**EXERCISES OSCP**

2.4.3.4 Exercises 1. Use man to look at the man page for one of your preferred commands. 2. Use man to look for a keyword related to file compression. 3. Use which to locate the pwd command on your Kali virtual machine. 4. Use locate to locate wce32.exe on your Kali virtual machine. 5. Use find to identify any file (not directory) modified in the last day, NOT owned by the root user and execute ls -l on them. Chaining/piping commands is NOT allowed!

3.1.3.2 Exercises

1. Inspect your bash history and use history expansion to re-run a command from it.

2. Execute different commands of your choice and experiment browsing the history through the shortcuts as well as the reverse-i-search facility.

3.2.5.1 Exercises

1. Use the cat command in conjunction with sort to reorder the content of the /etc/passwd file on your Kali Linux system.

2. Redirect the output of the previous exercise to a file of your choice in your home directory.

3.3.5.1 Exercises

1. Using /etc/passwd, extract the user and home directory fields for all users on your Kali machine for which the shell is set to /bin/false. Make sure you use a Bash one-liner to print the output to the screen.

The output should look similar to Listing 53 below:

kali@kali:~$ YOUR COMMAND HERE...

The user mysql home directory is /nonexistent The user Debian-snmp home directory is /var/lib/snmp The user speech-dispatcher home directory is /var/run/speech-dispatcher The user Debian-gdm home directory is /var/lib/gdm3 Listing 53 - Home directories for users with /bin/false shells

2. Copy the /etc/passwd file to your home directory (/home/kali).

3. Use cat in a one-liner to print the output of the /kali/passwd and replace all instances of the “Gnome Display Manager” string with “GDM”.

3.5.3.1 Exercises

1. Download the archive from the following URL https://offensive-security.com/pwkfiles/scans.tar.gz

2. This archive contains the results of scanning the same target machine at different times. Extract the archive and see if you can spot the differences by diffing the scans.

3.6.3.1 Exercises 1. Find files that have changed on your Kali virtual machine within the past 7 days by running a specific command in the background. 2. Re-run the previous command and suspend it; once suspended, background it. 3. Bring the previous background job into the foreground. 4. Start the Firefox browser on your Kali system. Use ps and grep to identify Firefox’s PID. 5. Terminate Firefox from the command line using its PID.

3.7.2.1 Exercises

1. Start your apache2 web service and access it locally while monitoring its access.log file in real-time.

2. Use a combination of watch and ps to monitor the most CPU-intensive processes on your Kali machine in a terminal window; launch different applications to see how the list changes in real time.

3.8.3.1 Exercise

1. Download the PoC code for an exploit from https://www.exploit-db.com using curl, wget, and axel, saving each download with a different name.

3.9.3.1 Exercises

1. Create an alias named “..” to change to the parent directory and make it persistent across terminal sessions.

2. Permanently configure the history command to store 10000 entries and include the full date in its output.

4.2.4.1 Exercises

1. Use socat to transfer powercat.ps1 from your Kali machine to your Windows system. Keep the file on your system for use in the next section.

2. Use socat to create an encrypted reverse shell from your Windows system to your Kali machine.

3. Create an encrypted bind shell on your Windows system. Try to connect to it from Kali without encryption. Does it still work?

4. Make an unencrypted socat bind shell on your Windows system. Connect to the shell using Netcat. Does it work?

Note: If cmd.exe is not executing, research what other parameters you may need to pass to the EXEC option based on the error you receive.

4.3.8.1 Exercises

1. Use PowerShell and powercat to create a reverse shell from your Windows system to your Kali machine.

2. Use PowerShell and powercat to create a bind shell on your Windows system and connect to it from your Kali machine. Can you also use powercat to connect to it locally?

3. Use powercat to generate an encoded payload and then have it executed through powershell. Have a reverse shell sent to your Kali machine, also create an encoded bind shell on your Windows system and use your Kali machine to connect to it.

