**LABS**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Command** | **Description** | | **B** | **P** | | |
| **# find / -perm /2000** | Find files with the SGID permissions in newer distros | | 6 | 38 | | |
| **# find / -perm /4000** | Find files with SUID permissions in newer distros | | 6 | 37 | | |
| **# find / -perm +2000** | Find files with the SGID permissions in older distros | | 6 | 38 | | |
| **# find / -perm +4000** | Find files with SUID permissions in older distros | | 6 | 37 | | |
| **|** | Pipe | | 6 | 54 | | |
| **cat** | Reads what is inside | |  |  | | |
| **cat /etc/shadow** | Where we can see password hashes stored | |  |  | | |
| **grep** |  | | 6 | 54 | | |
| **ls** | Shows you a list of files on the system and helps identify modification or changes to a file | | 6 | 53 | | |
| **ls -l /etc/shadow** | Show permissions for the file | |  |  | | |
| **netstat** | Shows network connections, including outbound connections and the services that are main the connections | | 6 | 53 | | |
| **passwd (username)** | Sets a password for a user | |  |  | | |
| **ps** | Provides a list of running services and details about the services | | 6 | 53 | | |
| **su** | Switch user | |  |  | | |
| **sudo** | Superuser do it, based on sudoers configuration file | |  |  | | |
| **sudo bash** | Log in as root | |  |  | | |
| **sudo systemctl | grep httpd** | Search for httpd service on the system | | 6 | 56 | | |
| **sudo systemctl | grep running** | List all active services on the system | | 6 | 56 | | |
| **sudo systemctl list-unit-files -type service -all** | List all services on the system | | 6 | 56 | | |
| **tail** |  | | 6 | 54 | | |
| **useradd (x)** | Add user | |  |  | | |
| **apt-get update** | Updates the system database of software updates that are available | | 6 | 67 | | |
| **apt-get install** | Installs the specified software with all associated dependencies | 6 | | | 67 |
| **apt-get remove package** | Removes the specified software but does not remove dependencies | 6 | | | 67 |
| **apt-get upgrade** | Upgrades all the software with the available updates | 6 | | | 67 |
| **sysctl -a --pattern randomize** | View current settings for ASLR | 6 | | | 75 |
| **last -f /var/log/wtmp** |  | 6 | | | 85 |
| **last -f /run/utmp** |  | 6 | | | 85 |
| **utmpdump /run/utmp** |  | 6 | | | 85 |
| **/etc/sysctl.conf** | Recommended way to edit the kernel variables is to edit, permanently commits the variable changes so that they load at boot time; whereas changing them while the system is running with the sysctl command only affects settings in memory, which are lost after a reboot. | 6 | | | 73 |
| **tail -c 600 /var/log/messages** | Displays the last 600 lines of a file | 6 | | | 54 |
|  |  |  | | |  |
| **Tcpdump** |  |  | | |  |
| **Tcpdump -r/path/to/file.pcap** | This command reads a pcap file. |  | | |  |
| **tcpdump --help** | List common commands for tcpdump | L | | | 4 |
| **man tcpdump** | View tcpdump man pages | L | | | 4 |
| **sudo su -** | Switch to root | L | | | 6 |
| **tcpdump -i lo** | This start a tcpdump and have it sniff your loopback (lo) | L | | | 6 |
| **ping 127.0.0.1 -c 1** | This runs the ping command against your loopback interface, also known as your localhost. We will send one ICMP Echo request using the “-c” option. The default IP address for your loopback interface is | L | | | 7 |
| **tcpdump -i eth0 port 21 -c 3** | This starts tcpdump with at appropriate interface (eth0). Port 21 selects only that port because windows can be chatty, and this will record the TCP 3-way handshake (-c 3) | L | | | 8 |
| **ftp 10.10.10.10** | This command makes an FTP connection to 10.10.10.10 (my windows VM) | L | | | 9 |
| **tcpdump -X -i eth0 port 21 -c 4** | -X will capture and display the hex and ascii output and –c4 changes the number of characters to 4 so that the FTP banner is displayed. | L | | | 10 |
| **tcpdump -a -i eth0 port 21 and src 10.10.10.20** | The –a option prints out the ASCII and not the hexadecimal version of the packets and this time we are only asking for packets coming from slingshot on port 21. | L | | | 12 |
| **tcpdump -i lo tcp port 333** | This is making a tcpdump interface connection to loopback to a port that is not listening (333) | L | | | 14 |
