CI5235: COURSEWORK 2 – PYTHON CYBER SECURITY USE CASE

Windows Event Log Analytics

Learning Outcomes:	• Build and modify Python scripts that can be used in a cyber security context. • Demonstrate the practical use of python for surveillance and information gathering • Demonstrate the practical use of python for directory navigation, file search, file copying, file opening, file reading, file saving and time stamping • Demonstrate the practical use of python for data structuring, analysis and visualisation
Percentage of Module:	40%
Maximum Marks:	100
Set Date:	6th January 2020
Deadline:	11th March 2020
Feedback and Marks:	8th April 2020

Table of Contents

SCENARIO

You are a member of a pen testing team, where you are an active contributor to reconnaissance and information gathering activities. As a python "enthusiast" with a developing interest in the application of data science to cyber security, you have been assigned the task of scrutinising a raw dataset of ***evtx*** (Windows Event Log) files.

For this coursework, you will ***design***, ***write*** and ***implement*** python scripts that will be utilised to:

1) Acquire data
2) Explore and clean the data
3) Analyse and structure the data by turning it into useful information
4) Build a simple visualisation of analysed data

PREPARATION

Before you begin any coding, it is important to plan your coursework activities and to manage the delivery of milestones in the context of the submission deadline. Therefore, you are strongly advised to:

- READ THIS COURSEWORK DOCUMENT CAREFULLY!
- Fully understand the aims and objectives of this coursework. If you are unclear on any aspect of this assignment, please speak to your course instructors as soon as possible.
- Understand the marking guidelines (see pages 10 and 11 of this document).
- Understand requisite technologies by actively participating in lab sessions and researching topics and terminology to support your learning.
- Establish and manage an appropriate coding environment to support your delivery of this coursework
- Use appropriate tools to help you manage this coursework. Examples of such tools include meistertask.com and dynalist.io.
- As this is a live and dynamic project, aspects of the coursework may be subject to updates and change. THEREFORE, REFER TO CANVAS FOR UPDATES TO THIS DOCUMENT REGULARLY.

COURSEWORK RESOURCES

For this coursework, you will require the following resources:

RESOURCE	INFORMATION AND INSTRUCTIONS
Mint19_Anaconda_CW This VM contains: • The Anaconda Python development environment • Visual Studio Code, Juypter and Spyder IDEs • Appropriate python modules and addons	This VM is available from the Cyber Lab VM Repository (i.e. ***create_vm.sh***) **User Name:** user **Password:** abc123 **Network:** Wired Connection 1 (NAT)
CI5235_KUNumber_FirstName folder • This is the coursework folder that will be used for the creation of python scripts and for the submission of coursework.	The **CI5235_KUNumber_FirstName** folder must be created in the "user" directory on the **Mint19_Anaconda_CW** VM. See example below: /home/user/**CI5235_K1234567_Jack**
evtx_logs.zip • This zip file contains a directory structure of **evtx** files to be analysed. • Each evtx file contains a minimum of 1 event log incident. Most contain many more.	This file is available from Canvas. The ***evtx_logs.zip*** folder should be downloaded directly to the **Mint19_Anaconda_CW** VM and extracted in the **CI5235_KUNumber_FirstName** folder. The path should be as follows: /home/user/CI5235_KUNumber_FirstName/**evtx_logs**
evtx_dump.py • This python script is used to convert **evtx** files to **xml** files.	This file is available from Canvas. The ***evtx_dump.py*** file should be downloaded directly to the **Mint19_Anaconda_CW** VM and saved to the **CI5235_Coursework** folder. The path should be as follows: /home/user/CI5235_KUNumber_FirstName/**evt_dump.py**

COURSEWORK TASKS

This section defines three python scripts that you will create as part of this coursework. Each script has a maximum of 25 marks. Please attempt to complete all three scripts.

A further 25 marks are awarded for "wow" factor. This is reserved for exemplary demonstrations of creativity, engagement, independence, additional features or functionality.

