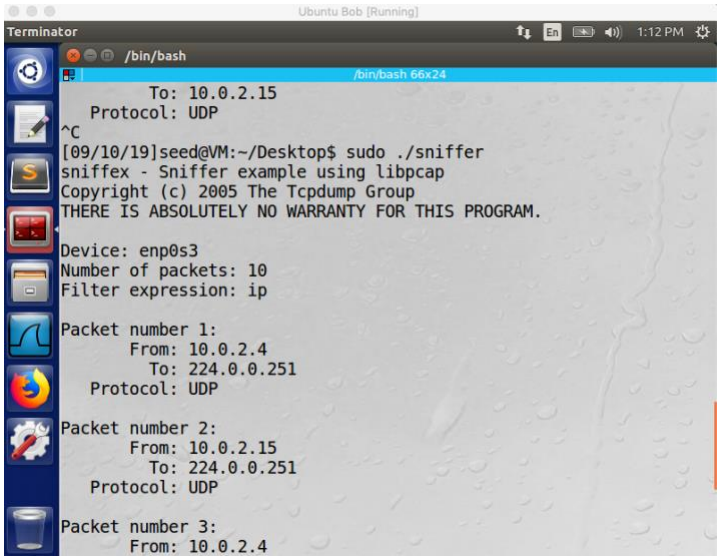
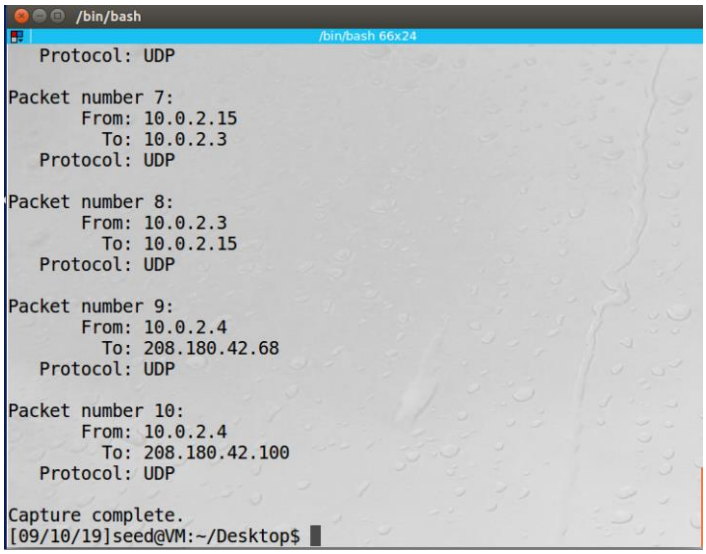
Cody Williams
CSCE 465
September 10, 2019

Homework 1 Report

PART 1: PACKET SNIFFER

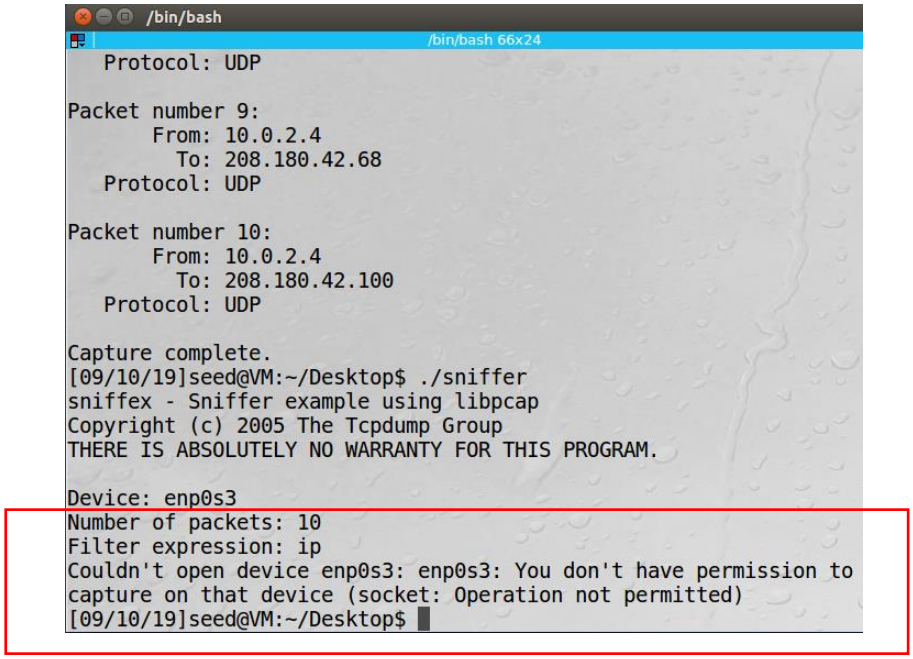
- **Please download the sniffex.c program from the tutorial mentioned above, compile and run it. You should provide screendump evidence to show that your program runs successfully and produces expected results.**





