

Python

Offensive and Defensive Tool Construction

Table of Contents

[Objectives 2](#_Toc63628744)

[Background Reading 2](#_Toc63628745)

[Important Information 2](#_Toc63628746)

[Problem 1 (30 pts) 3](#_Toc63628747)

[Questions: 4](#_Toc63628748)

[Problem 2 (40 pts) 5](#_Toc63628749)

[To be completed 5](#_Toc63628750)

# EVALUATION:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Problem 1 | 30 |  |
| 2 | Problem 2 | 40 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | TOTAL MARK | 70 |  |

Offensive and Defensive Tool Construction

Python Programming III

Objectives

This lab focuses on the following objectives:

* Analyze the Linux filesystem using Python.
* Explore the use of python in building basic tools to gather information about the filesystem.
* Use variables, expressions and statements, lists, dictionary, functions in Python.
* Use built-in and external modules to assist in the development of Python Tools.

Background Reading

* Read chapters 6–10 in *How to Think Like a Computer Scientist: Learning with Python*, available at [www.greenteapress.com/thinkpython/thinkCSpy.pdf](http://www.greenteapress.com/thinkpython/thinkCSpy.pdf).
* <https://docs.python.org/3.8/>

# Important Information

**YOU MUST PRESENT IMAGES OF YOUR CODE BEING EXECUTED. DO NOT SUBMIT YOUR ANSWERS IN THE DOCUMENT. CREATE A BLANK DOCUMENT AND SUBMIT YOUR ANSWERS THERE.**

**YOU WILL LOSE MARKS FOR NOT FOLLOWING THE ABOVE REQUIREMENTS.**

All scripts must have the following elements:

1. File and Header comments, which follows the following format:

***# Filename: m##XXX.py***

***# Author: Thunder Cat***

***# Course: ITSC203***

***# Details: This exercise checks to see if students read the suggested items or***

***# prior to class or doing the labs.***

***# Resources: https://www.cs.siue.edu/programming-style-guide***

1. Comments on lines where you used some unique computation that might be tricky to comprehend a month later.

***list1 = [x for x in range(20) if x % 4 == 1] # Using list comprehension to ….***

# Problem 1 (30 pts)

During an Incident Response event you come across a folder that you suspect could have malware files.

Perform the following actions: (**CREATE A SNAPSHOT prior to beginning this lab!!!!**)

1. Make a folder in your home directory called Lab3 and change directory to the new folder.
2. From D2L download the program **createfiles**, to your VM and save it to the Lab3 directory
3. Run the program in the ~/Lab3 folder: **./createfiles**
4. Confirm that a new folder was created and that the folder contains files.

Writing a python program, you will do the following:

1. You need to enumerate the entire folder in such a way that it is easy to see the folder structure. The table below is just an example, you can present the data as a table or in another layout format that presents the data is an easy to read format.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parent directory** | **Sub level 1 with files** | **Sub level 2 with files** | **Sub level 3 with files** | **Sub level 4 with files** |
| myfolder | Sub1-1 | Sub2-1 | Sub3-1 | File12  File22 |
| File41 |  |
| File12  File22 |  |  |
| Sub1-2 | File13  File24 |  |  |
| File01  File02 |  |  |  |

You can try to use the **PrettyPrint** module or another module that provides a nice table structure. Or you can build your own custom format.

1. Your program will also use the **hashlib** module to generate a hash of all the files in the folder and subfolders.

|  |  |
| --- | --- |
| **File** | **SHA512** |
| /myfolder/sub1-1/sub2-1/sub3-1/file12 | 1111111111abcd111… |
| /myfolder/sub1-1/sub2-1/sub3-1/file22 | 1111111111abcd111… |

1. Create a new folder for each type of file extension found
   1. This new parent folder **MUST NOT MODIFY** the suspicious folder containing the potential malware artifacts.
   2. Copy the files to the folder based on the file extension and rename them so that it uses the hash instead of the filename. For example
      1. Myfile.txt will be copied to the TXT folder as 1234123598abdc.txt
      2. Myfile.zip will be copied to the ZIP folder as 94523049adcde3.zip

**DO NOT SAVE THE COPY IN THE SAME FOLDER AS THE ORIGINAL FILES,**

**The original folder and files must not be modified in any way while performing your work**.

**THE ORIGINAL FILES/FOLDER SHOULD NOT BE MODIFIED OR MOVED BUT YOU CAN CREATE COPIES WITH YOUR PROGRAM.**

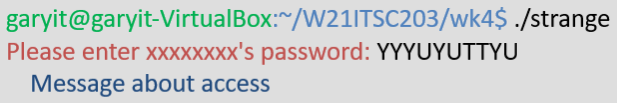
## Questions:

1. Why is the hash calculated for a file? **1pt**
2. Name 2 other existing hashing algorithm, other than **md5** or **sha256**? **2pts**
   1. Is there a reason why you would choose **sha512** over **sha256** or **md5**
3. Is the file extension a good way of tracking the true nature (**type**) of the file? **1pt**
4. Are there other methods that can be used other than file extension or hash to determine if a file is unique or is correctly identified? **1pt**
5. Why should you not use the original suspected folder to perform any analysis in? **1pt**

# Problem 2 (40 pts)

For this exercise you will need to download the file **strange.tx** from D2L and do the following:

1. mv strange.tx strange
2. chmod u+x strange
3. ./strange

During a digital forensics investigation of a computer you found an executable file (**strange**). You execute it a few times in a controlled environment, to see the following output:

Your analysis provides the following information:

1. **strange** - is the name of the program
2. **YYUYUTTYU** – Is the password guess made by the user
3. **Message about access** – is the message that tells you if login was successful or a failure.

After doing some static analysis you find out a few things:

1. There are **multiple users** that the program randomly asks you to login as
2. Each user has a **unique** password
3. There is a total of **10 messages** that will be presented by the program**, 5 – for successful login and 5 for failed login attempts**
4. The program uses the following algorithm to determine if the password is correct:
   1. (userinput[offset] ^ 0x22) \* (length – offset)
      1. *Userinput is an array*
   2. Each character in the user input is xor’d with 0x22 and multiplied by the length of the string minus the offset

## To be completed

Write a python script using the following criteria:

1. You **MUST** use the **pexpect** module
2. Your program will determine how many total users are available for login to the program
3. Your program will determine and print the (**Message About Access**) strings presented by the program.
4. Determine programmatically if there are other passwords that will gain access
   1. This is challenging because you have to keep track of which user you attempted the password on. One correct guess, for the username/password combination, is sufficient

HINT: **child = pexpect.spawn(‘/bin/bash’) child.sendline(‘ls’) child.expect(‘\r’) child.readline().decode(‘utf-8’)**