Erik Macik

Computer Networks

Assignment 3

10/26/2020

Packet Sniffing

1. Sniffing packets using Scapy
   1. Text

      Description automatically generatedText

      Description automatically generatedIn order to run the provided sample Python code in my virtual machine, I created a python file and ran the command sudo python3 scapy\_sniff.py. I was able to demonstrate that I can indeed receive packets, seen in the left screenshot below. In the second screenshot on the right, I ran the program without root privileges as just python3 scapy\_sniff.py. This yields an “Operation not permitted error,” which probably indicates a protection in place by the operating system.
   2. I set filters on the packets received by passing a string to the optional “filter” parameter. I have added the output of the program for both of these filters to show that it worked. There are no packets captured because I was not in the process of capturing packets at the time.
      1. Text

         Description automatically generatedText

         Description automatically generatedCapture only the ICMP packet: ***"icmp"*** (pictured left)
      2. Capture any TCP Packer that comes from a particular IP and with a destination port number: ***"tcp and src host 8.8.8.8 and dst port 8"*** (pictured right)
2. Writing a C-Program to Sniff Packets
   1. Print out source and IP addresses:
      1. As I see them at the API layer, the sequence of library calls that are essential for sniffer programs are: ***open a pcap session, compile the pcap filter, set the filer, and assign a listener function***. We can then decide to do with the packet we sniffed in the listener function.
      2. A screenshot of a computer

         Description automatically generatedPackets are a very personal resource and contain just about all data that is transferred over a network, which can be pretty scary if someone gets a hold of. It is for this reason that sniffer programs require root privileges before they can be used, otherwise they will be blocked by the operating system. If not run with root privileges, the program crashes at the API call pcap\_compile(/\*\*/).
   2. Filter expressions, screenshots below
      1. (left screenshot) ICMP packets between specific hosts: ***"icmp and dst host 10.0.2.15 and src host 142.250.68.174"*** (IP address of pinging “google.com”, although this IP varies between runs)
      2. Text

         Description automatically generatedGraphical user interface, text

         Description automatically generated(right screenshot) TCP packets with destination port number in range: ***“src portrange <argv[0]>-<argv[1]>”.*** In my program, I enter this port range from command like arguments.
   3. A picture containing diagram

      Description automatically generatedAlthough I did not send any kind of real password to by sniffed through my program, in the screenshot below I highlight where it is possible to see some information about a password including what it might actually be in the screenshot below circled in red. Although it looks sort of like gibberish at the moment we could definitely see some private information here if the information is not well protected or encoded.

**DISCLAIMER:** In order to get my program to work, I downloaded the code from the tcpdump.org website (sniffex.c) and included the file to use the structs explained in the tutorial provided. I also made some minor changes to avoid compiler redefinition errors. Writing these structs directly into my file unfortunately did not work for me.

Code evidences

Scapy\_sniff.py:

#!/usr/bin/python3

from scapy.all import \*

def print\_pkt(pkt):

pkt.show()

pkt = sniff(prn=print\_pkt, filter="icmp")

#pkt = sniff(prn=print\_pkt, filter="tcp and src host 8.8.8.8 and dst port 8")

mySniffer.c

#include <pcap.h>

#include <stdio.h>

// Code provided by tcpdump.org

#include "sniffex.c"

// /\* ethernet headers are always exactly 14 bytes \*/

#define SIZE\_ETHERNET 14

const struct sniff\_ethernet \*ethernet; /\* The ethernet header \*/

const struct sniff\_ip \*ip; /\* The IP header \*/

const struct sniff\_tcp \*tcp; /\* The TCP header \*/

const char \*payload; /\* Packet payload \*/

u\_int size\_ip;

u\_int size\_tcp;

int packetNum = 0;

/\* This function will be invoked by pcap for each captured packet.

We can process each packet inside the function.

\*/

void got\_packet(u\_char \*args, const struct pcap\_pkthdr \*header, const u\_char \*packet){

printf("Got a packet %d\n", packetNum++);

ip = (struct sniff\_ip\*)(packet + SIZE\_ETHERNET);

size\_ip = IP\_HL(ip)\*4;

printf("\tSource: %s\n", inet\_ntoa(ip->ip\_src));

printf("\tDestination: %s\n", inet\_ntoa(ip->ip\_dst));

tcp = (struct sniff\_tcp\*)(packet + SIZE\_ETHERNET + size\_ip);

printf("\tSource port TCP: %d\n", ntohs(tcp->th\_sport));

printf("\tDestination port TCP: %d\n", ntohs(tcp->th\_dport));

payload = (u\_char \*)(packet + SIZE\_ETHERNET + size\_ip + size\_tcp);

int size\_payload = ntohs(ip->ip\_len) - (size\_ip + size\_tcp);

if (size\_payload > 0) {

printf("----Payload (%d bytes):\n", size\_payload);

print\_payload(payload, size\_payload);

printf("\n");

}

}

int main(int argc, char \*argv[]){

pcap\_t \*handle;

char errbuf[PCAP\_ERRBUF\_SIZE];

struct bpf\_program fp;

//char filter\_exp[500];

char filter\_exp[500];

if (argc == 1) {

strcpy(filter\_exp, "icmp and dst host 10.0.2.15 and src host 142.250.68.174");

} else if (argc == 3) {

strcpy(filter\_exp, "src portrange ");

strcat(filter\_exp, argv[1]);

strcat(filter\_exp, "-");

strcat(filter\_exp, argv[2]);

} else {

printf("Expected 0 or 2 arguments.");

exit(0);

}

printf("FILTER %s\n",filter\_exp);

bpf\_u\_int32 net;

// Step 1: Open live pcap session on NIC with name ethx

// you need to change "eth3" to the name

// found on their own machines (using ifconfig).

handle = pcap\_open\_live(/\*"ethx"\*/"enp0s3", BUFSIZ, 1, 1000, errbuf);

// Step 2: Compile filter\_exp into BPF psuedo-code

pcap\_compile(handle, &fp, filter\_exp, 0, net);

pcap\_setfilter(handle, &fp);

// Step 3: Capture packets

pcap\_loop(handle, -1, got\_packet, NULL);

pcap\_close(handle); //Close the handle

return 0;

}

// Note: donâ€™t forget to add "-lpcap" to the compilation command.

// For example: gcc -o sniff sniff.c -lpcap

Reference:

Carstens, T. (n.d.). Programming with pcap (1011599288 778628960 G. Harris, Ed.). Retrieved October 24, 2020, from https://www.tcpdump.org/pcap.html