TASK 1	convert.py	Max Marks
	Create a python script that achieves the following: 1) Conditionally test to see if a folder called "**CI5235_Logs**" exists in the /home/user/ CI5235_KUNumber_FirstName/ directory. If it does not exist, create it. 2) Read all folders and **evtx** files in a directory called **evtx_logs** 3) Conditionally test to see if any **xml files** exist in the **evtx_logs** folder. If any xml files do exist, prompt the user to delete them before conversion starts. 4) If no **xml files** exist, utilise **evtx_dump.py** to convert all **evtx files** in the **evtx_logs** folder from **evtx** to **xml**. REQUIREMENTS • The script must get a **count** of all folders within the **evtx_logs** folder (including sub directories) • The script must get a **count** of all **evtx files** found within the **evtx_logs** folder • The script must **display feedback** of progress to the user while it is processing • The script must include a **timestamp** • The script must include a **time duration**. • A log file of the script conversion activity should be created and saved in a folder called /home/user/ CI5235_KUNumber_FirstName/CI5235_logs • The name of the log file should be "**convert_log_**" concatenated with a time stamp. • The script must display a summary of count totals, duration and log file information once running of the script has concluded. IMPORTANT: See Canvas for script output and log file examples.	25

TASK 2	analyse.py	Max Marks
	Create a python script that achieves the following:	25

1) Search for **xml files** in the **evtx_logs** folder.

2) For each xml file found, read each file line by line and search for a means to isolate an "Event ID".

3) Sanitise the Event ID line so that only the Event ID remains (i.e. an integer)

4) Compare the sanitised Event ID found in each file, to values in the Event ID table below:

1102	4611	4624	4634	4648	4661	4662	4663	4672	4673
4688	4698	4699	4702	4703	4719	4732	4738	4742	4776
4798	4799	4985	5136	5140	5142	5156	5158		

Review the link below to find out what each Event ID represents:
https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/

5) Where a match occurs, display a line that reads
MATCHED Event ID: "X"
(where "X" is the Event ID that has been matched)

6) Where a match does not occur, display a line that reads
NO MATCH For Event ID: "Y"
(where "Y" is the Event ID that has not been matched).

REQUIREMENTS

- The script must get a **count** of all **xml files** found within the **evtx_logs** folder
- The script must get a **count** of all **Event IDs** found within the **xml files**
- The script must get a **count** of all **Matched Event IDs** found within the **xml files**
- The script must display feedback of progress to the user while it is processing.
- A log file of the scripts analysis activity should be created and saved in a folder called /home/user/ CI5235_KUNumber_FirstName/CI5235_logs/
- The name of the log file should be "**analyse_log_**" concatenated with a time stamp.
- The script must display a summary of count totals, timestamp and log file information once running of the script has concluded.

IMPORTANT: See Canvas for script output and log file examples.

TASK 3	visualise.py	Max Marks

Preparation:

25

Create a python script that includes a nested python dictionary containing entries for each of the Event IDs listed in the table below:

1102	4611	4624	4634	4648	4661	4662	4663	4672	4673
4688	4698	4699	4702	4703	4719	4732	4738	4742	4776
4798	4799	4985	5136	5140	5142	5156	5158		

Each Event ID, should be paired with an additional dictionary value that has a Key entry of "Count" and a Value entry of 0 (where 0 = is zero).

Create Python script that includes the python dictionary described above, and that achieves the following:

1) Open a specified "**analyse_log_**" file (created as an output of analyse.py).

2) Read each line of a specified "**analyse_log_**" file and identify the occurrence of the string "MATCHED Event ID:" Once identified, sanitise the line so that only the Event ID remains (i.e. an integer)

3) Compare the **sanitised Event ID**, to your nested **dictionary of Event IDs** and iterate the appropriate dictionary count value, by 1.

4) Once the process of **assigning count values** to the **Event ID dictionary** is complete, write each dictionary Event ID value and its associated count, to a **log file**.

5) Once the **log file** has been saved, **open** it and **read** each line.

6) For each occurrence of an **Event ID**, **append** it to an Event ID **list variable**.

7) For each occurrence of an **Event IDs count value**, **append** the count to an appropriately named **list variable** (e.g. EventIDCount)

8) Use Matplotlib to create and display a horizontal bar graph, based on the list variables created in the last two steps.

