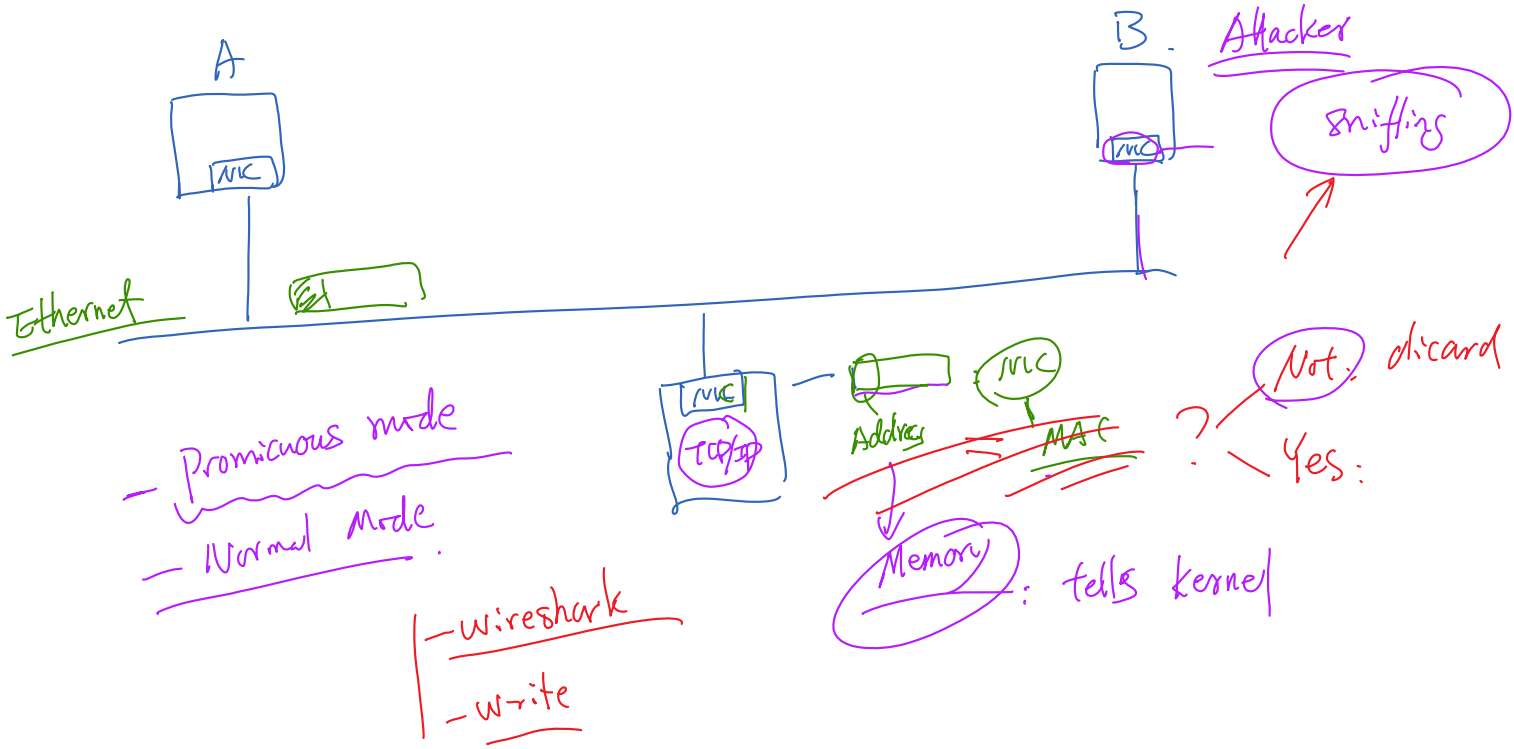
Sniffing and Spoofing

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

Sniffing and Spoofing Overview

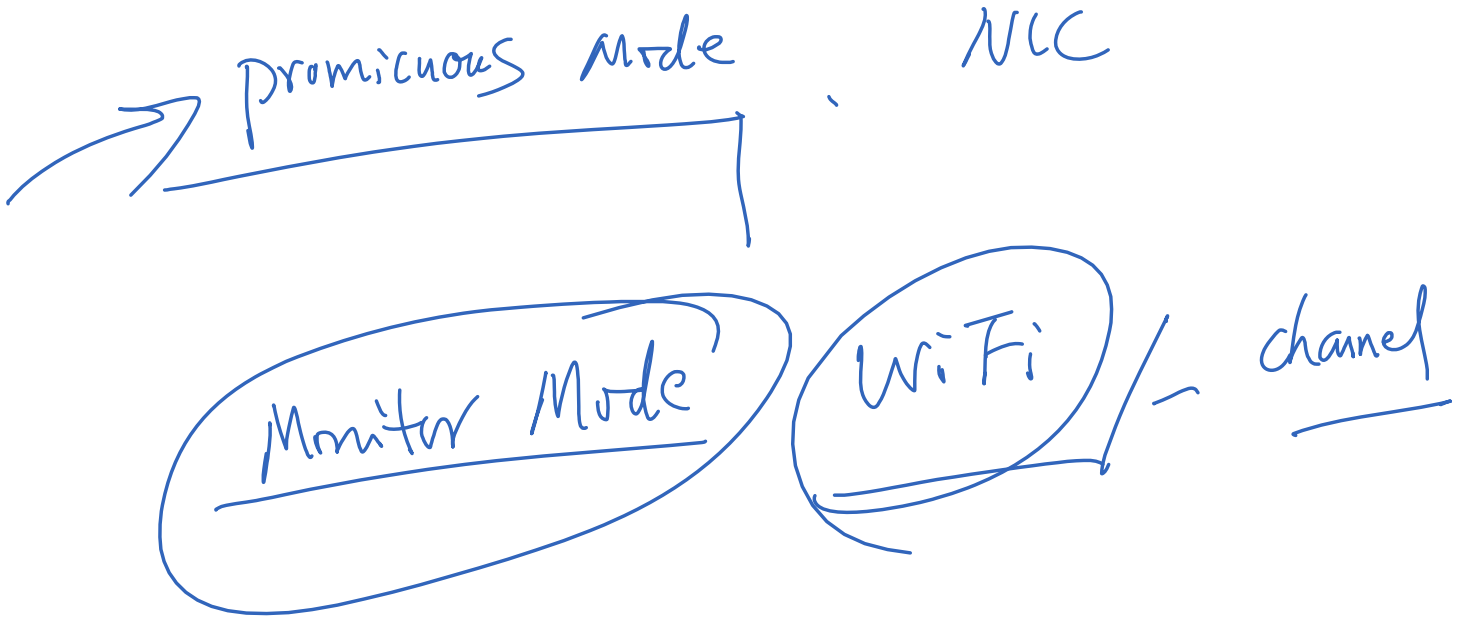


SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

Packet Sniffing

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

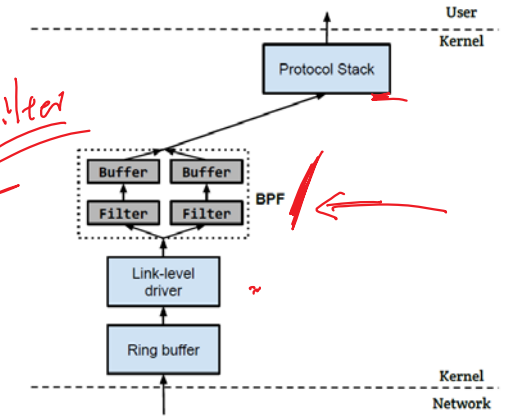
Packet Sniffing



Packet Capturing Using Raw Socket

Listing 3: Packet Capturing using raw socket

```
1  //Creation of the socket.
2  sock_fd = socket( PF_PACKET , SOCK_RAW , htons(ETH_P_ALL));
3  ...
4  //Setting up the device into promiscuous mode and binding the socket to the
       device.
5  struct packet_mreq mr;
6  ...
7  mr.mr_type = PACKET_MR_PROMISC;
8  setsockopt(sock_fd, SOL_PACKET, PACKET_ADD_MEMBERSHIP, &mr, sizeof(mr));
9
10 //Setup the BPF packet filter.
11 ...
12 setsockopt(sock_fd, SOL_SOCKET, SO_ATTACH_FILTER, &bpfcode, sizeof(bpfcode));
13
14 //Capturing data from the socket
15 ...
16 while(1)
17 {
18   recvfrom(sock_fd , buffer , 65536 , 0 , &saddr , &saddr_size);
19 }
```



Capture Packets Using PCAP API

❖ Set up the packet-capturing logic.

```c
int main()
{
  pcap_t *handle;
  char errbuf[PCAP_ERRBUF_SIZE];
  struct bpf_program fp;
  //char filter_exp[] = "port 23";
  char filter_exp[] = "";
  bpf_u_int32 net;

  //Open live pcap session on NIC with name eth0
  handle = pcap_open_live("eth18", BUFSIZ, 1, 1000, errbuf);

  //Compile filter_exp into BPF psuedo-code
  pcap_compile(handle, &fp, filter_exp, 0, net);

  pcap_setfilter(handle, &fp); //Setup BPF code on the socket
  pcap_loop(handle, -1, got_packet, NULL); //Capture packets
  pcap_close(handle);   //Close the handle
  return 0;
}
```

