Report

2061245 Betty

November 29, 2021

Contents

1 General

This project is used to capture and analyse web packets, output suspicious malicious packets to users. PCAP library is used for capturing packages. pcap_loop function will pass packet and user input to dispatch, dispatch will control the multi threads and let analyse to handle the packet. analyse.c will using parameters of packets like port number, protocol type or reading raw data in it to determine if the packet is malicious. Due to high network traffic, the single-thread process is not enough to support massive network packages analysis. Therefore, multi-threads is used in this project. After consideration, the thread pools strategy is chosen.

2 Threading strategy

There are two main strategies that can be implemented for this project. One thread per package and thread pools. Compared to One thread per package, thread pools offer these advantages:

1. Using an existing thread to serve a request cost less time than creating a thread.
2. Total resource occupation is limited for thread pool strategy.[1]

Hence, thread pools strategy is used.

3 Implementation

Thread pool keeps a set number of threads running and waiting to do something. An array is created to store id of each thread. And then use a for loop and pthread_create function, pass the function the thread should be worked on as argument.

```
pthread_t tid[THREADNUM];
    for (int i = 0; i < THREADNUM; ++i) {
pthread_create(&tid[i], NULL, analyse, NULL);
}
```

Next, implement a queue. The queue is basically a FIFO data type, use to pointer point to head and tail node. Structure job will be the node for the queue which indicate there is a packet that need thread and pass parameters to threads. In order to let threads sleep and wake properly, pthread_cond and pthread_mutex is used. Let's call this mutex lock queue lock, use to protect the job queue.

When threads on analyse.c get queue lock and find queue is empty. This thread will be blocked and release the lock, waiting for the condition specified by cond to be signalled or broadcast to.[2]

```
pthread_mutex_lock(&queueMutex);
while (isEmpty(q)){
pthread_cond_wait(&queueCond, &queueMutex);
}
// get the job
job *j = deQueue(q);
pthread_mutex_unlock(&queueMutex);
```

When a packet is captured and pass to dispatch.c using pcap_loop, main thread will compete and get the queue mutex, create job node and enqueue, then broadcast to all blocked threads. These threads wake up and compete for mutex lock, one of them will get this packet and go on to analysis process, others will check if queue is empty and keep waiting for next broadcast.

```
job *j;
j = (job *)malloc(sizeof (struct job));
int verbose = (int)userdata;
createJob(j, header, packet, verbose);
pthread_mutex_lock(&queueMutex);
enQueue(j, q);
//  broadcast to all waiting thread
pthread_cond_broadcast(&queueCond);
```

```
pthread_mutex_unlock(&queueMutex);
```

In order to protect common variables that are used to count three types of malicious packets (SYN Flooding Attack, ARP Cache Poisoning, Blacklisted URLs), three mutex locks are used. When a thread needs to modify common data, it will acquire and lock the mutex, process the modification, and unlock. Because there is only one lock is required at a time, deadlock will not appear in this project. Note that the variables are used for different files, except one file, other files should use extern to declare these variables.

4 Testing

This project is well-tested using white box testing. And using the method provided by the coursework page.[3] The most useful test is the SYN Flooding Attack test. It requires hping3 to send 100 packets to local address. Using lo will prevent interference by Internet traffic, and massive packets can test if the common variables are well protected. It requires the output 100 SYN packets detected from 100 different IPs (syn attack). And the project give the same output as expected. Therefore, the project is working properly.

References

[1] Galvin, P., Silberschatz, A. and Gagne, G., 2005. Operating System Concepts, Seventh Edition. 7th ed. p.141.

[2] Ibm.com. 2021. IBM Docs. [online] Available at: <https://www.ibm.com/docs/en/i/7.4?topic=ssw_ibm_i_74/apis/users_78.htm> [Accessed 26 November 2021].

[3] University of Warwick. 2021. CS241 Coursework 2021-2022. [online] Available at: <https://warwick.ac.uk/fac/sci/dcs/teaching/material/cs241/coursework21-22/> [Accessed 26 November 2021].