Packet Sniffing and Spoofing

Lecture 04

Instructor: Dr. Cong Pu, Ph.D.

cong.pu@okstate.edu

Acknowledgment: Adapted partially from course materials from Dr. Wenliang Du at Syracuse University, Dr. Fengwei Zhang at Southern University of Science and Technology, and Dr. Steven M. Bellovin at Columbia University.

Sending Spoofed Packet Using Raw Sockets

- One special type of socket provided by most OS allowing app. to have more control: ***<u>raw socket</u>***
- ***<u>raw socket</u>***
  - construct the entire packet in a buffer (e.g., IP header and all of its subsequent headers)
  - give the packet to the socket for sending
  - * *enable user to set arbitrary values for header fields* *
- Two major steps in using ***<u>raw socket</u>***
  1. constructing the packet in a buffer
  2. sending the packet out

Sending Spoofed Packet Using Raw Sockets (cont.)

- Need to construct the entire packet before sending out spoofed packet using raw sockets
  - filling in a buffer with header info. and payload data
  - E.g., constructing ICMP Echo request message with *spoofed src. IP addr.*

```
char buffer[1500];

memset(buffer, 0, 1500);
```

create a buffer

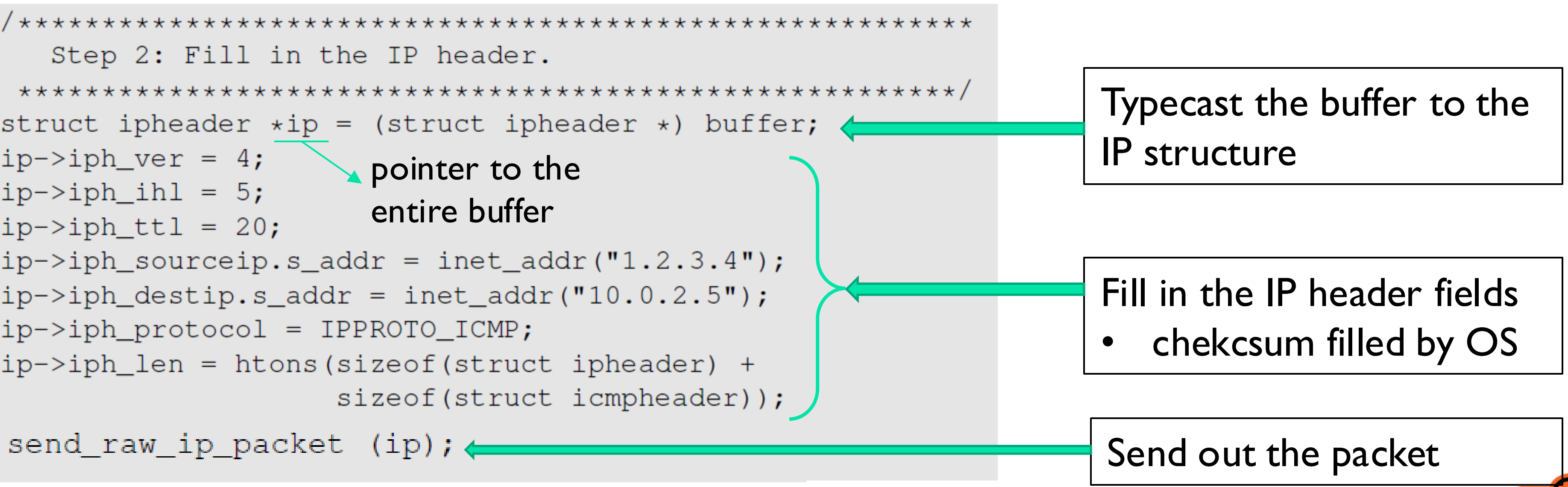
```
/*********************************************************
   Step 1: Fill in the ICMP header.
 ********************************************************/
struct icmpheader *icmp = (struct icmpheader *)
                          (buffer + sizeof(struct ipheader));
icmp->icmp_type = 8; //ICMP Type: 8 is request, 0 is reply.

// Calculate the checksum for integrity
icmp->icmp_chksum = 0;
icmp->icmp_chksum = in_cksum((unsigned short *)icmp,
                              sizeof(struct icmpheader));
```

Find the starting point of the ICMP header, and typecast it to the ICMP structure

Fill in the ICMP header fields
- type
- checksum
- payload (optional)

Sending Spoofed Packet Using Raw Sockets (cont.)