- **Problem 1: Please use your own words to describe the sequence of the library calls that are essential for sniffer programs. This is meant to be a summary, not detailed explanation like the one in the tutorial.**
  - **Set the device for sniffing:** This can be done through user provided input such as…
    - printf("Device: %s\n", dev);
  - **Open the device for sniffing:** This is where the user will indicate promiscuous mode or not
    - pcap_t *pcap_open_live(char *device, int snaplen, int promisc, int to_ms,char *ebuf)
  - **Compile the program:**
    - pcap_compile(pcap_t *p, struct bpf_program *fp, char *str, int optimize, bpf_u_int32 netmask)
  - **Set filters:** This library call is used to focus or remove unnecessary information from the pcap results
    - pcap_setfilter(pcap_t *p, struct bpf_program *fp)
  - **Capture the packet**: Capturing the actual information contained in the packet
    - u_char *pcap_next(pcap_t *p, struct pcap_pkthdr *h)
  - **Format the callback function for the packet capture**
    - void got_packet(u_char *args, const struct pcap_pkthdr *header, const u_char *packet);

- **Problem 2: Why do you need the root privilege to run sniffex? Where does the program fail if executed without the root privilege?**
    - The root privilege is needed to run sniffex because root privilege is required because pcap requires access to the network interface of the device. Accessing the network interface is a privileged operation. If sniffex is run without root privilege, the program will fail to execute when it attempts to open the device with the message …
      "Couldn't open device (device name) : you don't have permission to capture on that device"



- **Problem 3: Please turn on and turn off the promiscuous mode in the sniffer program. Can you demonstrate the difference when this mode is on and off? Please describe how you demonstrate this.**
    - Promiscuous mode can be turned on and off by changing the value of the "int promisc" value to 1 for PROMISCOUS ON or 0 for PROMISCOUOUS OFF in …
      pcap_t *pcap_open_live(char *device, int snaplen, int promisc, int to_ms, char *ebuf)
    - If promiscuous mode is turned ON, packets from all devices communicating over the wire will be captured
    - If promiscuous mode is turned OFF, the host will only sniff traffic that is related to the host (although unrelated traffic can slip through).
    - The left image is traffic captured by host 10.0.2.15 with promisc ON (unrelated traffic highlighted), the right image is the same host with promisc OFF

```
handle = pcap_open_live(dev, SNAP_LEN, 1, 1000, errbuf);
```

