CS241 COURSEWORK REPORT

Name: Omkar Satish Gadhave

University ID: u1806161

1. INTRODUCTION

This coursework involves the Implementation of a basic intrusion detection system that detects SYN attack, ARP poisoning attack and Blacklisted URL. The program records any malicious activity and prints a report when the program is interrupted using Cntrl+c. This Coursework is divided into three parts namely Packet sniffing, Intrusion detection and Multithreading.

2. PACKET SNIFFING

By using the template provided, built upon analysis.c so that it can parse the Ethernet, TCP, ARP and IP headers. The first header that was parsed was the Ethernet header followed by IP or ARP header depending on the value of the ether type. The TCP header comes after the IP header. So if the packet has an IP header, and the value of the IP header protocol is 6 then the TCP header is parsed. This part did not require any major decisions but required a thorough understanding of the packet structure which was achieved by reading the manual pages for all the packet headers. This part was tested by printing out important information from the packet headers such as the source and the destination address.

3. INTRUSION DETECTION

After parsing the individual packets, the next step is to analyze them. For this part, the program should record any malicious activity and print a report when the code is killed using Cntrl+c

3.1 SYN Flooding Attack

To detect a SYN flooding attack, Firstly all the IP addresses and the arrival time of the packets with only the syn bit 1 and the rest of the flags set to 0 are stored in a dynamic array. The reason a dynamic array is used because the number of packets to be received can vary and using a static array does not allow for resize. The dynamic array is implemented in such a way that the IP address and the arrival time of an individual packet is stored in a struct and the struct is stored in the array essentially meaning an array of structs. Such an implementation helps store all the data in one place which in turn makes it easy to access. In order to get the number of unique IP addresses, the array is sorted based on the IP address of the packets by using quick sort having a worst case time complexity of O(n^2). The quick sort used here is a standard library function and one of the arguments it takes is the comparator function. The comparator function is used for comparing elements when running quick sort As the array stores the arrival times of the packets, the arrival time of the first and the last packet was found and the time interval was calculated. Afterwards the rate was calculated and checked if it was more than 100 packets per second as well as checking if 90% of the SYN packets were generated from unique source IP address. If both the conditions are satisfied, then a SYN flood attack is possible and the full report is printed on pressing Cntrl+c. This part was tested by generating SYN flooding attack on the loopback interface using hping3. This will send 1000 packets each with the SYN flag enabled to port 80.

3.2 ARP Cache Poisioning

To detect an ARP poisoning attack, we increment the counter every time we encounter an ARP response. Testing was done by listening on the loopback interface after executing the python script present in the test file. The python script sent an ARP packet which incremented the ARP counter. After this the report is printed showing the number of ARP responses.

3.3 Blacklisted URL

For this part www.telegraph.co.uk has been identified as a suspicious domain which is to be monitored. After finding the payload for all TCP packets sent to port 80, search for the Blacklisted URL in the payload. If the Blacklisted URL is present in the payload then increment the counter. One of the ways the code can be tested is by using the wget command to retrieve a webpage which in this

case is www.telegraph.co.uk. Run the wget command while listening on the external enp0s3 interface.

4. MULTITHREADING

The two common approaches for Multithreading are One Thread per X model and the Threadpool model. In the dispatch.c file the One Thread per X model was used a it was simple to implement and it had a low overhead when dealing with constant light loads as no threads are kept idle. MultiThreading was implemented using the pthread library. A dynamic array was used wherein for every new thread a new packet was assigned and then the packet was pushed onto the array. A dynamic array was used as the number of packets received can change significantly. Whenever the program was interrupted using Cntrl+c the threads would get joined and then after the memory used by the array is freed. Freeing the memory used is very important to avoid any memory leaks. Mutex locks ensures synchronization among threads while sharing resources. This is because Mutex lock ensures that once a thread has locked a piece of code then no other thread can execute the same region until its unlocked by the thread that locked it in the first place. So to provide Thread safety and prevent race conditions, Mutex locks are used around reassigning operations.