

课程实践报告



**课程 网络攻防技术(314006040)**

**课 序 号 3**

**作业名称 TCP攻击实验**

**评分**

**姓名** 邓嘉怡 **学号** 2022141530010

# 作业题目

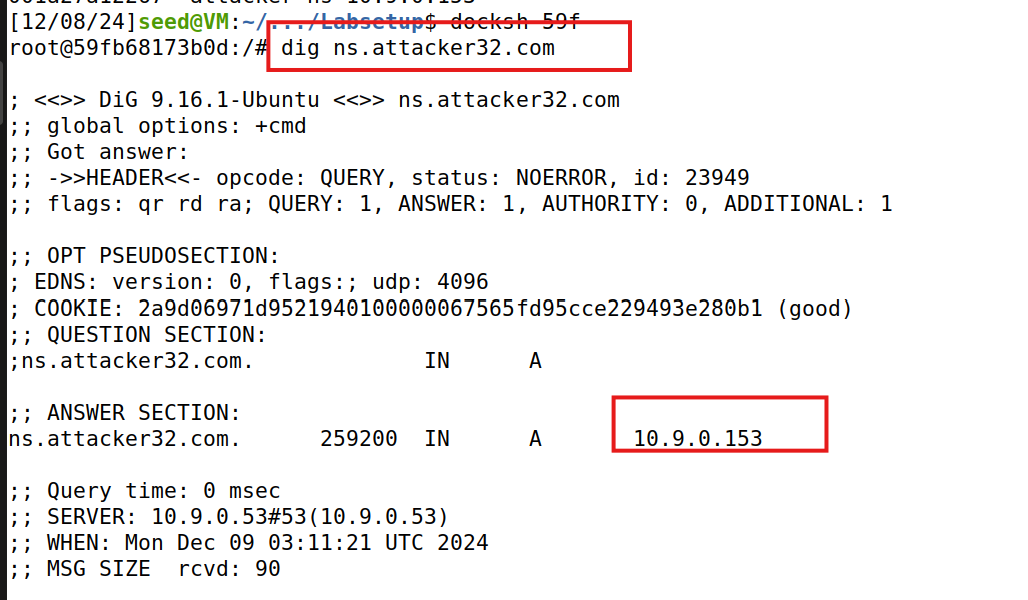
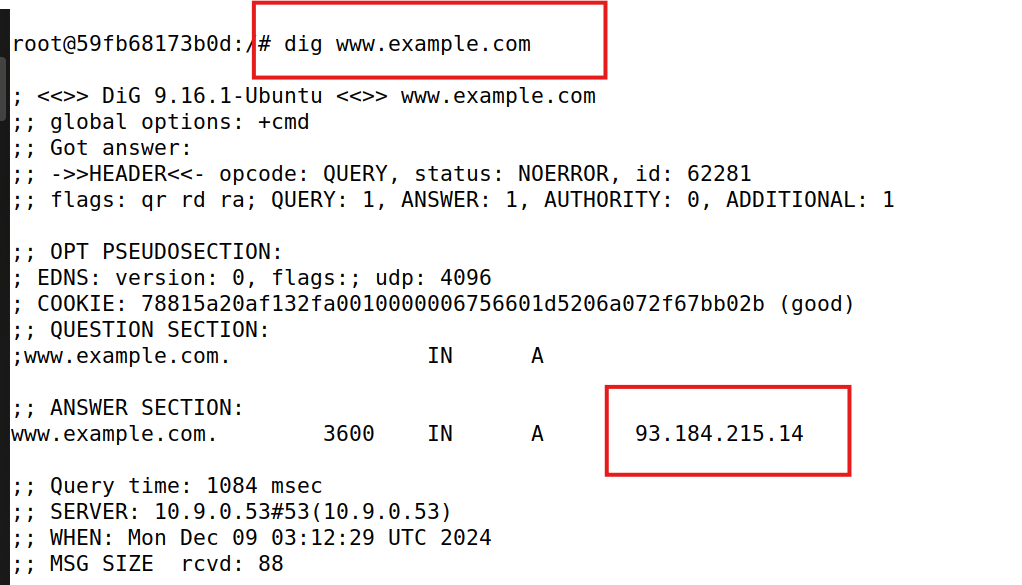
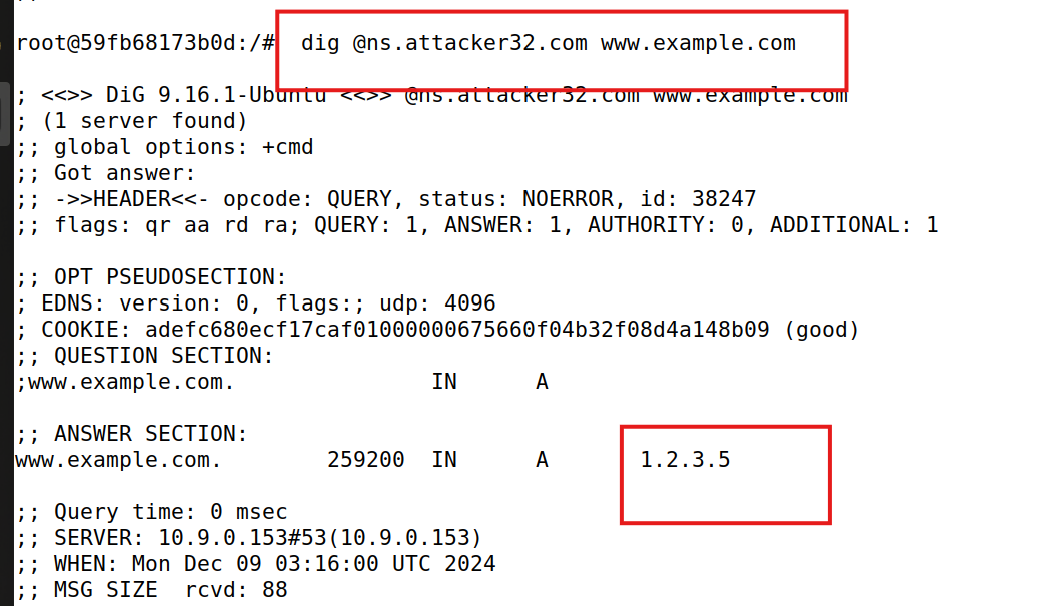
本实验的目标是让学生获得对DNS（域名系统）的各种攻击的第一手经验。DNS是互联网的电话簿；它将主机名转换为IP地址，反之亦然。这种转换是通过DNS解析实现的，这种解析发生在幕后。DNS欺骗攻击以各种方式操纵此解析过程，目的是将用户误导到其他目的地，这些目的地通常是恶意的。本实验室主要研究几种DNS欺骗攻击技术。学生将首先设置和配置DNS服务器，然后在实验室环境中的目标上尝试各种DNS欺骗攻击。

第一个大实验任务（本地DNS欺骗）中进行的攻击假设攻击者位于同一本地网络上，因此可以嗅探DNS数据包。这个假设是为了简化实验任务。第二个大实验任务为远程DNS攻击实验，攻击者在没有嗅探数据包的情况下发起远程欺骗攻击，远程攻击实验室比本地DNS欺骗实验更具挑战性。