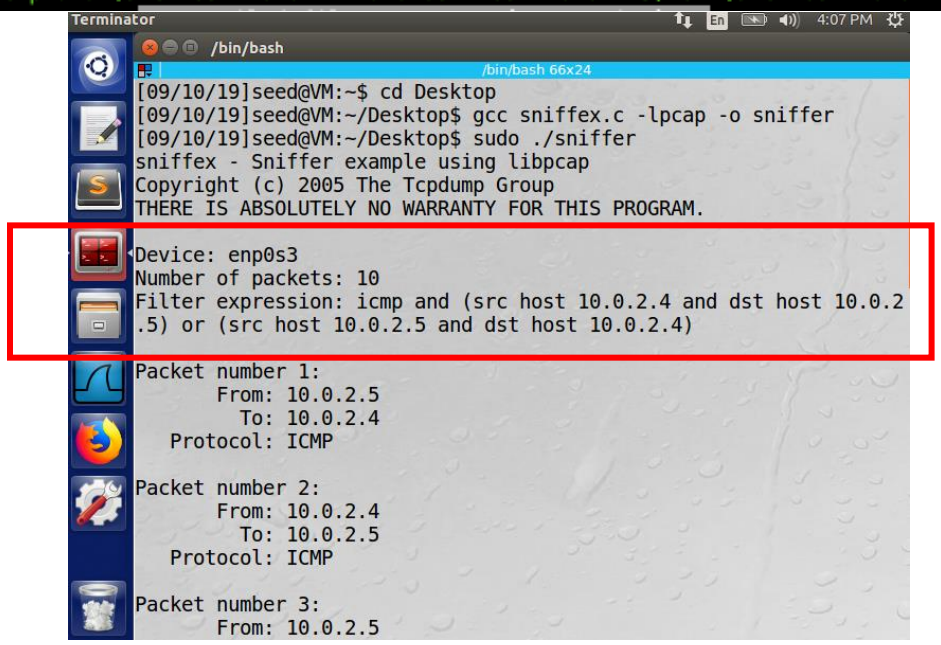
```
/bin/bash
/bin/bash 66x24
   Protocol: UDP

Packet number 7:
       From: 10.0.2.15
         To: 10.0.2.3
   Protocol: UDP

Packet number 8:
       From: 10.0.2.3
         To: 10.0.2.15
   Protocol: UDP

Packet number 9:
       From: 10.0.2.4
         To: 208.180.42.68
   Protocol: UDP

Packet number 10:
       From: 10.0.2.4
         To: 208.180.42.100
   Protocol: UDP

Capture complete.
[09/10/19]seed@VM:~/Desktop$
```

```
handle = pcap_open_live(dev, SNAP_LEN, 0, 1000, errbuf);
if (handle == NULL) {
```

```
/bin/bash
/bin/bash 66x24
   Protocol: UDP

Packet number 7:
       From: 208.180.42.100
         To: 10.0.2.15
   Protocol: UDP

Packet number 8:
       From: 208.180.42.68
         To: 10.0.2.15
   Protocol: UDP

Packet number 9:
       From: 10.0.2.15
         To: 104.73.84.18
   Protocol: TCP
   Src port: 33964
   Dst port: 80

Packet number 10:
       From: 104.73.84.18
         To: 10.0.2.15
   Protocol: TCP
   Src port: 80
```

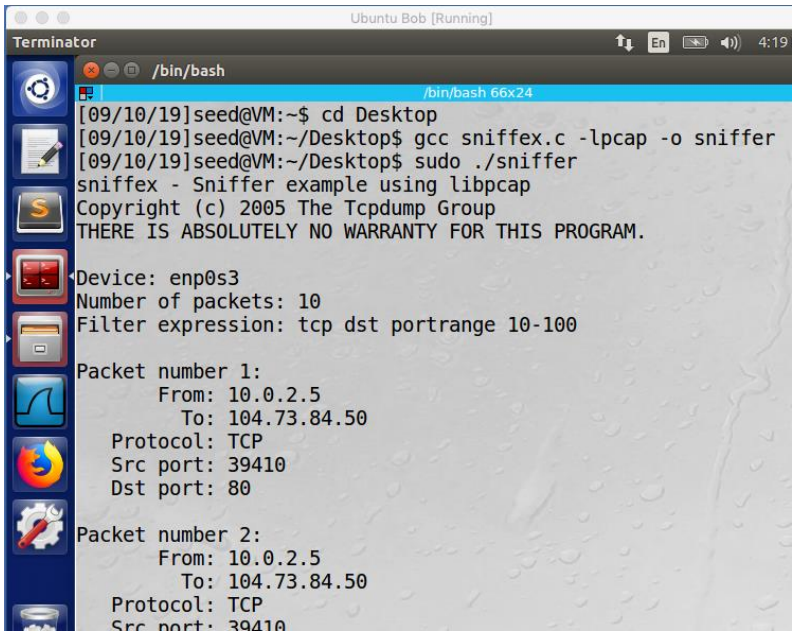
- **Problem 4: Writing Filters Please write filter expressions to capture each of the followings. In your lab reports, you need to include screendumps to show the results of applying each of these filters.**
  - **Capture the ICMP packets between two specific hosts**
  - The following change was made to sniffex.c

```
char filter_exp[] = "icmp and (src host 10.0.2.4 and dst host 10.0.2.5) or (src host 10.0.2.5 and dst host 10.0.2.4)";
```