4.4.5.1 Exercises

1. Use Wireshark to capture network activity while attempting to connect to 10.11.1.217 on port 110 using Netcat, and then attempt to log into it.

2. Read and understand the output. Where is the three-way handshake happening? Where is the connection closed?

3. Follow the TCP stream to read the login attempt.

4. Use the display filter to only monitor traffic on port 110. 5. Run a new session, this time using the capture filter to only collect traffic on port 110.

4.5.3.1 Exercises

1. Use tcpdump to recreate the Wireshark exercise of capturing traffic on port 110.

2. Use the -X flag to view the content of the packet. If data is truncated, investigate how the -s flag might help.

3. Find all ‘SYN’, ‘ACK’, and ‘RST’ packets in the password\_cracking\_filtered.pcap file.

4. An alternative syntax is available in tcpdump where you can use a more user-friendly filter to display only ACK and PSH packets. Explore this syntax in the tcpdump manual by searching for “tcpflags”. Come up with an equivalent display filter using this syntax to filter ACK and PSH packets.

5.7.3.1 Exercises

1. Research Bash loops and write a short script to perform a ping sweep of your target IP range of 10.11.1.0/24.

2. Try to do the above exercise with a higher-level scripting language such as Python, Perl, or Ruby.

3. Use the practical examples in this module to help you create a Bash script that extracts JavaScript files from the access\_log.txt file (http://www.offensive-security.com/pwkfiles/access\_log.txt.gz). Make sure the file names DO NOT include the path, are unique, and are sorted. 4. Re-write the previous exercise in another language such as Python, Perl, or Ruby

6.3.1.1 Exercise

1. Use the whois tool in Kali to identify the name servers of MegaCorp One.

6.4.1.1 Exercises 1. Who is the VP of Legal for MegaCorp One and what is their email address?

2. Use Google dorks (either your own or any from the GHDB) to search www.megacorpone.com for interesting documents.

3. What other MegaCorp One employees can you identify that are not listed on [www.megacorpone.com](http://www.megacorpone.com)?

6.5.1.1 Exercise

1. Use Netcraft to determine what application server is running on [www.megacorpone.com](http://www.megacorpone.com).

6.7.1.1 Exercise

1. Search Megacorpone’s GitHub repos for interesting or sensitive information.

6.12.1.1 Exercises

1. Use theHarvester to enumerate emails addresses for megacorpone.com.

2. Experiment with different data sources (-b). Which ones work best for you?

6.13.2.1 Exercise

1. Use any of the social media tools previously discussed to identify additional MegaCorp One employees.

7.1.6.3 Exercises

1. Find the DNS servers for the megacorpone.com domain.

2. Write a small script to attempt a zone transfer from megacorpone.com using a higher-level scripting language such as Python, Perl, or Ruby.

3. Recreate the example above and use dnsrecon to attempt a zone transfer from megacorpone.com.

7.2.2.9 Exercises

1. Use Nmap to conduct a ping sweep of your target IP range and save the output to a file. Use grep to show machines that are online.

2. Scan the IP addresses you found in exercise 1 for open webserver ports. Use Nmap to find the webserver and operating system versions.

3. Use NSE scripts to scan the machines in the labs that are running the SMB service.

4. Use Wireshark to capture a Nmap connect and UDP scan and compare it against the Netcat port scans. Are they the same or different?

5. Use Wireshark to capture a Nmap SYN scan and compare it to a connect scan and identify the difference between them.

7.3.2.1 Exercises

1. Use Nmap to make a list of the SMB servers in the lab that are running Windows.

2. Use NSE scripts to scan these systems for SMB vulnerabilities.

3. Use nbtscan and enum4linux against these systems to identify the types of data you can obtain from different versions of Windows.

