Scripting for Cybersecurity

Assignment 3: Malware Detection & Network Monitoring

The goal of this assignment is to create a simple malware detection and network monitoring tool in Python.

Task 1 (10%)

Create a new file called "my_detector.py". Add a main function. Add code to manually read command line arguments from the user using only the sys library. Add code to allow a user to pass in an argument "--file-scan". If the user passes this argument then your program should print a list of the contents of the current directory. Use the os library to achieve this.

Task 2 (20%)

Modify the code so that when a user passes in the "--file-scan" argument the program will compute the MD5 hash of all files in the current directory. Subdirectories and folders in your current directory can be ignored. Create a new file called "bad_hashes.txt". This file is to act as a database for the programs antimalware functionality. Select a file in the current directory and place its MD5 hash into this file. It may be a good idea to create a new file for this (e.g. my_fake_malware.txt). Modify your code so that when the user passes the "--file-scan" argument the program computes the MD5 hashes of the files in the current directory and compares each hash with the contents of the bad_hashes.txt file. If a hash is found which matches the hash in the bad_hashes.txt file then the user should be alerted via a printed message.

Task 3 (10%)

Add code to allow a user to pass in a new argument "--net-mon". A user should not be able to use both the "--file-scan" and "--net-mon" arguments at the same time. When a user passes in the "--net-mon" argument they must also pass in an interface name (e.g. eth0 or ens33). When this argument is passed in with an interface name the program should start sniffing traffic on this interface. The Scapy library should be used to achieve this. Create a function to print the source and destination IP addresses of traffic observed on the relevant interface. Traffic can be generated using the ping tool or by using the web browser.

Task 4 (20%)

Create a new file called "bad_ips.txt". This file will be used as a database for the programs network monitoring functionality. Add code to your program to read this file and parse its contents into a list. Add an IP address to this file. When the user passes in the "--net-mon" argument the program should compare the IP addresses in observed traffic with the IP addresses in the bad_ips.txt file. If a match is found then the user should be alerted via a printed message. To test this, add the IP address 10.0.2.30 to the file and use the ping tool to ping that address (i.e. "ping 10.0.2.30") while the program is running. An alternative address may be used to test.

Task 5 (20%)

Modify Task 2 so that the length of time taken to check the files is printed out (e.g. "Scan completed after 5 seconds"). Implement threading to speed up the MD5 hash comparison in task 2. Compare the time taken to perform the task when threading is used vs when it is not used. Document this comparison in comments in your code.

Task 6 (20%)

Create a new file “bad_words.txt”. Add code to your program to read this file and parse its contents into a list. Modify Task 4 to include code to read the payload of packets and search that payload for words in the “bad_words.txt” file. If a match is found then the user should be alerted via a printed message. This functionality can be tested by adding any word to the bad_words.txt file and using the Scapy command line (or in another script) to send a packet with a payload containing that word. Alternatively, add a word like “html”, “script”, or anything you may expect to be included in the source of a web page and use the web browser to navigate to a non-HTTPS site. Sites using HTTPS will have their payload encrypted and it therefore won’t be possible to check the payload.