CS241 OPERATING SYSTEMS AND COMPUTER NETWORKS

REPORT FOR CS 241 COURSEWORK 1

Name	Yishan Sun
Student No.	u1903826
Department	Computer Science
Email	u1903826@live.warwick.ac.uk
Date	December 7, 2020

Contents

Introduction

In this coursework, our main goal is to understand, comprehend, and implement a basic intrusion detection system. The two machines communicate with each other via a communication protocol. This protocol is the Internet protocol stack which is *the conceptual model and set of communications protocols used in the Internet and similar computer networks. It is commonly known as TCP/IP because the foundational protocols in the suite are the Transmission Control Protocol (TCP) and the Internet Protocol (IP).*[1]

There are five layers, in this report we mainly use the data link, network, and transport layers. The data link layer encapsulates the data into frames that are sent to the network layer via the IP protocol, and it can also convert the IP address to a local address via the ARP protocol (address resolution protocol). [2,3] Between the network layer and the transport layer, the TCP protocol is used to transfer data.

Methods, Results and Discussion

1. Sniff & Parsing

First we get a packet using the pcap_next() function of the provided sniff function, (the pcap_next() function captures a network packet and returns it as soon as it is received), then we parse the packet, including its headers.[4]

With the dump function we know that the parsing header is to find the position of the header in this entire data packet, (Ethernet header, followed by IP header, followed by TCP header). A pointer is then used to record the position of the individual headers or the other useful function. A header can be declared directly from this address when we need it, as in the following example.

```
/*Second is the Network Header*/
const unsigned char *ip_pointer = packet + 14; //14 is the unchanging length of ether_header which is 6+6+2
struct iphdr *ip_header = (struct iphdr *) ip_pointer;
```

In the second line, we create an ip_header by the address of ip header. The

address of ip header is founded by the address of ether header and add the length of ether header, which is showed on the first line, “packet +14”, 14 is the length of total ether header.

The TCP header is the same as the IP header, consisting of the first address of the IP header plus the total length of the IP header, but it should be noted that the total length of the IP header is obtained from a data * 4 in the IP header. This data is called ihl, which is short for IP header length.

```
/*Third is the Transport Header*/
unsigned char iphdr_length = (unsigned char) 4* (ip_header -> ihl); // ihl in the ip_header store the lendth of whole ip header.
const unsigned char *tcp_pointer = ip_pointer + iphdr_length;
struct tcphdr *tcp_header = (struct tcphdr *) tcp_pointer;
```

2. Syn Flooding Attack

For counting the total number of SYN packets sniffed, we need to detect all the SYN packets whose SYN bit is set to 1 and all other flag bits are set to 0, like the code below

```
unsigned long ip_saddr = ntohl(ip_header->saddr);
// get the source ipAddress to determine the syn flood attack
if((tcp_header -> syn) == 1 &&
     (tcp_header -> urg) ==
     (tcp_header -> ack) ==
     (tcp_header -> psh) ==
     (tcp_header -> rst) ==
     (tcp_header -> fin) == 0)
     {
         syn_count ++;
```

Meanwhile, we need to record the number of them having a unique source IP address, we use malloc() and realloc() function to implement an dynamically growing array, we malloc() a space the first time we receive a packet and then realloc a space each time we add the array. Every time a SYN packet comes in, check whether the IP address has ever been in the array. If not, count++ and add it to the array. Here we set the IP count, size and the array as global variables, which are not emptied of their data.

```
if (size == 0){  //first time Initialize the dynamically growing array
    size ++;
    dynamicArray = (long*) malloc(size*sizeof(long));  //build a dynamically growing array
    add_ipAddress(dynamicArray, ip_saddr);             //and add every new ip address
    unique_ip ++;       //count for unique source ip ++
} else{
    if (!contain_ipAddress(dynamicArray, ip_saddr, size)){  //if this ip address is a new unique address
        unique_ip ++;
        size ++;
        add_ipAddress(dynamicArray, ip_saddr);
    }
}
```