7.4.2.1 Exercises

1. Use Nmap to make a list of machines running NFS in the labs.

2. Use NSE scripts to scan these systems and collect additional information about accessible shares.

7.5.1.1 Exercises

1. Search your target network range to see if you can identify any systems that respond to the SMTP VRFY command.

2. Try using this Python code to automate the process of username discovery using a text file with usernames as input.

7.6.3.6 Exercises

1. Scan your target network with onesixtyone to identify any SNMP servers.

2. Use snmpwalk and snmp-check to gather information about the discovered targets.

8.2.4.2 Exercises

1. Follow the steps above to create your own unauthenticated scan of Gamma.

2. Run the scan with Wireshark open and identify the steps the scanner performed to completed the scan.

3. Review the results of the scan.

8.2.5.2 Exercises

1. Follow the steps above to create your own authenticated scan of your Debian client.

2. Review the results of the scan.

8.2.6.1 Exercises

1. Follow the steps above to create your own individual scan of Beta.

2. Run Wireshark or tcpdump during the individual scan. What other ports does Nessus scan? Why do you think Nessus scans other ports?

3. Review the results of the scan.

8.3.1.1 Exercise

1. Find an NSE script similar to the NFS Exported Share Information Disclosure that was executed in the “Scanning with Individual Nessus Plugins” section. Once found, run the script against Beta in the PWK labs.

9.3.4.1 Exercise

1. Spend some time reviewing the applications available under the Web Application Analysis menu in Kali Linux

9.4.1.3 Exercises

1. Use Burp Intruder to gain access to the phpMyAdmin site running on your Windows 10 lab machine.

2. Insert a new user into the “users” table.

9.4.2.5 Exercises

1. Exploit the XSS vulnerability in the sample application to get the admin cookie and hijack the session. Remember to use the PowerShell script on your Windows 10 lab machine to simulate the admin login.

2. Consider what other ways an XSS vulnerability in this application might be used for attacks.

3. Does this exploit attack the server or clients of the site?

9.4.3.2 Exercise

1. Exploit the directory traversal vulnerability to read arbitrary files on your Windows 10 lab machine.

9.4.4.5 Exercises

1. Obtain code execution through the use of the LFI attack.

2. Use the code execution to obtain a full Shell

9.4.4.7 Exercises

1. Exploit the RFI vulnerability in the web application and get a shell.

2. Using /menu2.php?file=current\_menu as a starting point, use RFI to get a shell.

3. Use one of the webshells included with Kali to get a shell on the Windows 10 target.

9.4.4.10 Exercises

1. Exploit the LFI vulnerability using a PHP wrapper.

2. Use a PHP wrapper to get a shell on your Windows 10 lab machine.

9.4.5.4 Exercises

1. Interact with the MariaDB database and manually execute the commands required to authenticate to the application. Understand the vulnerability.

2. SQL inject the username field to bypass the login process.

3. Why is the username displayed like it is in the web application once the authentication process is bypassed?

4. Execute the SQL injection in the password field. Is the “LIMIT 1” necessary in the payload? Why or why not?

9.4.5.9 Exercises

1. Enumerate the structure of the database using SQL injection.

2. Understand how and why you can pull data from your injected commands and have it displayed on the screen.

3. Extract all users and associated passwords from the database.

9.4.5.11 Exercises

1. Exploit the SQL injection along with the MariaDB INTO OUTFILE function to obtain code execution.

2. Turn the simple code execution into a full shell.

9.4.5.13 Exercises

1. Use sqlmap to obtain a full dump of the database.

2. Use sqlmap to obtain an interactive shell.

10.2.5 Exercises

1. Repeat the steps shown in this section to see the 12 A’s copied onto the stack.

2. Supply at least 80 A’s and verify that EIP after the strcpy will contain the value 41414141.

11.1.1.2 Exercises

1. Build the fuzzer and replicate the SyncBreeze crash.

2. Inspect the content of other registers and stack memory. Does anything seem to be directly influenced by the fuzzing input?

