CYB631 Automating Information Security with Python and Shell Scripting

Lab 2: Analyzing Logs and Other Administrators' Tasks

Your Name: **Vaibhav A Mayekar**

Exercises:

[Environment: Stating with PowerShell ISE]

1. Windows 10 environment is needed for the lab. Launch Windows PowerShell ISE. Click Start->Run, and then type PowerShell ISE.
2. Now, open a new PowerShell ISE and run as an administrator. Make sure that you navigate to the directory where your script is. Then shorten the command prompt and change the executive policy to RemoteSigned or Unrestricted. Remote Signed will required all scripts downloaded from the Internet to be signed digitally. Unrestricted will allow execution but provide prompts for users to confirm.

```
cd \\pace.edu\shares\users\lchen\Desktop\CYB631
function Prompt {"LCHEN PS> "}
Set-ExecutionPolicy -ExecutionPolicy Unrestricted
```

[Exercise I: Basic Object Operations]

3. In a variable, let us store the value of an object from a cmdlet. In the example below, **get-date** cmdlet returns a DateTime object. $today is an **instance** of DateTime.

```
$today=get-date
$today.Date
$today.DayOfWeek
$today.DayOfYear
$today.ToLongDateString()
```

4. To use static member of a class, for example, **System** class.

```
[System.DateTime]::Today

[System.DateTime]::DaysInMonth(2023,9)
```

5. You can also create an instance. For example, the example below creates a new instance of System.Random which has a method that generates random numbers.

```
$rnum=New-Object System.Random

$rnum.Next()
```

6. Paste the screenshot of your results from all of the commands in this exercise.

```
PS C:\Users\mayek\OneDrive\Desktop\python shell scripting> [System.DateTime]::Today

Monday, September 18, 2023 12:00:00 AM


PS C:\Users\mayek\OneDrive\Desktop\python shell scripting> [System.DateTime]::DaysInMonth(2021,6)
30

PS C:\Users\mayek\OneDrive\Desktop\python shell scripting> .\Lab2.ps1

Monday, September 18, 2023 12:00:00 AM
Monday, September 18, 2023
Monday, September 18, 2023 12:00:00 AM
30
183993981
```

[Exercise II: List, Array and Hashtables]

7. Let us create an array that holds several items.

   $myarray=1, 2, 3, "apple","banana"

   $myarray

8. Create an array of 10 strings and assign values to some of the strings.

   $myarray= New-object string[] 10

   $myarray[5]="apple"

   $myarray[2]="pear"

   $myarray

9. What is the results of the above step?

   pear

   apple

10. Create an array of multiple dimensions.

    $arr2=@((1,2,3,4),(5,6,7,8))

    $arr2[0][1]

11. What is the value of **$arr2[1][1]** ? **6**

12. Sort the array of lists.

    $list= “watermelon”,”pear”,”banana”,”apple”
    [Array]::sort($list)
    $list

13. What is the result of the step above?

    apple
    banana
    pear

watermelon

14. Hashtable in PowerShell is a collection of items with labels that you provided.

    $htable = @{}
    $htable[“Mary”]=5
    $htable[“John”]=7
    $htable[“Eric”]=6
    $htable

15. Paste the results of the command above.

```
PS C:\Users\mayek\OneDrive\Desktop\python shell scripting> .\Lab2.ps1

Name                           Value
----                           -----
Eric                           6
John                           7
Mary                           5
```

[Exercise III: Files and Directories]

16. Exam what files you have under the current directory.

    Get-ChildItem

17. Find files with a certain extension, such as *.ps1.

    Get-ChildItem -recurse -filter *.ps1

18. **Compare-Object** cmdlet can be used to compare the differences between two files or two objects that are returned by two commands. To compare two files,

    “Ten Apples Up on Top!”> .\file1.txt

    “Ten Apples Up on Top*&^!” > file2.txt

    $c1=Get-Content .\file1.txt

    $c2=Get-Content .\file2.txt

    Compare-Object $c1 $c2

19. Find files modified in the past 10 days and sort them by length of the file.

    $cdate=(get-date).AddDays(10)

    Get-ChildItem -Recurse | Where-Object{$_.LastWriteTime -lt $cdate}|Sort-Object -property length