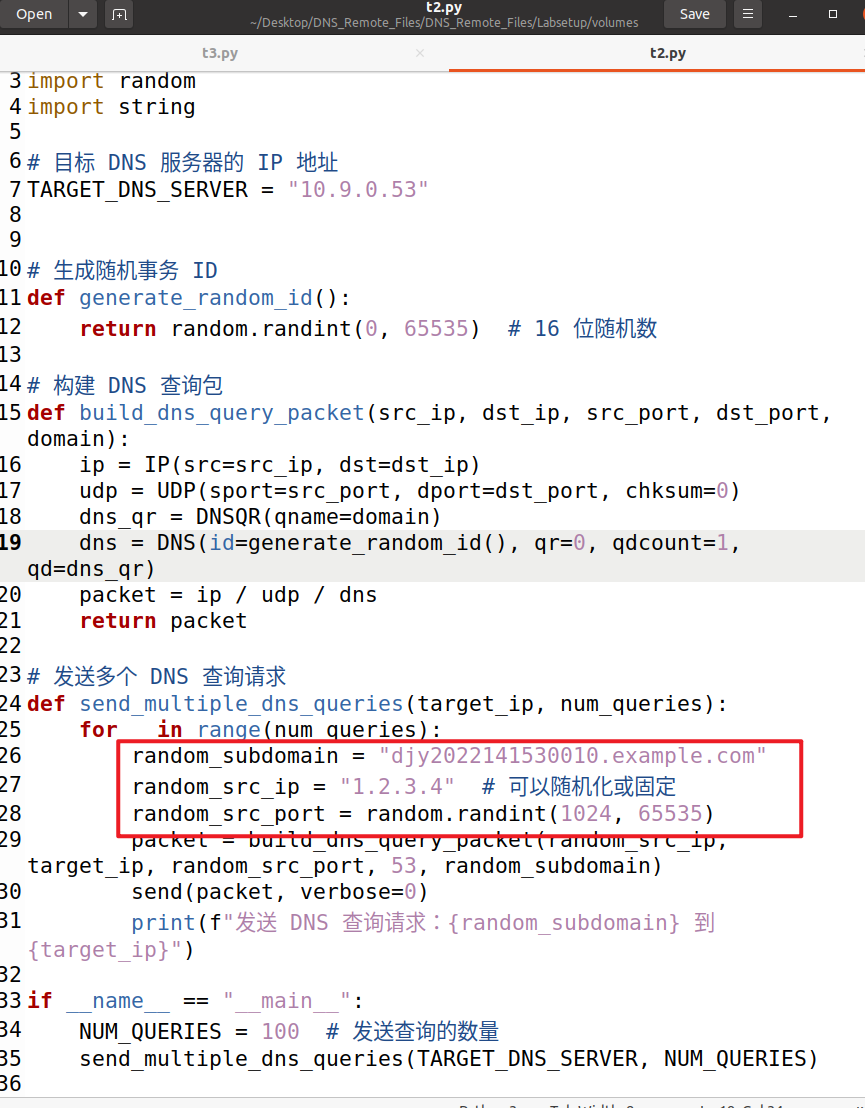
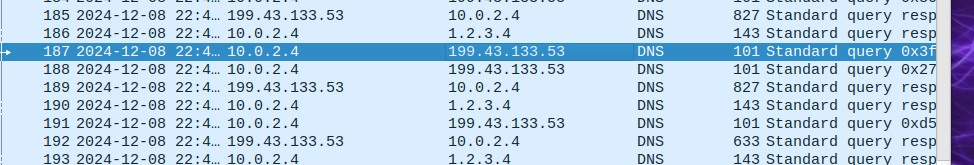
# 实验步骤及结果

**DNS Remote:**

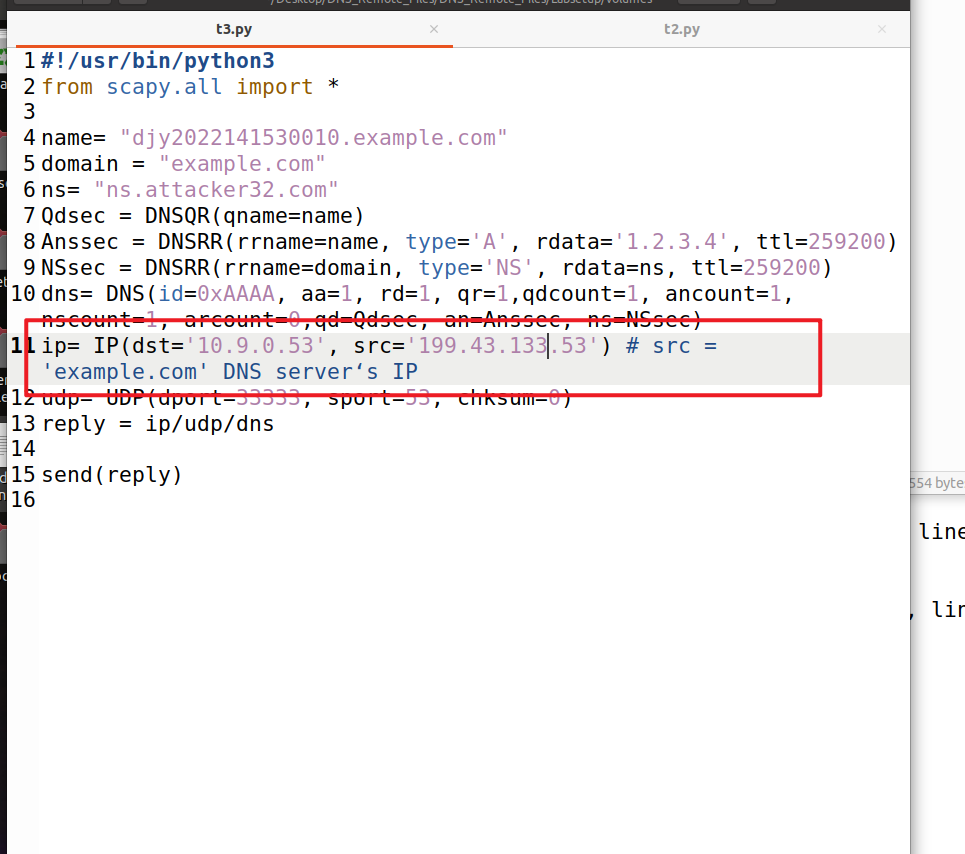
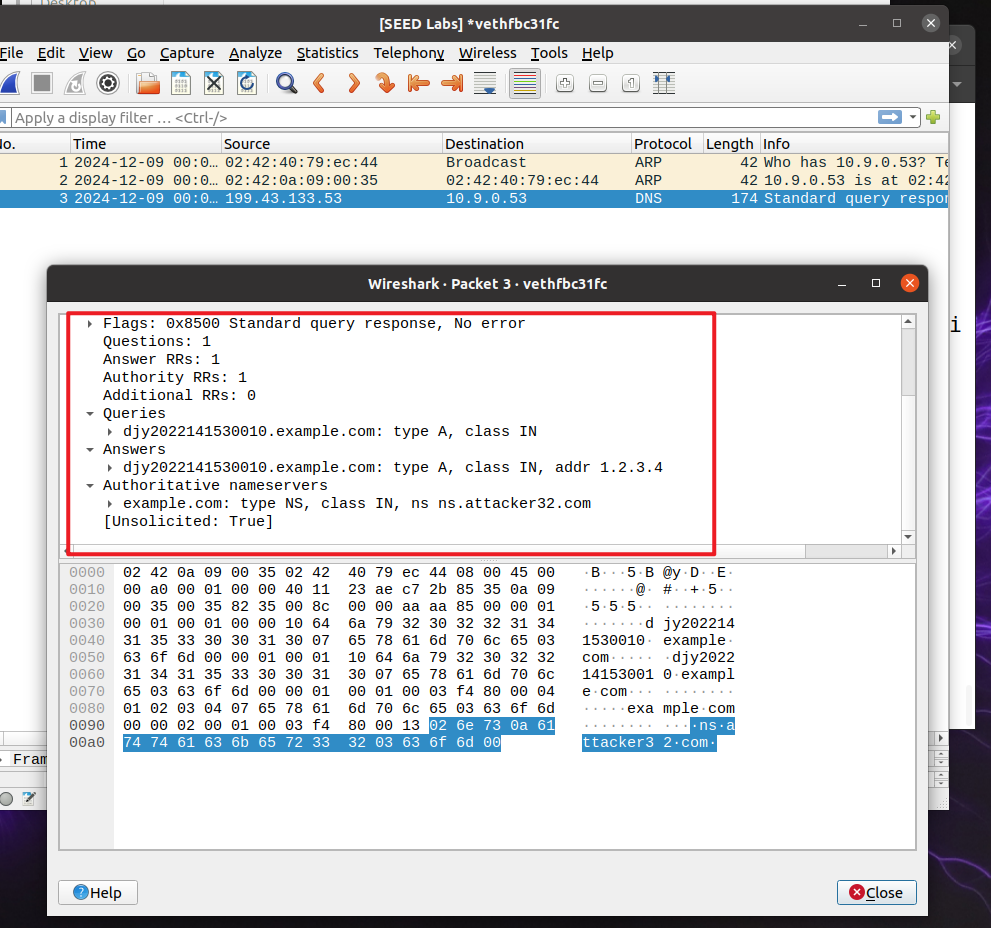
**准备：测试dns的搭建**

1. Get the IP address of ns.attacker32.com  
   当我们运行以下dig命令时，由于本地DNS服务器配置文件中添加的转发区域条目，本地DNS服务器将请求转发给攻击者名称服务器。  
   
2. Get the IP address of [www.example.com](http://www.example.com)  
   两个命名服务器现在托管example.com域，一个是该域的官方命名服务器，另一个是攻击者容器。我们将查询这两个名称服务器  
   官方服务器  
     