❖ Get a packet and process it.

```c
/********************************************************************************
  This function will be invoked by pcap, whenever a packet is captured.
********************************************************************************/
void got_packet(u_char *args, const struct pcap_pkthdr *header, const u_char *packet)
{
    struct ethheader *eth = (struct ethheader *)packet;
    if (eth->ether_type != ntohs(0x0800))  return; // not an IP packet

    struct ipheader* ip = (struct ipheader*)(packet + SIZE_ETHERNET);
    int ip_header_len = ip->iph_ihl * 4;

    printf("---------------------------------------\n");
    /* print source and destination IP addresses */
    printf("       From: %s\n", inet_ntoa(ip->iph_sourceip));
    printf("         To: %s\n", inet_ntoa(ip->iph_destip));

    /* determine protocol */
    if (ip->iph_protocol ==  IPPROTO_ICMP){
        printf("   Protocol: ICMP\n");
        spoof_icmp_reply(ip);
    }
}
```

Demonstration of the Sniffing Program

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

Packet Spoofing

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

Packet Sending

```c
int main()
{
  // Create socket
  int sockfd = socket(AF_INET, SOCK_DGRAM, 0);

  // Set the destination information
  struct sockaddr_in dest;
  memset(&dest, 0, sizeof(struct sockaddr_in));
  dest.sin_family = AF_INET;
  dest.sin_addr.s_addr = inet_addr("10.0.2.5");
  dest.sin_port = htons(9090);

  // Send data
  char *buffer = "Hello Server!\n";
  sendto(sockfd, buffer, strlen(buffer), 0,
         (struct sockaddr *)&dest, sizeof(dest));

  close(sockfd);
  return 0;
}
```

UDP

UDP

set up destination info.

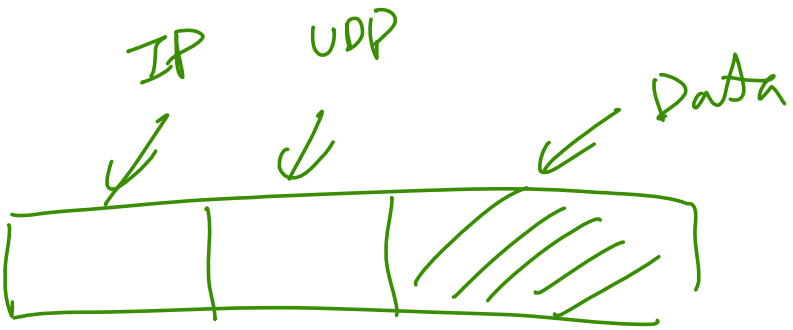
10.0.2.5

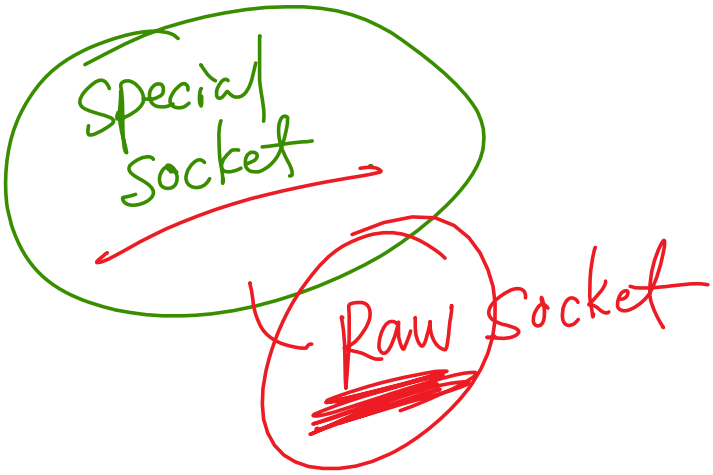
9090

system fills the other field.

SRC

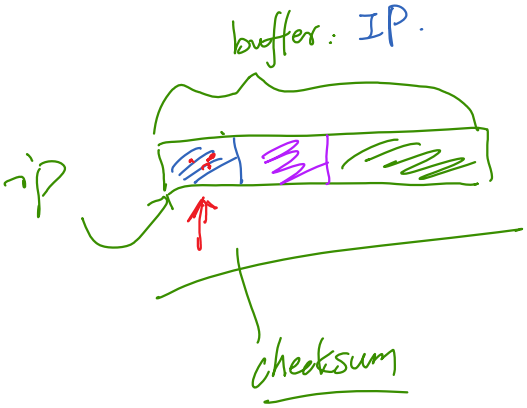
Packet Spoofing





Spoofing IP Packet: Code

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

    // Create a raw network socket, and set its options.
    int sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);
    setsockopt(sock, IPPROTO_IP, IP_HDRINCL, &enable, sizeof(enable));