11.2.3.1 Exercises

1. Write a standalone script to replicate the crash.

2. Determine the offset within the input buffer to successfully control EIP.

3. Update your standalone script to place a unique value into EIP to ensure your offset is correct.

11.2.5.1 Exercises

1. Repeat the required steps in order to identify the bad characters that cannot be included in the payload.

2. Why are these characters not allowed? How do these bad hex characters translate to ASCII?

11.2.7.1 Exercises

1. Locate the JMP ESP that is usable in the exploit.

2. Update your PoC to include the discovered JMP ESP, set a breakpoint on it, and follow the execution to the placeholder shellcode.

11.2.9.1 Exercises

1. Update your PoC to include a working payload.

2. Attempt to execute your exploit without using a NOP sled and observe the decoder corrupting the stack.

3. Add a NOP sled to your PoC and obtain a shell from SyncBreeze.

11.2.10.1 Exercise

1. Update the exploit so that SyncBreeze still runs after exploitation.

11.2.10.2 Extra Mile Exercises In the Tools folder of your Windows VM, there are three applications called VulnApp1.exe, VulnApp2.exe, and VulnApp3.exe, each containing a vulnerability. Associated Python proof of concept scripts are also present in the folder. Using the PoCs, write exploits for each of the vulnerable applications.

12.2.1.2 Exercises

1. Log in to your dedicated Linux client using the credentials you received.

2. On your Kali machine, recreate the proof-of-concept code that crashes the Crossfire server.

3. Attach the debugger to the Crossfire server, run the exploit against your Linux client, and confirm that the EIP register is overwritten by the malicious buffer

12.3.1.1 Exercises

1. Determine the correct buffer offset required to overwrite the return address on the stack.

2. Update your stand-alone script to ensure your offset is correct.

12.5.1.1 Exercises

1. Determine the opcodes required to generate a first stage shellcode using msf-nasm\_shell.

2. Identify the bad characters that cannot be included in the payload and return address.

12.6.1.1 Exercises

1. Find a suitable assembly instruction address for the exploit using EDB.

2. Include the first stage shellcode and return address instruction in your proof-of-concept and ensure that the first stage shellcode is working as expected by single stepping through it in the debugger.

12.7.1.1 Exercises

1. Update your proof-of-concept to include a working payload.

2. Obtain a shell from the Crossfire application with and without a debugger.

13.2.2.1 Exercises

1. Use msfvenom to generate a HTML Application and use it to compromise your Windows client. 2. Is it possible to use the HTML Application attack against Microsoft Edge users, and if so, how?

13.3.2.1 Exercise

1. Use the PowerShell payload from the HTA attack to create a Word macro that sends a reverse shell to your Kali system.

13.3.3.1 Exercise

1. Use the PowerShell payload to create a batch file and embed it in a Microsoft Word document to send a reverse shell to your Kali system.

13.3.4.1 Exercises

1. Trigger the protection by Protected View by simulating a download of the Microsoft Word document from the Internet.

2. Reuse the batch file and embed it in a Microsoft Publisher document to receive a reverse shell to your Kali system.

3. Move the file to the Apache web server to simulate the download of the Publisher document from the Internet and confirm the missing Protected View.

14.3.1.1 Exercises

1. Connect to your dedicated Linux client and start the vulnerable Apache James service using the /usr/local/james/bin/run.sh script.

2. Enumerate the target using port scanning utilities and use information from the banners and Internet searches to determine the software running on the machine.

3. Use the searchsploit tool to find exploits for this version on the online resources mentioned in this module.

4. Launch the exploit and verify that the payload is executed upon logging in to the machine.

5. Attempt to modify the payload variable in order to get a reverse shell on the target machine.

15.1.3.1 Exercises

1. Locate the exploit discussed in this section using the searchsploit tool in Kali Linux.

2. Install the mingw-w64 suite in Kali Linux and compile the exploit code.

15.1.4.1 Exercises

1. Modify the connection information in the exploit in order to target the SyncBreeze installation on your Windows client.

2. Recompile the exploit and use Wireshark to confirm that the code successfully initiates a socket connection to your dedicated Windows client.