20. Paste a screenshot with all of the results from this exercise.

```
PS C:\Users\mayek\OneDrive\Desktop\python shell scripting> Get-ChildItem


    Directory: C:\Users\mayek\OneDrive\Desktop\python shell scripting


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         9/13/2023   1:07 PM                cyb631
d-----         9/18/2023   8:26 PM                lab2
d-----         9/18/2023   4:18 PM                testdir1
-a----         9/18/2023   8:26 PM          29568 2.1.png
-a----         9/18/2023   8:31 PM           8091 2.2.png
-a----         9/18/2023   8:08 PM          18608 2.5.png
-a----         9/18/2023   8:09 PM          21707 2.6.png
-a----         9/14/2023   8:38 PM            374 cputime.ps1
-a----         9/14/2023   8:40 PM            374 cputime.txt
-a----         9/18/2023   8:37 PM             48 file1.txt
-a----         9/18/2023   8:37 PM             54 file2.txt
-a----         9/18/2023   4:19 PM           1659 Get-DiskUsage.ps1
-a----         9/18/2023   8:03 PM           1662 Get-DiskUsage1.ps1
-a----         9/13/2023   1:05 PM          26888 lab 1.1.png
-a----         9/13/2023   1:11 PM          28254 lab 1.2.png
-a----         9/13/2023   1:53 PM          79926 lab 1.3.png
-a----         9/13/2023   1:58 PM          14589 lab 1.4.png
-a----         9/13/2023   2:16 PM          43845 lab 1.5.png
-a----         9/14/2023   1:56 PM          53046 lab 1.6.png
-a----         9/14/2023   2:00 PM          17393 lab 1.7.png
-a----         9/14/2023   3:01 PM         616448 Lab1_Mayekarvaibhav.doc
-a----         9/18/2023   8:37 PM            301 Lab2.ps1
-a----         9/14/2023   1:48 PM            108 Script1.ps1
-a----         9/18/2023   8:06 PM            218 showtoday.ps1
-a----         9/18/2023   8:11 PM            247 ShowToday.psm1


PS C:\Users\mayek\OneDrive\Desktop\python shell scripting> Get-ChildItem -recurse -filter *.ps1


    Directory: C:\Users\mayek\OneDrive\Desktop\python shell scripting
```

```
    Directory: C:\Users\mayek\OneDrive\Desktop\python shell scripting


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         9/14/2023   8:38 PM            374 cputime.ps1
-a----         9/18/2023   4:19 PM           1659 Get-DiskUsage.ps1
-a----         9/18/2023   8:03 PM           1662 Get-DiskUsage1.ps1
-a----         9/18/2023   8:37 PM            301 Lab2.ps1
-a----         9/14/2023   1:48 PM            108 Script1.ps1
-a----         9/18/2023   8:06 PM            218 showtoday.ps1


PS C:\Users\mayek\OneDrive\Desktop\python shell scripting> .\Lab2.ps1

InputObject              SideIndicator
-----------              -------------
Ten Apples Up on Top*&^! =>
Ten Apples Up on Top!    <=
```

[Exercise IV: Set Parameter for Calculating Disk Usage]

21. We are going to try PowerShell scripts included in the lab2 folder. Make sure that you download lab2.zip from BrightSpace. Unzip the file. Navigate to where the scripts

are. Open Get-DiskUsage.ps1 in PowerShell ISE to exam it. Please read through the program to understand the program. In this program, you should learn how to obtain input parameter in PowerSehll script.

22. Since the file is downloaded from BrightSpace. You will need to save it first before you can run it or change the Execution Policy to Unrestricted.

23. To test Get-DiskUsage.ps1, we will create a folder (use mkdir)cunder your lab folder and create 2-3 files in the folder. And then, create a sub-folder under this new folder, and then create a file under the sub-folder. Below is an example.

    mkdir testdir1

    cd testdir1

    “Ten Apples Up on Top!!” > apples.txt

    "Hello World!!!" > hello.txt

    mkdir testdir2

    cd testdir2

    "Banana, Peach!!!" > fruit.txt

    cd ..

    cd ..

24. You can then now try the script. Once with no parameter option and the other with the parameter.

    .\Get-DiskUsage.ps1

    .\Get-DiskUsage.ps1 -IncludeSubdirectories

25. Discuss the differences in the results.

In .\Get-DiskUsage.ps1 the script lists the sizes of directories and files in the specified path but does not include subdirectories.

