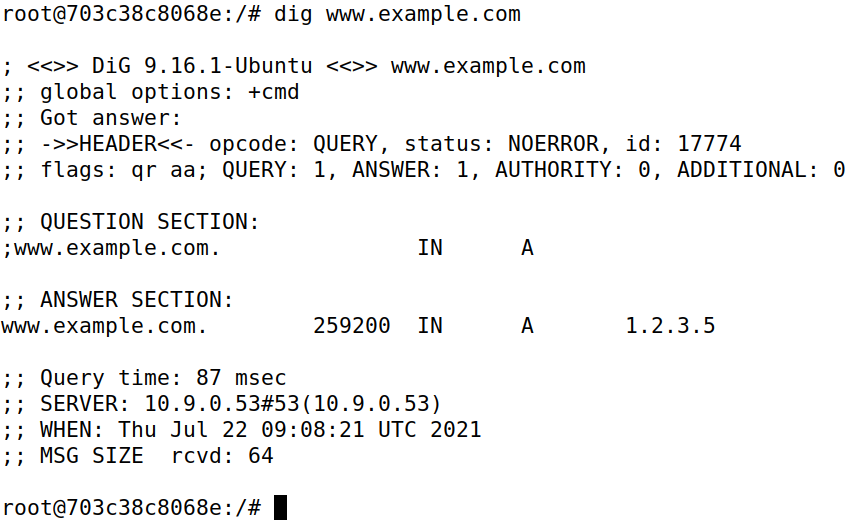
根据以上信息完成spoof.py程序：



清空本地DNS服务器上的缓存：



然后在attacker上执行spoof.py，在10.9.0.5上dig www.example.com。



可以看到攻击成功。

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

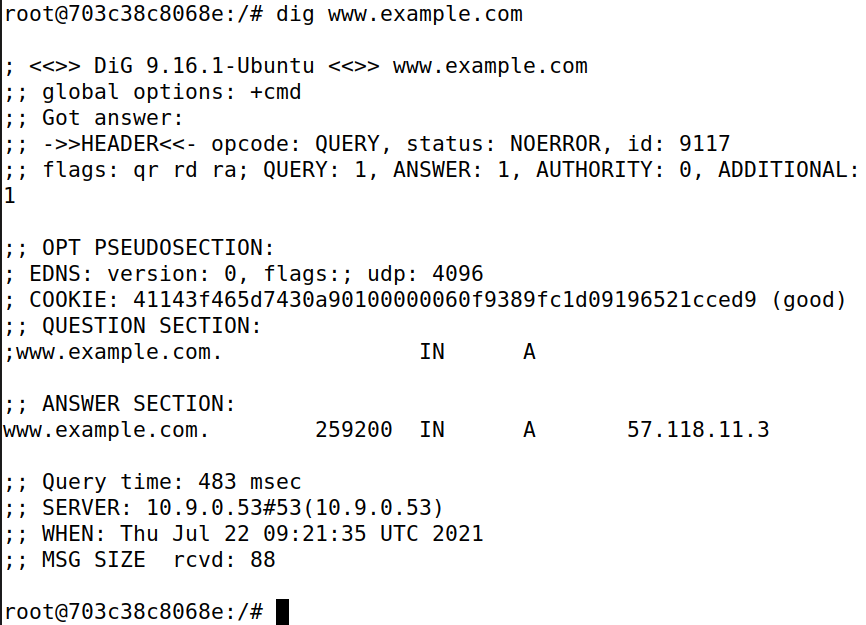
首先清空DNS服务器的缓存：



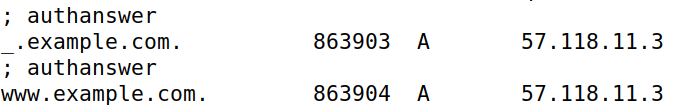
对Task1的程序进行修改，以回复本地DNS服务器收到未知DNS请求时由33333端口向外发送的DNS请求报文。伪造的答复报文中的IP地址为57.118.11.3。



