CSED353 Computer Networks Assignment #6

20160360 명성현

1. Capture packets using libpcap

My code is in "q1.c". Here are some key functions.

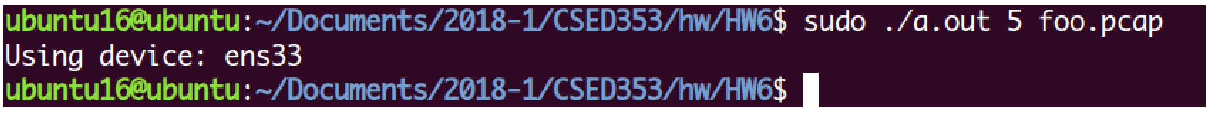
- `pcap_lookupdev()` finds a device that we can sniff packets.

- `pcap_openlive()` establishes a connection to the device that can bring packets.

- `pcap_compile()` and `pcap_setfilter()` are uses to catch only TCP, UDP, and ICMP packets which go through well-known ports(1~1024).

- `pcap_dump_open()` opens a dump file to output, and `pcap_dump()` prints a packet to the dump file.

Use "`gcc q1.c –lpcap`" to compile. The program expects two arguments – the period to capture packets in seconds and the name of file to print captured packets to. Because we need access to network interface cards, a sudo privilege may be required.



```
ubuntu16@ubuntu:~/Documents/2018-1/CSED353/hw/HW6$ sudo ./a.out 5 foo.pcap
Using device: ens33
ubuntu16@ubuntu:~/Documents/2018-1/CSED353/hw/HW6$
```

My version of packet capture is provided in "20160360.pcap". The time period is 30 seconds.

2. Packet analysis using libpcap

My code is in "q2.c". Here's how it works:

- The program is expecting one or two arguments. The first argument is the name of packet capture file. The second argument is the name of file to print endpoint analysis, which is optional. The endpoint analysis will be skipped if this argument isn't given.

- We begin by opening the capture file with `pcap_open_offline()`. Then parse the packets until the end using `pcap_next()`.

- Now we parse through the packet one by one.

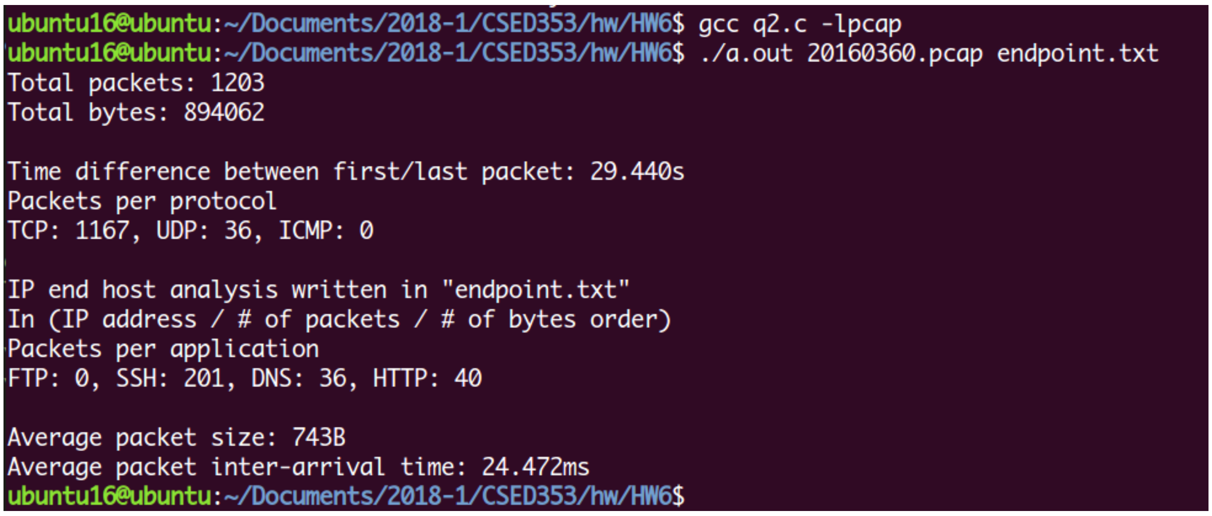
  - The timestamp and legth is given in libpcap header.

- Offset 0x17 of the packet holds information about which transport layer protocol is used.

- Since we don't know much about which application layer protocol is used, we guess it from the ports that are being used. Offsets 0x22~3 and 0x24~5 holds the source/destination port respectively. SSH will use port 22, DNS port 53, and HTTP port 80.

- Finally, read each end's IP address to update the endhost list. I have implemented it as a linked list – if the IP has previously, it will update the packet count and total bytes of that node. If not, then it will make a new node at the end of list with the new IP.