| **netcat 127.0.0.1 333** | This is making a netcat (TCP/UDP) connection to that address and port 333u=-0 | L | | | 14 |
| **-XX** | Tcpdump flag displays hex, ASCII, and the Ethernet header |  | | |  |
| **-n** | Tcpdump flag allows us to turn off hostname and port resolution |  | | |  |
| **SYN** | TCP flag is the only one set when initiating a connection |  | | |  |
| **$ netstat -natu | grep 'ESTABLISHED'** | Filters out “ESTABLISH” |  | | |  |
| **sudo tcpdump host 108.177.122.155** |  |  | | |  |
| **sudo tcpdump -i ens160** | Getting a dump from ens160 interface. This will overwhelm the system as there are no filters. |  | | |  |
| **sudo tcpdump port 22 -w SSH\_CAP** | Connecting to port 22 after establishing an SSH connection, and -w will write the output to a pcap file called “SSH\_CAP”. |  | | |  |
| **sudo tcpdump -i ens160 - nnSX udp port 9996** | No name resolution (make it go faster), show packets and the hexadecimal values on UDP port 9996 |  | | |  |
|  |  |  | | |  |
| **Wireshark** |  |  | | |  |
| **FTP and TFTP** | Wireshark: Two cleartext protocols did we analyze |  | | |  |
| **192.168.0.172** | The source IP address in packet #87 in the SEC401\_PCAP1.pcapng file |  | | |  |
| **TLSv1.2** | Protocol is used in packet #32 in the SEC401\_PCAP1.pcapng file |  | | |  |
|  | This is the screen right after clicking on ‘ethernet’ where you need to click on the blue fin and start capturing packets |  | | |  |
|  | This is a screen of ftp connections I established to start sniffing the packets on my Windows VM. I pinged my VM 10.10.10.10, used the username and password provided. |  | | |  |
|  | This is the screen after selecting ftp on the filter. After clicking on the first FTP packet, I selected follow, and I followed the TCP stream.  The screen is displaying username and password information. |  | | |  |
| **Ftp-data** | Selecting this filter will allow you to locate any data transmitted in the ftp connection. |  | | |  |
|  | To get to this screen, you need to apply the ftp-data on the filter section, follow the UDP stream and you will get the information retrieved in the packets. In this case, the pdf document. |  | | |  |
| **TFTP** | This filter will allow you to locate any TFTP packets |  | | |  |
|  | After locating the TFTP packets using a filter, select the first packet “Data Packet block”, follow the UDP stream and this will help you locate the secret hash. |  | | |  |
|  |  |  | | |  |
| **Aircrack-ng** |  |  | | |  |
| **aircrack-ng** | The primary cracking tool | L | | | 30 |
| **aireplay-ng** | Tool for injecting and replaying wireless frames | L | | | 30 |
| **airmon-ng** | Tool to enable and disable wireless interface monitoring | L | | | 30 |
| **airodump-ng** | Tool from the aircrack-ng suite captures wireless frames | L | | | 30 |
| **aircrack-ng -w all SEC401\_WPA2PSK.pcap -e SEC401** | Aircrack-ng is using the file “all” to crack the pcap file. | L | | | 34 |
| **Airmon-ng start wlan0** | The interface is started in monitor mode so that it is able to capture all desired traffic. | L | | | 30 |
| **Airodump-ng wlan0mon** | The airodump-ng tool is used to take a look at available wireless networks and clients. | L | | | 30 |
| **Aircrack-ng SEC401\_WEP.cap** | This command runs aircrack against the “SEC401.WEP.cap” file to crack the WEP key | L | | | 30 |
| **True** | To crack WPA, you must capture a valid WPA handshake | L | | | 35 |
| **2^24** | Keyspace associated with WEP IVs | L | | | 36 |
|  |  |  | | |  |
| **Hashcat** |  |  | | |  |
| **Straight mode** | **hashcat -m 500 -a 0 -o cracked.txt shadow /usr/share/wordlists/sqlmap.txt** | L | | | 39 |
| **Combination mode** | Set the path to the hashcat binary  HASHCAT= "/path/to/hashcat"  Set the path to the wordlist containing passwords to try  WORDLIST= "/usr/share/wordlists/sqlmap.txt"  Set the path to the file containing password hints or rules  RULES= "/path/to/rules.txt"  Set the hash mode as NTLM (hash type 500)  HASH\_MODE= 500  Set the file that contains the password hashes  HASH\_FILE= ”shadow"  Set the output file for cracked passwords  OUTPUT\_FILE= "cracked.txt"  $HASHCAT -m $HASH\_MODE -a 1 -r $RULES -o $OUTPUT\_FILE $HASH\_FILE $WORDLIST | L | | | 39 |
