Assessment #4: Software Based Solution

Name Joe Velardi

Student ID 10647326

Unit Details CYB6004 – Scripting Languages

Facilitator Rajesh Soni

**Description of Software Based Solution**

The software solution is based on breached websites, when they occurred, including the number of accounts and categories impacted. These are also aptly known as PWNED sites.

**Using the Software Based Solution**

The solution first prompts you to enter a password (defaults to foo-bar), whereby the system will determine if data has been downloaded. If not, the system will go ahead and download and cleanse the data, in readiness for review. Once processed, navigation is through a menu-based system. The menu will display the results of the cleansed data, displaying both full sets of data and also a ‘Top 20’ of results data. Results are in tabular format with a summary displaying the total records, totals per line, and a totals and average summary. Included is the ability to search for a breached record to display the results in detailed format. Addition to the statistical information is an administrative section that will allow you to refresh the downloaded, and the ability to change the default login password. This new password can be used the next time you enter the system. There is also an option to reset the environment, which will reset to the default password and will also perform a data download again.

**Challenges**

Defining the scope of work and what to include changed across the journey. I wanted to create a more complex login workflow, however time constraints ensured I focused on completion of the core components. Additionally, I attempted to create some Python graphs using matplotlib, but I was consuming too much time testing and learning the core concepts, thus dropped having graphs included. Technically there were challenges with cleansing the data, specifically using sed and regex. I had to “wash” the data a few times to ensure I had the data in the condition which allowed it to be read into awk easily. Where possible I created some functions to cater to common calls.

**Testing**

The most common control structure used was the case statement, which allowed for strict inputs and outcomes. This allowed for the core structure to be managed with a sense of control. Logical controls were tested manually, for both positive and negative outcomes, and in many cases the logical controls consumed much of the time, as each variant action had possible outcomes. Also tested were the outputs of the source data. I used the website as a source of truth, and once extracted, the data was imported into excel to facilitate sorting, filtering, summing and averaging data. This was useful as I noticed I was hitting an upper threshold using decimal, where I should have been using float to display very large numbers.

**Limitations & Dependencies**

In formatting numbers in awk, I had the option of including thousand separators, or excluding them, when cleansing the data. I opted to cleanse the data without having thousand separators so I could sum the data in awk. In wanting to add the thousand separators for display, I chose to use gawk, as I was seeing inconsistent results when using my local ubuntu and the version of ubuntu used for the labs.

**References**

Bash Cheat Sheet – <https://devhints.io/bash>

Bash Reference Manual

<https://www.gnu.org/savannah-checkouts/gnu/bash/manual/bash.html>

Create ASCI Banners – <https://manytools.org/hacker-tools/ascii-banner/>

GitHub: Awesome Hacking – <https://github.com/Hack-with-Github/Awesome-Hacking>

GitHub: Brute Force py – <https://github.com/notaSWE/brutepw/blob/main/bruteforce.py>

grep Cheat Sheet – <https://devhints.io/grep>

grep Pocket Reference

https://learning.oreilly.com/library/view/grep-pocket-reference/9780596157005/

itertools: functions for efficient looping – <https://docs.python.org/3/library/itertools.html>

sed Cheat Sheets – <https://linuxdigest.com/howto/sed-cheat-sheet/>

sed Cheat Sheets – <https://quickref.me/sed.html>

sed Cheat Sheets – <https://lzone.de/cheat-sheet/sed>

Shell Scripting Password Generator – <https://www.youtube.com/watch?v=P5552IJp7tU>

Stack Overflow: Bash regex for Strong Password

<https://stackoverflow.com/questions/4213453/bash-regex-for-strong-password>

regex101 – https://regex101.com/

regexp Cheat Sheet – <https://devhints.io/regexp>