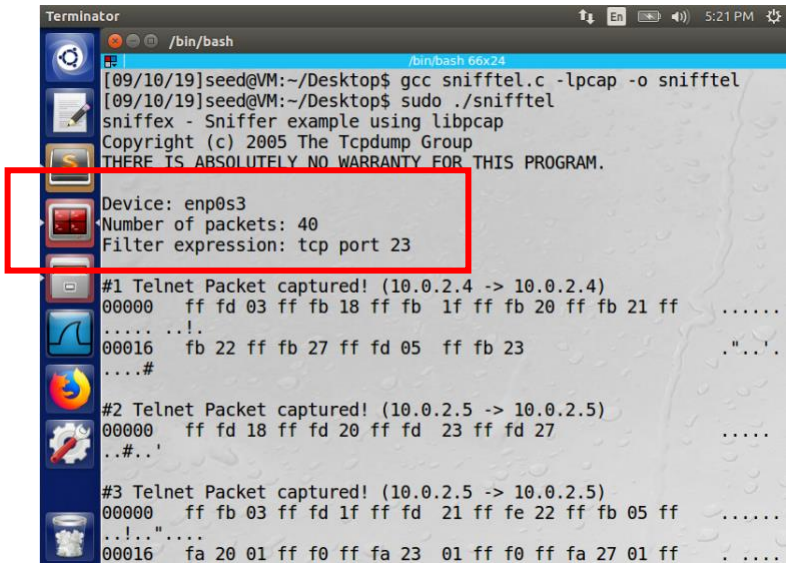


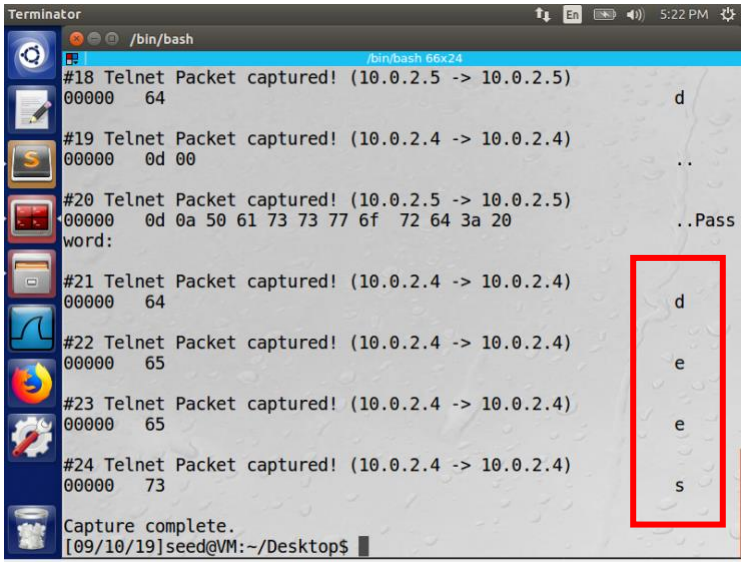
- **Capture the TCP packets that have a destination port range from to port 10 - 100.**
  - The following change was made to sniffex.c
  -

```
char filter_exp[] = "tcp dst portrange 10-100";
```



- **Problem 5: Sniffing Passwords Please show how you can use sniffex to capture the password(s) when somebody is using telnet on the network that you are monitoring. You can start from modifying sniffex.c to implement the function. You also need to start the telnetd server on your VM. If you are using our pre-built VM, the telnetd server is already installed; just type the following command to start it. sudo service openbsd-inetd start.**