| **Brute-force mode** | Set the path to the hashcat binary  HASHCAT= "/path/to/hashcat"  Set the hash mode as NTLM (hash type 500)  HASH\_MODE= 500  Set the file that contains the password hashes  HASH\_FILE= "shadow"  Set the output file for cracked passwords  OUTPUT\_FILE= "cracked.txt"  Run hashcat in brute-force attack mode with the specified parameters  $HASHCAT -m $HASH\_MODE -a 3 -o $OUTPUT\_FILE $HASH\_FILE ?a?a?a?a?a?a?a?a | L | | | 39 |
| **Hybrid Wordlist + Mask mode** | Set the path to the hashcat binary  HASHCAT= "/path/to/hashcat"  Set the hash mode as NTLM (hash type 500)  HASH\_MODE= 500  Set the file that contains the password hashes  HASH\_FILE= "shadow"  Set the output file for cracked passwords  OUTPUT\_FILE= "cracked.txt"  Set the path to the wordlist containing passwords to try  WORDLIST= "/usr/share/wordlists/sqlmap.txt"  Set the mask pattern for additional password variations  MASK\_PATTERN= "?l?l?l?l?l?l?l?l"  Run hashcat in hybrid attack mode with the specified parameters  $HASHCAT -m $HASH\_MODE -a 6 -o $OUTPUT\_FILE $HASH\_FILE $WORDLIST $MASK\_PATTERN | L | | | 39 |
| **Hybrid Mask + Wordlist mode** | Set the path to the hashcat binary  HASHCAT= "/path/to/hashcat"  Set the hash mode as NTLM (hash type 500)  HASH\_MODE= 500  Set the file that contains the password hashes  HASH\_FILE= "shadow"  Set the output file for cracked passwords  OUTPUT\_FILE= "cracked.txt"  Set the mask pattern for additional password variations MASK\_PATTERN= "?l?l?l?l?l?l?l?l"  Set the path to the wordlist containing passwords to try  WORDLIST= "/usr/share/wordlists/sqlmap.txt"  Run hashcat in hybrid attack mode with the specified parameters  $HASHCAT -m $HASH\_MODE -a 7 -o $OUTPUT\_FILE $HASH\_FILE $WORDLIST $MASK\_PATTERN | L | | | 39 |
| **hashcat --help | grep "Attack Modes" -A9** | This runs hashcat to get its usage information filtering the words Attack modes using grep. | L | | | 38 |
| **cat shadow** | To find out the right hash indicator to use with the hashes inside of the | L | | | 40 |
| **hashcat --help |grep "MD5 (Unix)"** | We now know that these are MD5 hashes and must find the corresponding value to use with Hashcat that represents MD5. | L | | | 41 |
| **hashcat -m 500 -a 0 -o cracked.txt shadow /usr/share/wordlists/sqlmap.txt** | The "-m 500" is the hash type argument. We are using the value 500 to specify md5 (UNIX). The "-a 0" argument indicates straight or dictionary mode. The "-o cracked.txt" argument says to write any cracked passwords to the file cracked.txt. The next item supplied is our shadow file, followed by the wordlist we are using. | L | | | 41 |
|  | This screen is the result from the previous command.  Not shown in the screen is the passwords that were extracted, however, there was one password that I wasn’t able to crack due to the password being salted. In order to crack a salted password, we need to use a rule that will add special characters. |  | | |  |
|  | We have created a new file called sec401-rules using the touch command and echoed in the special characters. |  | | |  |
| **hashcat.potfile** | File used by Hashcat to store cracked passwords |  | | |  |
| **Hash indicator** | 1 for MD5  5 for SHA-256  6 for SHA-512  Example: $1$YJhL5Btw$kR0Ceq4h1OQfF3Yol This hash is an MD5 since it has a $1$. |  | | |  |
| **hashcat -m 500 -r sec401-rules -a 0 -o cracked.txt shadow /usr/share/wordlists/sqlmap.txt** | Showing in the screen is the result of the code, where we used -r flag to insert the rules folder in order to crack the salted password. |  | | |  |
|  | this screen shows our output file cracket.txt and it shows all passwords being cracked included the salted one. |  | | |  |
|  |  |  | | |  |
| **Cain & Abel** |  |  | | |  |
|  |  |  | | |  |
| **AppLocker** |  |  | | |  |
| **SAM Database** | Password database on Windows |  | | |  |
| **Publisher** | Considered the most secure condition to which rules can be applied with AppLocker |  | | |  |
| **True** | Windows installer files can be blocked from executing with AppLocker |  | | |  |
| **Application Identity** | Name of the service that must be started for AppLocker to function |  | | |  |
|  |  |  | | |  |
| **Nmap** |  |  | | |  |
| **Nmap –iflist** | Checks the interface, routers information. |  | | |  |