- Need to construct the entire packet before sending out spoofed packet using raw sockets
  - filling in a buffer with header info. and payload data
  - E.g., constructing ICMP Echo request message with spoofed src. IP addr.



```
/*********************************************************
   Step 2: Fill in the IP header.
 *********************************************************/
struct ipheader *ip = (struct ipheader *) buffer;
ip->iph_ver = 4;
ip->iph_ihl = 5;
ip->iph_ttl = 20;
ip->iph_sourceip.s_addr = inet_addr("1.2.3.4");
ip->iph_destip.s_addr = inet_addr("10.0.2.5");
ip->iph_protocol = IPPROTO_ICMP;
ip->iph_len = htons(sizeof(struct ipheader) +
                    sizeof(struct icmpheader));

send_raw_ip_packet (ip);
```

pointer to the entire buffer

Typecast the buffer to the IP structure

Fill in the IP header fields
- chekcsum filled by OS

Send out the packet

Sending Spoofed Packet Using Raw Sockets (cont.)

```
/*************************************************************
  Given an IP packet, send it out using a raw socket.
*************************************************************/
void send_raw_ip_packet(struct ipheader* ip)
{
    struct sockaddr_in dest_info;
    int enable = 1;

    // Step 1: Create a raw network socket.
    int sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);

    // Step 2: Set socket option.
    setsockopt(sock, IPPROTO_IP, IP_HDRINCL,
                     &enable, sizeof(enable));

    // Step 3: Provide needed information about destination.
    dest_info.sin_family = AF_INET;
    dest_info.sin_addr = ip->iph_destip;

    // Step 4: Send the packet out.
    sendto(sock, ip, ntohs(ip->iph_len), 0,
           (struct sockaddr *)&dest_info, sizeof(dest_info));
    close(sock);
}
```

IPv4 (AF_INET6 for IPv6)

raw socket

we supply IP header

passed to OS when sending packet

communication facility

pointer to packet in the buffer

des. IP addr.

packet size

no flag set

* For security reason, only root processes and processes with CAP_NET_RAW capabilities can create raw sockets
- use sudo to run program

Use *setsockopt()* to enable *IP_HDRINCL (header included)* on socket.

For raw socket programming, since the des. info. is already included in the provided IP header, no need to fill all the fields

Since the socket type is raw socket, the system will send out the IP packet as it is.

Constructing UDP Packets

- Constructing UDP packets is similar, except the need of payload

```
memset(buffer, 0, 1500);
struct ipheader *ip = (struct ipheader *) buffer;
struct udpheader *udp = (struct udpheader *) (buffer +
                                      sizeof(struct ipheader));

/*********************************************************
   Step 1: Fill in the UDP data field.
 ********************************************************/
char *data = buffer + sizeof(struct ipheader) +
                      sizeof(struct udpheader);
const char *msg = "Hello Server!\n";
int data_len = strlen(msg);
strncpy (data, msg, data_len);

/*********************************************************
   Step 2: Fill in the UDP header.
 ********************************************************/
udp->udp_sport = htons(12345);
udp->udp_dport = htons(9090);
udp->udp_ulen = htons(sizeof(struct udpheader) + data_len);
udp->udp_sum =  0; /* Many OSes ignore this field, so we do not
                      calculate it. */
```

create a buffer for packet

calculate the offset for the payload

placing data into the payload region inside the buffer

send "Hello Server!" msg to the server

UDP header:
- src port #
- des. Port #
- size
- checksum

Constructing UDP Packets (cont.)

- Constructing UDP packets is similar, except the need of payload

```
/*************************************************************
   Step 3: Fill in the IP header.
 ************************************************************/

...... /* Code omitted here; same as that in Listing 12.6 */
ip->iph_protocol = IPPROTO_UDP; // The value is 17.
ip->iph_len = htons(sizeof(struct ipheader) +
                    sizeof(struct udpheader) + data_len);
```

Testing:

- use the nc command to run a UDP server on 10.0.2.5.
- spoof a UDP packet from another machine.

Output:

- the spoofed UDP packet was received by the server machine.

```
seed@Server(10.0.2.5):$ nc -luv 9090
Connection from 1.2.3.4 port 9090 [udp/*] accepted
Hello Server!
```