Compile the same way as part 1, and run the program with the pcap file from part 1. The endpoint analysis is also provided in "endpoint.txt".



```
ubuntu16@ubuntu:~/Documents/2018-1/CSED353/hw/HW6$ gcc q2.c -lpcap
ubuntu16@ubuntu:~/Documents/2018-1/CSED353/hw/HW6$ ./a.out 20160360.pcap endpoint.txt
Total packets: 1203
Total bytes: 894062

Time difference between first/last packet: 29.440s
Packets per protocol
TCP: 1167, UDP: 36, ICMP: 0

IP end host analysis written in "endpoint.txt"
In (IP address / # of packets / # of bytes order)
Packets per application
FTP: 0, SSH: 201, DNS: 36, HTTP: 40

Average packet size: 743B
Average packet inter-arrival time: 24.472ms
ubuntu16@ubuntu:~/Documents/2018-1/CSED353/hw/HW6$
```

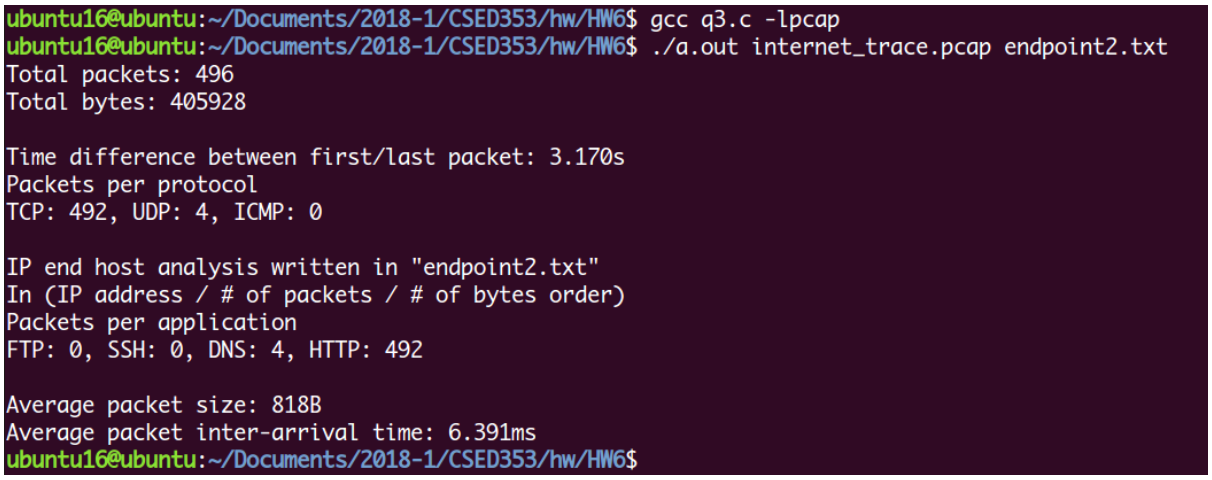
3. Packet decapsulation using libpcap

My code is "q3.c", which identical with part 2 with just minor adjustments. Since we want to know what's inside the IP packet capsulated by GTP protocol, we need to recalculate the offsets. Here are some hacks that I used. Note that these may not work in other trace files other than the given one in the course web page.

- Every packet in this trace file will have two IP headers: the original one and another one inside the GTP protocol. It turns out that data outside the 'inner IP packet' is 40 bytes – 36 on the front and 4 bytes of Ethernet protocol footer on the back. Therefore, we can just subtract 40 from the outer packet length to obtain the inner packet's length.

- Also, the offset difference between the inner IP packet and the outer IP packet is constant – this means we can add 40 to the packet pointer to access the inner packet the same way. So at the beginning of the `while` loop, I added "`packet += 0x28;`".

Compile and run the same way as part 2, and here’s the result. Endpoint analysis is in “endpoint2.txt”.



```
ubuntu16@ubuntu:~/Documents/2018-1/CSED353/hw/HW6$ gcc q3.c -lpcap
ubuntu16@ubuntu:~/Documents/2018-1/CSED353/hw/HW6$ ./a.out internet_trace.pcap endpoint2.txt
Total packets: 496
Total bytes: 405928

Time difference between first/last packet: 3.170s
Packets per protocol
TCP: 492, UDP: 4, ICMP: 0

IP end host analysis written in "endpoint2.txt"
In (IP address / # of packets / # of bytes order)
Packets per application
FTP: 0, SSH: 0, DNS: 4, HTTP: 492

Average packet size: 818B
Average packet inter-arrival time: 6.391ms
ubuntu16@ubuntu:~/Documents/2018-1/CSED353/hw/HW6$
```