| **Nmap -sn 192.168.0.3/24** | Runs a simple scan, without having to use sudo. It will stop once nmap finds an open port. | L | | | 110 |
| **Sudo nmap -sS 192.168.100.139** | This is a half open scanning. This is a TCP half connection, as it only sends a SYN flag and not a SYN – ACK to stablish a connection | L | | | 110 |
| **Sudo nmap -sV 192.168.100.139** | To probe open ports and to get the servers VERSION information. In this case, port 22 is using openSSH 7.6p1. | L | | | 111 |
| **Sudo nmap -O 192.168.100.139** | Remotely detects the operating system using what’s known as TCP stack fingerprinting. Nmap sends a TCP and UPD to the target and examines every bit of the response and compares it with its nmap database. | L | | | 112 |
| **nmap -n -sT -O 10.10.10.10 -p21,80** | The default TTL on a Windows system is 128. If Nmap inspects the TTL of a packet received from the target device and it is 120, Nmap could infer that the system is likely to run Windows because many Linux OS’ and macOS default to a TTL of 64 and other OS such as OpenBSD default to a TTL of 255. Remember, this is just an example and Nmap uses many different checks to try and determine the OS, such as IPID sampling and TCP window size checking. | L | | | 113 |
| **nmap -n -sT -A 10.10.10.10 -p21,80** | You saw previously that TCP port 80 listens on 10.10.10.10; however, you do not know what service runs on this port. TCP Port 80 is the default and well-known port for HTTP, but you need to confirm this assumption. You also want to know what web server version runs on this port. To do this, you can use Nmap's application version scanning. Actually, if you use the -A option, Nmap performs both OS fingerprinting and application version scanning. | L | | | 113 |
| **ls /usr/share/nmap/scripts/p\*** | A large number of NSE scripts are on your system by default, which are stored in the "/usr/share/nmap/scripts/" folder. Just to look at a small sample of the scripts available, enter the command "ls /usr/share/nmap/scripts/p\*" to get all scripts that begin with the letter "p". | L | | | 114 |
| **cat community.lst** | The community string wordlist used to run a dictionary attack against the target system. From your "/home/sec401/labs/401.3/nmap" folder, enter the following command "cat community.lst" and press enter. | L | | | 115 |
| **nmap -sU -p161 --script snmp-brute 10.10.10.10 --script-args snmp-brute.communitiesdb=community.lst** | To run a specific NSE script, use the --script option, followed by the script name and arguments. If you want to run the most common scripts, use the -sc option. Scripts are broken into various categories, some of which could potentially do harm to a system or service. Run the snmp-brute script and provide it with a wordlist that resides in your current directory called "community.lst". Note that the command is long and wraps to two lines. Be careful typing it, as typos cause the command to fail or give the wrong results. | L | | | 116 |
| **snmpcheck -t 10.10.10.10 -c publ1c | grep "User accounts" -A12** | The snmpcheck tool requires that you supply it with the IP address of the target system and the community string that you recovered. A large amount of data displays on the screen; therefore, we will use grep to look at specific results. You can exclude the grep command to see the complete response | L | | | 117 |
| **oX** | Nmap option enables you to write the results in XML format |  | | |  |
| **-sS** | Nmap scan type performs a Stealth Scan |  | | |  |
| **Lua** | Language are NSE scripts written |  | | |  |
|  |  |  | | |  |
| **Malicious Software** |  |  | | |  |
| **strings -n 14 trojan1 | more** | This tool parses through any input file, searching for ASCII characters. We are using the "-n" option to only display strings 14 characters or longer. Feel free to change this value. The output may still be rather large, so we will pipe the results to the "more" tool so that we only get one page at a time. | L | | | 122 |
|  |  |  | | |  |
| **Command Injection** |  |  | | |  |
| **Segmentation fault** | Message did we get during the buffer overflow | L | | | 128 |
| **./** | We prepend to a program to ensure it runs from the current folder | L | | | 128 |
| **rbash** | Rbash is a restricted shell that limits many of the commands you can normally run. | L | | | 131 |
| **python cmd\_vuln.py** | Run the "cmd\_vuln.py" Python script, to get autocomplete permissions in a restricted shell. | L | | | 131 |
| **system** | Name of the function enabling this command injection bug |  | | |  |
| **False** | You used the | (pipe) symbol to append on an additional command |  | | |  |
| **rbash** | Command did you use to go to a restricted shell |  | | |  |
|  |  |  | | |  |
| **Hping3** |  |  | | |  |
| **hping3 --help | grep Mode -A7** | To view the various modes supported by hping3 |  | | | 138 |
| **hping3 --help | grep "\-\-spoof" -A7 -B1** | To see a sampling of the IP options with hping3 You got these results using the grep command as you have many times; however, a couple new things have appeared. First, you want to grep only for “--spoof", but the hyphens are treated as special characters. To handle this issue, use the “\” character to escape them so that they are not interpreted as special. Next, we added the -B1 option, which simply displays the line before the matched pattern. You do this only to get specific data from the help menu printed to the screen |  | | | 138 |
| **hping3 --help | grep "\-\-base" -A15 -B1** | Look at a sampling of TCP and UDP options. |  | | | 139 |
| **hping3 -S 10.10.10.10 -p 21 -c 1** | Simple packet with hping3 that can send a SYN packet to TCP port 21 on 10.10.10.10. To do this, |  | | | 141 |
| **hping3 -S 10.10.10.10 -a 10.11.12.13 -p 21 -c 1** | Spoof the IP address of your outgoing packet so that the source is from an address not belonging to the Slingshot VM. To do this, we set the "-a" flag, followed by the address you want to use as the source IP address. Use the address 10.11.12.13. With tcpdump still up and running in one Terminal window, |  | | | 144 |
| **-a** | Hping3 option performs IP source address spoofing |  | | |  |
| **true** | Hping3 can transfer files covertly. |  | | |  |
| **TTL** | "-T" flag with hping3, what can we set the value |  | | |  |
|  |  |  | | |  |
| **Steganography** |  |  | | |  |
| **embed** | Steganography mode |  | | |  |
| **four times** | Using the Pre-Scale option increases the host size by how many times |  | | |  |
| **PNG** | Host file format supported by Image Steganography |  | | |  |
|  |  |  | | |  |
| **GNU** |  |  | | |  |
| **gedit** | Is the default text editor for the GNOME desktop environment on Linux. |  | | | 171 |
| **GNU Privacy Assistant** | The name of the GUI you can use to manage GPG |  | | |  |
| **Signer's Private** | Key is used to encrypt the hash for a digital signature |  | | |  |
| **RSA** | Public key crypto algorithm was used by GPG in this exercise |  | | |  |
|  |  |  | | |  |
| **SNORT** |  |  | | |  |
| **Sudo snort -v** | Runs in verbose mode with root privileges. |  | | |  |
| **Ip a** | Will show your internet address |  | | |  |
| **Sudo gedit /etc/snort/rules/local.rules** | Local rules come up in gedit. This is how we set rules for alerts. |  | | |  |
| **Alert icmp any any -> 192.168.100.0/24 any (msg: IICMP even”; sid: 1000009;)** | This is a snort rule created in gedit. |  | | |  |
| **Sudo snort -T -c /etc/snort/rules/local.rules** | This is how snort validates the configuration |  | | |  |
| **Sudo snort -d – l ./ -c /etc/snort/rules/local.rules** | Dump to the application layer, -l log to the directory, -c to the config file. |  | | |  |
| **cd /etc/snort** | Configuration file for Snort is quite large and is available at /etc/snort/snort.conf.  Snort is currently set up to write the log and alert files to the default location at /var/log/snort. |  | | | 186 |
| **snort -c /etc/snort/snort.conf -i eth0 -A full** | To start up Snort. The "-c" option enables you to specify the configuration file to use; the "-i" option enables you to specify the interface; and the "-A full" options sets alerting to full mode. |  | | | 190 |
| **xxd snort.log** | Xxd tool to look at the "snort.log" file. The xxd tool simply dumps the contents of a file in hexadecimal. |  | | | 193 |
| **snort -c /etc/snort/snort.conf -r /home/sec401/labs/401.4/snort/snort.pcap -A full** | Run Snort against the "snort.pcap" file located in your "/home/sec401/labs/401.4/snort" folder. Use the "-r" option to tell Snort to read from an existing PCAP file. Just like before, a lot of information should stream by on the screen, and then snort should automatically stop. |  | | | 196 |
