Week 4 Ass 2 Task 1 Peer Programming

Task Overview

You will work in a group to implement and test a defensive cybersecurity technology.

Code Reuse

You are encouraged to search for and reuse good quality code in this unit. You are encouraged to use tools such as ChatGPT in this unit. All sources of code should be referenced. For websites and web services, the inclusion of a commented URL in the reused code is a sufficient reference. Deprecated code, for example, PowerShell scripts that use ping, should not be reused.

Sharing of artefacts, for example, code or virtual machines, between groups is not permitted. You can share artefacts with other members of your group.

Repository

Create a private Git repository, for example, on GitHub. Invite your tutor, the unit coordinator and your group members to the private repository.

Due

The Peer Programming task is due in Week 4. You should submit a draft to your Git repository by the end of tutorial, but you have the rest of the week to finalise your submission.

Return

The Peer Programming task will be marked with the Defend task (due in Week 6). Feedback will be provided within 2 weeks of the due date. You should ask your tutor for informal feedback during or in a subsequent tutorial.

Submission Overview

Submit artefacts to both your private code (Git) repository and to the unit website. Submit a link to your private repository to the unit website. All group members must submit.

Criteria Overview

You will be marked on aspects such as the quality of your scripts including your testing scripts including functionality, modularity, style, code reuse, documentation, lack of deprecated features, level of automation and use of code repository tools.

You will also be marked on your teamwork including your individual contributions; your communication quality and quantity; your interpersonal style; and the performance of your team including its decision making and self-management. Example behaviours that we will be looking for include:

- How well do team members learn from and respond to each other?
- Does one person dominate the group and the contributions?
- Are decisions made by one person sufficiently challenged by other members?
- Does the team manage dysfunctional behaviours such as free riders?

- Time management - we want you to have fun in your group but … does the team manage excessive distractions?

Scenario

You work at the Monto Caravan and Cabin Park. Your boss has asked you to improve the cybersecurity of her workstation. You will develop a portfolio of evidence of the implementation and testing of a safeguard. Your evidence will include PowerShell scripts and a PowerPoint slideshow containing, for example, screenshots of your testing.

Topic 1 Implement CIS 12.8 Restrict Admin Internet

Your boss uses the DC virtual machine. Download the DC virtual machine from the unit website and import it into Virtualbox. Setup a shared folder on DC for your scripts.

CISv8 Safeguard *12.8 Network Infrastucture Management > Use dedicated resources for Admin* recommends that Internet access should be removed from admin accounts. You decide to setup your PowerShell admin accounts to use a dummy proxy to limit Internet access. The following instructions are a guide. You can make improvements as you see fit.

1. Create a PowerShell script called *defend*.ps1.
2. Pre-test: write a function *testInternetAccess():*
    a. Use *Invoke-WebRequest* to check for Internet access by downloading a webpage. An HTTP status code of 200 means that a webpage was successfully obtained. You might need to use the option *-UseBasicParsing*.
    b. The function should interpret the HTTP status and output to the console whether the command was successful or not.
3. Document your code.
    a. Clean up your code using PSScriptAnalzyer
    ```
    PS> Invoke-ScriptAnalyzer defend.ps1
    ```
    b. Run the code and collect testing screenshots for your portfolio.
    c. Commit your code to your Git repository.
4. Create a function *enableRestrictInternet()* that removes Internet access by setting the proxy to a fake, non-existent server “proxy”:
    ```
    [System.Net.HttpWebRequest]::DefaultWebProxy = `
      New-Object System.Net.WebProxy("http://proxy",$true)
    ```
5. Add code to your script that checks the first argument of the command line:
    a. If the first argument is *testInternetAccess*, call the *testInternetAccess*() function. For example, the following command line would call *testInternetAccess*():
    defend.ps1 *testInternetAccess*
    b. Improve your code to call the other functions in your script based on the command line argument.
6. Post-test: After running this function, rerun *testInternetAccess*(). Internet access should now be disabled due to the fake web proxy. Run the code and collect screenshots for your portfolio.
7. Add a *resetRestrictInternet*() function to your script that re-enables Internet access by removing the fake proxy:
    ```
    [System.Net.HttpWebRequest]::DefaultWebProxy = `
      New-Object System.Net.WebProxy($null)
    ```

Collect Evidence

Prepare a PowerPoint presentation that collects evidence of the implementation of your safeguard. Include evidence such as screenshots of:

1. Testing: including pre-tests and post-tests for each safeguard
2. Style: include output of PSScriptAnalzyer for each script.

Task Submission

Include a link to your private repository in your PowerPoint slideshow. Commit your PowerPoint slideshow to your private Git repository. Submit your PowerPoint slideshow to the unit website. All group members must submit.

Task Criteria

Each of the following marking criteria have equal weighting.

Criteria	Indicative of 100%	75%	50%	25%	0%
Automation and Testing	Excellent level of automated installation and configuration ← Pre- and Post-tests demonstrate effect of safeguards ←			→ Insufficient evidence of automation → Insufficient testing or poorly chosen test cases	→ No automation or no testing
Code functionality, style, documentation & repository management	Consistent, reasonable layout← Excellent functions documentation ← Git commits showing regular script & function development by all group members ←		→Lack of modularity, e.g. poor or no functions →Uses deprecated functionality	→ Insufficient Git commits → Not all group members committing to Git → Scripts not available via Git repository	
Peer Programming Teamwork	Excellent individual contributions, communication & team management← Challenges groupthink ←		→Free riders are not confronted →Decisions not sufficiently challenged →One person dominates		→No evidence of teamwork