3. ARP cache poisoning

Detecting ARP cache poisoning is easy, we only need to record the number of ARP responses. ARP messages can be detected in the ether-head. There is a data called "*type*", which records the type of the rest of the packet, if it is 0x0806 then it is the ARP mode, which is 2054 in decimal. When this data is equal to 2054 then the ARP data is ++.

```
/*first we entered the Data-Link Header*/
struct ether_header *eth_header = (struct ether_header *) packet;
unsigned short ether_type = ntohs(eth_header->ether_type);
if(ether_type == 2054){   // 2054 is 0x0806 in decimal. which is the arp mode
    arp_count++;        //arp mode, then the arp mode counter ++
}
```

4. Blacklisted URLs.

In order to link from the TCP socket to the HTTP request head, we need a data in the TCP header called "*destport*", the destination port, if this port is 80, meaning that the TCP header payload is HTTP.

The next step is to find the captured packet with " www.google.co.uk " in it, which should be flagged up as malicious. There are two ways to do this.

Method one is using function **char *strstr(const char *haystack, const char *needle)** Find the first occurrence of the string **needle** in the string **haystack**, without the terminator '\0'. So if the result is not NULL, that's means we find the google.

```
if (tcp_destport == 80){ // 80 is http port
    // printf("test in 80\n");
    /*Method 1 easy way explain in the report */
    char *httpRequestHead = ( char *)tcp_pointer + tcphdr_length;
    char blacklist_webpage_String[] ="www.google.co.uk";
    if(strstr(httpRequestHead, blacklist_webpage_String) != NULL){
        printf("success in strstr\n");
        url_count ++;
    }
```

Method 2 is to use function strtok() and strcmp(). char *strtok(char *str, const char *delim) The decomposed string str is a set of strings and

delim is a delimiter. With this function we can separate each line into its own lines. int strcmp(const char *str1, const char *str2) compares the string pointed to by str1 with the string pointed to by str2. It will return 0, if they are the same.

```
/*method 2, explain in the report */
int length = (*header).len - (14 + iphdr_length + tcphdr_length);

readableHead = (char*) malloc(length*sizeof(char));

int i;
for (i = 0; i < length; i++) {
    char byte = (httpRequestHead[i]);
    readableHead[i] = byte;
}
char *pureSentence;
pureSentence = strtok(readableHead,"\r\n");
char blacklist_webpage_String[] = "Host: www.google.co.uk";
while(pureSentence != NULL){
        if (strcmp(pureSentence,blacklist_webpage_String) == 0){
            url_count ++;
            return;
        }
        pureSentence = strtok (NULL,"\r\n");
}
```

5. Global variables

In all three of the above vignettes, we needed to record something and print it out when we exited the programme. This required a global variable, which we placed in the analysis,h file and initialised and assigned in analysis.c.

```
/*extern is used to call this value in another file. */
extern long syn_count;        //count for total syn packet
extern long unique_ip;        //count for unique source ip
extern long arp_count;        //count for ARP Cache Poisoning
extern long url_count;        //count for Blacklisted URLs
extern long *dynamicArray;    //dynamically growing array
extern int size;              //size of dynamically growing array
```

Then in the sniff.c we use signal function to detect “ctrl + c” command, print it out after that.[5]

```c
/*reference: Linux Signals 锟斤拷 Example C Program to Catch Signals (SIGINT, SIGKILL, SIGSTOP, et
/*website page: https://www.thegeekstuff.com/2012/03/catch-signals-sample-c-code/  */
void sig_handler(int signo)
{
 if (signo == SIGINT){
    printf( " \n "
        " Intrusion Detection Report:\n "
        " %ld SYN packet detected from %ld different IPs(syn attack)\n ", syn_count, unique_ip);
    printf(" %ld ARP responses (cache poisoning)\n ", arp_count);
    printf("  %ld URL Blacklist violations\n ", url_count);
  _exit(0);
 }
}
```

```c
 /*reference: Linux Signals 锟斤拷 Example C Program to Catch Signals (SIGINT, SIGKILL, S
 /*website page: https://www.thegeekstuff.com/2012/03/catch-signals-sample-c-code/  */
if (signal(SIGINT, sig_handler) == SIG_ERR)
printf("\ncan't catch SIGINT\n");
 // A long long wait so that we can easily issue a signal to this process
```

6. Threads

For thread, we use One Thread per X Model, with POSIX.