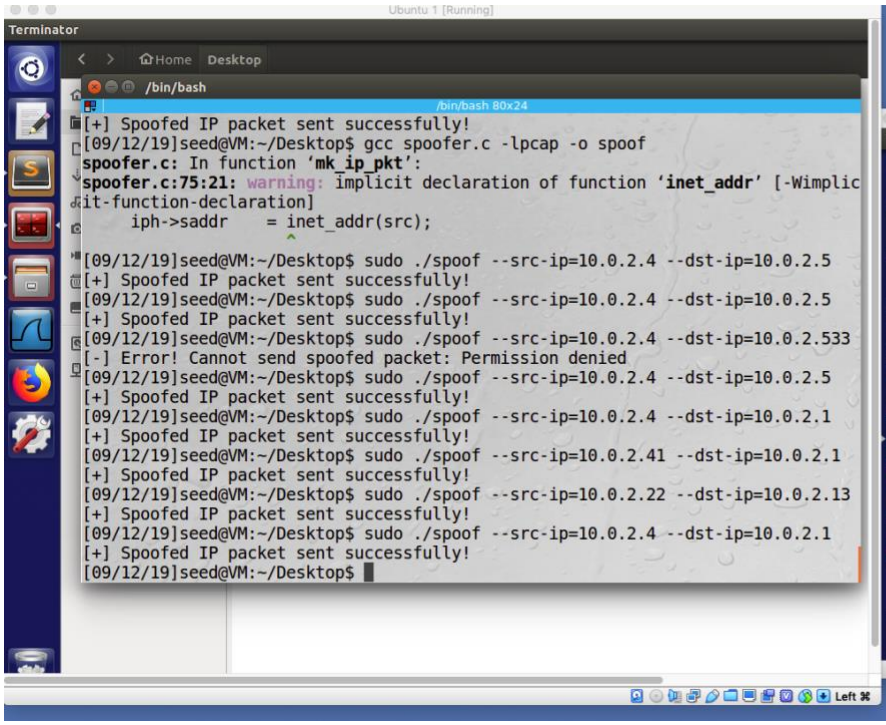


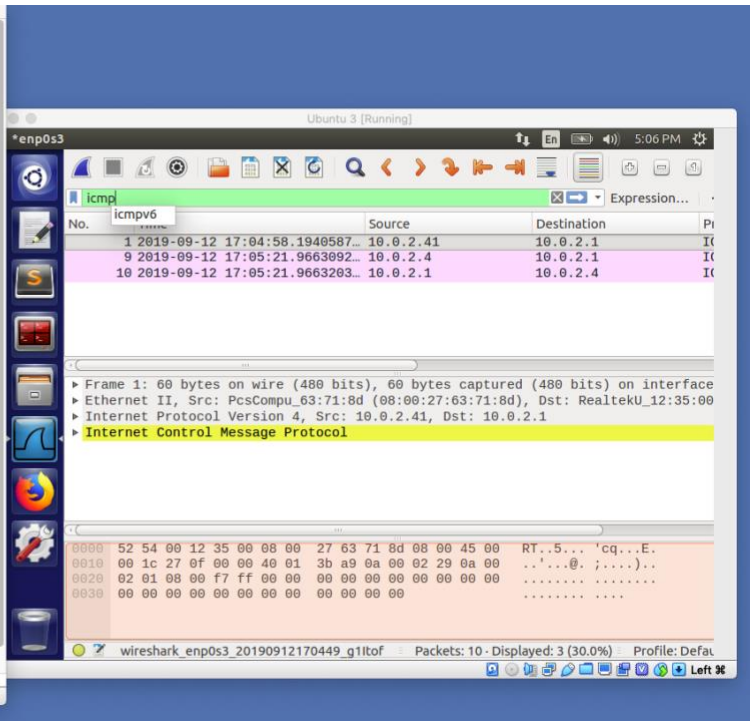
PART 2: PACKET SPOOFER

Task 2.a: Write a spoofing program. Please write *your own* packet spoofing program in C. You need to provide evidences (e.g., Wireshark packet trace) to show us that your program successfully sends out spoofed IP packets.

