Joshua Church
Homework 3 – Network Intrusion Detection System

INSTALLATION INSTRUCTIONS

In order to run this program, you will need to work in an environment that has the following installed:

- Python 2.7+ for the base coding language
- Scapy for the various functionality

If additional help is needed after installation, add '-h' in the command line argument. This will display a custom help page.

DESIGN

Given the task to design an Intrusion Detection System, I decided to develop the system with separate function calls. Starting out, the user needs to pass in a set of configuration files via the command line. Each argument is check in the *parse()* function, which checks for specific strings. If help is needed, the user can pass in the *'-h'* in the argument to display a custom help menu. Depending on the configuration files passed in the argument, the system will check either a .pcap file or monitor live traffic. Error checking is embedded within the *parse()* in order to prevent the user from passing an invalid text file.

After parsing, the system begins sniffing for packets using Scapy's built-in packet sniffer. Once a packet is obtained, the system will check a variety of elements. This is solely dependent on the configuration file(s) passed in by the user. Each configuration file must follow the one-item-per-line rule in order to function correctly. Otherwise, the system will face errors parsing the files correctly. One a packet is passed into the *checkPacket()* function, a variety of checks happen. If certain files are passed in via the command line, then the packet will undergo those checks.

Different classes are made to make object arrays to keep up with different parts of a packet. I found this to be easier until trying to implement the port scanning function. I decided to switch to a python dictionary implementation, which was cleaner.

DIFFICULTIES

Intrusion detection is not a simple task. Much research had to be done in order to figure out how to get much of the functionality working. The first issue was simply setting up Scapy on my MacBook. It was taking an unnecessarily long amount of time, so I decided to switch to my Virtual Machine and use Kali Linux. Luckily, Kali Linux pre-installs Scapy, so that was one issue out of the way. Learning how the layers worked in Scapy, along with parsing packets was

problematic. There were some issues with imports and different Python functionality. Generally, though, most of the difficulties resulted in the lack of knowledge in intrusion detection.

Nonetheless, I learned a plethora of new ideas. I know now to start implementation sooner because simply reading and researching took too long. Because too much time went to reading and setting up the environment, I did not reserve adequate time to implement everything I wanted.

LIMITATIONS

Since the program is developed in Python, speed and memory management is not as efficient as C or C++. However, Python is much easier to work with and handles a variety of probable issues. While working in my Kali VM, I was limited to the amount of main memory devoted to the VM, which was 2 GB. Also, the entire computer only has a 1.4 GHz processor, and that was split with the VM.

Since my code has some overhead with multiple packet checking and handling, it tends to drop some packets. The program timestamp checking for the alerting to screen function is slightly wrong. I accidently stored the *datetime.datetime.now().minutes*. However, the problem is when checking the time happens, an abs(datetime.datetime.now() – stored_time) can make the system alert to screen when it should not. It does store the correct timestamp in the log file, though. While signature checking, the packet may not decode correctly, resulting a blacklisted string being missed or resulting in a false positive.

The network port scanning was coming to completion, but there was some parsing error within the function. The dictionary / key values were working as intended, but it may have ultimately been an overhead issue. For now, the function call is commented out, but the function portScanning() is still left in.

When parsing the .pcap file, the log stores the current time and not the timestamp stored in the file itself.

TESTING

I tested the program against a self-developed .pcap file. It was able to parse through and return the correct functionality. It was sluggish reading the file, which contained approximately 2,000 packets. The only stress-testing done in the program was manually loading a web browser with specified domains and IP addresses. I continued to refresh the page, attempting to simulate a flooding of packets. However, I understand that I cannot send as many packets as a script could do.

Other testing done was checking if the passed in files were valid. If not, a specified error was thrown.