15.1.5.1 Exercise

1. Find any valid return address instruction and alter the one present in the original exploit.

15.1.6.1 Exercises

1. Generate a reverse shell payload using msfvenom while taking into account the bad characters of our exploit.

2. Replace the original payload with the newly generated one.

3. Attach the debugger to the target process and set a breakpoint at the return address instruction.

4. Compile the exploit and run it. Did you hit the breakpoint?

15.1.7.1 Exercises

1. Fix the overflow buffer such that the EIP register will be overwritten by your chosen return address instruction.

2. Install the ASX to MP3 Converter application located under the C:\Tools\fixing\_exploits directory; download the exploit for ASX to MP3 Converter from EDB391 and edit it in order to get a shell on your dedicated Windows machine.

15.2.3.1 Exercises

1. Connect to your dedicated Linux lab client and start the apache2 service; the target web application is located under /var/www/https/.

2. Modify the original exploit and set the base\_url variable to the correct IP address of your dedicated Linux lab client as well as the protocol to HTTPS.

3. Get familiar with the requests Python library and adjust your exploit accordingly to avoid SSL verification.

4. Edit the username and password variables to match the ones from our test case (username “admin”, password “HUYfaw763”).

5. Try to run the exploit against the Linux lab client, does it work? If not, try to explain why.

15.2.4.1 Exercises

1. Observe the error that is generated when running the exploit.

2. Attempt to troubleshoot the code and determine why the error occurs.

3. Modify the exploit in order to avoid the error and run it against your dedicated Linux client.

4. Verify that your exploit worked by attempting to execute the whoami command using the remote php shell.

5. Attempt to obtain a fully interactive shell with this exploit.

17.3.3.2 Exercises

1. Review the code from the PowerShell script and ensure that you have a basic understanding of how it works.

2. Get a meterpreter shell back to your Kali Linux machine using PowerShell.

3. Attempt to get a reverse shell using a PowerShell one-liner rather than a script.432

17.3.3.4 Exercises

1. Inject a meterpreter reverse shell payload in the WinRAR executable.

2. Transfer the binary to your Windows client and ensure that it is not being detected by the antivirus.

3. Run the WinRAR installer and migrate your meterpreter shell to prevent a disconnect.

4. Attempt to find different executables and inject malicious code into them using Shellter.

18.1.1.13 Exercise

1. Perform various manual enumeration methods covered in this section on both your dedicated Windows and Linux clients. Try experimenting with various options for the tools and commands used in this section.

18.1.2.1 Exercises

1. Inspect your Windows and Linux clients by using the tools and commands presented in this section in order to get comfortable with manual local enumeration techniques.

2. Experiment with different windows-privesc-check and unix\_privesc\_check options.

18.2.3.2 Exercise

1. Log in to your Windows client as the admin user and attempt to bypass UAC using the application and technique covered above.

18.2.4.1 Exercises

1. Log in to your Windows client as an unprivileged user and attempt to elevate your privileges to SYSTEM using the above vulnerability and technique.

2. Attempt to get a remote system shell rather than adding a malicious user.

18.3.2.1 Exercise

1. Log in to your Debian client as an unprivileged user and attempt to elevate your privileges to root using the above technique.

18.3.3.1 Exercise

1. Log in to your Debian client with your student credentials and attempt to elevate your privileges by adding a superuser account to the /etc/passwd file.

19.4.2.1 Exercises

1. Use Mimikatz to extract the password hash of an administrative user from the Windows client.

2. Reuse the password hash to perform a pass-the-hash attack from your Kali system and obtain code execution on your Windows client.

20.1.1.1 Exercises

1. Connect to your dedicated Linux lab client and run the clear\_rules.sh script from /root/port\_forwarding\_and\_tunneling/ as root.

2. Attempt to replicate the port-forwarding technique covered in the above scenario.

20.2.1.1 Exercises

1. Connect to your dedicated Linux lab client and run the clear\_rules.sh script from /root/port\_forwarding\_and\_tunneling/ as root.