REQUIREMENTS

- The script must get a **count** of all Events IDs and their associated counts
- The script must display feedback of progress to the user while it is processing.
- A log file of the scripts analysis activity should be created and saved in a folder called /home/user/ CI5235_KUNumber_FirstName/CI5235_logs/
- The name of the log file should be "**visdata_log_**" concatenated with a time stamp.
- The script must display a visualisation of the list data that has been processed.

IMPORTANT: See Canvas for script output and log file examples.

TASK 4	Wow Factor	Max Marks

	Additional marks awarded for exemplary demonstrations of creativity, engagement, independence, additional features or functionality.	25

WHAT YOU WILL NEED TO SUBMIT

For this coursework, you will need to submit your coursework directory (CI5235_KUNumber_FirstName), containing the 3 python files that you have designed and created. Collectively they represent stages of data acquisition and conversion; data restructuring, data exploration; information analysis and visualisation. They should be named as follows:

1) convert.py (data acquisition and conversion)
2) analyse.py (data restructuring AND data exploration)
3) visualise.py (information analysis and visualisation)

All three files should be created and run from the /home/user/ CI5235_KUNumber_FirstName/ folder as shown below:

1) */home/user/ CI5235_KUNumber_FirstName/**convert.py***
2) */home/user/ CI5235_KUNumber_FirstName/**analyse.py***
3) */home/user/ CI5235_KUNumber_FirstName/**visualise.py***

Once you have completed the coursework tasks, you will need to zip your work folder (CI5235_KUNumber_FirstName) using the following name structure for the name of your zip file:

YourKUnumber_YourName_CI5235_Coursework2.zip.

Once you have successfully zipped your folder, you will need to upload it to the coursework submission page on Canvas.

MARKING GUIDELINES

TASK 1: convert.py (25 Marks)	
No script submitted or an incorrect file format was submitted	0
An attempt to create the script has been made. The script does not run due to coding errors (e.g. tab indentation errors, logical errors etc.).	1 - 9
The script runs partially and returns a very limited number of expected outcomes in accordance with the task description and requirements.	10 - 12
The script runs and returns a limited number of expected outcomes in accordance with the task description.	14 - 13
The script runs and returns some of the expected outcomes in accordance with the task description and requirements	15 - 16
The script runs and returns most of the expected outcomes in accordance with the task description and requirements.	17 - 19
The script runs fully in accordance with the task description and requirements References have been included in the code Clear and concise explanatory comments have been included in the code	20 - 25

TASK 2: analyse.py (25 Marks)	
No script submitted or an incorrect file format was submitted	0
An attempt to create the script has been made. The script does not run due to coding errors (e.g. tab indentation errors, logical errors etc.).	1 - 9
The script runs partially and returns a very limited number of expected outcomes in accordance with the task description and requirements.	10 - 12
The script runs and returns a limited number of expected outcomes in accordance with the task description.	14 - 13
The script runs and returns some of the expected outcomes in accordance with the task description and requirements	15 - 16
The script runs and returns most of the expected outcomes in accordance with the task description and requirements.	17 - 19
The script runs fully in accordance with the task description and requirements References have been included in the code Clear and concise explanatory comments have been included in the code	20 - 25

TASK 3: visualise.py (25 Marks)	
No script submitted or an incorrect file format was submitted	0
An attempt to create the script has been made. The script does not run due to coding errors (e.g. tab indentation errors, logical errors etc.).	1 - 9
The script runs partially and returns a very limited number of expected outcomes in accordance with the task description and requirements.	10 - 12
The script runs and returns a limited number of expected outcomes in accordance with the task description.	14 - 13
The script runs and returns some of the expected outcomes in accordance with the task description and requirements	15 - 16
The script runs and returns most of the expected outcomes in accordance with the task description and requirements.	17 - 19
The script runs fully in accordance with the task description and requirements References have been included in the code Clear and concise explanatory comments have been included in the code	20 - 25

Wow Factor: (25 Marks)	
Additional marks awarded for exemplary demonstrations of creativity, engagement, independence, additional features or functionality.	0 - 25