CIS/CSE644: quizz1a student name: Kevin Martin

1(10): How to get the address in C language? such as how to get address from normal integer variable?

To get the address of a variable in the C language, simply prepend a "&" to the beginning of that variable. In the below example, the address of i is &i.

Int i =10;

Address of i= ?

If you want to use a pointer to hold the value of this address, now first try to declare a pointer(what type?) and use it to save the address of i. Then print out the value of (int i) using this pointer you just declared. (do not need a compiling code, just the major lines.

To use a pointer to hold the value of this address, we will initialize a new variable that is an integer pointer, then assign it to the address of i as shown below:

```
int i = 10;
int *p;
p = &i;
printf("%p", p);
```

2:(10) what does " (int*) &a " mean: ("a" can be any type except integer type).

In this example, the pointer to a is being type casted to an integer pointer. Most likely, another variable is being declared (of type int *) which will now point tot he first byte of &a, only now it will interpret this as an "int" object.

3: (10)

char **buffer**[LENGTH]; Initialize the char buffer of size LENGTH (previously defined)
struct ipheader ***ip** = (struct ipheader *) buffer; Typecast buffer, which was previously declared as a char, as a user-defined structure ipheader and set the new vairable, *ip. Now the ip can be set to whatever it needs to be.
struct udpheader ***udp** = (struct udpheader *) (buffer + sizeof(struct ipheader); After assigning the *ip, must move pointer to the next portion of the buffer, the udp. An additional typecast of struct udpheader * is also done. Again, now the upd can be set ot whatever it needs to be.
Char ***data** = buffer + sizeof(struct ipheader) + sizeof(struct udpheader); Pointer is moved once again, starting with the buffer, moving past the ip and the upd. The data field can also be updated to whatever the user would like. Data does not need to be typecasted again as buffer and data are both chars.

explain above code, what they are doing? also what is type casting in C language?

The above code is initializing a buffer, and setting up three variables (ip, udp, and data) which can then later be manipulated by the user to include a custom (raw socket) packet.

Type casting is the conversion of one data type to another. For example, converting a user defined structure type into a char.

3b(5): what is the return of sizeof(); in what unit?

The sizeof(); function returns the size of whatever **type** the variable that it's being called on is. For example, if it is being called on a char, it will return bytes. However, if it was a type int, it would return 4 bytes multiplied by the size of the variable.

The function will return an answer in bytes.

4:(10) steps you need to think about in order to write sniffer? reasons for each step.(briefly)

First, make sure you are running the program as the super user (root).

Second, create/initialize a raw socket.

Third, must turn on Promiscuous mode on the NIC card. Can be doen with the set socket option (setsockopt).

Fourth, set up a filter. If a filter is not set, than EVERY packet will be caught by the sniffer and could overwhelm the computer. The filter sits between the datalink layer and the TCP/IP stack.

Fifth, enter an infinite loop that continually polls the network and displays those packets that pass through the filter.

4b:(10) what is promiscuous mode? What is normal mode? How do these two modes make difference in sniff tool? What is necessary for sniffing and why?

Promiscuous mode is when the matching of the NIC card address and the address in the packet are ignored, thus taking in everything on the wire.

Normal mode is when the address of the NIC card is compared against the address in the MAC address to determine if that particular packet was meant for that machine.

While originally designed to observe everything on the wire, Promiscuous mode is essentially a sniffing tool.

This was designed to observe traffic on a specific wire, but it is also what an attacker would do for sniffing. It is necessary because the attacker is not the intended recipient and can use this method to maliciously intercept packets.

5:(10) what is raw socket? Why Prof. Du spend time talking about raw socket? What is special about raw socket?

A raw socket is a specific type of socket that overrides the automatically generated socket information (provided by the OS). The user will provide the all the raw data that is needed.

The professor spends time talking about the raw socket because it is an integral part of the sniffing/spoofing architecture. Without it, the user could not manipulate the packet contents to properly execute a spoofing program.

It is special because it gives the user complete flexiblity about the headers of a packet.

5b(5): what is critical in the function: send_raw_ip_packet()? (without it, the packet will not be sent), and explain the reason?

The critical function in the function is the destination information section, specifically the address:

dest_info.sin_addr = ip→iph_destip;

Even though that destination information is already in the buffer, the OS will not actually check inside the buffer and retrieve it. The user must manually include this in order for the packet to be successfully sent.

6:(10) relationship between raw socket programming and pcap library programming? Why do we need the root privilege to run?

We use the pcap library to easily intialize and work with a raw socket. Once we have one created, the functionality to manipulate the contents of the raw socket is very easy.

We need root privilege to run because this is an unusual request: usually we would want the operating system to fill in the details of the socket. With a raw socket, we (the user) want to enter the details.