In .\Get-DiskUsage.ps1 -IncludeSubdirectories the script lists the sizes of directories in the specified path and includes subdirectories' sizes as well. The total size includes the sizes of the subdirectories and their contents.

26. We will now mimic the format of this script to write a new program that uses parameter option. Write a short script called **showtoday.ps1** to take one input parameter **ShowWeek**. If ShowWeek is included in the parameter, the program shows only the day of the week for today (i.g. Sunday), otherwise it will show all date information for today.

27. Paste a screenshot of **showtoday.ps1** here.

```
param (
    [switch]$ShowWeek
)

if ($ShowWeek) {
    $DayOfWeek = (Get-Date).ToString("dddd")
    Write-Host "Today is $DayOfWeek."
} else {
    $Today = Get-Date
    Write-Host "Today's Date: $Today"
}
```

28. Paste a screenshot of the results by running **showtoday.ps1** to show your script works in both ways.

```
PS C:\Users\mayek\OneDrive\Desktop\python shell scripting> .\showtoday.ps1
Today's Date: 09/21/2023 18:06:52

PS C:\Users\mayek\OneDrive\Desktop\python shell scripting> .\showtoday.ps1 -ShowWeek
Today is Thursday.
```

[Exercise V: Package Commands in a Module]

29. PowerShell enables you to package a set of common commands in a module and re-use later like a cmdlet. We will learn how to turn a ps1 file into a module.

30. Let us turn the **ShowToday.ps1** into a module. Open the file. Add the follow to wrap the codes of the script.

    function Show-Today

    {

    }

31. By doing so, the script becomes a function that you can call later. Save the file as **Show-Today.psm1**.

32. Now, we will move the file to your Modules directory, defined by PSModulePath environment variable.

    $env:PSModulePath

33. The first one for the users should look like

    C:\Users\cybusr\Documents\WindowsPowerShell\Modules

    or

    \\pace.edu\shares\users\lchen\Documents\WindowsPowerShell\Modules

    Check to see if the directory exist by moving to the directory. Sometimes, the **\WindowsPowerShell\Modules** directory does not exist yet and you will have to create it yourself.

34. Once you have the default directory for modules. You will need to move to that directory and create a sub-directory called "**Show-Today**" under it.

35. Then, you navigate back to your working directory (where Show-Today.psm1 is). You can move the file to that directory.

    mv Show-Today.psm1 \\pace.edu\shares\users\lchen\Documents\WindowsPowerShell\Modules\Show-Today

36. After moving the file to the module directory, you can import it as a module.

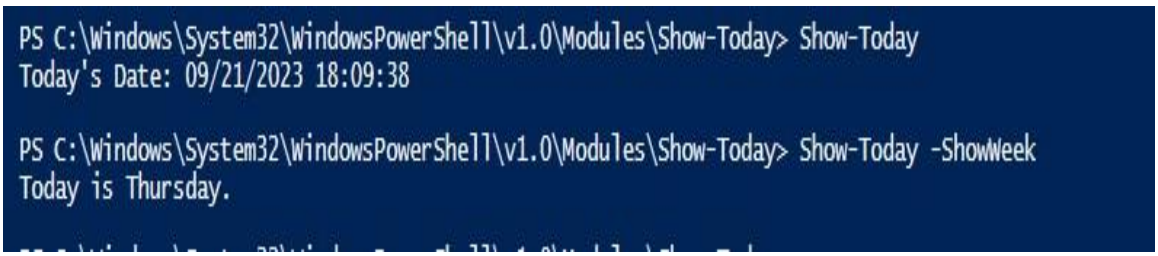
    Import-Module Show-Today

37. Once Show-Today is imported as a module, you can run it like all other PowerShell cmdlet.

    Show-Today

    Show-Today -ShowWeek

38. Paste a screenshot to show that you have successfully run the module above.



[Exercise VI: Event Logs]

39. Determine Even Logs in the system.

Get-EventLog -List

Get-EventLog -list |select logdisplayname, MaximumKilobytes

40. Determine Application and Service logs.

    Get-WinEvent -ListLog *|select logname, recordcount

41. Obtain the most recent 10 entries from the security event log.

    Get-EventLog security -Newest 10 |Format-Table index, source, message –AutoSize

42. List the most recent 100 system events that contain specific text, such as “disk”.

    Get-EventLog system –Newest 100|Where-Object{$_.Message –match “service”}

43. Write down PowerShell commands that can be used to list all of the system events that are generated today.

    Get-EventLog -LogName System -After (Get-Date -f '2023-09-21')

    Get-WinEvent -ListLog System -After (Get-Date -f '2023-09-21')

44. Why is it important for host security to analyze logs?

    Detect malicious activity, such as unauthorized access, suspicious changes to system files, and anomalous network activity.