   攻击者服务器  
   

**实验二：Construct DNS request**  
攻击者在发送DNS查询之后，立马发送伪造的 DNS 响应，持续攻击 DNS 服务器，从而绕过 DNS 缓存机制并成功执行 DNS 缓存投毒。这里我们需要自动化发送DNS查询

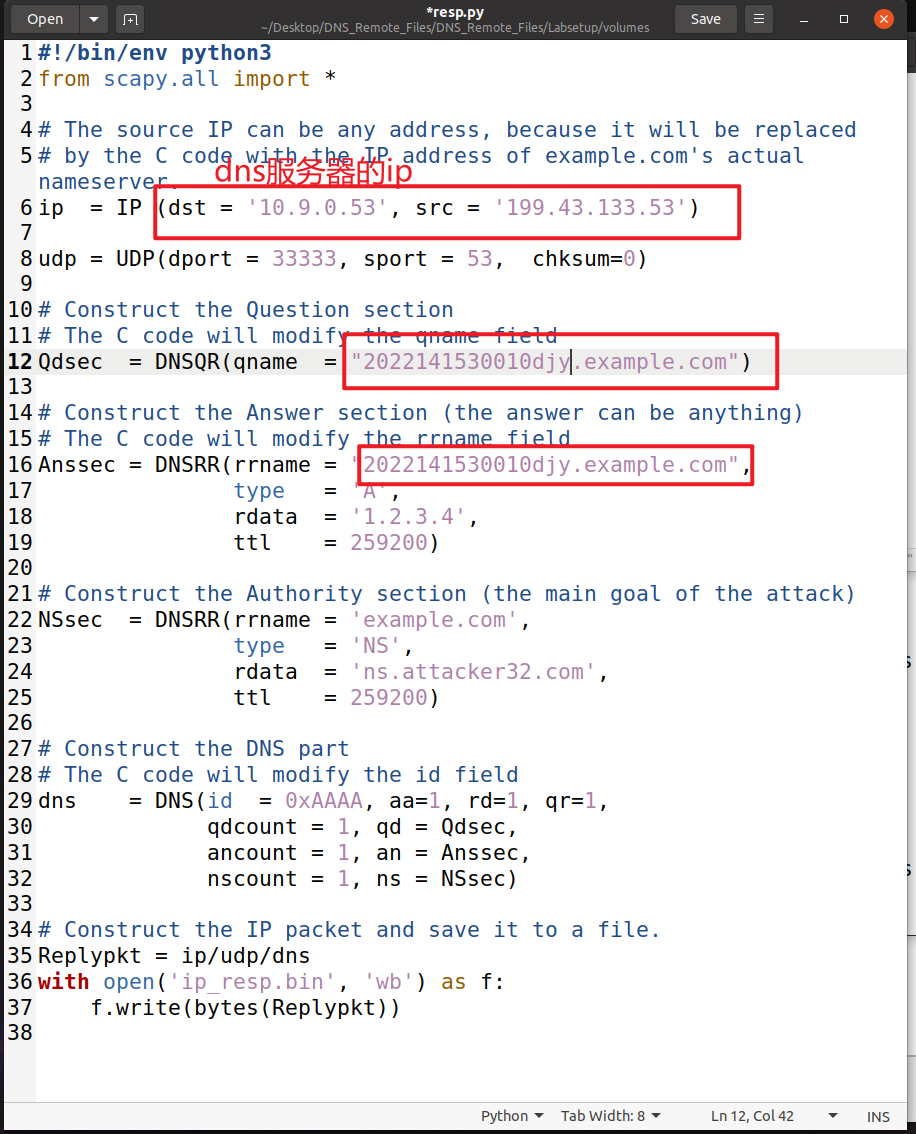
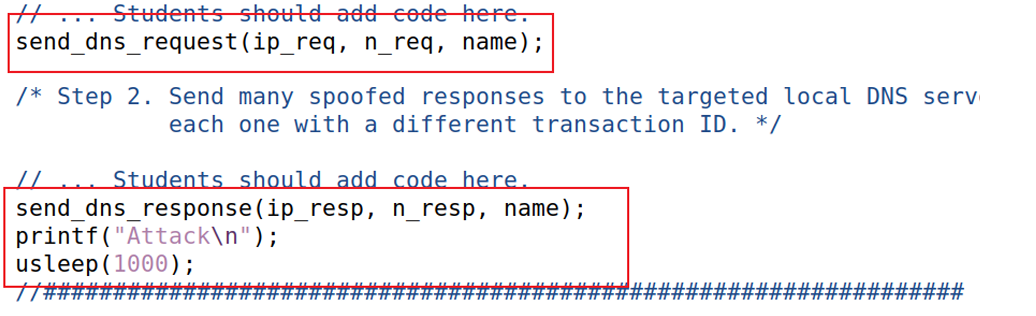
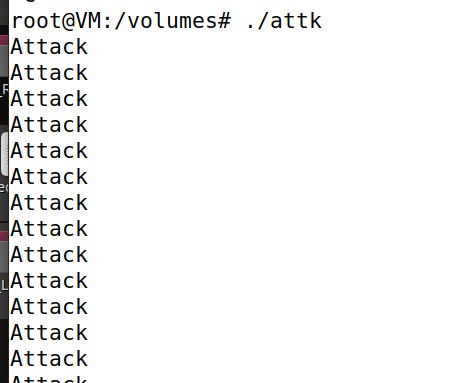
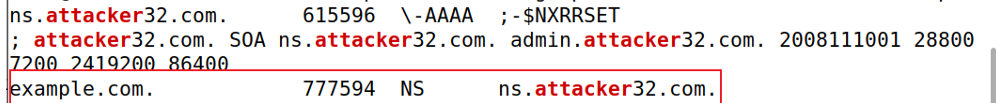
1. 编写dns请求代码  
   这是一个自动化的程序，用python编写  
   
2. 运行  
   使用wireshark抓包，发现 Local DNS Server接受到伪造的DNS 请求后，向外发送DNS请求。证明伪造成功。  
   

**实验三：Spoof DNS Replies**  
在这个任务中，我们需要在 Kaminsky 攻击中欺骗DNS回复。由于我们的目标是example.com，我们需要从这个域名服务器上伪造回复。

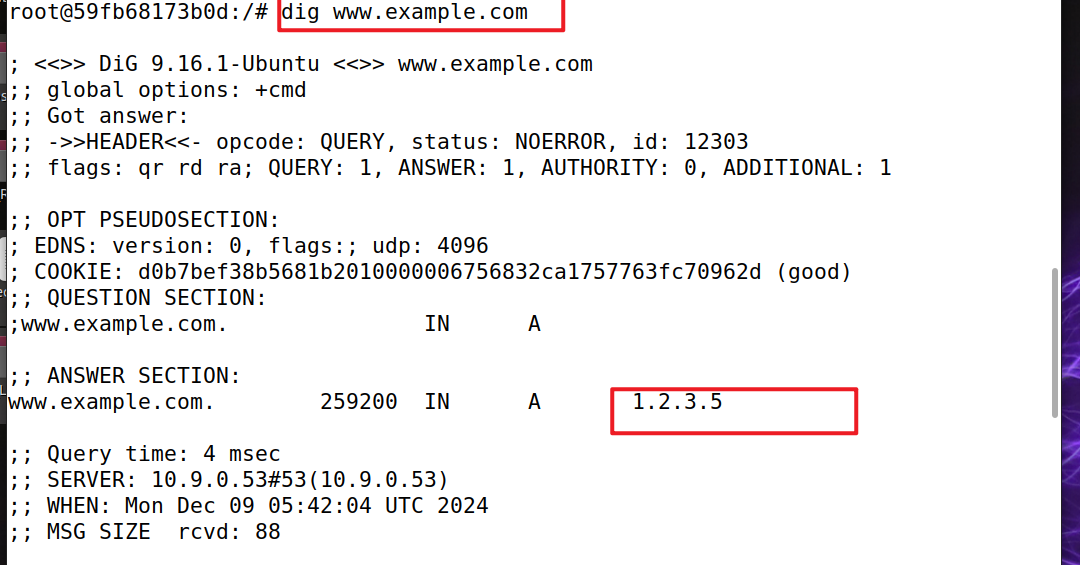
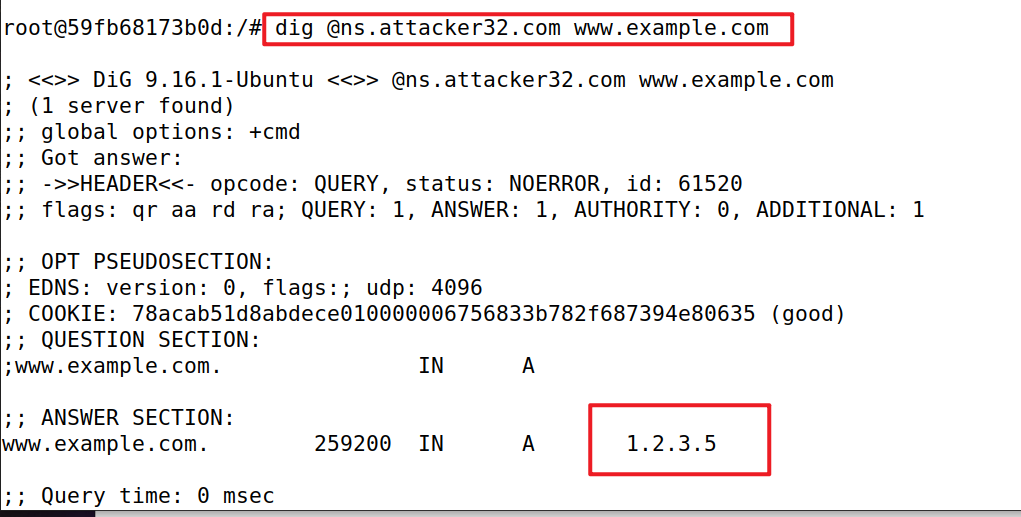
1. 从Task2 的 wireshark 抓包结果，我们可以得到 example.com 服务器的IP地址为 199.43.133.53 。dns服务器的地址是10.9.0.53  
   编写t3.py  
   
