**CSCI 5742-Cybersecurity Programming-Lab 04-Powershell Programming**

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Lucas Fulmer**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Goals: Understand Powershell Programming

Deliverables: Lab Writeup, lastnameLab4.txt

Scenario: System administrators can not possibly look at all security issues associated thousands of clients on a network. Clearly the key is automation, but there are few tools available for Windows. Most tools are focused around operations not specific security. An answer is powershell. Note that this lab has you dealing with settings for an individual computer, but if it is attached to an active directory, with identifying information like the computer name and domain, you can essentially use a for loop and do everything (and more) that you can do in this lab.

This lab just gives you a basic idea that if you can set it via an Active Directory Server, you can set it and get values using powershell with the right permissions.

**Part I - Accessing the Registry**

* Open Windows 10. Go to VM->Snapshot and go to Initial Config. Set it to Host Only. Run it Log in as Student: **Give it an incorrect password twice** before putting in the correct password: Student123
* Verify that you can drag and drop. Create a file on the desktop and move it to the host desktop.
* Powershell is available on all Windows 7 and above operating systems. Often system administrators automate operating system tasks with it rather than having to use the GUI. Powershell is scripting language that is far more powerful than a Bash (for Linux) or the Windows Command Line. Powershell is generally not case or space sensitive except for strings, but it is a good practice to use standard conventions.
* Powershell has four execution policies: Restricted, All Signed, RemoteSigned and unrestricted. There are ways hackers use to bypass these policies, and depending on how the policies are set up in active directory, many times are successful at bypassing the security.
* We won't attempt these for this lab, but if you want to be an uber-hacker, on your own time verify these ways by setting up a series of Active Directory policies. (Of course you also have to escalate your privileges to a local admin, which usually requires some kind of exploit if you are a bad guy)
* <https://blog.netspi.com/15-ways-to-bypass-the-powershell-execution-policy/>
* For now just go to start and in the search box type in **powershell**, right click and run the windows powershell as administrator. Note that you could also run scripts from the standard command line by putting the word powershell.exe infront of script commands.
* **Get-ExecutionPolicy** (tells you what is currently set)…note tabs will complete statements once they are unique
* **Set-ExecutionPolicy Unrestricted** (If it doesn't let you do it, you are not running as an administrator)
* **$MaximumHistoryCount=200** (In order to help motivate you to test every command in this lab, you will print out your history for this session at the end. If you close the powershell window you will lose all of these commands, so you should go to the end and type in the command)
* **cd c:\users\student\documents**
* Define in your own words the purpose of the Microsoft Windows Registry

**To reduce access of configuration files as well as establish an easy-to-use database for the system to retrieve configuration files.**

* First, we want to check out the GUI version of the Windows Registry
* **regedit**
* Note that the registry is set up as a directory tree starting at Computer, then 5 top level "keys" List those 5 top level keys

**HKEY\_CLASSES\_ROOT**

**HKEY\_CURRENT\_USER**

**HKEY\_LOCAL\_MACHINE**

**HKEY\_USERS**

**HKEY\_CURRENT\_CONFIG**

* Let's say that you want to find what software is loaded on the computer. If you know what software is running, you may better understand the vulnerabilities and possible exploits available to a system. (We will do everything on one machine, but note that with some minor script changes you could verify information for an entire domain
* Most professional software has an entry in the Registry. From the Registry Editor you could go to HKEY-LOCAL-MACHINE->Software and see all of the software loaded.
* Now for some help with powerscript commands. There are a number of parameters/information we can get from the system with Get and Set commands. For a list of commands, at the prompt
* **Get-Command**
* **Get-Help Get\***
* In addition to these hundreds of commands, powershell has all of the usual script constructs like conditionals, comparisons, For loops, ForEach-Object loops, while, do-while, do-until
* From the PS prompt first we want to see if there are any aliases set up (for easier typing)
* **Get-PSDrive**
* What is the alias for HKEY-LOCAL-MACHINE? **HKLM**
* Powershell uses pipes | to direct output of one command into another
* Most of these commands will work with file systems and registry items
* **Get-ChildItem HKLM:\SOFTWARE**
* or write to a file:
* **Get-ChildItem HKLM:\SOFTWARE > software.txt**
* This lists all of the software loaded into the Registry (which is most of the professional software on the machine). You will see that there is a ton of Microsoft software. To get that breakdown, go further down the hierarchy
* **Get-ChildItem HKLM:\SOFTWARE\Microsoft**
* Now we may want to find more information about what version of Microsoft Internet Explorer the machine is using (If we know the version, we may be able to determine vulnerabilities or possible exploits)
* **Why might you think you are in a virtual machine?**

**Using the following command will let you know that you are in a VM:**

**Get-wmiobject -computer LocalHost win32\_computersystem**

* Why might a hacker care to know if they are inside a virtual machine?