在attacker上执行上述程序，同时在10.9.0.5上dig www.example.com，查看结果。



可以看到攻击成功。将DNS服务器缓存转储到文件中并进行查看：



可以看到，攻击成功。

**Task 3: Spoofing NS Records**

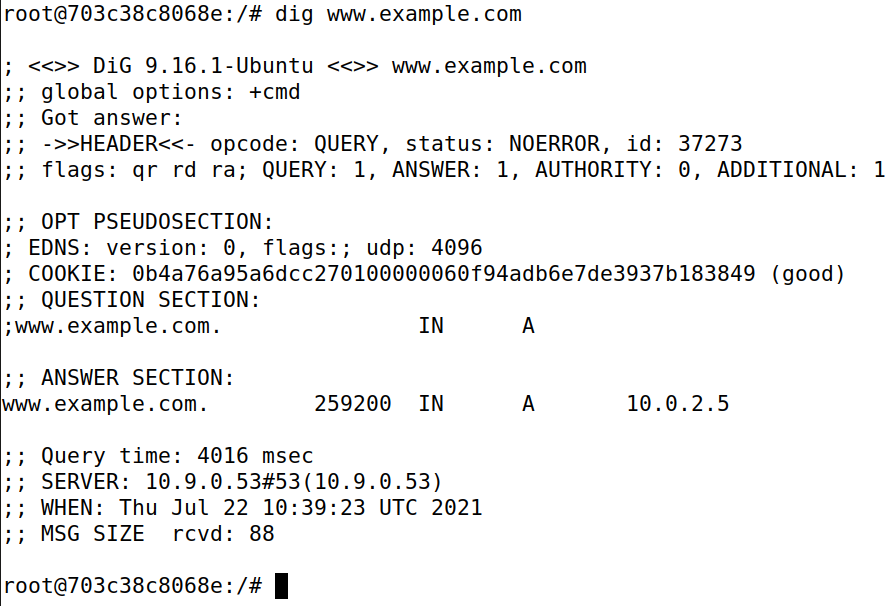
首先修改dns\_sniff\_spoof.py代码为如下：



清空DNS服务器缓存：



同时在attacker上执行上述代码，在host上执行dig www.example.com：



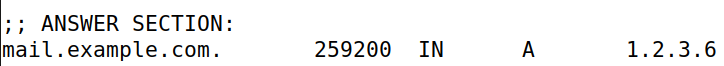
显示攻击成功。

然后停止dns\_sniff\_spoof.py程序运行，查看DNS服务器缓存。

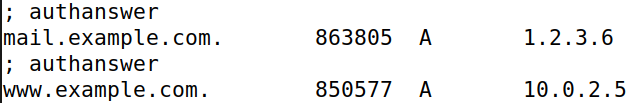


显示NS攻击成功。

在host中输入dig mail.example.com命令，查看结果：



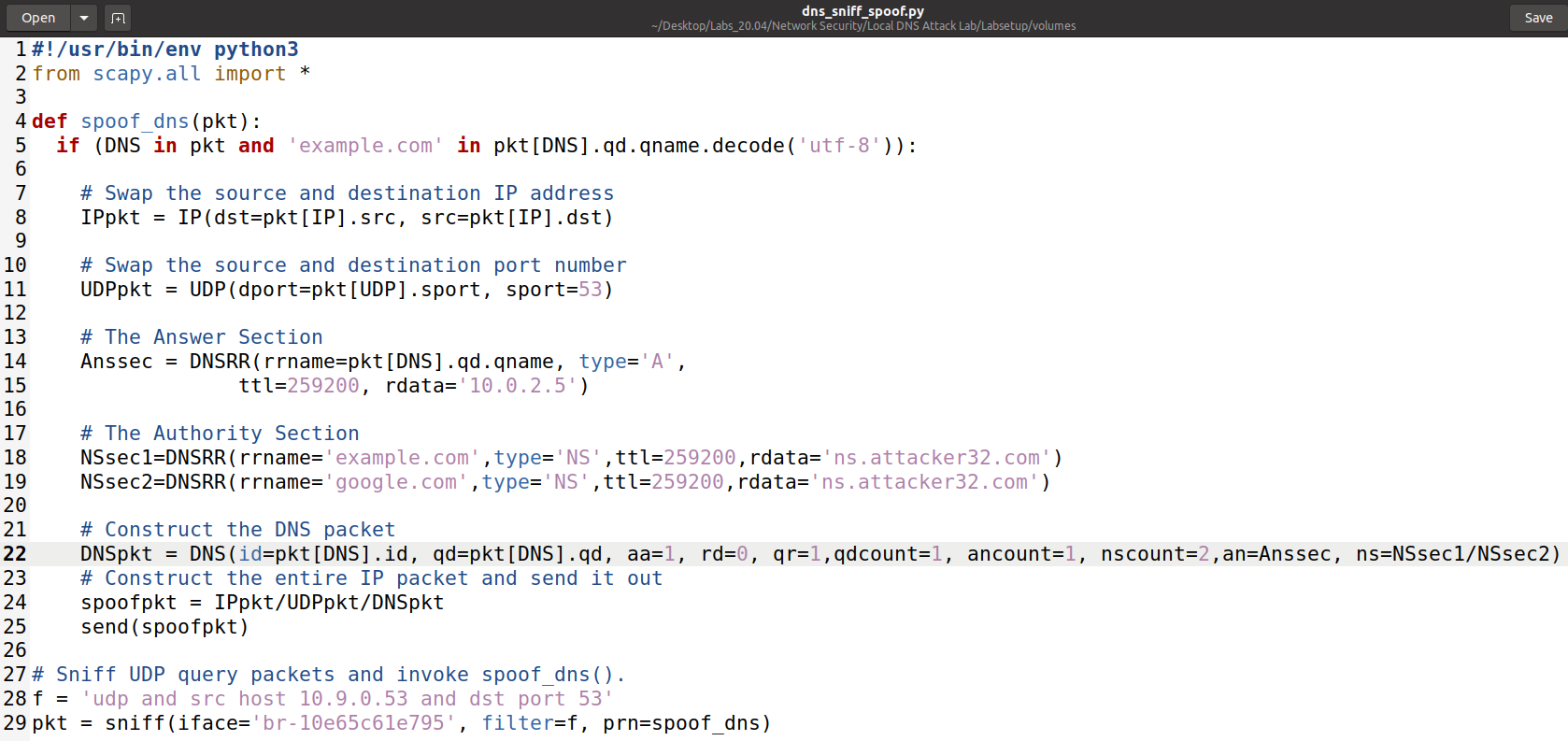
再次查看本地DNS服务器的缓存为：



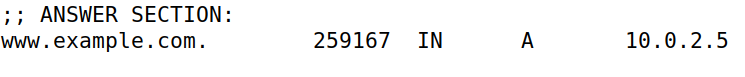
如图所示，攻击成功，成功对域名进行污染。mail.example.com解析后的结果1.2.3.6来自于先前的配置。

