Virginia Tech ■ ECE 5480 ■ Fall 2018

Project 3

Before starting this first homework assignment, please be sure that you have completed all of the following activities.

☐ ·Review the Graduate Honor System at https://graduateschool.vt.edu/academics/expectations/graduate-honor-system.html. Review the Graduate Honor System Constitution, especially Articles I (Sections 1, 2, and 3), V, VI, VII, VIII, and IX.

Please note the following.

- Solutions must be clear and presented in the order assigned. Solutions must show work needed, as appropriate, to derive your answers. Written answers should be concise, but sufficiently complete to answer the question. Neat hand drawings, where needed, are acceptable. Your final solution for each problem must be easily identified.

- At the top of the first page, include: your name (as recorded by the university); your email address; and the assignment name (e.g., "ECE 5480, Project 3"). Do not include your Virginia Tech ID number or your social security number.

- Homework must be submitted as a PDF (.pdf) file with the file name *lastname_firstname*_PROJ3.pdf, where *lastname* is your last or family name and *firstname* is your first or given name, along with requested files.

- Submit your assignment using the Assignments area of the class website.

INTRODUCTION

One benefit of automation is that it helps to reduce manual error introduced when trying to repeat many complicated steps in exactly the same way for multiple iterations. Scripting languages — eg Python — are useful here. By automating tasks, Python can help you be more efficient and accurate, especially for routine and error-prone tasks. Also, automation scripts can be helpful if you are coming back to a job after a long time has passed, or if someone new is trying to repeat your steps.

In the security arena, there are often many jobs that could benefit from automation, such as making sure patches for IoS devices are downloaded and installed correctly and periodically, vulnerabilities in web applications or network devices are thoroughly scanned and monitored, and log files are explored and analyzed. (We also got a taste of this in a previous project where we automated testing the network availability of multiple IP addresses by calling `ping`. For legal reasons (https://nmap.org/book/legal-issues.html) we will not be automating `nmap` scans, but you can imagine how this could be done with Python.)

In this project we will perform two tasks to explore these automation ideas.

TASK 1

In a security context, we often have a need to be able to extract useful information from a given system. For example, we might want to test a system (desktop computer, IOT device, web server) for vulnerabilities, or we might want to identify files and directories of certain types. So the purpose of this exercise is to give you exposure to Python systems programming and experience programmatically exploring local file systems.

In particular, this exercise will explore the use of the Python os module (https://docs.python.org/3.4/library/os.html), which has a number of useful functions that, for example, allow us to:

* check the name of the current working directory
* change the current working directory
* list the files in a directory
* create new directories
* create/delete existing files or directories
* walk through a directory and perform operations on every file in that directory based on user-defined criteria

You can imagine that these abilities would be useful for cybersecurity defensive as well as offensive operations. For example, you could traverse directories and files to collect and analyze device or process log files, automate web or device penetration testing activities, or collect web/process/network data and write them to files in specific directories for later analysis.

PROGRAM SPECIFICATION:

Create a python program to accept input of a base directory path, then create a list of all files and directories in that base directory, then as output create two separate text files, one file containing the list of files found, and one file containing the list of directories found. Note that the found files and directories should be specified with their full paths, and the initial base directory should be specified as a full path.

An example base directory on UNIX systems might be:

```
/Users/Kendall/Projects/python
```

while an example base directory in Windows might be:

```
c:\\Projects\\python
```

NOTE: You do not need to fully "traverse" the initial base directory, though that would be an obvious extension. In other words, if `/Users/Kendall/Projects/python` also contains the directories `/Users/Kendall/Projects/python/ex1`, `/Users/Kendall/Projects/python/ex2`, `and /Users/Kendall/Projects/python/ex3`, you only need to find the files and directories in `/Users/Kendall/Projects/python`, not also in `/Users/Kendall/Projects/python/ex1`, `/Users/Kendall/Projects/python/ex2`, or `/Users/Kendall/Projects/python/ex3`.

As a starting point, you are given the outline of a .py file.

Submit in your assignment .pdf the output when running your program on an example input base directory, and submit with your .pdf your .py code (include the actual code file, not copy/paste in the pdf), and the resulting files and directories text files from your example run.

TASK 2

Excel is often used for data analysis, but tools like spreadsheets can hide the sequences and transformations performed on the data, leading to unreproducible results — at a later time or when analyzed by someone else, you (or they) are not able to figure out what was done to the data to reach a certain result. Another use of Python, especially in security, is to help automate the exploration and processing of log files, such as from web servers, processes, DNS servers, etc. These files are often large and messy, so programming scripts can help, and the script serves as a document of the exact transformations and operations performed on the data, so a well-documented Python script can help ensure reproducibility (this is inspired by "literate programming," introduced by Donald Knuth, that unfortunately we will not be able to explore further in this course).

To explore some of these concepts, you have been provided a log file from a DNS server (we will explore attacking DNS systems in Project 7) and we want to use Python to help us understand and explore the contents of this file.

A typical record looks like the following:

```
07-Nov-2011 00:14:19.671 queries: info: client
7.204.241.161#49698: query: smtp.usna.bluenet IN A +
```

However, note that not all records follow this format, eg:

```
07-Nov-2011 00:25:25.273 xfer-in: error: transfer of 'hq.bluenet/IN'
from 10.1.10.5#53: failed to connect: connection refused
```

Here, the first two fields are standard, but after the second field the record deviates from the typical record.

There are easy methods for exploring datasets in standard formats (we will look at some of these in a later project). However, in this project we will look at exploring large datasets in non-standard formats.

Your task is to write a Python script to read the log file and print out answers to the following questions for the subset of valid records:

1. Earliest record time and date

2. Last record time and date

3. Total number of records

4. Number of unique client IP addresses (ignore port #)

5. Number of unique query domains (ignore query IP addresses)

6. Most common client address and number of occurrences

7. Most common query domain and number of occurrences

8. Unusual query domains

There are a number of ways this can be implemented in Python, but a rather basic and straightforward implementation requires only the use of Python lists, dictionaries, for loops and if/then conditionals.

Note: do not modify the original data file, but ignore non-typical records as illustrated above to create a list of “valid” records which you will then process. Your script should read in the original data file, perform necessary operations to parse and isolate desired answers to the listed questions, and print the questions and answers to the screen. For this task please submit your .py program output (copy/paste or screenshot), answers to the above questions, and your .py file.

Finally, I’ve added examples for Python lists, dictionaries, and strings to the Python tutorial page: https://www.kendallgiles.com/2017/01/quick-python-tutorial/, though of course you can consult other Python resources as well.