CS241 Coursework

Ross Gilmore

SYN Attack

A SYN attack relevant to this program was taken to mean more than 1 SYN request to the network from the same IP. We decided to store all MAC source addresses as global pointer, the pointer would store an array of the MAC addresses on the heap. This way every time a packet was scanned, if it was a SYN packet, by comparing the MAC address to each in the array we could test if it was from a client who'd already sent a SYN packet. As the array is on the heap, if a packet contains an unknown address we can use the **realloc** library function to dynamically change the size of the array to contain another MAC address.

The MAC addresses were cast to strings and then added to the array as strings. To check if one MAC address was equal to another, we then simply had to compare if two strings were equal. This was achieved using the strcmp function, which if it returns 0 implies two C strings are equal. As an array of strings, the array was a 2 dimensional array of characters, with each MAC address 18 characters long. Since it is a dynamic array, it functions as a pointer of pointers.

The size of the array of MAC addresses would be of course equal to the number of unique IPs which was also stored as a running variable and the number of SYN packets processed by the program was also counted as it was running. This allowed when the signal interrupt was processed to print both the number of SYN packets analysed by the program but also the number of unique IPs.

Arp Attack

The specification states to simply to count all ARP responses on the network as an arp response is rare due to caching and then output the number of ARP responses after the program is finished running.

Blacklisted URL

To meet this requirement of the specification, a count was kept of all packets with requests to “ www.google.co.uk “.

Multithreading

For the multithreading aspect of the code the idea that we aimed to achieve was have each thread be used to analyse each individual packet. The dynamic array and the variables used to count the number of unique IPs, SYN packets, ARP poisoning attacks and Blacklisted URL detection would be accessed by each individual thread under a mutex lock. This way the threads would not overwrite the values while adding to the results. Ultimately a working solution of this was not developed.

This ultimately would have allowed the program to continue to analyse packets and accept new packets at the same time which is obviously a desired feature for any network program. There would always be a hardware constraint however on the maximum number of packets (number of threads currently being executed) the computer could analyse at any given time, this may also lead the program to not count the actual total number of packets at any given time if the processor cannot process all current packets.

Interface (option 2)

The linux distribution I am using required me to change the main file to use the enp11s0 network interface.

References

https://www.devdungeon.com/content/using-libpcap-c#packet-info

https://wiki.archlinux.org/index.php/Network_configuration#net-tools

https://www.imperva.com/learn/application-security/arp-spoofing/

http://yuba.stanford.edu/~casado/pcap/section2.html