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

修改Task3的python程序中的负载部分，先尝试对本地DNS服务器进行攻击：



清除DNS服务器缓存，执行上述程序，并在host中dig www.example.com。



查看DNS服务器缓存：

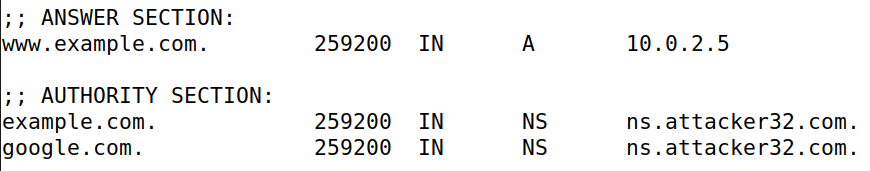


发现仅将example.com地址缓存，没有google.com的缓存，根本没有authority section。是因为使未知权威域名服务器掌管任意域是不安全的。

将程序过滤规则改为：



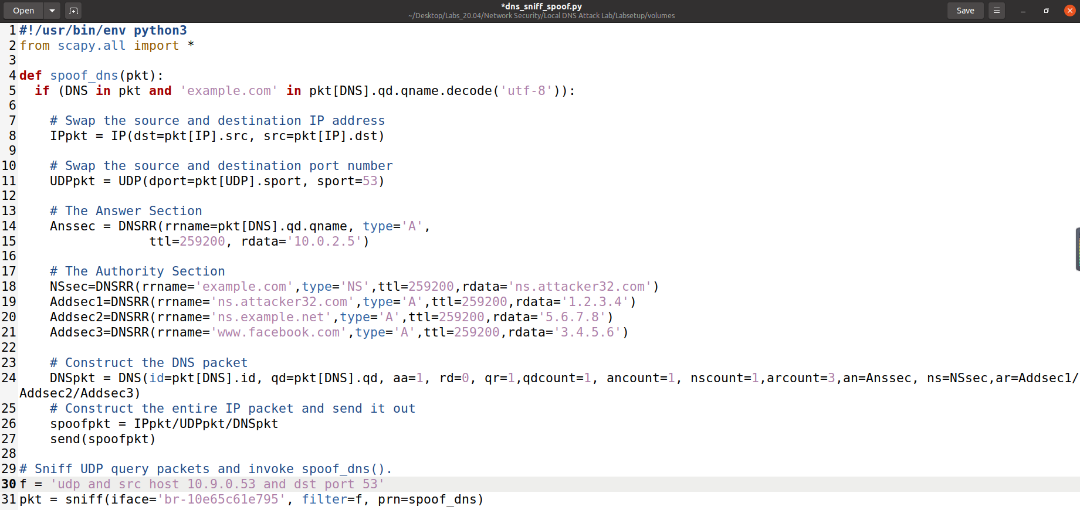
清除缓存，运行修改过滤后的脚本：



发现攻击成功。

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

修改Task3中程序：



清理DNS服务器缓存后，执行结果为：



查看DNS服务器缓存：

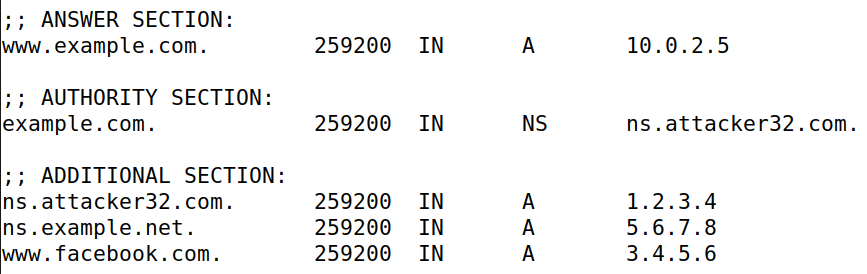


仅有www.example.com一条缓存，没有我们手动添加的additional section。

将程序过滤规则修改为：



执行结果为：



攻击成功。