2. 使用wireshark监听local dns server，运行t3.py，发现local dns server接受到伪造的dns响应  
   

**实验 4: Launch the Kaminsky Attack.**

实现完整的 Kaminsky Attack。

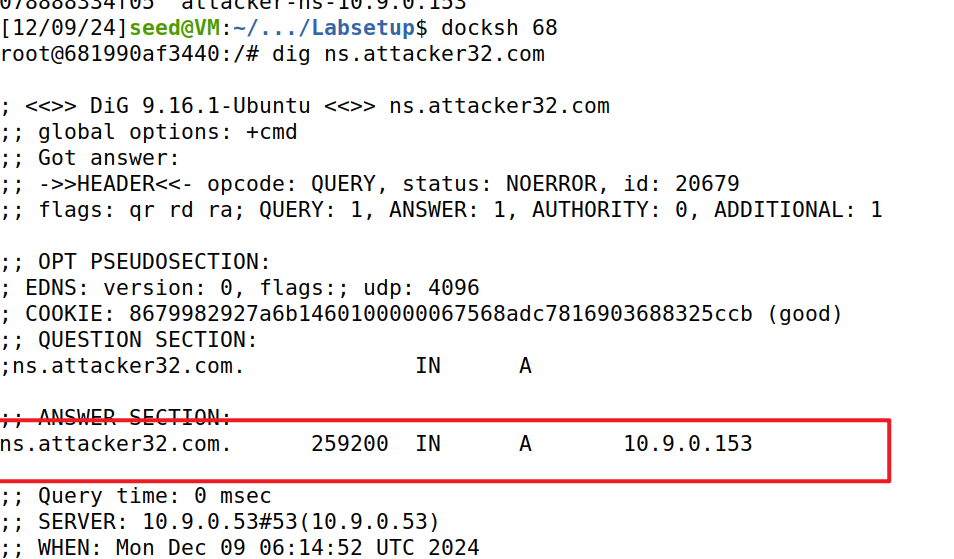
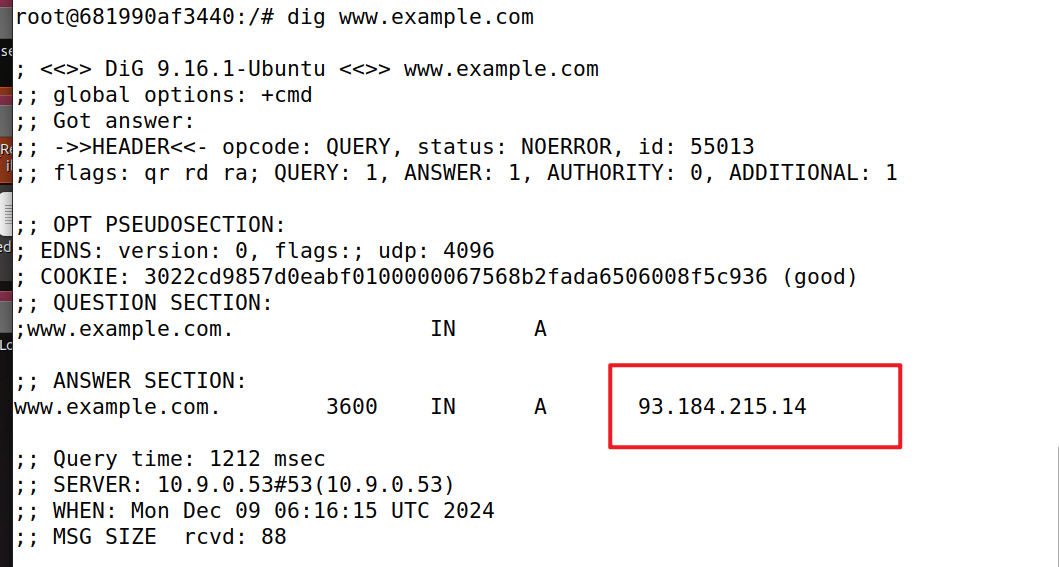
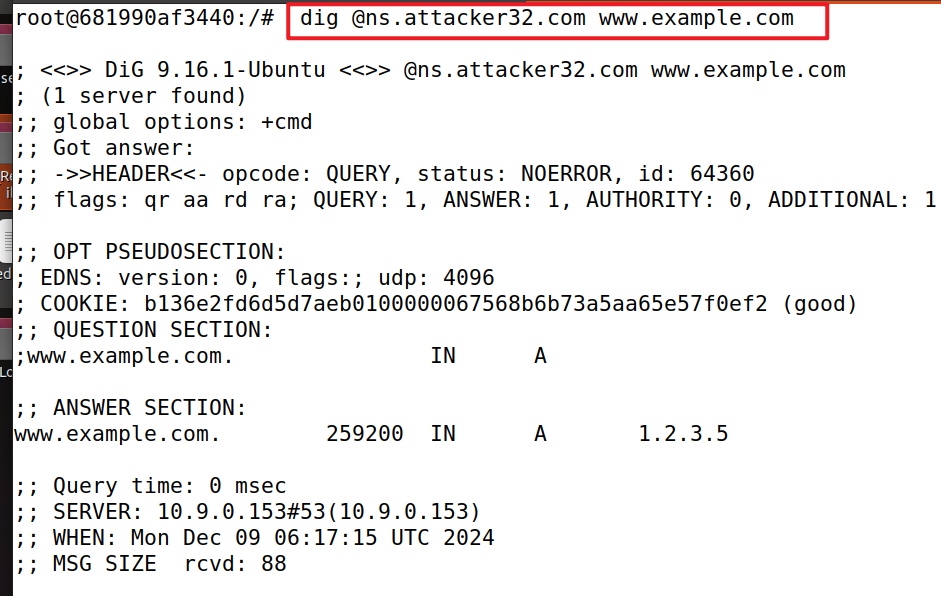
1. 将代码分为三个部分，第一个部分是dns请求，第二部分是伪造的响应，第三部分是攻击者的攻击程序  
   将第一部分和第二部分伪造的请求保存下来，第三部分调用请求，进行多次发送  
   第一部分：  
     
   第二部分：  
     
   第三部分：  
     
   （全部代码见附录）
2. 编译运行  
     
   查看缓存，攻击成功  
   

**实验5: Result Verification**

1. 在uesr容器中，运行dig [www.example.com](http://www.example.com)  
   
