

Introduction:

During penetration testing and post-exploitation activities shell access is the backbone of the procedures. Bind shells listen on the target system for incoming connections, while reverse shells get the victim to call you back by connecting to the attacker's machine. My understanding of bind/reverse shells will be enforced by the activities I do in this lab. Specifically, I will create various types of shell connections, identify and exploit shell vulnerabilities, implement shell stabilization techniques, and apply these concepts to real-world penetration testing scenarios.

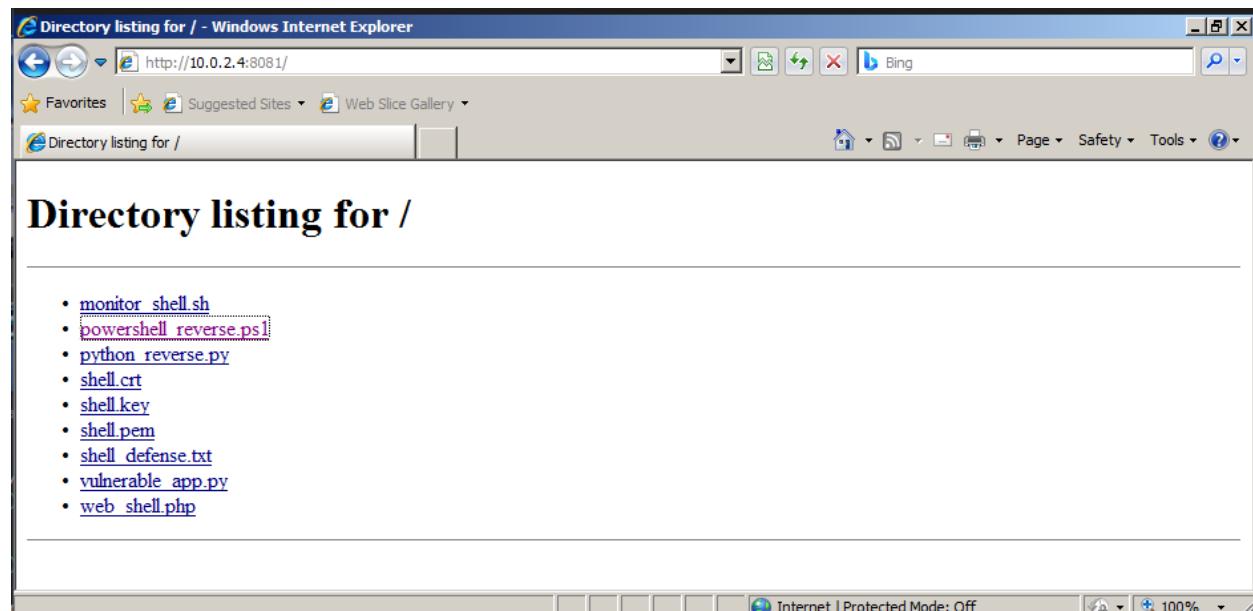
Exercise: PowerShell reverse shell

Setting up script and opening a web server

```
(cyberjackson㉿kali-attacker)-[~/cybersec-labs/shell-lab]
$ nano powershell_reverse.ps1

(cyberjackson㉿kali-attacker)-[~/cybersec-labs/shell-lab]
$ python3 -m http.server 8081
Serving HTTP on 0.0.0.0 port 8081 (http://0.0.0.0:8081/) ...
10.0.2.15 - - [29/Oct/2025 15:21:36] "GET / HTTP/1.1" 200 -
10.0.2.15 - - [29/Oct/2025 15:21:36] code 404, message File not found
10.0.2.15 - - [29/Oct/2025 15:21:36] "GET /favicon.ico HTTP/1.1" 404 -
```

Navigating to webserver on metasploitable3 (Windows 2k8 machine) to download powershell script



Begin listening and execute script to establish shell connection

The screenshot shows two windows. The top window is a terminal on a Kali Linux machine (cyberjackson@kali-attacker) with the command:

```
$ nc -nvlp 9001
listening on [any] 9001 ...
connect to [10.0.2.4] from (UNKNOWN) [10.0.2.15] 49366
whoami
PS C:\Users\vagrant\downloads> systeminfo
```