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

    // Send the packet out.
    printf("Sending spoofed IP packet...\n");
    sendto(sock, ip, ntohs(ip->iph_len), 0, (struct sockaddr *)&dest_info, sizeof(dest_info));
    close(sock);
}
```

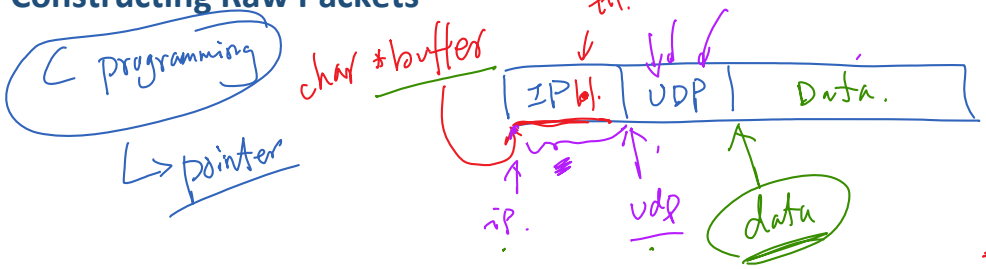


SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

Constructing Raw Packets

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

Constructing Raw Packets



Structure. IP header
UDP header.

Type cast → a structure.

```
/* IP Header */
struct ipheader {
        unsigned char      iph_ihl:4, iph_ver:4; //IP Header length & Version.
        unsigned char      iph_tos; //Type of service
        unsigned short int iph_len; //IP Packet length (Both data and header)
        unsigned short int iph_ident; //Identification
        unsigned short int iph_flag:3, iph_offset:13; //Flags and Fragmentation offset
        unsigned char      iph_ttl; //Time to Live
        unsigned char      iph_protocol; //Type of the upper-level protocol
        unsigned short int iph_chksum; //IP datagram checksum
        struct  in_addr    iph_sourceip; //IP Source address (In network byte order)
        struct  in_addr    iph_destip;//IP Destination address (In network byte order)
};
```

ip->iph_ttl = 20

udp-> [ ] = [ ]

```
char buffer[LENGTH];
struct ipheader *ip = (struct ipheader *) buffer;
struct udpheader *udp = (struct udpheader *) (buffer + sizeof(struct ipheader);
Char *data = buffer + sizeof(struct ipheader) + sizeof(struct udpheader);
```

char * 20 sizeof(char)

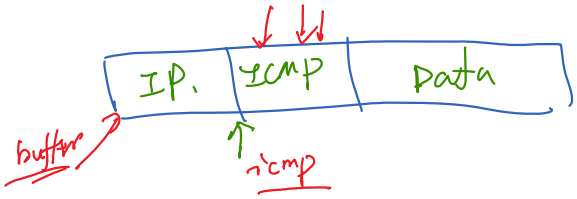
SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

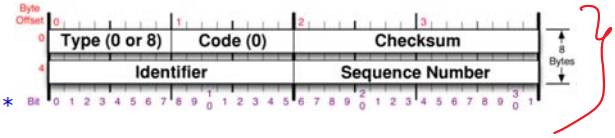
Spoofing Packets: Code and Examples

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

Spoofing ICMP Echo Request

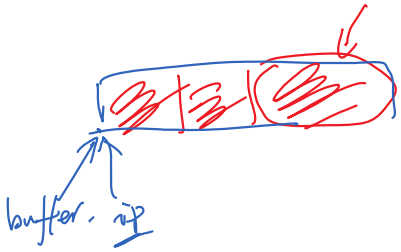
❖ Step 1: Fill in the ICMP header.






```c
/*****************************************************************
  Spoof an ICMP echo request using an arbitrary source IP Address
*****************************************************************/
int main() {
   char buffer[PACKET_LEN];

   memset(buffer, 0, PACKET_LEN);

   /*********************************************************
      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));
```

❖ Step 2: Fill in the IP header.




```c
   /*********************************************************
      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(SRC_IP);
   ip->iph_destip.s_addr = inet_addr(DEST_IP);
   ip->iph_protocol = IPPROTO_ICMP; // The value is 1, representing ICMP.
   ip->iph_len = htons(sizeof(struct ipheader) + sizeof(struct icmpheader));

   // No need to set the following fileds, as they will be set by the system.
   // ip->iph_chksum = ...
```

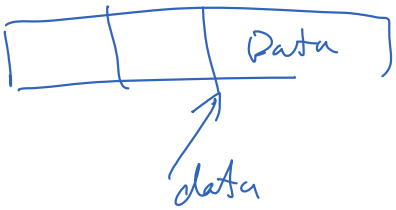
❖ Step 3: Send the raw IP packet.