2. Run the ssh\_local\_port\_forwarding.sh script from /root/port\_forwarding\_and\_tunneling/ as root.

3. Take note of the Linux client and Windows Server 2016 IP addresses shown in the Student Control Panel.

4. Attempt to replicate the smbclient enumeration covered in the above scenario.

20.2.2.2 Exercises

1. Connect to your dedicated Linux lab client via SSH and run the clear\_rules.sh script from /root/port\_forwarding\_and\_tunneling/ as root.

2. Close any SSH connections to your dedicated Linux lab client and then connect as the student account using rdesktop and run the ssh\_remote\_port\_forward.sh script from /root/port\_forwarding\_and\_tunneling/ as root.

3. Attempt to replicate the SSH remote port forwarding covered in the above scenario and ensure that you can scan and interact with the MySQL service.

20.2.3.1 Exercises

1. Connect to your dedicated Linux lab client and run the clear\_rules.sh script from /root/port\_forwarding\_and\_tunneling/ as root.

2. Take note of the Linux client and Windows Server 2016 IP addresses.

3. Create a SOCKS4 proxy on your Kali machine, tunneling through the Linux target.

4. Perform a successful nmap scan against the Windows Server 2016 machine through the proxy.

5. Perform an nmap SYN scan through the tunnel. Does it work? Are the results accurate?

20.3.1.1 Exercises

1. Obtain a reverse shell on your Windows lab client through the Sync Breeze vulnerability.

2. Use plink.exe to establish a remote port forward to the MySQL service on your Windows 10 client.

3. Scan the MySQL port via the remote port forward.

20.4.1.1 Exercise

1. Obtain a reverse shell on your Windows lab client through the Sync Breeze vulnerability.

2. Using the SYSTEM shell, attempt to replicate the port forwarding example using netsh

20.5.1.1 Exercises

1. Connect to your dedicated Linux lab client as the student account using rdesktop and run the http\_tunneling.sh script from /root/port\_forwarding\_and\_tunneling/ as root.

2. Start the apache2 service and exploit the vulnerable web application hosted on port 443 (covered in a previous module) in order to get a reverse HTTP shell.599 3. Replicate the scenario demonstrated above using your dedicated clients.

21.2.1.1 Exercise

1. Connect to your Windows 10 client and use net.exe to lookup users and groups in the domain. See if you can discover any interesting users or groups.

21.2.2.1 Exercises

1. Modify the PowerShell script to only return members of the Domain Admins group.

2. Modify the PowerShell script to return all computers in the domain.

3. Add a filter to only return computers running Windows 10.

21.2.3.1 Exercises

1. Repeat the enumeration to uncover the relationship between Secret\_Group, Nested\_Group, and Another\_Nested\_Group.

2. The script presented in this section required us to change the group name at each iteration. Adapt the script in order to unravel nested groups programmatically without knowing their names beforehand.

21.2.4.1 Exercises

1. Download and use PowerView to perform the same enumeration against the student VM while in the context of the Offsec account.

2. Log in to the student VM with the Jeff\_Admin account and perform a remote desktop login to the domain controller using the Jeff\_Admin account. Next, execute the Get-NetLoggedOn function on the student VM to discover logged-in users on the domain controller while in the context of the Jeff\_Admin account.

3. Repeat the enumeration by using the DownloadString method from the System.Net.WebClient class in order to download PowerView from your Kali system and execute it in memory without saving it to the hard disk.

21.2.5.2 Exercises

1. Repeat the steps from this section to discover the service principal name for the IIS server.

2. Discover any additional registered service principal names in the domain.

3. Update the script so the result includes the IP address of any servers where a service principal name is registered.

4. Use the Get-SPN script638 and rediscover the same service principal names

21.3.3.1 Exercises

1. Use Mimikatz to dump all password hashes from the student VM.

2. Log in to the domain controller as the JefLAdmin account through Remote Desktop and use Mimikatz to dump all password hashes from the server.

21 .3.4. 1 Exercises

1. Repeat the manual effort of requesting the service ticket. exporting it. and cracking it by using the tgsrepcrack.py Python script.