2. 运行dig @ns.attacker32.com www.example.com，结果相同，说明攻击成 功  
   

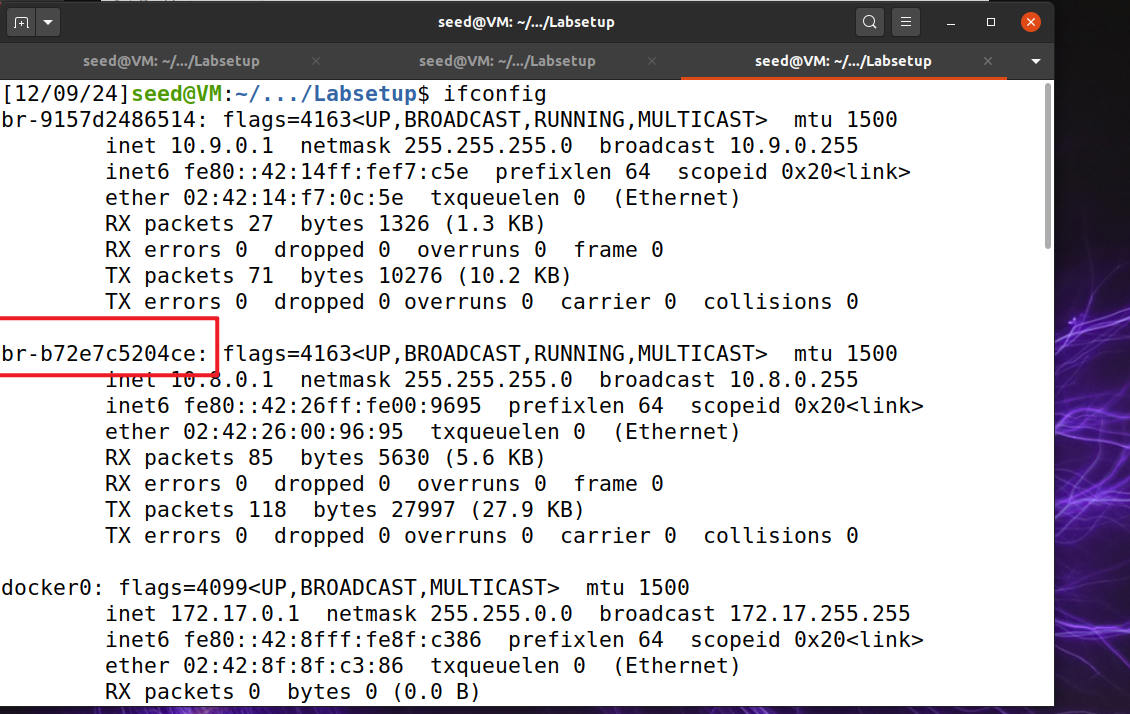
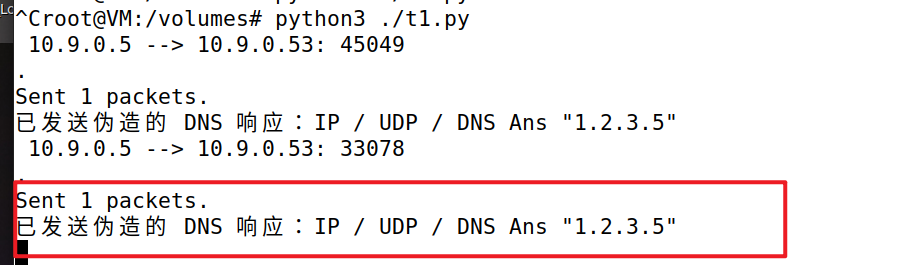
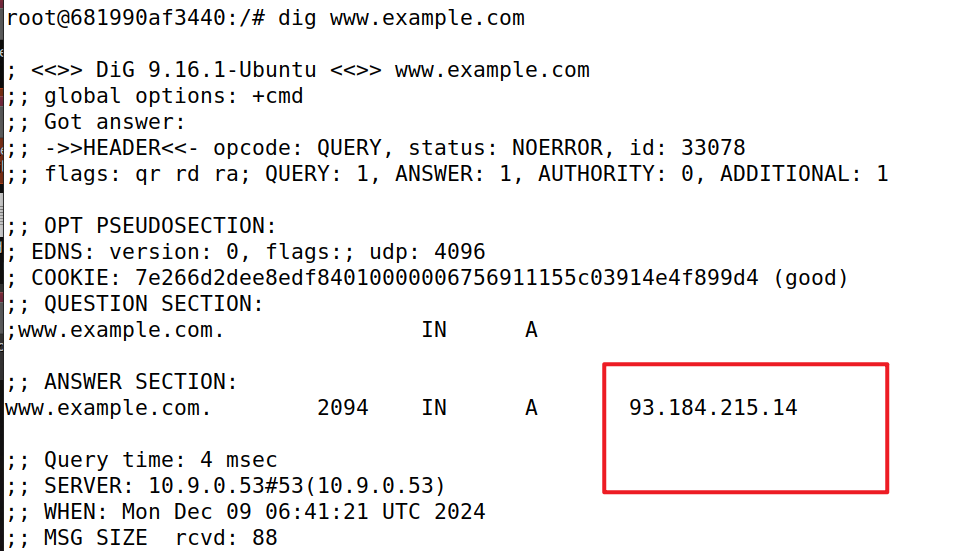
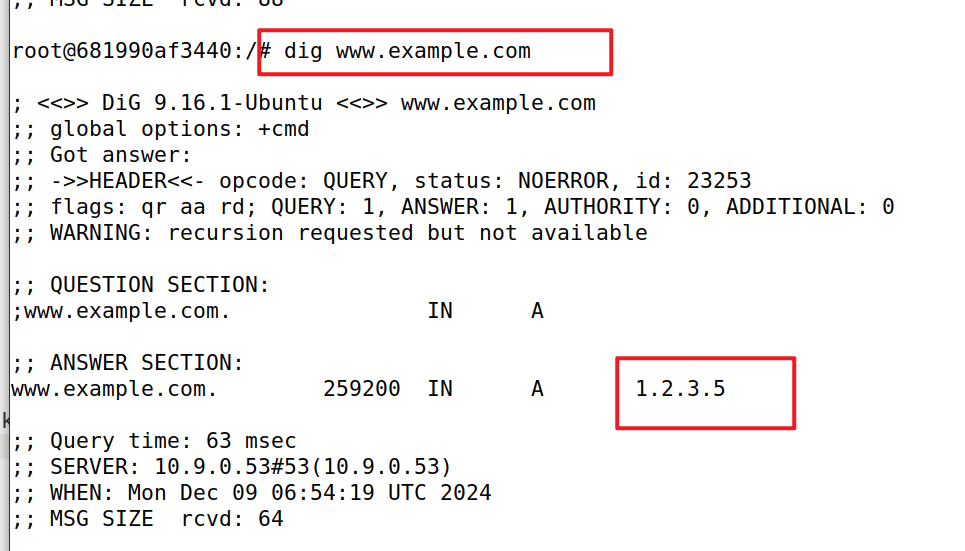
**本地DNS实验**

**Testing the DNS Setup**

1. 得到ns.attacker32.com的ip  
   在user容器中执行 dig ns.attacker32.com。由于本地DNS的配置文件中添加了转发区域条目，所以本地DNS服务器会将请求转发到攻击者域名服务器。  
   
2. 得到[www.example.com的ip](http://www.example.com的ip)  
   有两个域名服务器托管 example.com，一个是官方的域名服务器，另一个是攻击者容器。在user 容器中分别查询这两个域名服务器。  
   
3. 向ns.attacker32.com这个域名服务器查询 www.example.com 的IP地址。  
     
   得到1.2.3.5

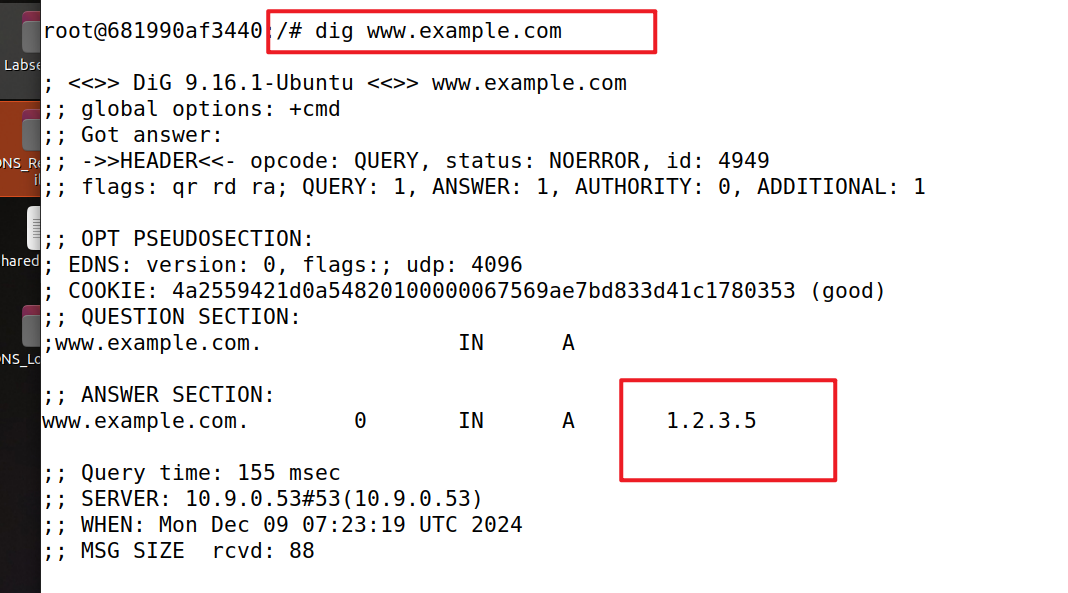
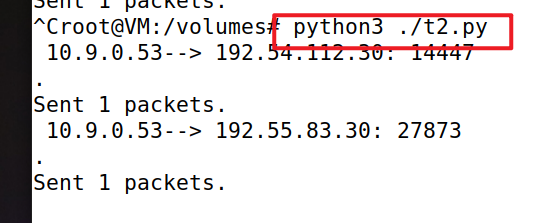
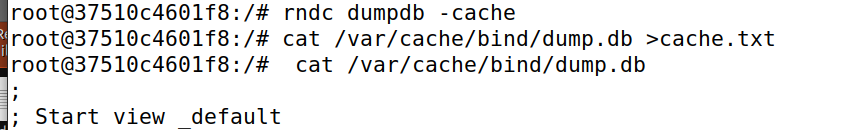
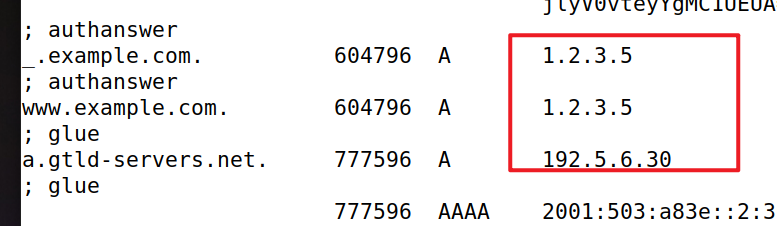
**Task 1: Directly Spoofing Response to User**