```c
   /*********************************************************
      Step 3: Finally, send the spoofed packet
    *********************************************************/
   send_raw_ip_packet (ip);
```

Spoofing UDP Packet

❖ The code

```
char buffer[PACKET_LEN];

memset(buffer, 0, PACKET_LEN);
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 UDP\n";
int data_len = strlen(msg);
strncpy (data, msg, data_len);


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



❖ Test it

- On another machine (e.g., 10.0.2.16)

  ```
  $ nc -l -u -v 9090
  ```

- Send a spoofed UDP packet to 10.0.2.16:9090

Spoofing TCP Packet

❖ Construct TCP data and header.

```
int main() {
   char buffer[PACKET_LEN];

   srand(time(0)); // We need to use random numbers for some attacks

   memset(buffer, 0, PACKET_LEN);

   struct ipheader *ip = (struct ipheader *) buffer;
   struct tcpheader *tcp = (struct tcpheader *) (buffer + sizeof(struct ipheader));

   /*********************************************************
      Step 1: Fill in the TCP data field.
    *********************************************************/
   char *data = buffer + sizeof(struct ipheader) + sizeof(struct tcpheader);
   const char *msg = TCP_DATA;
   int data_len = strlen(msg);
   strncpy (data, msg, data_len);


   /*********************************************************
      Step 2: Fill in the TCP header.
    *********************************************************/
   tcp->tcp_sport = htons(SRC_PORT);
   tcp->tcp_dport = htons(DEST_PORT);
   tcp->tcp_seq = htonl(SEQ_NUM);
   tcp->tcp_offx2 = 0x50;
   tcp->tcp_flags = 0x00;
   tcp->tcp_win =  htons(20000);
   tcp->tcp_sum =  0;
```

Source port								Destination port
Sequence number								
Acknowledgment number								
TCP header length		URG	ACK	PSH	RST	SYN	FIN	Window size
Checksum								Urgent pointer

❖ Construct IP header and compute TCP checksum.

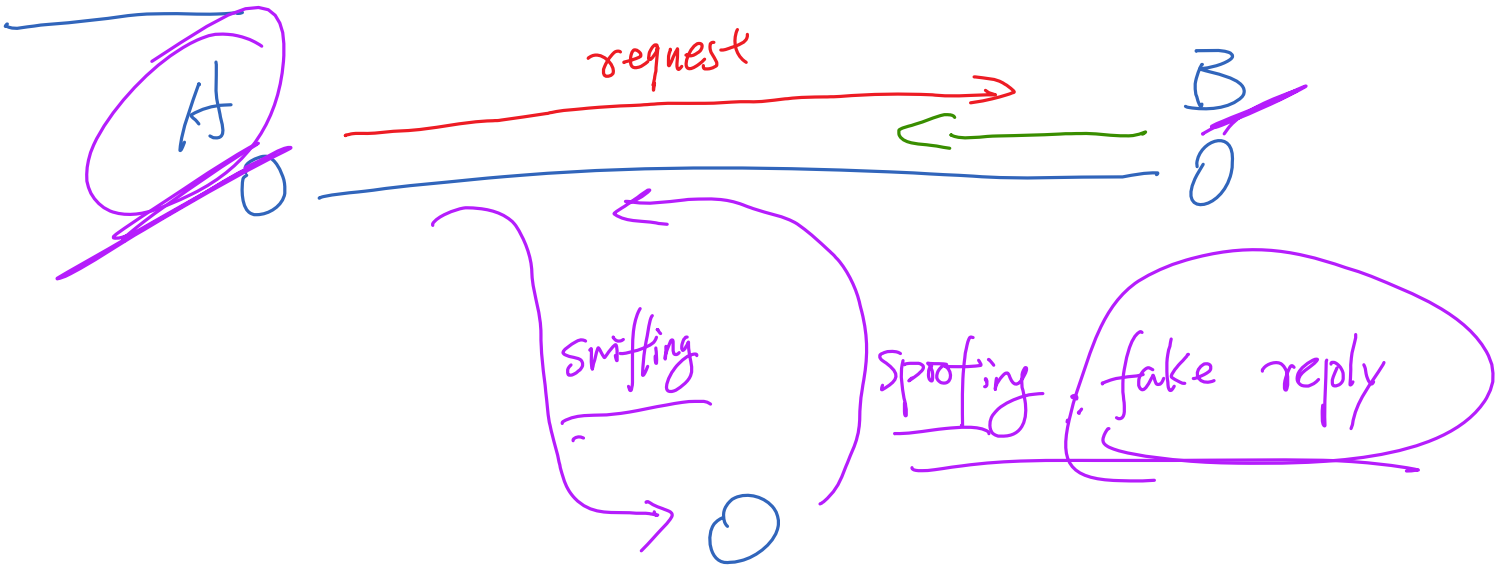
```
   /*********************************************************
      Step 3: Fill in the IP header.
    *********************************************************/
   ip->iph_ver = 4;   // Version (IPV4)
   ip->iph_ihl = 5;   // Header length
   ip->iph_ttl = 20;  // Time to live
   //  ip->iph_sourceip.s_addr = rand(); // Use a random IP address
   ip->iph_sourceip.s_addr = inet_addr(SRC_IP); // Source IP
   ip->iph_destip.s_addr = inet_addr(DEST_IP);  // Dest IP
   ip->iph_protocol = IPPROTO_TCP; // The value is 6.
   ip->iph_len = htons(sizeof(struct ipheader) + sizeof(struct tcpheader) + data_len);

   // Calculate tcp checksum here, as the checksum includes some part of the IP header
   tcp->tcp_sum = calculate_tcp_checksum(ip, data_len);

   // No need to fill in the following fileds, as they will be set by the system.
   // ip->iph_chksum = ...
```