| **true** | Snort can read existing tcpdump PCAP files |  | | |  |
| **Cisco Systems** | Sourcefire was acquired by |  | | |  |
| **content** | Snort signature syntax to examine application layer data |  | | |  |
|  |  |  | | |  |
| **Hashing** |  |  | | |  |
| **hashing algorithm**  **fixed-length output** | A grey rectangular sign with black text  Description automatically generated |  | | | 199 |
| **md5sum < file1.txt** | Run the file through a couple of hashing algorithm tools |  | | | 200 |
| **sha1sum < file1.txt** | Sha1sum tool |  | | | 202 |
| **SHA1** | Hashing algorithm used in the lab has an output length of 20 bytes |  | | |  |
| **Collision** | Called when two different files produce the same hash |  | | |  |
| **Tripwire** | Commercial integrity checking tool |  | | |  |
|  |  |  | | |  |
| **Process Hacker** |  |  | | |  |
| **The Terminator** | Is a Process Hacker tool that will attempt to terminate a process |  | | |  |
| **bright magenta** | Color is used to highlight "Packed Processes" by default |  | | |  |
| **the Token tab** | N the properties of a process, what tab shows privileges |  | | |  |
|  |  |  | | |  |
| **NTFS Permissions Reporter** | The tool scans the Windows file system and reports all permissions associated with users and groups. When users have administrative access to their system it is not uncommon for changes to be made to file system permissions. These changes may be intentional or unintentional. Regardless of the reason behind the changes made, what may be seemingly harmless could in fact result in a compromised system. |  | | | 227 |
| **TrustedInstaller** | It is a part of Windows Resource Protection whose goal is to prevent unauthorized file system and registry changes as well as maintain the integrity of the file system. |  | | | 236 |
| **TrustedInstaller** | User account is part of Windows Resource Protection |  | | |  |
| **Windows\System32** | File system location where DLL files are stored |  | | |  |
|  |  |  | | |  |
| **SECEDIT.EXE** |  |  | | |  |
| **SECEDIT.EXE** | The SECEDIT.EXE command line tool can be used to automate the application of an INF security template to reconfigure a Windows computer. The SECEDIT.EXE tool may also be used to audit a Windows computer by comparing the settings of the computer to the settings defined in the INF template. The output of an audit like this is a plaintext log file. This log file can be searched for keywords that indicate mismatches found between the computer and the template. These commands can be done in the old CMD.EXE command shell, but PowerShell is the future of Windows scripting and command shell execution. |  | | | 241 |
| **secedit.exe /analyze** | Command line switches of the secedit.exe tool for analyzing a computer |  | | |  |
| **secedit.exe /analyze /db temp.sdb /cfg SecurityTemplate.inf /log log.txt** | Compare the settings defined in the security template against the local computer, saving the output to a text log file named "log.txt". |  | | |  |
| **secedit.exe /configure** | Review the command line switches of the secedit.exe tool for reconfiguring a computer |  | | |  |
| **secedit.exe /configure /db temp.sdb /cfg SecurityTemplate.inf** | Reconfigure the computer by applying the securitytemplate.inf file. The computer has now been reconfigured to match the security template. Let's reaudit the machine again with the template. |  | | |  |
| **ise .\log.txt** | Open the log file created by secedit (log.txt) in PowerShell |  | | |  |
|  |  |  | | |  |
| **PowerShell** |  |  | | |  |
| **Get-Content .\log.txt | Select-String -Pattern "Mismatch"** | Get the contents of the log.txt file and pipe (" | ") the lines from that file into the PowerShell Select-String command to compare each line against a regular expression pattern (very similar to grep on Linux). |  | | |  |
| **powershell\_ise.exe** | Command is used to launch the graphical PowerShell ISE editor |  | | |  |
| **Mismatch** | Keyword do we look for in secedit.exe log files to find mismatches |  | | |  |
| **ise** | Command is used to open a text file in the PowerShell ISE editor |  | | |  |