We use thread in dispatch.c to call the analysis function, In order to do this, we modified the parameters of the analysis function, since pthread_create can only accept input from one function and one parameter, we created a struct as an incoming parameter in the dispatch.h file. And Free the thread after using.

```c
struct thread_args{
    struct pcap_pkthdr *header;
    const unsigned char *packet;
    int verbose;
}
```

```c
pthread_t threads;
struct thread_args test_args;
pthread_create(&threads, NULL, &analyse, &test_args);
free(threads);
```

These parameters are then passed back again in analysis.c.

```c
pthread_mutex_t muxlock = PTHREAD_MUTEX_INITIALIZER;

void analyse (void *arg) {
    // TODO your part 2 code here
    /*we need to define a pointer to a struct of type thread_args to reference
    struct thread_args *test_args = (struct thread_args *) arg;
    struct pcap_pkthdr *header = (struct pcap_pkthdr *) test_args -> header;
    const unsigned char *packet = (const unsigned char *) test_args -> packet;
    int verbose = (int) test_args -> verbose;
```

The muxlock is used to lock the global variables in case this value is also used by other threads.

```
  printf("ether type is %hd\n", ether_type);
if(ether_type == 2054){   // 2054 is 0x0800 in decimal. which is the arp mode
    pthread_mutex_lock(&muxlock);
    arp_count++;        //arp mode, then the arp mode counter ++
    pthread_mutex_unlock(&muxlock);
}

char blacklist_webpage_String[] ="www.google.co.uk";
if(strstr(httpRequestHead, blacklist_webpage_String) != NULL){
    printf("success in strstr\n");
    pthread_mutex_lock(&muxlock);
    url_count ++;
    pthread_mutex_unlock(&muxlock);
}
```

Conclusion

In this report we mainly explain and implement how to parse the header of each layer. With the data we have parsed, we can do some basic intrusion detection. Our code is not yet resistant to these attacks, but it can find hidden dangers.

Unfortunately, however, the threaded changes did not work, and when the code was written and compiled it was found that there was a bug that could not be solved. Meanwhile that the call to strstr() method did not work properly, whereas the student who also used this method could. We added a number of printf's (which were removed for aesthetic reasons) to check that the program was working correctly.

Reference & Appendices

1. Wikipedia, The Free Encyclopedia, Internet protocol suite [online]; 2020 [2020 Dec 07]. Available from: https://en.wikipedia.org/wiki/Internet_protocol_suite (Accessed 07 Dec 2020)
2. D.C. Plummer, Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware, RFC826.
3. Plummer, D, "An Ethernet Address Resolution Protocol', RFC-826, Symbolics Cambridge Research Center, November 1982.
4. Mike 江，*Linux 网络编程—— libpcap 详解* [online]; 2020[2020 Dec 07]. Available from: https://blog.csdn.net/tennysonsky/article/details/44811899 (Accessed 07 Dec 2020)
5. HIMANSHU ARORA, Linux Signals - Example C Program to Catch Signals (SIGINT, SIGKILL, SIGSTOP, etc.), on MARCH 9, 2012
6. Cs241 coursework 2020-2021,[online]; [online]; 2020 [2020 Dec 07]. Available from: https://warwick.ac.uk/fac/sci/dcs/teaching/material/cs241/coursework20-21/ (Accessed 07 Dec 2020)