SYRACUSE
UNIVERSITY
ENGINEERING
& COMPUTER
SCIENCE

Sniffing and Spoofing: Code and Examples

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

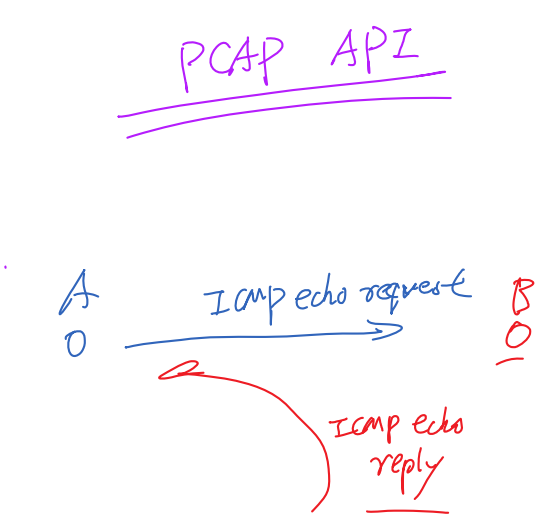
Snoofing: Sniffing and Spoofing



Snoofing ICMP Echo Request/Reply Messages

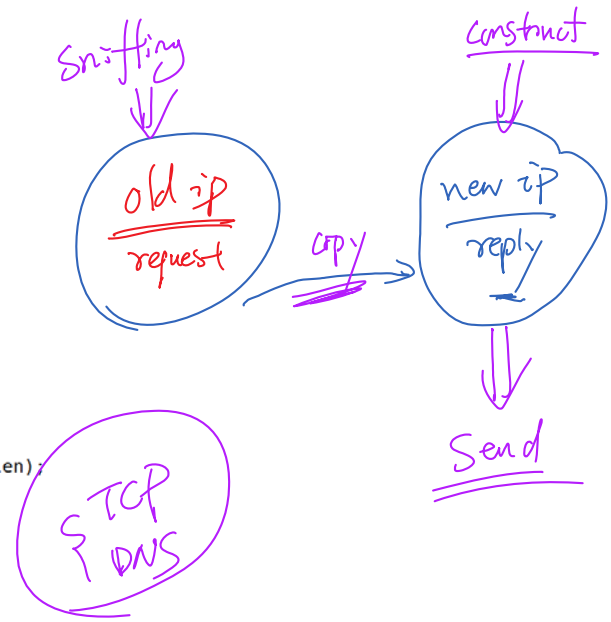
❖ Sniffing the ICMP request

```
/*********************************************************************
  This function will be invoked by pcap, whenever a packet is captured.
*********************************************************************/
void got_packet(u_char *args, const struct pcap_pkthdr *header, const u_char *packet)
{
    struct ethheader *eth = (struct ethheader *)packet;
    if (eth->ether_type != ntohs(0x0800))  return; // not an IP packet

    struct ipheader* ip = (struct ipheader*)(packet + SIZE_ETHERNET);
    int ip_header_len = ip->iph_ihl * 4;

    printf("-----------------------------------\n");
    /* print source and destination IP addresses */
    printf("       From: %s\n", inet_ntoa(ip->iph_sourceip));
    printf("         To: %s\n", inet_ntoa(ip->iph_destip));

    /* determine protocol */
    if (ip->iph_protocol ==  IPPROTO_ICMP){
         printf("   Protocol: ICMP\n");
         spoof_icmp_reply(ip);
    }
}
```