| **PowerShell Scripting** | The future of Windows automation scripting is PowerShell. PowerShell is a replacement for the old CMD.EXE command shell. PowerShell also includes an object-oriented scripting language that is designed to be as easy to learn as possible (easier than C# at least, but "easy" is a relative term---it can still be hard) |  | | | 243 |
| **Get-Process** | Command to see a list of running processes |  | | |  |
| **NPM(K)** | Non-Paged Memory in kilobytes. |  | | |  |
| **PM(K)** | Ageable Memory in kilobytes. |  | | |  |
| **WS(K)** | Working Set memory in kilobytes. |  | | |  |
| **CPU(s)** | Total number of seconds the process has consumed across all CPU cores |  | | |  |
| **Id** | He Process Identifier number (PID) |  | | |  |
| **SI** | Session ID, where 0 is for all services and 1 represents the first logged on user |  | | |  |
| **ProcessName** | Name of the process, which is usually the EXE name. |  | | |  |
| **Get-Process -Name lsass | Format-List \*** | Command to see all the properties of just the LSASS process |  | | |  |
| **mspaint.exe**  **Get-Process -Name mspaint | Format-List \*** | Launch Microsoft Paint (MSPAINT.EXE) and show its details with these commands |  | | |  |
| **$PaintApp = Get-Process -Name mspaint** | Capture the object representing the mspaint process in a new variable named $paintapp. Instead of displaying the object representing the mspaint process, we have captured that object inside a variable ($paintapp), which can now be used later in the shell or in a script. |  | | |  |
| **$PaintApp.Kill()** | Terminate the mspaint process by invoking the Kill() method on the object that represents it by typing the name of the variable ($paintapp), followed by a period ("."), followed by the word "Kill", and then a pair of parentheses with no space characters before or in between them ("()") |  | | |  |
| **IntelliSense** | Pop-up list of properties and commands/methods when you typed the period (".") at the end of the $paintapp variable |  | | |  |
| **Get-Process | Select-Object Name,Id,Path | Export-Csv -Path ProcList.csv** | Save the Name, Id, and Path properties of all running processes to a comma-delimited text file. You got all the process objects just like before, then piped ("|") all those objects into the Select-Object command. The Select-Object command strips away all the other properties of those objects except those you specify, for example, Name, Id, and Path. Then the objects are piped into Export-Csv, which saves the data as a comma-delimited text file named "proclist.csv" |  | | |  |
|  | Open the proclist.csv text file in a new tab |  | | |  |
| **Get-Process | Select-Object Name,Id,Path | Out-GridView** | The Out-gridview application shows the same data, but in a graphical application where you can click column headers to sort by column; filter by typing in a name at the top or add your own custom filter criteria by clicking on the "Add Criteria" button. |  | | |  |
| **cls** | Like clear in Linux |  | | |  |
| **Get-Service** | Display a list of all background services and their current statuses |  | | |  |
| **Clear-DnsClientCache** | When you clear the Dnscache, it clears from memory any cached name-to-IP address mappings that were previously resolved through DNS. The next time a Fully Qualified Domain Name (FQDN) needs to be resolved, the computer will do a fresh DNS query to get the correct IP address. |  | | |  |
| **Get-Service | Select-Object DisplayName,Status | ConvertTo-Html | Out-File -FilePath Services.html** | O save your list of services to an HTML file that can be opened in a web browser, service objects have all of their properties stripped away by the Select-Object command except for displayname and Status, then these objects are piped into convertto-Html, which does exactly what its name implies: converts object data into an HTML table. Sometimes HTML is better for human consumption than CSV or XML. You can pipe the output of almost any command into convertto-Html! Finally, the HTML text is piped into Out-File, which saves the HTML text to a new file named "Services.html". This file can be opened in your web browser. |  | | |  |
| **.\Services.html** | O open the Services.html file in your browser |  | | |  |
| **dir** | List all files in the current directory |  | | |  |