嗅探用户发送给本地DNS服务器的DNS请求，嗅探到后立即伪造DNS响应数据包，在真实的响应到达用户前，将伪造的虚假DNS响应包发送回用户，造成DNS攻击。

1. 查看本机ip和网卡  
     
   br-b72e7c5204ce
2. 根据题目的框架编写程序  
   当我们抓取到发送包时，根据发送包中的信息来 伪造我们的回应包，伪造的相应IP为1.2.3.5  
   同时设置过滤器，只捕获来自usr的包，丢弃其他包  
   
3. 尝试运行  
   攻击主机运行t1.py  
     
   用户机发送请求  
     
   发现运行失败,考虑未清除缓存
4. 解决问题  
   清除dns服务器中的缓冲  
   
5. 再次尝试  
   成功  
   

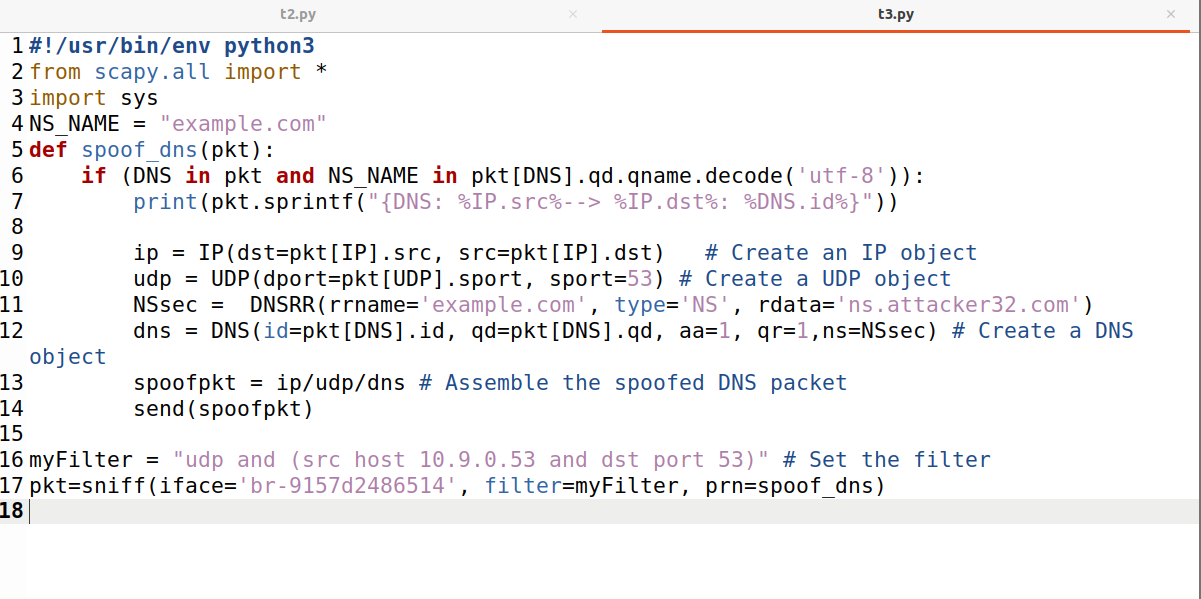
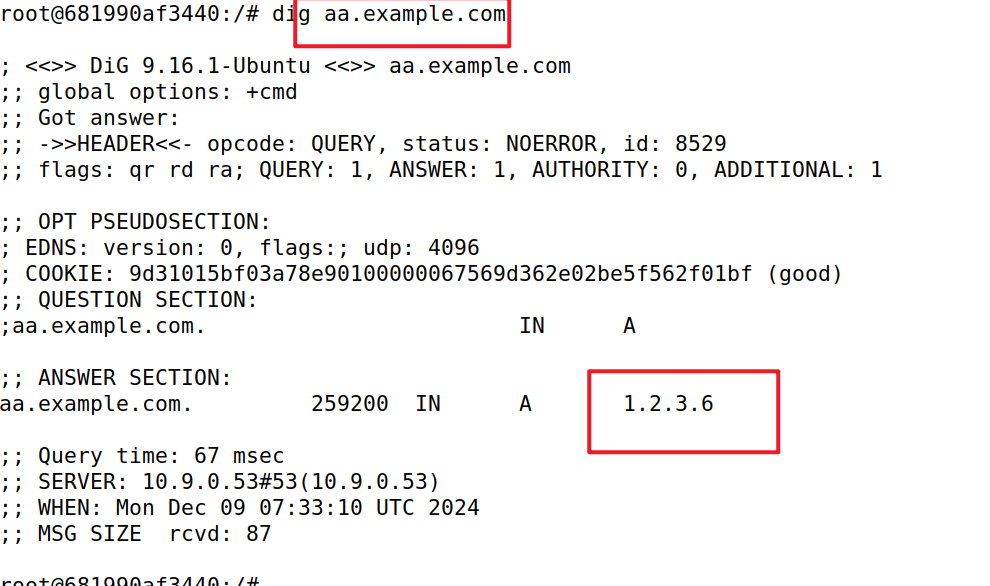
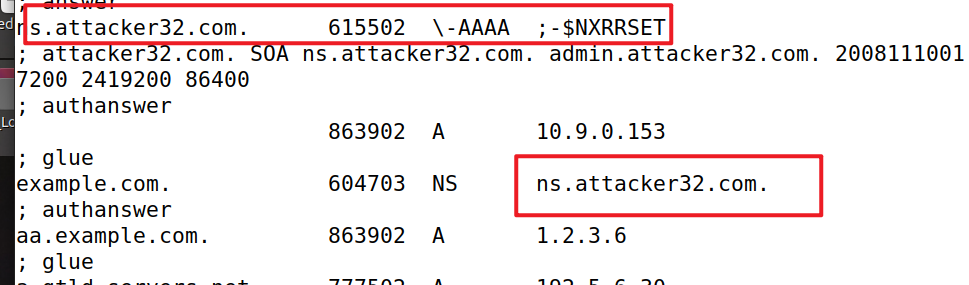
**Task 2: DNS Cache Poisoning Attack – Spoofing Answers**

task2将目标聚焦于DNS server，会是更高效的方式，即DNS缓存投毒：如果攻击者伪装从其他DNS server来的响应，该DNS Server就会把内容存储在缓存中，会保留相当长的一段时间。在该期间的请求都会直接返回缓存内的结果。我们只需spoof一次，影响就会持续直到下次缓存更新。

1. 目标为伪造local dns server发出的向其他dns server的查询的响应包  
   
2. 运行代码，user发起一次查询，成功实现欺骗（local dns server先flush缓存）。  
     
   
3. 查看缓存情况，确认投毒结果  
   local dns server中  
     
   