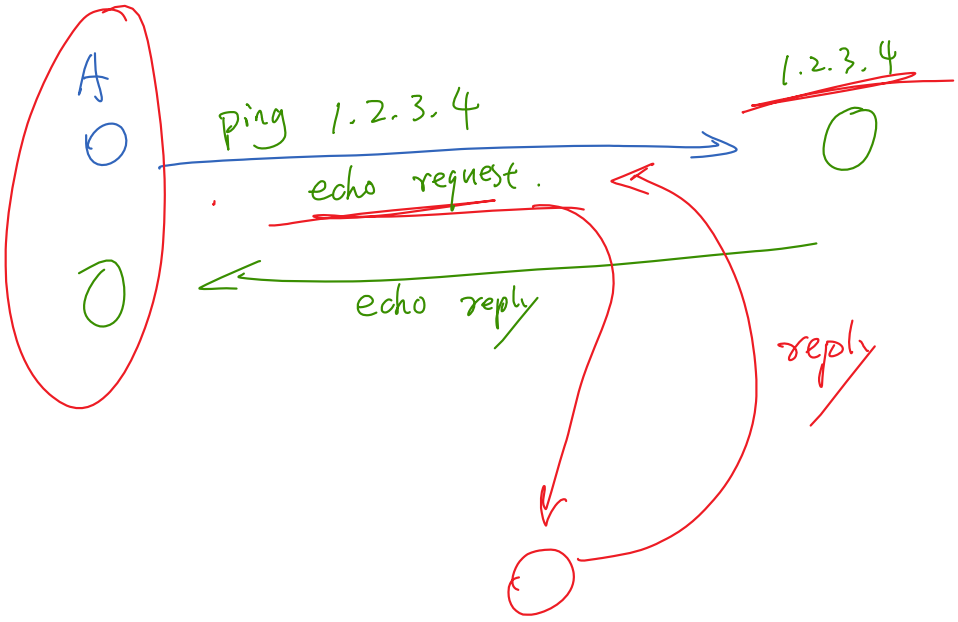


❖ Spoofing the ICMP reply

```
/*********************************************************************
  Given a captured ICMP echo request packet, construct a spoofed ICMP
  echo reply, which includes IP + ICMP (there is no data).
*********************************************************************/
void spoof_icmp_reply(struct ipheader* ip)
{
    int ip_header_len = ip->iph_ihl * 4;
    const char buffer[BUFSIZE];

    struct icmpheader* icmp = (struct icmpheader *) ((u_char *)ip + ip_header_len);
    if(icmp->icmp_type!=8) { // only process ICMP echo request
        printf("Not an echo Request\n");
        return;
    }

    // make a copy from original packet to buffer(faked packet)
    memset((char*)buffer, 0, BUFSIZE);
    memcpy((char*)buffer, ip, ntohs(ip->iph_len));
    struct ipheader   * newip    = (struct ipheader *) buffer;
    struct icmpheader * newicmp = (struct icmpheader *) ((u_char *)buffer + ip_header_len);

    // Construct IP: swap src and dest in faked ICMP packet
    newip->iph_sourceip = ip->iph_destip;
    newip->iph_destip = ip->iph_sourceip;
    newip->iph_ttl = 20;
    newip->iph_protocol = IPPROTO_ICMP;

    //Fill in all the needed ICMP header information.
    //ICMP Type: 8 is request, 0 is reply.
    newicmp->icmp_type = 0;

    //Calculate the checksum for integrity. ICMP checksum includes the data
    newicmp->icmp_chksum = 0; // Set it to zero first
    newicmp->icmp_chksum = in_cksum((unsigned short *)newicmp,
                                    ntohs(ip->iph_len) - ip_header_len);

    send_raw_ip_packet(newip);
}
```