**Hacker will likely not be able to do any damage or exploit an actual network if he is within a VM**.

**(Get-ItemProperty 'HKLM:\SOFTWARE\Microsoft\Internet Explorer').Version**

* From the internet connected host go to <https://web.nvd.nist.gov/view/vuln/search>
* Look up that version (just the integer, not the decimal) of Internet Explorer. Use the advanced search, vendor Microsoft, product ie, version 9….this fill in database is VERY slow so have patience, once character at a time
* How many critical vulnerabilities would there be for this version of Internet Explorer? **0**
* (If there are zero, explain why this might be)

**It may be that patches have already been released for a potential vulnerability which will downgrade its level. It could also be that the vulnerability has not yet been discovered.**

* Click in to the latest High Vulnerability CVE link. Go down to the References to Advisories, Solutions and Tools. Click on the technet.microsoft link
* What is the recommended solution, patch etc to fix this vulnerability?

**KB2699988**

* Now we want to see the history of typed URLs in Microsoft Internet Explorer
* Note that it We need to combine the Get-ChildItem with all of the items in Get-ItemProperty like:
* **Get-ItemProperty "hkcu:\software\microsoft\internet explorer\TypedUrls"**
* Why might a hacker want to see a list of the most recent typed urls?

**If a hacker is able to see what urls you have accessed recently, he can check the recent keystrokes to try and identify passwords. He may also be able to log directly into a website which you have recently visited if your password is saved from the last visit.**

**Part II - Determine Patch Levels**

* To determine which patches are active on the machine
* **Get-HotFix | sort-object -Descending InstalledOn**
* If you had a list of all HotFixes for a given MS operating system, list psuedocode for a script to verify what is missing on a given machine.

**Part III - Event Log Parsing**

* There are thousands of Events logged in a session. These events are logged in different logs
* From the search bar type in secpol.msc. Go to Local Policies->Audit Policy->Audit Account Logon Events. Check the Failures box
* From command prompt, gpupdate /force
* Sign out. Sign back in with at lease 2 incorrect passwords before signing in with Student123
* **Get-WinE**vent (Control + c) to stop
* **Get-WinEvent -ListLog \***
* There are many things that you can learn from regular viewing of logs, especially if you define events that could have an operational or security impact.
* **$end=get-date**
* **$start=$end.AddDays(-1)**
* **Get-WinEvent -FilterHashtable @{logname='security';StartTime=$start; EndTime=$end} | Where { $\_.Message -Match "failed" }| Format-List**
* Notice that you can see who failed attempts, where they logged in from etc.
* Look through the system log with the pipe to Format-List. List something that is being tracked which could be of some security value (and describe why).

**The date, time, and user are logged. This can be used to establish a pattern of use. A hacker could exploit this by attacking the host while the user is not likely to be active.**

**Part IV - Processes and directories**

* Though the registry can give you a lot of information, a hacker needs to understand the current status of a machine. You could do most of these tasks from the command line, but can be more valuable when combined with other Powershell functions
* To see what files are in a specific directory, you can
* **Get-ChildItem c:\users\student**
* **Get-ChildItem -r c:\users\student** (control C to quit)
* Now you may want to find what processes are running. If you are a good guy, you want to verify any processes or services that should/should not be running. If you are a bad guy, you may want to inject some evil code into an existing process
* Describe what type of script that you might run daily to see if any changes or additions have been made to the c:\windows\system32 directory (which contains a lot of the critical libraries, and should be regularly analyzed)

**Create a .txt file of the existing directory. Create a script which compares the current directory to the directory that you have saved in a .txt file and identifies discrepancies.**

* **Get-Process**
* You can cat the file to verify that it converted it to xml. Note there is a lot of information, including exactly when each command was entered.
* Turn in this xml file with this lab writeup to Canvas. (You will have to email it to yourself…make sure that the image is in NAT, and the ip address is automatic)
* **Get-Process**
* You could further limit this output (or any other objects created by these get commands by piping the output to a where delimiter:
* **Get-Process | Get-Member**
* This command shows you the member aliases, variables and methods available to that commandlet. This will allow you to select a subset of the records.
* **Get-Process | where {$\_.name -eq "svchost"}**
* **Get-Process | where-object {$\_.name -like "\*svc\*"}**
* The $\_. (the current object. The dot gives access to the object member
* Now we want all of the information known about each process. This might be used if you are trying to determine if you have malicious software running on that machine
* **Get-Service**
* **Get-Service | Findstr Running**
* Now you are going to make an xml formatted history of your last powershell commands (only keeps the last 64) most of these outputs can be piped into an XML format, so other xml tools can parse them.
* **Get-History -Count $MaximumHistoryCount | Export-CliXML c:\users\student\lastnameLab4.xml**
* **Get-History -Count $MaximumHistoryCount | Out-File c:\users\student\lastnamelab4.txt**
* Read them with notepad What additional information do you have by exporting to xml?

**Date time group for when commands were executed.**