Quiz 1b

CSE-644 INTERNET SECURITY

DR. SYED SHAZLI

1/26/2023

Anthony Redamonti
SYRACUSE UNIVERSITY

1(10): Without optional part, normally what will be the size of the header of IP packet? How can you get to the data part in IP packet, given a pointer to the beginning of an IP packet?

Answer: Without the optional part, the size of the header of the IP packet is 20 bytes. To find the data part in the IP packet given the pointer to the beginning of the IP packet, simply increment the pointer by the length of the IP header (20 bytes).

2(5): What is TTL in IP header? How do you use it?

Answer: Time To Live (TTL) is a mechanism inside the IP header that sets the lifetime of the packet. Each time the packet is forwarded by a router, the TTL inside the packet is decremented. If the destination is unreachable with the given TTL, the packet is dropped, and an error message is returned to the sender. TTL ensures that if the destination IP is unreachable, the packet will not be stuck in an infinite loop trying to reach the destination. An attacker can also use it in a traceroute program.

3(5): Is there a limit to the length of IP packet? What is that and why?

Answer: Yes, each IP packet must not exceed 2^16 (65536) bytes of data. The reason is because there is a portion of the IP header that defines the size of the packet. The portion is called the "total length" and it equals the header and payload length in bytes. The portion is 16 bits long, so 2^16 is the maximum packet length.

4(5): What is IP fragmentation? What causes that?

Answer: IP fragmentation is needed because of the maximum transmit unit (MTU). The MTU is the largest packet size in bytes that can be sent over the data link layer (DLL). It is typically 1500 bytes. Therefore, if a packet is 65536 bytes long, it will not fit into one packet because of the 1500-byte MTU. Therefore, fragmentation is needed to divide the packet's payload into multiple packets that fit inside the MTU.

5(10): Consider IP fragments for one big IP packet. What part is really fragmented? What do all those IP fragments have in common?

Answer: The part of the large IP packet that is fragmented is the payload (data). All the fragments will have the same 16-bit identification (ID) number.

6(5): What are those lines of codes doing below? In sniff or spoof do you need access source or destination address, why?

```
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);
```

Answer: The first line creates a buffer used to store the packet. Its size should be the MTU (1500). The second line casts the buffers starting address to an IP structure pointer type. It is useful when populating the struct with the required packet header information. The third and fourth lines populate the source and destination IP addresses of the IP packet header. They use

the inet.addr method to convert the IP addresses from IPv4 (1.2.3.4) notation to a binary data format that can be used as a network address.

In sniff or spoof, the access to the destination or source is not needed. When spoofing an IP address, the attacker can mimic an IP address by altering the source IP in the IP header in the packet sent to the victim. Therefore, it does not have to access the IP address it is impersonating. When sniffing the network, the attacker must open a NIC on the network to sniff for incoming packets. The attacker can sniff any or all packets on the network or can filter to find only packets of a particular type or originating from a particular source IP address.

7(10): How do you get length of TCP header? How can you move *tcp(beginning of the tcp) to the data part?

Answer: To calculate the size of the TCP header, read the data offset field. The data offset field corresponds to the number of 32-bit (4-byte) words in the TCP header. So, if the data offset field is 4, then the TCP header is 4*4 bytes = 16 bytes.

The following line of C code will locate the beginning of the TCP header and cast it to a pointer of type tcpheader.

```
struct tcpheader *tcp = (struct tcpheader *)(packet + size_ethernet + size_ip);
```

The following line of C code will locate the payload (data) portion of the TCP packet.

```
const char* payload = (u_char *)(tcp + size_tcp);
```

The payload resides directly after the TCP header in the packet.