Quizz1b student name: Kevin Martin

1(10): without option part, normally what will be the size of the header of IP packet? how to get to the data part in IP packet giving a pointer to the beginning of a IP packet? (you need ip header graph.3.2 IP protocol).

Without the option part, the normal size of the header will be 20 octets (8 bits per octet). To get to the data, we look inside the data field and see which type of data is specified (TCP, UDP, etc.). This type will be indicated by the IP header.

2(5) what is TTL in IP header? what is the uses of TTL and how to use it ?

TTL is “time to live”, and it decides how long a packet can live. Because the internet is so complicated, there could be a loop in the routing path. Theoretically, the packet could be stuck in this loop for years.

The TTL does not measure “time”, but rather the number of hops (routers) that the packet will go through. It specifies the maximum number, so if the packet goes through more than this, it will die (indicating that something is most likely wrong with with the routing).

Using Traceroute, we can measure the number of hops it takes a packet to reach our intended destination by sending packets with increasing TTL’s until we get to the destination.

3: (5) is there limit length of IP packet? What is that and the reason?

The total length of an IP packet is 2^{16} (or 65k). The reason for that is that there are only 16 bits available for the total length size in the protocol. This is already a very large packet, so any bigger and it would be better to split into multiple packets.

4(5) What is IP fragmentations? What cause that?

IP fragmentations are IP packets that are split into multiple pieces when being sent. They now become their own IP packets.

They are caused by the physical limitations of the wire (such as ethernet). Defined by the MTU: maximum transmit unit, and varies between different types of physical connections are present.

5(10): among IP fragments for one big IP packet, what part is really fragmented? Among all those ip fragments, what do they have in common?

The part of the big IP packet that is really fragmented is the offset. There is a 3 bit flag indicating whether or not the packet can be fragmented and if there are more fragments coming. Then there is a 13 bit fragment offset which helps the router put the packets in order.

Among the IP fragments, they have in common the same 16 bit identification (ID) number.

6(10) how to design attacks according to the method of fragmentation(2-3 examples)?

1. Tie Up Target’s Resources: this targets a powerful server with lots of processing power. The attacker needs to find an asymetric attack where a little bit of processing power can bring down a much more powerful system. The attacker sends one single fragment that is indicated as the

last piece by setting the offset close to 2^{16}, it causes the server to reserve all 64k of memory for the rest of the packet, even though it will never come.

2. Create a Super Large Packet: the attacker again focuses on the last fragment, specifically the offset. By indicating that the last piece is very large, it will create a buffer overflow (crash).

3. Create Abnormal Situation: test wheter a computer can handle these “unreal” conditions. For example, create overlapping fragments to see if receiver can handle. If the end of the second fragment is less than the end of the first, it will create a negative number. If the sytem uses an unsigned integer, it will flip to a huge number, causing a ccrash.

7(10) what is ICMP stands for? How many purpose of ICMP? What is ping’s purpose?

ICMP stands for Internet Control Messasge Protocol. It has two purposes: control message and error message. Ping’s purpose is to communicate the echo request and echo reply messages.

8: (10)what are the types of ping ICMP in our lab1 (lecture code)? What are they? How to set it ?

The types of ping ICMP in lab1 were Type 0 (echo reply) and Type 8 (echo). They are used to test whether or not a server is “alive”. The host computer will send an echo, and if it successfully reaches and active server, it will return an echo reply. They are set automatically depending on the type of program. For example, explicitly entering the command “ping” to a specific IP address will set the ICMP type to 8. They can also be set deliberatly, as in the case of the spoofing program. The field ICMP.type can be manually set to 0 to spoof a reply.

9: (10) explain what is Spoof ICMP ECHO request in our lab1? what you need to do to achieve it(pseudo code should work here) and how to test your work?

The spoof ICMP echo request sends a fake reply back to the computer initially requesting the response. It can be implemented with the following:

```
a=IP()
a.dst = ‘10.0.2.3’
b=ICMP()
p-a/b
send(p)
```

To test,. By using a third party monitoring software like Wireshark, we could observe that the echo replies were coming from the malicious computer and not from the intended server.

10: (10): what is our snoof in lab1? what do you need to do in order to achieve snoof(action sequence)? what is the difference between spoof in this task and spoof in the previous task? (spoof ICMP Echo request). How to test your spoof?

Snoof from Lab1 is the combination of sniffing and spoofing. The action sequence can be as follows:

```
def spoof(pkt):
        p=copy.deepcopy(pkt[IP])
        p.src=pkt[IP].dst
```

```
p.dst=pkt[IP].src
p[ICMP].type = 0
send(p)
```

Note the bolded line, which sets the response to an ICMP echo reply.

The difference is that the two computers on the same ethernet would be needed, one sending a normal ping request to a remote server, and one running a sniffing AND spoofing program would be needed. To test, again a third party monitoring software like Wireshark could be used to demonstrate where the responses are coming from.

11: (5) What are those lines of codes doing below? In sniff or spoof do you need access source or destination address, why?

The lines of code below are setting up a buffer to store the IP header. In this code, we are recording what the source ipheader is and the destination ipheader, most likely to use in a spoof attack. By using these values, we can fool the sending computer into thinking this is a legitiment response.

```
char Buffer[length];

struct ipheader * ip = (struct ipheader*)buffer ;

ip->iph_sourceip.s_addr = inet.addr(SRC_IP);

ip->iph.destip.s_addr = inet.addr(DES_IP);
```

In sniffing, you need access only to the wire that the computers share in order to observe the packets being sent. In spoofing, you need to know the source and destination so you can create a legitimate looking response.

12(10): how to get TCP header length?? How to move *tcp(beginning of the tcp) to the data part?

In order to get the TCP header length, look at the number specifying it's length and multiply by 4. The integer that indicates its length is specified within the packet, before the checksum and data portions.

To move *tcp to the data part, we simply increment the pointer value to the sequence number of the first item in the data. There is no need to specify the sequence number for all the components of the data as they are consecutive.