    Investigate security incidents to determine how they happened, what actions the attacker took, and when the attack occurred.

[Exercise VII: System Service]

45. Get-Service cmdlet retrieve information regarding services running on the system.

    get-service|Where-Object{$_.Status -eq "Running”}

46. Run the command above but sort the results by the number of services depending on them.

get-service|Where-Object{$_.Status -eq "Running"}|Sort-Object -Descending {$_.DependentServices.Count}

47. Paste the screenshot of the first 5 services.

```
Running  RpcEptMapper       RPC Endpoint Mapper
Running  DcomLaunch         DCOM Server Process Launcher
Running  RpcSs              Remote Procedure Call (RPC)
Running  nsi                Network Store Interface Service
Running  BFE                Base Filtering Engine
Running  BrokerInfrastru... Background Tasks Infrastructure Ser...
Running  ProfSvc            User Profile Service
Running  mfemms             McAfee Service Controller
```

48. Why is it important for host security to know what services are running on the system?

    Identifying and disabling unnecessary services: Many systems have services running that are not needed for everyday operation. These services can be potential security vulnerabilities, as they may have known exploits or be used by attackers to gain access to the system.

[Exercise VIII: Develop your system admin script]

49. It is important for a system administrator/security analyst to collect system information on a regular basis to monitor host activities. Write a PowerShell script, sys_admin.ps1, for system administrators. When used, the script should print out a report (an output file) that contains the following information.

    a. The date and time of running the report.

    b. Use an input parameter, like what we did in **Get-DiskUsage.ps1**. If **ShowService** parameter is included, list **services** running on the system sorted by the number of services depending on them (only the first 5 entries).

    c. If **ShowService** parameter is not included, do not list services running.

    d. In both cases (either b or c), list names and count of entries from the **security event log**, grouped and sorted by name (only the first 5 entries)

Paste a screenshot of **sys_admin.ps1**.

```
param(

  [switch] $ShowService
)

Set-StrictMode -Version 3

$date = Get-Date

Start-Transcript -Path "sys_admin_report.txt" -Append

Write-Host "System Information Report"
Write-Host "Generated on: $date"


$services = Get-Service


if($ShowService)
{
  $services | Sort-Object -Property DependentServices.Count -Descending |
Select-Object -First 5 | Write-Host
}

$securityEvents = Get-EventLog -LogName Security | Group-Object -Property
EventId | Select-Object -Property Name, Count | Sort-Object -Property Count -
Descending | Select-Object -First 5 | Write-Host


Stop-Transcript
```

50. Paste a screenshot of running **sys_admin.ps1** and shows the report, generated by the same script.

```
PS C:\Users\mayek\OneDrive\Desktop\python shell scripting> .\sys_admin.ps1
Transcript started, output file is sys_admin_report.txt
System Information Report
Generated on: 09/21/2023 15:36:22
@{Name=5379; Count=18889}
@{Name=4798; Count=7076}
@{Name=4907; Count=2000}
@{Name=4624; Count=1422}
@{Name=4672; Count=1321}
Transcript stopped, output file is C:\Users\mayek\OneDrive\Desktop\python shell scripting\sys_admin_report.txt
```

```
PS C:\Users\mayek\OneDrive\Desktop\python shell scripting> .\sys_admin.ps1 -ShowService
Transcript started, output file is sys_admin_report.txt
System Information Report
Generated on: 09/21/2023 22:27:17
AarSvc_179277e
shpamsvc
ShellHWDetection
SharedRealitySvc
SharedAccess
@{Name=5379; Count=18658}
@{Name=4798; Count=7381}
@{Name=4907; Count=2000}
@{Name=4624; Count=1410}
@{Name=4672; Count=1304}
Transcript stopped, output file is C:\Users\mayek\OneDrive\Desktop\python shell scripting\sys_admin_report.txt
```