- Spoofer code provided in spoofer.c
- Main components of the program
    - Chesksum function: Validate the size of the packets provided to function calls
    - Make ping packet: Generates and ICMP ping packet
    - Make IP packet: generates an ip packet used to encapsulate the ping/ ICMP packet
    - Send packet: sends out the constructed packet
    - Main function: takes in command line args for src and ds tip used for the construction and send ing of the ICMP packet

Task 2.b: Spoof an ICMP Echo Request. Spoof an ICMP echo request packet on behalf of another machine (i.e., using another machine's IP address as its source IP address). This packet should be sent to a remote machine on the Internet (the machine must be alieve). You should turn on your Wireshark, so if your spoofing is successful, you can see the echo reply coming back from the remote machine.





Task 2.C: Questions. Please answer the following questions.

1. **Can you set the IP packet length field to an arbitary value, regardless of how big the actual packet is?**
    a. Yes. If the packet length differs in size of the actual packet, the packet will be padded with zeroes.
2. **Using the raw socket programming, do you have to calculate the checksum for the IP header?**
    a. Yes. The IP header checksum is used to protect against errors

3. **Why do you need the root privilege to run the programs that use raw sockets? Where does the program fail if executed without the root privilege?**
    a. Raw socket access is generally needed for protocols which are generally a part of the Kernel, which requires root privileges. Additionally, spoofed custom packets may interfere with inbound traffic which can be problematic.

3.3 Task 3: Sniff and then Spoof

In this task, you will combine the sniffing and spoofing techniques to implement the following sniff-and- then-spoof program. You need two VMs on the same LAN. From VM A, you ping an IP X. This will generate an ICMP echo request packet. If X is alive, the ping program will receive an echo reply, and print out the response. Your sniff-and-then-spoof program runs on VM B, which monitors the LAN through packet sniffing. Whenever it sees an ICMP echo request, regardless of what the target IP address is, your program should immediately send out an echo reply using the packet spoofing technique. Therefore, regard- less of whether machine X is alive or not, the ping program will always receive a reply, indicating that X is alive. You need to write such a program, and include screendumps in your report to show that your program works. Please also attach the code (with adequate amount of comments) in your report.

- Code included in sniffspoof.c (Main components pictured below)
    - Inclusion of spoofer functions into sniffer:
        - Checksum function
        - Make ping packet function
        - Make Ip packet function
        - Send packet fuction
    - A call to send packet was included in in the get packet function of sniffer.c after retrieving ingormation from the ping packet created by the active host

```c
void
got_packet(u_char *args, const struct pcap_pkthdr *header, const u_char *packet)
{

    static int count = 1;                   /* packet counter */

    /* declare pointers to packet headers */
    const struct sniff_ethernet *ethernet;  /* The ethernet header [1] */
    const struct sniff_ip *ip;              /* The IP header */
    const struct sniff_tcp *tcp;            /* The TCP header */
    const char *payload;                    /* Packet payload */

    int size_ip;
    int size_tcp;
    int size_payload;

    printf("\nPacket number %d:\n", count);
    count++;

    /* define ethernet header */
    ethernet = (struct sniff_ethernet*)(packet);

    /* define/compute ip header offset */
    ip = (struct sniff_ip*)(packet + SIZE_ETHERNET);
    size_ip = IP_HL(ip)*4;
    if (size_ip < 20) {
        printf("   * Invalid IP header length: %u bytes\n", size_ip);
        //return;
    }

    // print source and destination IP addresses */
    printf(" Ping From: %s\n", inet_ntoa(ip->ip_src));
    printf(" to: %s\n", inet_ntoa(ip->ip_dst));

    char  *srcip=NULL, *dstip=NULL, *payload1=NULL;
    int   type, opt, longidx = 0;
    arr_t pkt = { .d = NULL, .l = 0 };

    srcip = (char*)&ip->ip_src;
    //printf(" to: %s\n", srcip);

    dstip = (char*)&ip->ip_dst;
    //printf(" to: %s\n", dstip);

    //send the ICMP packet
    snd_pkt(srcip, mk_ip_pkt(dstip,srcip, IPPROTO_ICMP, mk_ping_pkt(pkt)));
```

10.0.2.2 is a nonexistent / inactive host. The ping packet sent was able to be sent successfully however error messages appeared at random? Possibly due to buffer length/ packet header errors ?