2. Perform the same action with any other SPNs in the domain.

3. Crack the same service ticket using John the Ripper.

4. Use the lnvoke-Kerberoast.ps1 script to repeat these exercises.

21.3.5. 1 Exercises

1. Use the PowerShell script in this module to guess the password of the jefLadmin user.

2. Use the Spray-Passwords.ps1 tool to perform a lookup brute force attack of all users in the domain from a password list.

21.4.2. 1 Exercise

1. Execute the overpass the hash attack above and gain an interactive command prompt on the domain controller. Make sure to reboot the Windows 10 client before starting the exercise to clear any cached Kerberos tickets.

21.4.3. 1 Exercises

1. Create and inject a silver ticket for the iis\_service account.

2. How can creating a silver ticket with group membership in the Domain Admins group for a SQL service provide a way to gain arbitrary code execution on the associated server?

3. Create a silver ticket for the SQL service account.

21.4.4. 1 Exercises

1. Repeat the exercise of launching Notepad using Excel and DCOM.

2. Improve the attack by replacing the VBA macro with a reverse shell connecting back to Netcat on your windows student VM.

3. Set up a pivoting channel from the domain controller to your Kali machine and obtain a reverse shell.

21 .5. 1. 1 Exercises

1. Repeat the steps shown above to dump the krbtgt password hash and create and use a golden ticket.

2. Why is the password hash for the krbtgt account changed during a functional level upgrade from Windows 2003 to Windows 2008?

22. 7. 3. 7 Exercises

1. Start the postgresql service and launch msfconsole.

2. Use the SMB, HTTP, and any other interesting auxiliary modules to scan the lab systems.

3. Review the hosts' information in the database.

22.2. 1. 1 Exercise

1. Exploit SyncBreeze using the existing Metasploit module.

22.3.3.2 Exercise

1. Take time to review and experiment with the various payloads available in Metasploit.

22.3.7. 7 Exercises

1. Create a staged and a non-staged Linux binary payload to use on your Kali system.

2. Setup a Netcat listener and run the non-staged payload. Does it work?

3. Setup a Netcat listener and run the staged payload. Does it work?

4. Get a Meterpreter shell on your Windows system. Practice file transfers.

5. Inject a payload into plink.exe. Test it on your Windows system.

6. Create an executable file running a Meterpreter payload and execute it on your Windows system.

7. After establishing a Meterpreter connection, setup a new transport type and change to it.

22.4. 7. 7 Exercise

1. Create a new Metasploit module for your SyncBreeze exploit.

22. 5.4. 1 Exercise

1. Use post-exploitation modules and extensions along with pivoting techniques to enumerate and compromise the domain controller from a meterpreter shell obtained from your Windows 10 client.

22. 6. 7. 7 Exercise

1. Create a resource script using both a second stage encoder and autorun scripts and use it with the meterpreter payload.

23. 7. 3. 7 Exercises. Now that we've walked through the basic features of PowerShell Empire, try these exercises on your own to solidify your knowledge.

1. Install and start PowerShell Empire on your Kali system.

2. Create a PowerShell Empire listener on your Kali machine and execute a stager on your Windows 10 client.

3. Experiment with the PowerShell Empire agent and its basic functionality.

23.3. 1. 1 Exercises

1. Set up a PowerShell Empire listener and stager and obtain a working agent.

2. Perform enumeration on the domain using various modules.

3. Perform a remote desktop login with the account JefLAdmin to ensure the credentials are cached on the Windows 10 client and then dump the credentials using PowerShell Empire.

4. Experiment with the different lateral movement modules.

24.2.2.2 Exercise

1. Use sqlmap to exploit the SQL injection and extract the username and password.

24.5. 1. 1 Exercises

1. Modify the original Python exploit and capture the reverse shell. Penetration Testing with Kali Linux 2.0 2. The original UDF exploit is advertised as a privilege escalation exploit. Why are we getting an unprivileged shell?