The bottom window is a Metasploitable3-win2k8 host (Administrator: Windows PowerShell) with the following session log:

```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

Loading personal and system profiles took 510ms.
PS C:\Users\vagrant> cd downloads
PS C:\Users\vagrant\downloads> ls

    Directory: C:\Users\vagrant\downloads

Mode                LastWriteTime         Length Name
----                <-----              ----- 
-a---       10/29/2025 12:23 PM            723 powershell_reverse.ps1

20 PS C:\Users\vagrant\downloads> ./powershell_reverse.ps1
.\powershell_reverse.ps1 : File C:\Users\vagrant\downloads\powershell_reverse.ps1 cannot be loaded. The file
C:\Users\vagrant\downloads\powershell_reverse.ps1 is not digitally signed. You cannot run this script on the curr
system. For more information about running scripts and setting execution policy, see about_Execution_Policies at
http://go.microsoft.com/fwlink/?LinkID=135170.
At line:1 char:1
+ ./powershell_reverse.ps1
+ CategoryInfo          : SecurityError: (<>[], PSSecurityException
+ FullyQualifiedErrorId : UnauthorizedAccess
PS C:\Users\vagrant\downloads> ./powershell_reverse.ps1
```

Exercise: Shell Detection and Prevention

Monitor network connections (Terminal 3 Monitor below, Attacker terminal above)

The screenshot shows a terminal window on a Kali Linux machine (cyberjackson@kali-attacker) with the command:

```
$ nc 10.0.2.4 4444
whoami
cyberjackson
```

Below this, the terminal continues with:

```
File Actions View Help
(cyberjackson@kali-attacker)-[~]
$ sudo tcpdump -i any -n 'port 4444 or port 5555 or port 6666'
[sudo] password for cyberjackson:
Sorry, try again.
[sudo] password for cyberjackson:
tcpdump: WARNING: any: That device doesn't support promiscuous mode
(Promiscuous mode not supported on the "any" device)
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
10:04:36.366271 lo  In IP 10.0.2.4.49980 > 10.0.2.4.4444: Flags [S], seq 751999881, win 65495, options [mss 6549
5,sackOK,TS val 1814021051 ecr 0,nop,wscale 7], length 0
10:04:36.366423 lo  In IP 10.0.2.4.49980 > 10.0.2.4.4444: Flags [S.], seq 2559928724, ack 751999882, win 65483,
options [mss 65495,sackOK,TS val 1814021051 ecr 1814021051,nop,wscale 7], length 0
10:04:36.366456 lo  In IP 10.0.2.4.49980 > 10.0.2.4.4444: Flags [.], ack 1, win 512, options [nop,nop,TS val 181
4021051 ecr 1814021051], length 0
10:04:43.606265 lo  In IP 10.0.2.4.49980 > 10.0.2.4.4444: Flags [P.], seq 1:8, ack 1, win 512, options [nop,nop,
TS val 1814028291 ecr 1814021051], length 7
10:04:43.606387 lo  In IP 10.0.2.4.4444 > 10.0.2.4.49980: Flags [.], ack 8, win 512, options [nop,nop,TS val 181
4028291 ecr 1814028291], length 0
10:04:43.613966 lo  In IP 10.0.2.4.4444 > 10.0.2.4.49980: Flags [P.], seq 1:14, ack 8, win 512, options [nop,nop
,TS val 1814028299 ecr 1814028291], length 13
10:04:43.614104 lo  In IP 10.0.2.4.49980 > 10.0.2.4.4444: Flags [.], ack 14, win 512, options [nop,nop,TS val 18
14028299 ecr 1814028299], length 0
```

