Network & Internet Forensics Week 7/8

Lab 7/8 – Forensic Tool Development

Overview

This tutorial introduces the concepts and libraries required to parse and visualise PCAP files using Python. Building on your knowledge and understanding of the techniques you employed to analyse PCAP files using Wireshark in previous weeks.

The tasks in this tutorial will help you to develop the components required to visualise network traffic, facilitating the identification of attacks.

Tips & Tricks

- Simply copying and pasting code from the examples in this tutorial and code found online will not help you learn to code.
- Read the documentation associated with the libraries you require, and structure a logical approach to solving the problem before beginning coding.
- Break down the problem you are solving into smaller problems, these form natural functions for you to develop. Consider the input and output required/generated from each of your functions.
- It is OK to reuse and modify code to create your solution, however you must acknowledge the original author, and make clear which code is not your own.

Task 1 – File Input/Output

Reading data from a file is a fundamental of programming. Review the documentation from the link below:

https://docs.python.org/2/tutorial/inputoutput.html

1. Create a text file containing one number per line.
2. Write a Python script that reads each line from the file and displays in on the screen.
3. Write a Python script that take a file name from the terminal and displays each line from the file on the screen. You will need to research the use of `sys.argv[1]`
   a. Tip – `import sys`

Task 2 – Data Structures

Creating and managing data structures is a key requirement of many programs. Review the documentation from the link below:

https://docs.python.org/3/tutorial/datastructures.html

1. Create a list.
2. Append the values from the file you used in task 1 to the list.
3. Display the list on the screen.

Task 3 – Using Functions

Functions are a logical way of separating functionality in your programs that allow code reuse. Functions specify the parameters they are expecting, you pass arguments to the function and it may or may not return a value. Review the tutorial on functions below:

http://www.tutorialspoint.com/python/python_functions.htm

1. Write a script containing two functions
2. The first function function will take a file name and return a list containing the lines from inside the specified file.
3. Call the function first function from the second function and display the list it returns.

Task 4 – Using Matplotlib

Visualisation is a powerful way of representing and allowing trends to be spotted or information inferred more easily. Review the pyplot tutorial below:

http://matplotlib.org/users/pyplot_tutorial.html

1. Install Matplotlib using the following command:
    a. `pip install matplotlib --user`
2. Write a script and create two lists containing random numbers.
3. Use the Matplotlib to generate a graph using the values from the lists.
    a. Tip – `import matplotlib.pyplot as plt`
4. Manipulate the values in the lists a few times and save the resulting graphs as png files.

Task 5 – Using dpkt & Socket

dpkt is a simple packet parsing and creation tool developed by Dug Song et al. You will use it to develop a script to parse and analyse pcap files. The documentation for dpkt is available from the link below:

https://dpkt.readthedocs.io/en/latest/api/api_auto.html#module-dpkt.pcap

1. Install dpkt using the following command:
    a. `pip install dpkt --user`

As you have probably realised, the documentation for dpkt is a little sparse. Fortunately, Jon Oberheide posted an excellent blog post on using dpkt. Review the blog post below:

https://jon.oberheide.org/blog/2008/10/15/dpkt-tutorial-2-parsing-a-pcap-file/

2. Write a script that opens a pcap file and displays the data associated with each packet in the file.
    a. Tip 1 – `import dpkt` at the top of your script
    b. Tip 2 – you must check that the data is TCP to avoid generating errors and crashing your script.
3. Update your script so that it only shows the destination port numbers associated with each packet.

The data returned by dpkt for the packets within a pcap file allow you to identify trends such as which destination port numbers traffic is bound for. If you read the documentation on Oberheide's page you will notice that the source and destination IP addresses are stored in hex, this is denoted by them beginning with \x. In order to convert this data to its more recognisable form (dotted-quad string e.g. 192.168.0.1) the socket library is required:

https://docs.python.org/2/library/socket.html

Search the above documentation for "socket.inet_ntoa(packed_ip)" this is the function required carry out the conversion.

4. Update your script to display the IP addresses of the packets from the pcap file in dotted-quad string format.
    a. Tip – `import socket`

Summary

This lab session introduces all of the Python libraries and concepts required to parse, analyse and visualise the data stored in pcap files.

Ensure you complete this lab session prior to undertaking the coding task in the coursework. You need to be familiar with the libraries and concepts covered in order to complete the assignment.