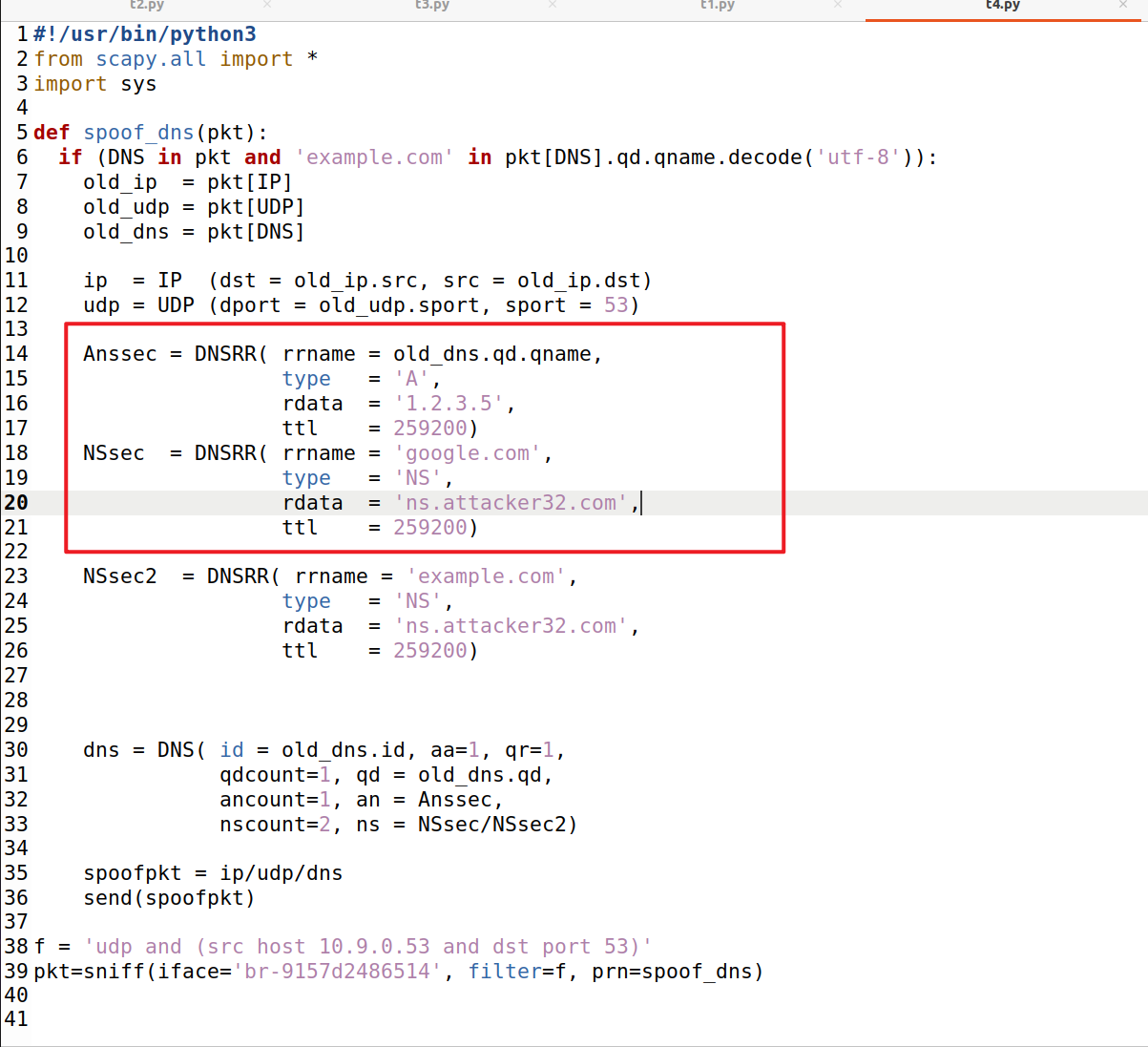
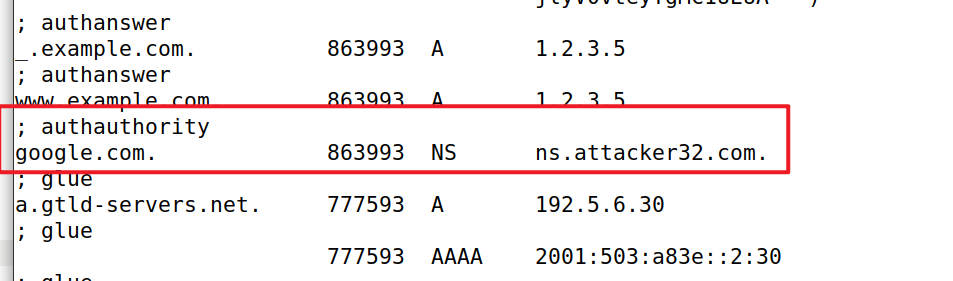
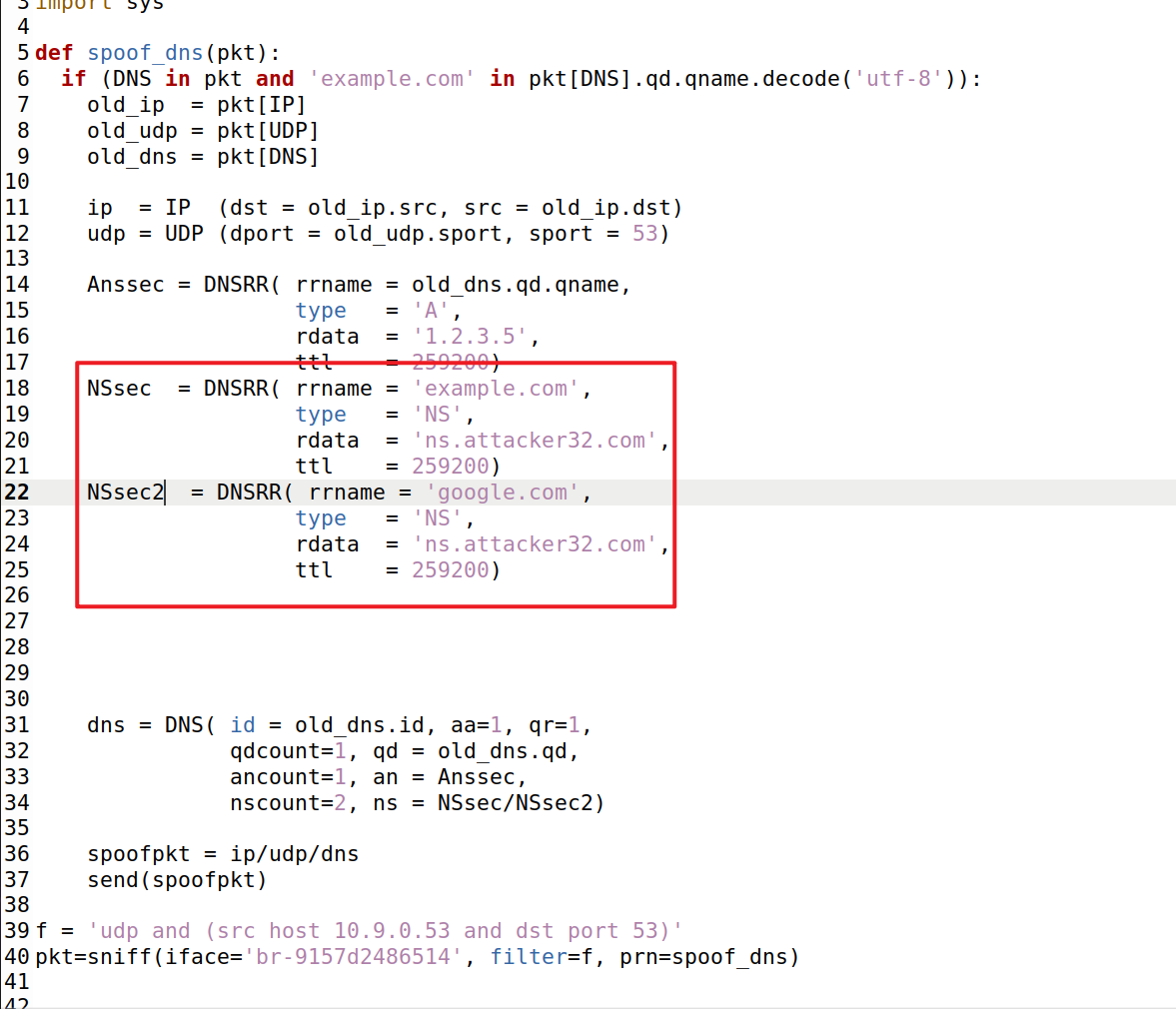
**Task 3: Spoofing NS Records**

进一步修改task2代码，实现NS记录的欺骗，使本地DNS服务器将攻击者控制的域名服务器（例如ns.attacker32.com）作为example.com域的权威域名服务器进行缓存，以达成**任何example.com**的子域名dns请求都会返回被某恶意权威域名服务器的控制的结果。

1. 编写程序  
   
2. 在user上执行 dig example.com 命令。可以看到运行结果，我们的攻击初步成功。接下来执行，dig www.example.com 命令和dig 2022141530010.example.com. 发现均指向我们伪造的虚假IP，说明攻击成功。  
     