| **dir .\Services.html | Format-List \*** | To display all the properties of the Services.html file, including those properties not shown by default. |  | | |  |
| **dir | Sort-Object CreationTime | Select-Object CreationTime,FullName** | To sort the listed files by the date and time they were created |  | | |  |
| **Get-Process -name “notepad” | Format-list \*** | This command will help you retrieve Notepad’s process information, but first type “notepad.exe” on the shell to launch it. |  | | |  |
| **Copy-Item -Path .\Services.html -Destination .\Copied.html**  **dir \*.html** | O make a copy of the Services.html file and list all \*.html files, |  | | |  |
| **Get-Content .\temp.txt|Select-String -Pattern “Mismatch”** | This command will extract the word “Mismatch” from temp.txt |  | | |  |
| **Get-FileHash -Algorithm SHA256 -Path \*.html** | O compute the SHA-256 hashes of all HTML files in the current directory |  | | |  |
| **"AAA" | Out-File -Append -FilePath .\Copied.html**  **Get-FileHash -Algorithm SHA256 -Path \*.html** | The Get-filehash command could be used in a scheduled script to check the integrity of files on a web server, then alert you when a file changes (perhaps by sending you an email message with the built-in Send-mailmessage command).    Append some text ("AAA") to the end of the Copied.html file, then compute hashes again. |  | | |  |
| **Get-Content -Path .\Copied.html** | O view the contents of a file inside the command shell |  | | |  |
| **ise .\Show-ComputerInfo.ps1** | Pen the Show-computerinfo.ps1 script in a new ISE tab |  | | |  |
| **Windows Management Instrumentation** | WMI) service is installed and running by default on all Windows computers. PowerShell can talk to the WMI service on local and remote machines. Among the many things you can do with WMI, you can query all sorts of useful information. WMI service queries that look like SQL queries to a database, for example, "SELECT \* FROM Win32\_computersystem". This is because WMI organizes its data into virtual database tables called "classes", and "Win32\_computersystem" is one of these classes/tables of information. |  | | |  |
| **Set-ExecutionPolicy RemoteSigned** |  |  | | |  |
| **Get-WmiObject -Query "SELECT \* FROM Win32\_BIOS" -ComputerName LocalHost** | Query BIOS information from a remote computer. The data output by this command is the same you saw earlier when you ran the script, which includes this command. The difference this time is the "-computername localhost" part. When you do not specify a remote computer name, powershell defaults to the local computer automatically, but you can provide a remote computer name if you wish. |  | | |  |
| **Get-WinEvent -ListLog \* | Select-Object LogName** | Run the following command to see the names of all local Windows Event Logs |  | | |  |
| **Get-WinEvent -LogName System -MaxEvents 10 | Select-Object TimeCreated,Id,Message** | How the last 10 events from the System log, showing only the time each event was written to the log, the ID number of each event, and the message payload of each event, |  | | |  |
| **Get-WinEvent -LogName System -MaxEvents 10 -ComputerName LocalHost | Select-Object TimeCreated,Id,Message | Export-Csv -Path LogData.csv** | Run the prior command again (hit the "Up Arrow" button on your keyboard) but edit the command to pretend to query the System log on a remote computer, and then save the output to a CSV file. |  | | |  |
| **ise .\LogData.csv** | Open that logdata.csv text file in a new tab |  | | |  |
| **Get-Process and Get-Service** | PowerShell commands show processes and services |  | | |  |
| **Export-Csv** | PowerShell command can export objects to a CSV text file |  | | |  |
| **Select-Object** | PowerShell command strips away properties we don't care about |  | | |  |
|  |  |  | | |  |
|  | **imagesteganography.codeplex.com.** |  | | |  |
|  | **hping.org** |  | | |  |
|  | **secedit.exe** |  | | |  |
|  | **www.aircrack-ng.org** |  | | |  |
|  | **hashcat.net** |  | | |  |
|  | **docs.microsoft.com** |  | | |  |
|  | **nmap.org** |  | | |  |
|  | **www.cjwdev.com** |  | | |  |
|  | **processhacker.sourceforge.net** |  | | |  |
|  | **www.snort.org** |  | | |  |
|  | **www.tcpdump.org** |  | | |  |
|  | **www.wireshark.org.** |  | | |  |