**COIT11241**

**Cyber Security Technologies**

**Term 2 – 2022**

**Assessment 2**

Lecturer – Jamie Shield

Student Name – Adam Donovan

Student ID – S0257097

GIT HUB EPORTFOLIO - <https://github.com/DarthFeenik/ePortfolio>

Contents

[Task 1 2](#_Toc114382136)

[Question 1 2](#_Toc114382137)

[Question 1 Output 3](#_Toc114382138)

[Task 2 3](#_Toc114382139)

[Question 2 3](#_Toc114382140)

[Question 2 output 4](#_Toc114382141)

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

# Task 1

The task for Powershell is to write a script that scans a users PC and prints results if there are any or sends an alert if necessary. The purpose of this task is to plan a real-world scenario in case a System Administrator needs to check for services not running, services using too many resources, unapproved applications or services running and more.

Below are examples of each task with an outline of what the expected result it, issues with the coding or errors and successful results if any.

### Question 1

**List any processes that are hogging the CPU. Provide an explanation of your test in your code’s documentation. Submit an example run of your script detecting a CPU hogging process.**

The requirement for this task is to write the code to search for CPU properties in the “CimInstance” Class Win32\_processor, ideally this will be able to be used on multiple computers utilizing different classes of CPU but will need to be running Windows 10 or 11.

The below code takes the processor information while comparing it to the processes and if any are above a certain percentage should list those programs by name and the amount of cpu usage they are using above the specified amount. In this instance the comparison is any process over 1% however the output contain too much data as can be seen in Figure 1 below.

function CPUHog  
{

# Need to calc the total number of cores for the CPU

$CPUNum = (Get-CIMInstance -class Win32\_processor | Measure-Object -Sum NumberOfLogicalProcessors).Sum

#Need to see what processes are using the CPU and calculate if their use is too high

$CPUCount = (Get-Counter "\Process(\*)\% Processor Time").CounterSamples | Where-Object {$\_.CookedValue / $CPUNum -gt 1} | Select InstanceName,CookedValue

#Divides the processes use by the number of cores and lists the top processes that are hogging the cpu

foreach($\_ in $CPUCount)

{

Write-Host $\_.InstanceName, ($\_.CookedValue / $CPUNum) | Out-File -FilePath "CPUHog.txt"

}

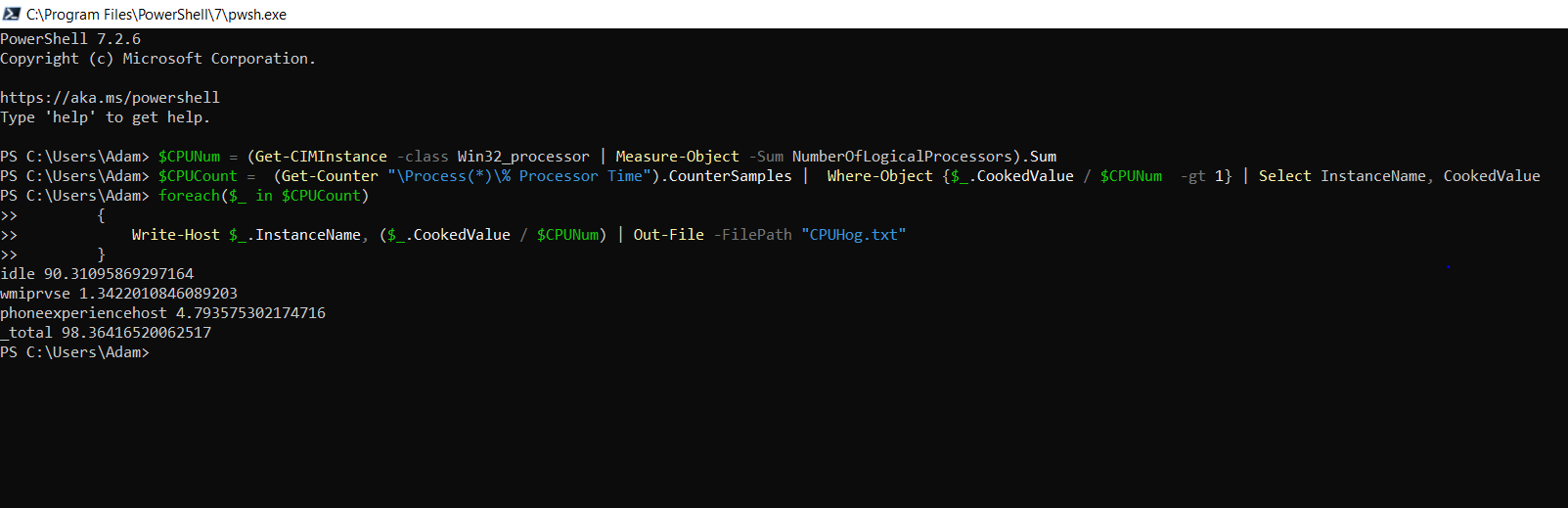
}

CPUHog

### Question 1 Output

The below image is the result of the working code with the exception of extra information not needed that was giving an incorrect result of the “\_total” amount being used as well as the “idle” amount of CPU being used.

Figure



After many attempts to correct the output and multiple rewrites trying to limit the amount of data being output the correct code can finally be seen below;

Figure

A computer screen capture

Description automatically generated with medium confidence

# Task 2

## Question 2

**Check that Windows Defender antimalware toolkit is enabled.**

## Question 2 output

*function CheckAMService{*

*#Check if AMServiceEnabled is enabled if not displays error*

*if((Get-MpComputerStatus).AMServiceEnabled -ne 1)*

*{*

*Write-Host "AMServiceEnabled Not Enabled"*

*}*

*}*

*CheckAMService*

As the code is written to detect is the Antimalware service is NOT active the code will be run as it is written, then with the code re-written to test if the AMService is enabled but with the same output. The testing of the code output for the correct code will not have any output as the AMService should be kept running to keep the device safe from any attacks.

The below image will show both outputs,the first with the correct code will be checking if AMSerice “is NOT equal to 1” and has no output. The second code has been changed to check if AMService “IS equal to 1” and shows the output of “AMService Not Enabled”

