University of Warwick

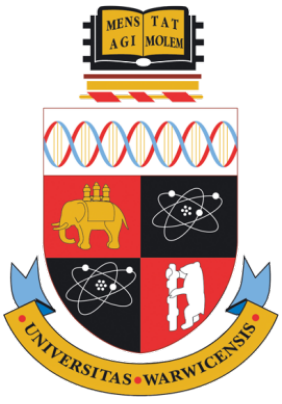
CS241

Operating Systems and Computer Networks

Threaded Packet Sniffer

Nathan Hall 1707704

November 22, 2018



Contents

1 Introduction

This system is a multi-threaded, intrusion detection system for XMAS tree scans, ARP poisoning and blacklisted URL requests.

2 Design

Firstly, the packets coming into the system must be broken down into their constituting headers for the multiple layers of the OSI model[2]. The system uses the previous header to determine what protocol is next in the network stack. It identifies ethernet, IP, ARP and TCP headers, breaking down the packet as shown in Figure 1.

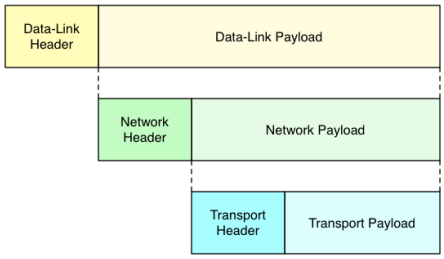


Figure 1: OSI Model - Pakcet Breakdown [2]

Once the packets are broken down the system can then analyse them for suspicious activity, there are 3 types that the system will detect:

- Xmas Tree Scan: This attack attempts to identify information about a system by sending an unexpected combination of TCP Flags. OS's respond differently to this combination for different ports, revealing information on the OS running [3]. To detect the attack the system checks that TCP flags for FIN, URG and PSH are all set, incrementing a counter if so.
- ARP Poisoning: By spoofing ARP replies attackers can implement MITM attacks by altering the cache addresses for devices on a network, redirecting traffic through themselves. This requires an influx of ARP request packets to ensure the attackers request is service. To test for this attack the system checks if the operation of the header is reply and if so will increment the ARP attack counter.
- Blacklisted URLs: When accessing a website a packet is sent out requesting the webpage information. The address of the request is stored in the HTTP header. The system uses the strstr method [9] to search for the string www.bbc.co.uk. If this returns a value then the counter is incremented.

In high load systems packets are constantly flooding in and this is something the system should be able to deal with. A single threaded system is limited by the down-time of requests and wastes CPU running time. As such the system implements a threadpool. Packets entering the system are added to a queue where they wait until a thread requests a packet to process. This allows for concurrent processing without the cost of rebuilding a new thread for every packet. With threading, the system must be careful of race conditions, as such it implements mutex locks for shared memory, as only binary semaphores are applicable in this model.

3 Implementation

The retrieval of packets is handled by the libpcap interface [6], pcap_loop() continually retrieves packets and calls the packet_handler() function where the packet can be sent to dispatch() to be processed. Dispatch makes a memory copy of the packet, as under high loads libpcap can overwrite its own buffer. From here a packet element struct is initialised and added to the queue (under mutex lock).

When the system initialises it sets-up the threads and packet queue:

- The threads use the pthread library [1]. The number of threads is controlled by the constant THREAD_NO, the systems development VM runs with 1 core, so 2 threads is ideal. The threads are initialised to run the function thread_code continually until FLAG_RUN is set to 0. This flag must also be mutex locked as it is edited by the master thread when the system closes (possible race condition).

- The packet queue makes use of a linked list structure, implemented using structs.

An aim for the system was to reduce the number of mutex locks required, for simplicity and to reduce the wait time for each thread. One way this is done is by using thread local attack counters. Each thread tracks the attacks it has discovered and only on termination are the threads counters summed. This requires a modification to the original analyse function, to return a pointer to a struct of the counters for each packet analysed, which is then added to the thread local counter. To avoid memory leaks it is important to free the counters, packet and packet struct once analysed.

The analyse function makes use of defined structs in the netinet package files [8] and the nature of pointer arithmetic to manipulate the packet into individual structs, containing each relevant protocol header.

While the system is executing the user can send a SIGINT signal, telling it to stop. When received the system must clean up all memory allocations, close all threads, stop pcap_loop and provide the user with a final report.

4 Testing

Multiple testing methods were used through the development of the packet sniffer system:

- Verbose - Due to the systems modified verbose mode, using -v when running idsniff revealed details on the packets and helped to determine if the structs had successfully captured the packet headers.

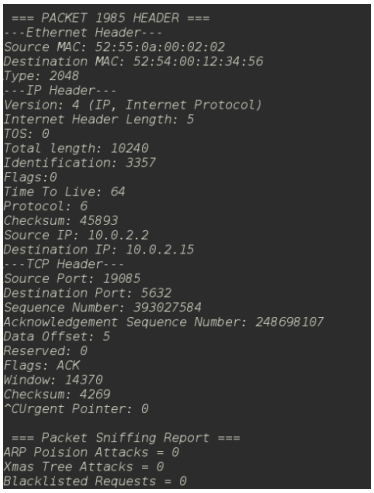


Figure 2: Verbose Output

- Valgrind - Valgrind is a debugging tool[10] which details the memory affect on a computer the system has, including any memory leaks (caused by improper freeing of heap memory).

