

Python

Offensive and Defensive Tool Construction

Table of Contents

[EVALUATION: 1](#_Toc72257793)

[Objectives 2](#_Toc72257794)

[Background Reading 2](#_Toc72257795)

[Important Information 2](#_Toc72257796)

[Problem 1 (14 pts) 3](#_Toc72257797)

[Question 1: 4](#_Toc72257798)

[Problem 2 (20 pts) 5](#_Toc72257799)

[Problem 3 (20 pts) 5](#_Toc72257800)

[Questions: 6](#_Toc72257801)

[Problem 4 (20 pts) 6](#_Toc72257802)

[Problem 5 (20 pts) 6](#_Toc72257803)

# EVALUATION:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Problem 1 | 14 |  |
| 2 | Problem 2 | 10 |  |
| 3 | Problem 3 | 16 |  |
| 4 | Problem 4 | 12 |  |
|  | TOTAL MARK | 52 |  |

Offensive and Defensive Tool Construction

Python Programming I

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 in Python.
* Use built-in 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: Craig Mac***

***# 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 (14 pts)

During exploitation you will often find yourself needing to generate a non-repeating pattern to be used to determine where the return address is located on the stack. In a 32 or 64bits system the address you are trying to find is 4 or 8 bytes in length. If each character requires 1 byte to represent it, you will need 4 or 8 characters. **You can use the random module**.

Write a Python program named **m2p1.py** (module 2, problem 1).

Your program will do the following tasks (**10pts**):

1. Generate a non-repeating sequence of either 4 or 8 bytes. You can use alphanumeric characters **(a-z, A-Z, 0-9)**
2. Your program should take as input the length of the sequence to generate.
   1. The sequence can be a minimum of 100 and a maximum of 1024.
3. The program will also accept as input the number of bytes in your address. In this case 4 or 8.
   1. 4 or 8 represents the number of bytes in the address, 32 or 64bits respectively.
   2. If the user doesn’t enter 4 or 8 for the sequence length; your program will print a usage message to let them know how to use your program.
   3. The 4 or 8 byte sequence should unique within the 100 – 1024 generated non-repeating pattern.
4. Your program will then wait for you to enter a short subsection of that original sequence. It will generate an offset value representing where that sub-sequence was found.
5. You will also generate a message at the end to say how many times that 4 or 8 byte sequence was found in your random pattern

Example of non-repeating sequence:

1. a1a2a3a4b1b2b3b4
2. 1122334455661213141516aabbccddee

Notice that **a1a2** or **11223344** doesn’t repeat again.

## Question 1:

1. Imagine you entered a sequence like AAA**AAAA**AAAA and the red A’s represent the sequence you are seeking from that specific location.

Why is it not possible to tell exactly which offset your **AAAA** pattern begins at? **1pt**

1. Is it possible to put the bytes that represent characters below **0x20** and above **0x7E** into your non-repeating sequence? **2pts**
2. Will uppercase be treated differently from lowercase letters when placed on the stack? Why? **1pts**
3. **Submit a screenshot of your program executing along with the python code. To test your program, simply generate a sequence then select a part of that sequence.**

**Example execution:**

**Linux\_prompt~$** **Please enter a seq length and architecture:** **45 4**

Your generated sequence of length 45 is:

Lajsdfpioq***peip****qif*posfadpqwierfasdfwepr1234

Please enter a substring: ***peip***

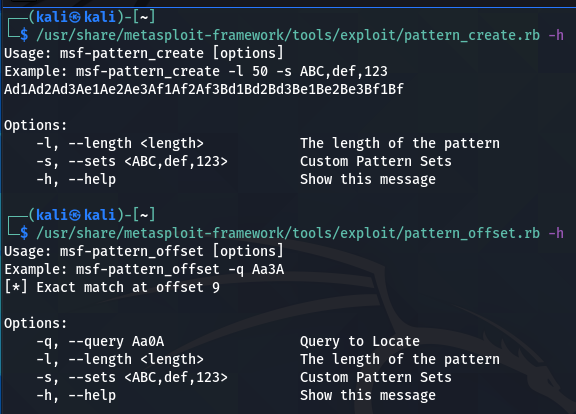
Sub-sequence was found at offset: **10**

Your pattern peip was not found anywhere else.

Happy Penetration Testing!!! Remember to always get permission; it’s the right thing to do 😊

## Interesting Note (no marks)

Kali Linux has 2 tools that perform a similar operation to this. You can see them in action by following the steps below:



**Using the information provided**:

1. Generate a sequence
2. Find a pattern of any length.

# Problem 2 (10 pts)

Write a Python program named **m2p2.py** (module 2, problem 2) that takes a single command line parameter, a directory name.

The program will then go to that directory and produce the following information about the files in that folder:

File Name : <filename>

File Path : <absolute path of the file>

File Size : 125 bytes

Inode : 2356755

Last Mod : Tues Jan 8 07:00:53 2019

What are inodes: https://www.youtube.com/watch?v=\_6VJ8WfWI4k

**HINT**: Modules you may want to investigate (datetime, pathlib, os, glob)

**Submit the code and screenshots of the program executing.**

# Problem 3 (16 pts)

Write a Python program named **m2p3.py** (module 2, problem 3) that dumps the **PE** file header and identifies its parameters (use the link for more details https://en.wikipedia.org/wiki/Portable\_Executable).

Print the parameters using the following format: **6pts**

Report generated by: John Smith

Contact : john.smith@edu.sait.ca

Date/Time : Jan-12-2021 09:27:33 (utc: 232142341342)

File : mytest.exe

Magic : 0x5A4D

PE Header Offset : 0x456

Format : 64 bit

Endian : big

Machine : x86-64

Entry Point : 0x401000

To accomplish this task you can copy one of the following files from your Windows host or VM:

1. **cmd.exe or**
2. **notepad.exe**

You will use 2 approaches to complete the task:

1. You will open the file using the open function and retrieve the information manually.
2. You will use the pefile module and functions to extract the same information.

**You can also use modules (pefile and datetime which should be available by default in your Linux installation)**

## Questions:

1. What is the significance of the Entry point of an executable file? **2pts**
2. What is the significance of the Magic number? In other words what is it used for? **1pt**
3. If the magic number is changed to say all F’s will the program still run? **1pt**
4. Test your hypothesis to see if changing the magic number affects the program execution. **1pt**
5. Evaluate the ease of using the pefile module vs development without it. To help try retrieving something like the time the program was compiled or the name of the compiler used from the PE file. **5pts**

**Submit the code and screenshots of the program executing.**

# Problem 4 (12 pts)

You have been told by your boss to determine which computers are currently on the network. She further tells you that there are 2 subnets **10.110.16.128**/**26** and **10.110.17.128**/**26**.She wants you to generate a list of IP Addresses to scan. She itemizes what she believes you should do as follows:

**(Do not use a third-party module to complete this)**

1. Create a program called **m1subnet.py**
2. Write code that will generate a list of all possible IP Address (including the subnet network address and subnet broadcast address).
3. The output should have the following format:

Subnet Network Address: **10.110.XX.XX**

Subnet First Address: 10.110.XX.XX

Subnet Last Address: 10.110.XX.XX

Subnet Broadcast Address 10.110.XX.XX

Range of IP Address to be scanned: 10.100.XX.XX, 10.100.XX.XX, 10.100.XX.XX, … and so on.

**Submit the code and screenshots of the program executing.**