Exercise: Firewall rules to block shells

```
(cyberjackson㉿kali-attacker)-[~/cybersec-labs/shell-lab]
└─$ sudo iptables -A INPUT -p tcp --dport 4444 -j DROP
└─$ sudo iptables -A INPUT -p tcp --dport 5555 -j DROP
└─$ sudo iptables -A OUTPUT -p tcp --dport 4444 -j DROP

(cyberjackson㉿kali-attacker)-[~/cybersec-labs/shell-lab]
└─$ sudo iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source          destination
DROP      tcp  --  0.0.0.0/0        0.0.0.0/0          tcp dpt:4444
DROP      tcp  --  0.0.0.0/0        0.0.0.0/0          tcp dpt:5555

Chain FORWARD (policy DROP)
target     prot opt source          destination
DOCKER-USER all  --  0.0.0.0/0        0.0.0.0/0
DOCKER-ISOLATION-STAGE-1 all  --  0.0.0.0/0        0.0.0.0/0
ACCEPT    all  --  0.0.0.0/0        0.0.0.0/0          ctstate RELATED,ESTABLISHED
DOCKER    all  --  0.0.0.0/0        0.0.0.0/0
ACCEPT    all  --  0.0.0.0/0        0.0.0.0/0
ACCEPT    all  --  0.0.0.0/0        0.0.0.0/0

Chain OUTPUT (policy ACCEPT)
target     prot opt source          destination
DROP      tcp  --  0.0.0.0/0        0.0.0.0/0          tcp dpt:4444

Chain DOCKER (1 references)
target     prot opt source          destination

Chain DOCKER-ISOLATION-STAGE-1 (1 references)
target     prot opt source          destination
DOCKER-ISOLATION-STAGE-2 all  --  0.0.0.0/0        0.0.0.0/0
RETURN    all  --  0.0.0.0/0        0.0.0.0/0

Chain DOCKER-ISOLATION-STAGE-2 (1 references)
target     prot opt source          destination
DROP      all  --  0.0.0.0/0        0.0.0.0/0
RETURN    all  --  0.0.0.0/0        0.0.0.0/0

Chain DOCKER-USER (1 references)
target     prot opt source          destination
RETURN    all  --  0.0.0.0/0        0.0.0.0/0

(cyberjackson㉿kali-attacker)-[~/cybersec-labs/shell-lab]
└─$ █
```

Exercise: Practical Shell Scenarios

Web application command injection to shell

The screenshot shows a terminal window with two tabs. The top tab is titled '(cyberjackson㉿kali-attacker)-[~]' and contains the following command and its output:

```
$ curl -X POST -d "host=127.0.0.1; whoami" http://localhost:8080/ping
<pre>PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.158 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.026 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.027 ms
— 127.0.0.1 ping statistics —
3 packets transmitted, 3 received, 0% packet loss, time 2032ms
rtt min/avg/max/mdev = 0.026/0.070/0.158/0.061 ms
cyberjackson
</pre><br><a href='/'>Back</a>
```

The bottom tab is titled 'cyberjackson@kali-attacker: ~/cybersec-labs/shell-lab' and contains the following session:

```
File Actions Edit View Help
(cyberjackson㉿kali-attacker)-[~]
$ nc -nlvp 8888
listening on [any] 8888 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 41704
(cyberjackson㉿kali-attacker)-[~/cybersec-labs/shell-lab]
$ python3 vulnerable_app.py &
python3 vulnerable_app.py &
[1] 142997

(cyberjackson㉿kali-attacker)-[~/cybersec-labs/shell-lab]
$ * Serving Flask app 'vulnerable_app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a producti
on WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:8080
* Running on http://10.0.2.4:8080
Press CTRL+C to quit
* Restarting with watchdog (inotify)
* Debugger is active!
* Debugger PIN: 134-409-077
127.0.0.1 - - [29/Oct/2025 12:52:11] "POST /ping HTTP/1.1" 200 -
kill $WEBAPP_PID
kill $WEBAPP_PID
kill: usage: kill [-s sigspec | -n signum | -sigsig] pid | jobspec ... or kill -l [sigspec]
```

Conclusion:

After completing the lab I feel much more comfortable and knowledgeable about terminal shells. It is obvious to me now how shell techniques are critical during penetration testing or red teaming to gain initial access and control or simulate APT tactics. For incident response, security assessment, malware analysis understanding attack vectors and investigating how the shells were used are important techniques to the job.

Through this lab I have learned the difference between bind and reverse shells, created various types of shell connections, implemented shell stabilization techniques, used encrypted shells, detected shell-based attacks using system monitoring tools, and applied

defensive measures to prevent shell attacks. All of this comes together to bring me a practical understanding of the shell in penetration testing scenarios.

The most challenging part of the lab was setting up the Powershell reverse shell and the web-based reverse shell. In both cases I had trouble running and compiling the scripts and was tripped up with how the shells should be set up attacker vs. Target.

The most interesting part of the lab for me was implementing shell stabilization techniques. At first it was tricky to me to understand whether the connection was maintained after stabilizing the shell. But, after getting my head around what each of the stabilizing methods were doing and seeing the terminals connection maintained via running test commands. I learned that the stabilization techniques are important for getting more reliability and use out of your shell connections.