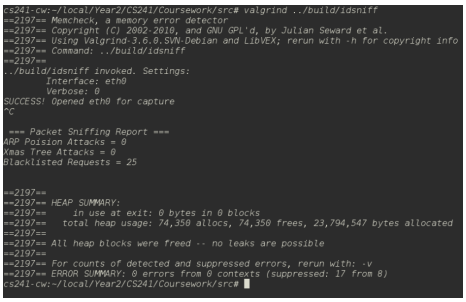


Figure 3: Valgrind Example Output

- Helgrind - An tool extension to Valgrind [7], this details any possible race conditions due to improper mutexing (DRD is another useful tool [4])

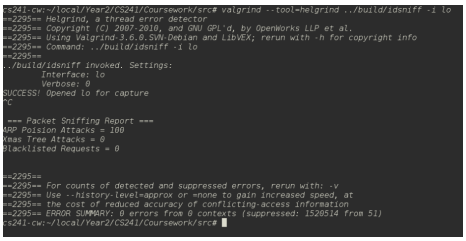


Figure 4: Helgrind Example Output

- GDB - The GNU Debugger allows for more specific debugging, by running the code line by line, with the option to include breakpoints, dump variable values and much more. [5]

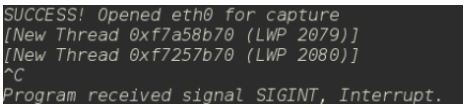


Figure 5: GDB Example, not including breakpoints

- Git - Documentation of errors is vital in a threading project as they may occur rarely. The error documentation on git is a perfect tool for monitoring these errors

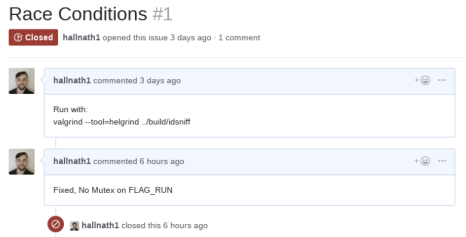


Figure 6: Git Error Log Example

- Stress Testing - The system is designed to be able to manage high loads and therefore requires stress testing. For each attack a test has been created:
  - Xmas Tree Attack - nmap -sX localhost -p 1-65535 examines all ports and sends over 60,000 XMAS packets
  - ARP Poisoning - Modifying the supplied python script to take an argument, allows for control over the number of ARP packets sent
  - Blacklisted Sites - A bash script runs wget x times, where x is a command line argument.

```
 === Packet Sniffing Report ===
ARP Poision Attacks = 0
Xmas Tree Attacks = 65539
Blacklisted Requests = 0
```

Figure 7: Xmas Stress Testing Output

```
for x in range(int(sys.argv[1])):
    send(arp)
```

Figure 8: ARP Poisoning Stress Testing Script

```
#! /bin/bash
for ((i=0;i<=$1;i++))
do
    wget www.bbc.co.uk/news
done
rm news*
```

Figure 9: Blacklisting Stress Testing Script

5 Conclusion

There are many intricacies to threaded systems and as such the system is not perfect and development is ongoing. The system takes the theory of packet headers and network intrusion attacks, and implements them using an elegant thread pooling solution. This is a system with high adaptability and could easily be extended to cover more attacks or provide additional functionality in future iterations.

References

[1] Jacqueline Proulx Farrell Bradford Nichols Dick Buttlar. *Pthreads Programming A POSIX Standard for Better Multiprocessing*. O'Reilly, September 1996. ISBN: 1565921151.

[2] Dr Adam Chester. *CS241: Network Primer*. URL: `https://warwick.ac.uk/fac/sci/dcs/teaching/material/cs241/coursework18-19/network-primer/`. (accessed: 21.11.2018).

[3] Andrew Whitaker Daniel P. Newman. *Penetration Testing and Network Defense*. Cisco Press, October 2005. ISBN: 1587052083.

[4] *DRD*. URL: `http://valgrind.org/docs/manual/drd-manual.html`. (accessed: 21.11.2018).

[5] *gdb*. URL: `https://www.gnu.org/software/gdb/`. (accessed: 21.11.2018).

[6] TCP Dump Group. URL: `https://github.com/the-tcpdump-group/libpcap`. (accessed: 21.11.2018).

[7] *Helgrind*. URL: `http://valgrind.org/docs/manual/hg-manual.html`. (accessed: 21.11.2018).

[8] *Netinet Library*. URL: `https://svnweb.freebsd.org/base/head/sys/netinet/`. (accessed: 21.11.2018).

[9] *StrStr Function*. URL: `http://www.cplusplus.com/reference/cstring/strstr/`. (accessed: 21.11.2018).

[10] *Valgrind*. URL: `https://linux.die.net/man/1/valgrind`. (accessed: 21.11.2018).