   首先测试www.example.com  
   再测试其他子域名  
     
   实现成功
3. 缓存情况确认  
   被修改的ns记录  
   

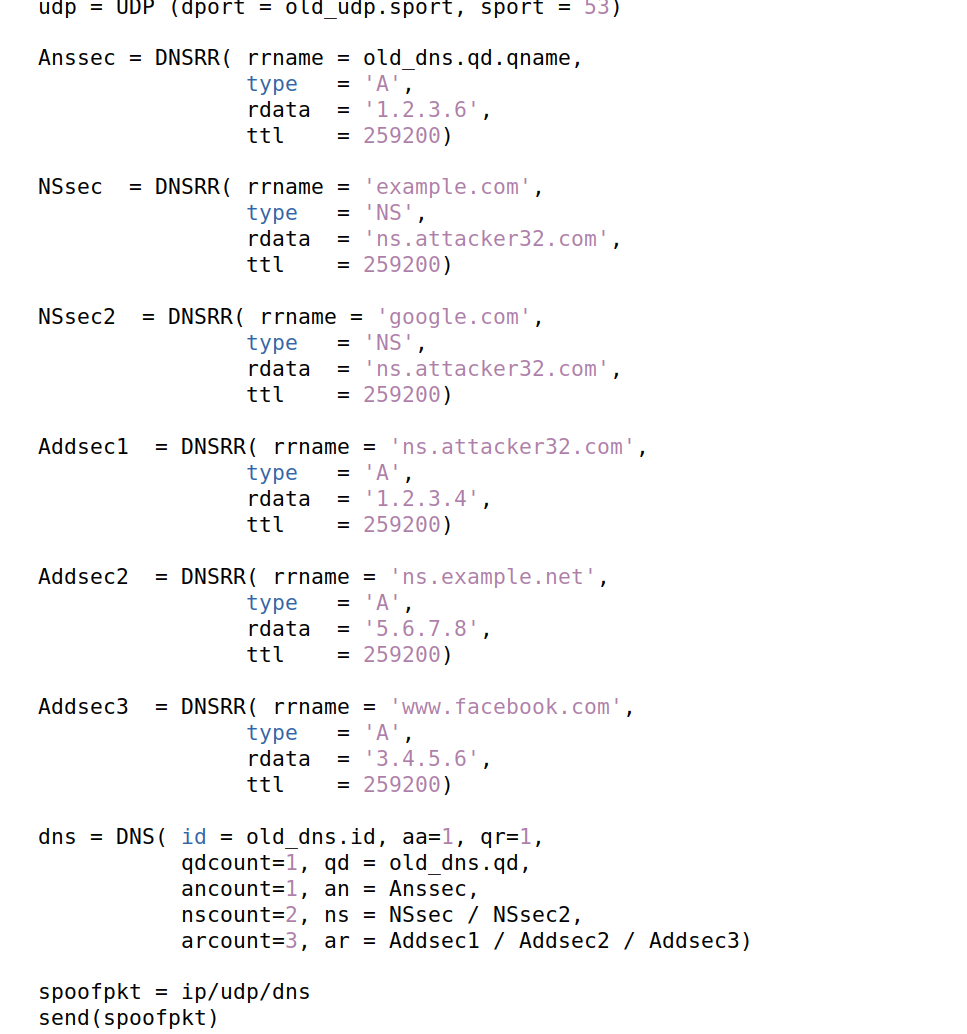
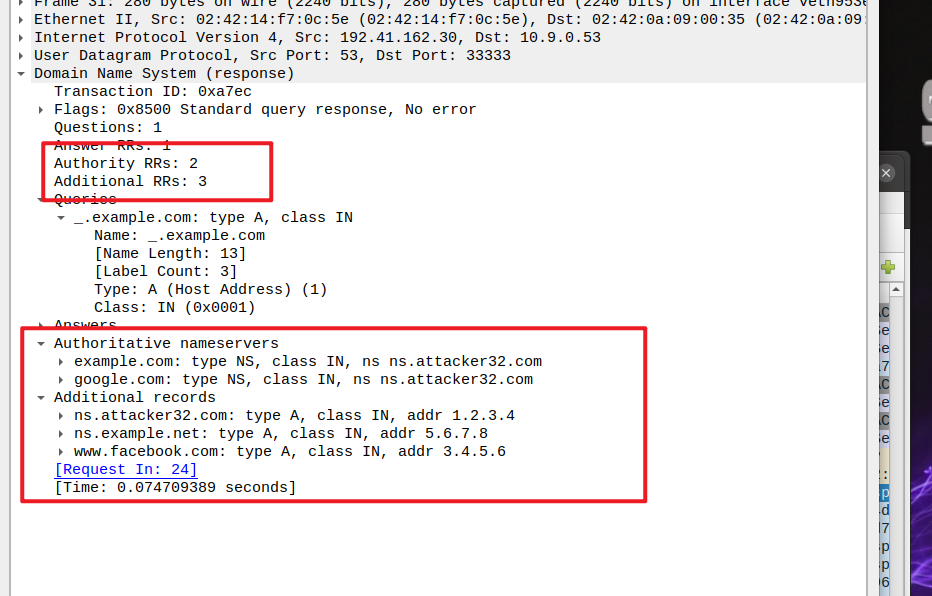
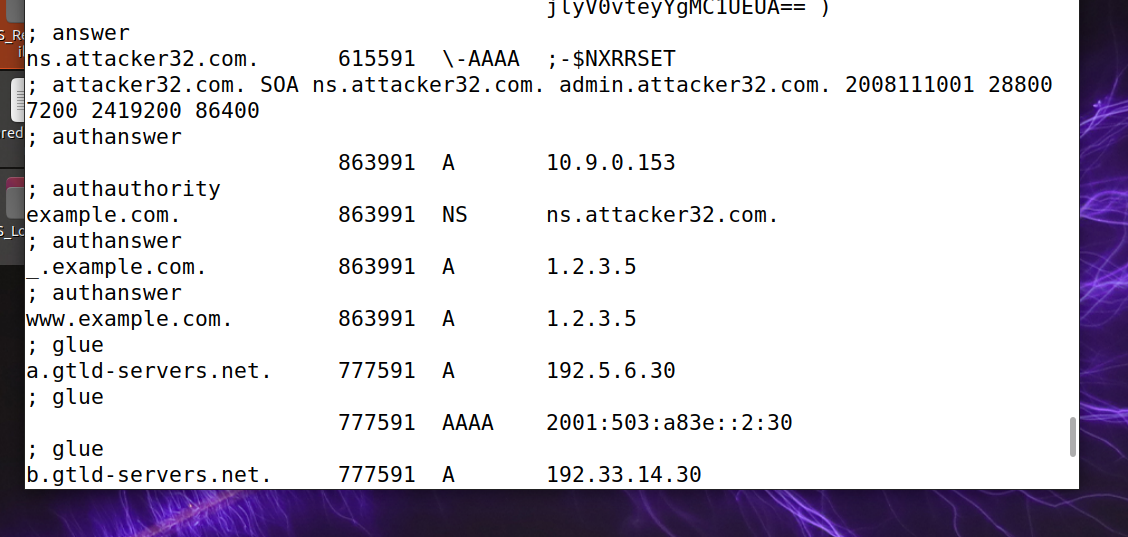
**Task 4: Spoofing NS Records for Another Domain**

通过Authority Section，将影响扩充到google.com。实验

1. 编写task4.py，，增加NS记录，其他保持一致。  
   
2. 清除缓存，attacker运行代码，user端dig www.example.com。查看缓存  
     
   但是，当我们在缓存中查看，却只有一条NS记录
3. 当我们修改程序，将第一条NS记录改为example.com时，我们再次运行程序，查看缓存  
     
   再次查看缓存  
     
   由此，可以判断DNS缓存中，只会保存第一条我们添加的NS记录

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

编写程序伪造一个 DNS 响应包，能够对Local DNS Server的本地缓存中的 AUTHORITY SECTION 和 ADDITIONAL SECTION进行修改：在 AUTHORITY SECTION 中，添加两个 NS 记录让example.com指向 ns.attacker32.com 和 ns.example.com；在 ADDITIONAL SECTION 中，提供相关的 A 记录，包括 ns.attacker32.com、ns.example.net 和 www.facebook.com 的 IP 地址。

1. 修改代码  
   
2. 使用wireshark抓包  
   
3. 但在缓存中并没有显示，猜测其不会显示与example.com无关域名  
     
   两条信息都被存下来了。这是因为它们两个都是域相关的。同时与 Task4 进行比较，我们调整facebook在拼接的ns中的位置，进行实验。发现出现了跟Task4一样的情况。

# 实验总结

**远程DNS攻击：**

成功实施了DNS缓存投毒和Kaminsky攻击，通过多次伪造响应绕过缓存机制，用户查询www.example.com时返回了攻击者指定的虚假IP地址。

**本地DNS欺骗：**

成功伪造了直接响应，影响了用户的DNS解析结果。

实现了DNS缓存投毒，修改了本地DNS服务器的缓存内容。

通过伪造NS记录，成功将子域名解析指向攻击者控制的服务器。

在扩展攻击至其他域名时，发现DNS缓存机制对NS记录的处理有一定限制，仅保存首条有效记录。