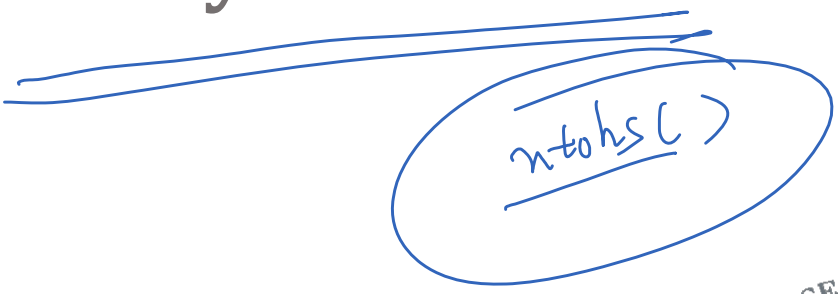


Demonstration of ICMP Snoofing



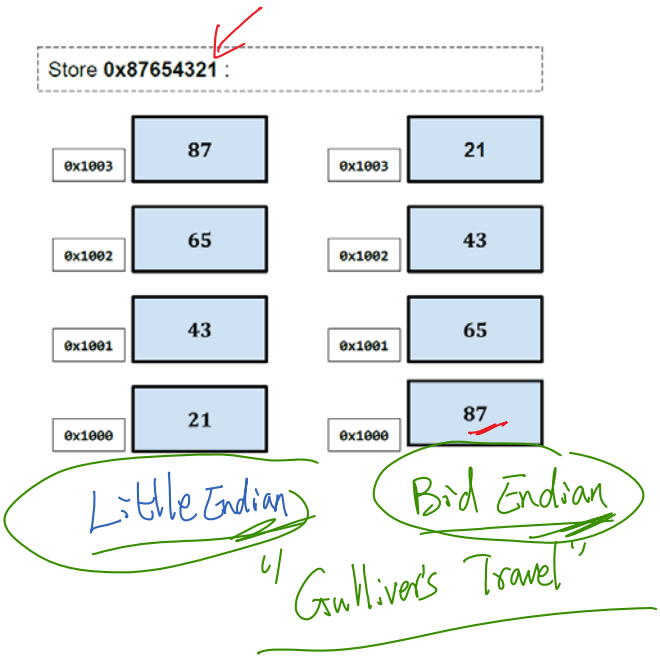
SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

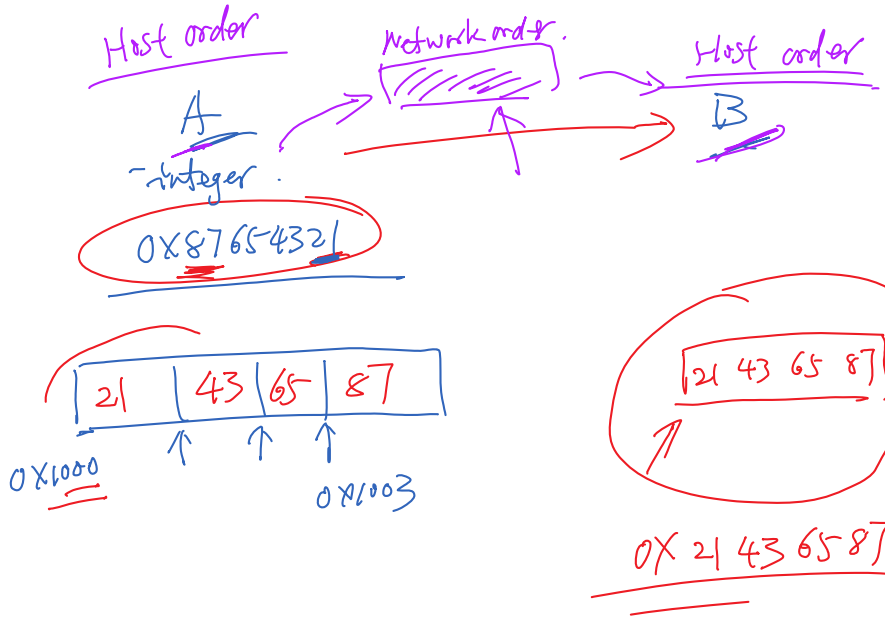
Byte Order



SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

Byte Order


Store 0x87654321 :
0x1003
87
0x1002
65
0x1001
43
0x1000
21
0x1003
21
0x1002
43
0x1001
65
0x1000
87
Little Endian
Bid Endian
Gulliver's Travel


Host order
A
integer
0X87654321
Network order
Host order
B
21 | 43 | 65 | 87
0X1000
0X1003
21 43 65 87
0X 21 43 65 87

Byte-Order Conversion

Macro	Description	Functionality
`htons()`	Host to Network Short	Used to convert unsigned short integer from Host byte-order to Network byte-order
`htonl()`	Host to Network Long	Used to convert unsigned integer from Host byte-order to Network byte-order
`ntohs()`	Network to Host Short	Used to convert unsigned short integer from Network byte-order to Host byte-order
`ntohl()`	Network to Host Long	Used to convert unsigned integer from Network byte-order to Host byte-order

h: host
n. network.

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

Summary

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

Summary

- Packet sniffing using pcap library
- Packet spoofing using raw socket
- Sniffing and spoofing
- Byte order

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE