Packet Sniffing and Spoofing

Lecture 02

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.

Packet Sniffing

- ***Packet sniffing***: *capturing* live data as they flow across network
  1. understand network characteristics (administrator)
  2. diagnose faulty networks and configurations (administrator)
  3. reconnaissance and exploitation (attackers)

Packet Sniffing

- ***Packet sniffing***: capturing live data as they flow across network

1. understand network characteristics (administrator)
   - network traffic analysis
     - bandwidth usage; traffic pattern; protocol distribution
   - performance monitoring
     - latency; packet loss; retransmissions
   - etc.
2. diagnose faulty networks and configurations (administrator)
3. reconnaissance and exploitation (attackers)

Packet Sniffing

- ***Packet sniffing***: *capturing* live data as they flow across network

1. understand network characteristics (administrator)
2. diagnose faulty networks and configurations (administrator)
   - identifying connectivity issues
     - dropped packets; delay; route tracing
   - Etc.
3. reconnaissance and exploitation (attackers)
   - reconnaissance
     - gathering information; identifying services; unencrypted data
   - exploitation
     - session hijacking; man-in-the-middle attack; password sniffing

Packet Sniffing

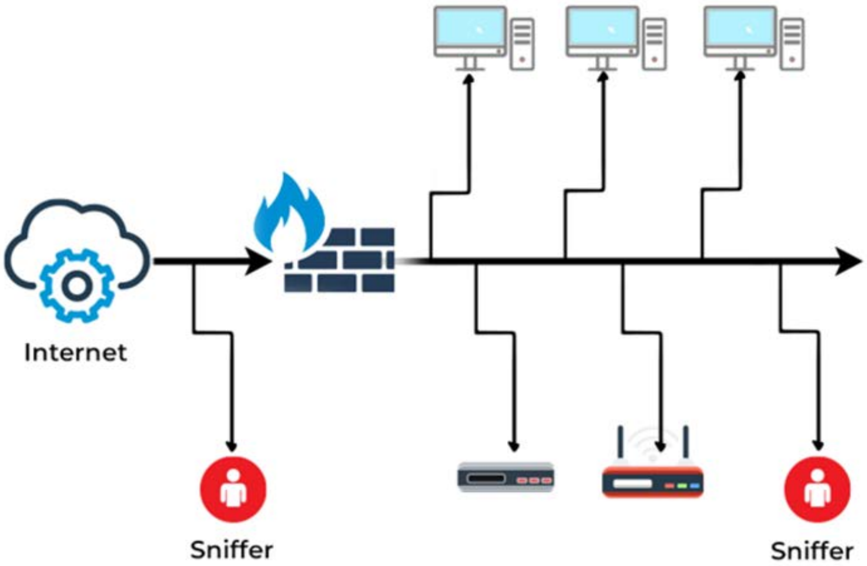
- ***Packet sniffing***: *capturing* live data as they flow across network

1. understand network characteristics (administrator)
2. diagnose faulty networks and configurations (administrator)
   - identifying connectivity issues
     - dropped packets; delay; route tracing
   - Etc.
3. reconnaissance and exploitation (attackers)
   - reconnaissance
     - gathering information; identifying services; unencrypted data
   - exploitation
     - session hijacking; man-in-the-middle attack; password sniffing

Packet Sniffing

- ***Packet sniffing***: *capturing* live data as they flow across network
  1. understand network characteristics (administrator)
  2. diagnose faulty networks and configurations (administrator)
  3. reconnaissance and exploitation (attackers)

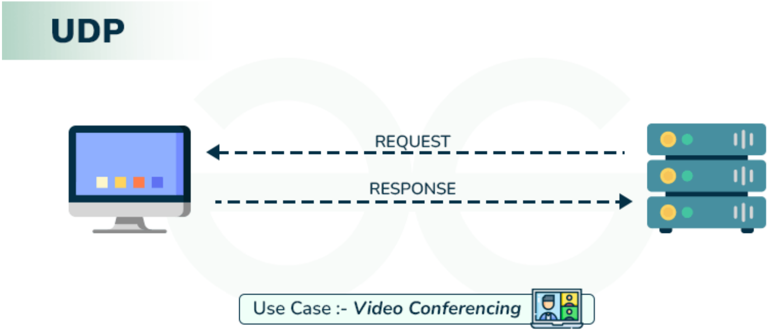
- packet sniffing tools
  ||
  packet sniffers



Receiving Packets Using Sockets

- How program normally receiving packets?
  - A UDP server program:
    1. *Socket Creation*: The program creates a network socket, which serves as an endpoint for sending and receiving data.
    2. *Binding*: The socket is bound to a specific IP address and port number, enabling the program to listen for incoming packets on that address and port.
    3. *Listening*: The program puts the socket into listening mode, waiting for incoming connections or packets.
    4. *Receiving Packets*: When packets arrive, the program receives them through the socket using functions.

Receiving Packets Using Sockets (udp_server.c)



- A UDP server program

create socket

```
// Step ①
int sock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
```

return socket descriptor

socket type (e.g., datagram socket)

protocol type (e.g., UDP)

protocol family (e.g., IPv4)

provide information about server

```
// Step ②
memset((char *) &server, 0, sizeof(server));
server.sin_family = AF_INET;
server.sin_addr.s_addr = htonl(INADDR_ANY);
server.sin_port = htons(9090);

if (bind(sock, (struct sockaddr *) &server, sizeof(server)) < 0)
    error("ERROR on binding");
```

fill a block of memory with a particular value

assigns a local protocol address (IP + port #) to a socket

receive packets

```
// Step ③
while (1) {
    bzero(buf, 1500);
    recvfrom(sock, buf, 1500-1, 0,
               (struct sockaddr *) &client, &clientlen);
    printf("%s\n", buf);
}
```

erase data in buf (1,500 bytes)

- ref.: https://www.tutorialspoint.com/unix_sockets/socket_core_functions.htm

Packet Sniffing Using Raw Sockets

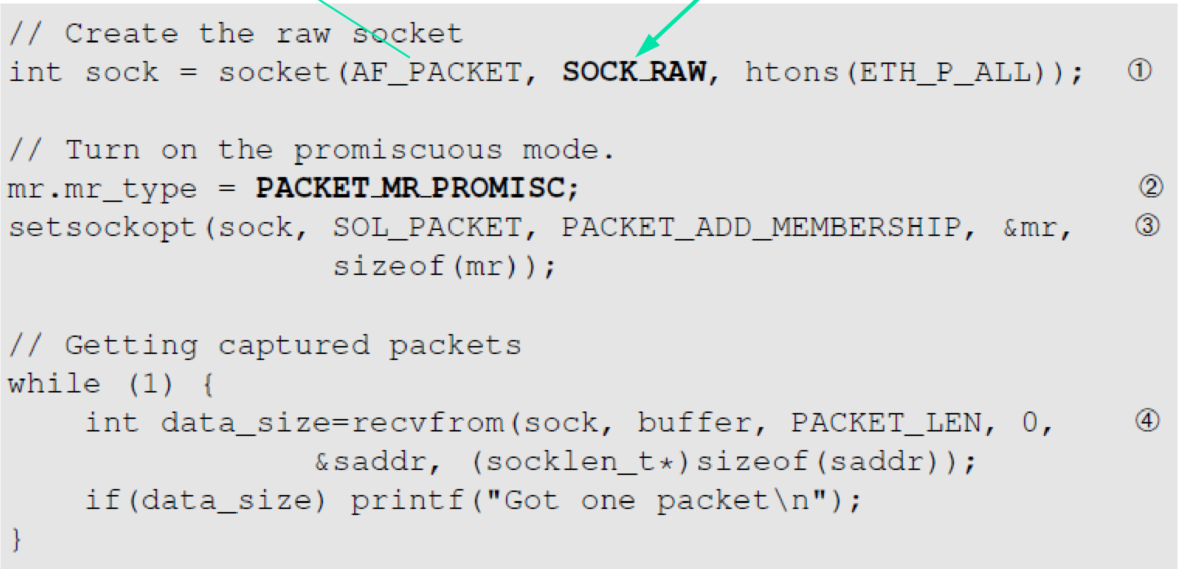
- "***Issue***" in the previous program: receiving packets that are *intended* for it
  - if the *des. IP addr.* or the *des. port #* is ***not matching***, no packets are captured
- What we want: capturing ***ALL*** packeting flowing on the cable, regardless of *des. IP addr.* or *des. port #*
  - ***raw socket***
    - allows access to the underlying transport provider
    - allows user to send and obtain packets of information from the network without interacting with OS

Packet Sniffing Using Raw Sockets

- Packet capture using ***raw socket***
  - other than setting up raw socket, the rest of the program is similar to normal socket program

protocol family (e.g., IPv4)

creating a ***raw socket***



```
// Create the raw socket
int sock = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_ALL));   ①

// Turn on the promiscuous mode.
mr.mr_type = PACKET_MR_PROMISC;                             ②
setsockopt(sock, SOL_PACKET, PACKET_ADD_MEMBERSHIP, &mr,    ③
                 sizeof(mr));

// Getting captured packets
while (1) {
    int data_size=recvfrom(sock, buffer, PACKET_LEN, 0,     ④
                &saddr, (socklen_t*)sizeof(saddr));
    if(data_size) printf("Got one packet\n");
}
```

- normal socket

  kernel receives packet
  ↓
  pass packet via protocol stack
  ↓
  pass to applications

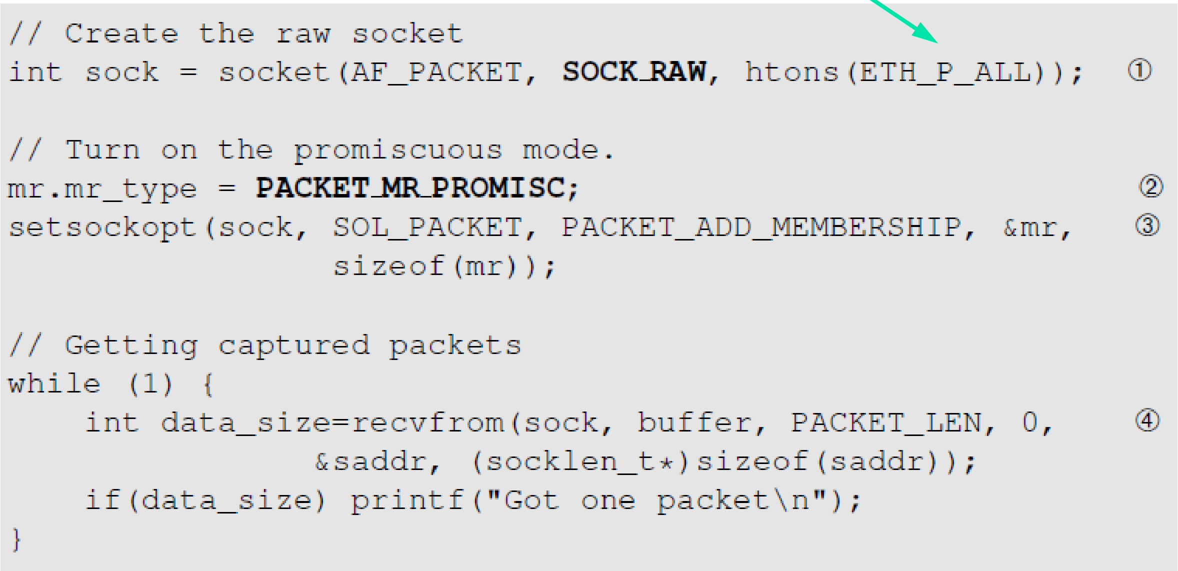
- raw socket

  kernel receives packet
  before passing to protocol stack ↓
  pass a ***copy*** of packet to socket

Packet Sniffing Using Raw Sockets

- Packet capture using ***raw socket***
  - other than setting up raw socket, the rest of the program is similar to normal socket program

capture all types of packets (or all protocols)

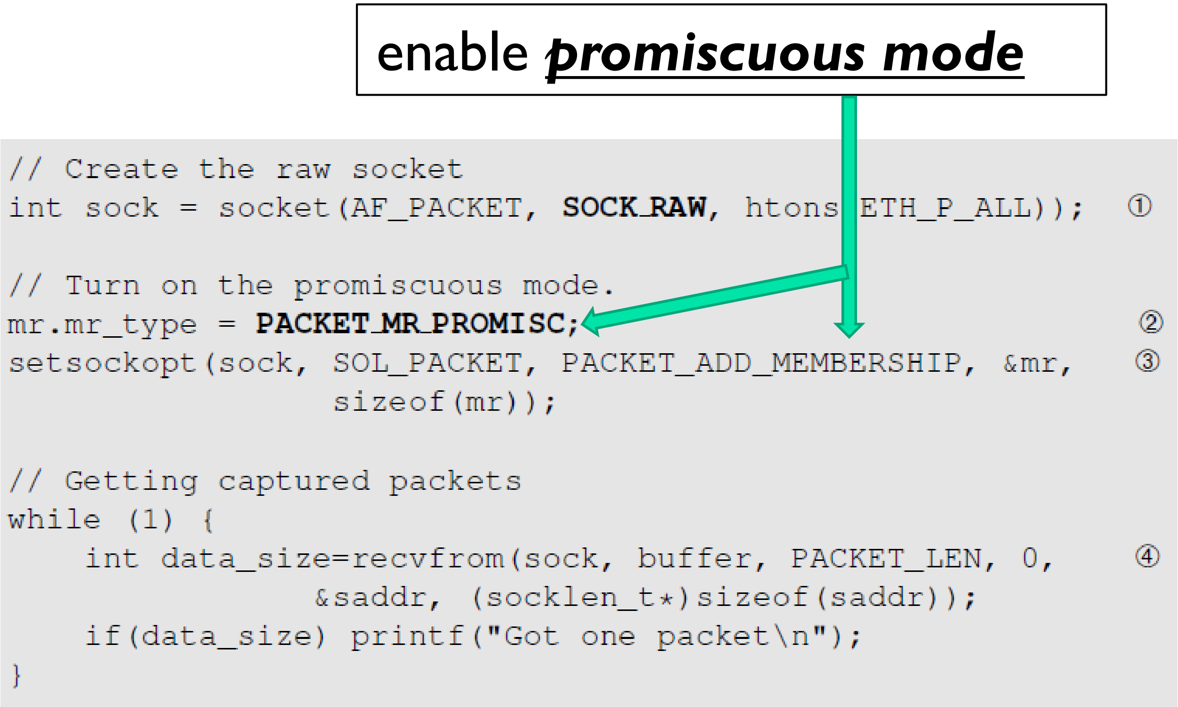


```
// Create the raw socket
int sock = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_ALL));   ①

// Turn on the promiscuous mode.
mr.mr_type = PACKET_MR_PROMISC;                              ②
setsockopt(sock, SOL_PACKET, PACKET_ADD_MEMBERSHIP, &mr,     ③
                 sizeof(mr));

// Getting captured packets
while (1) {
    int data_size=recvfrom(sock, buffer, PACKET_LEN, 0,      ④
                &saddr, (socklen_t*)sizeof(saddr));
    if(data_size) printf("Got one packet\n");
}
```

- for raw socket, need to specify the type of packets to receive
- protocol is specified the third arg of socket()
- *htons*(ETH_P_ALL)
  - packets of all protocols should be passed to socket
- *htons*(ETH_P_IP)
  - only IP packets will be passed to socket

Packet Sniffing Using Raw Sockets

- Packet capture using ***raw socket***
  - other than setting up raw socket, the rest of the program is similar to normal socket program



enable ***promiscuous mode***

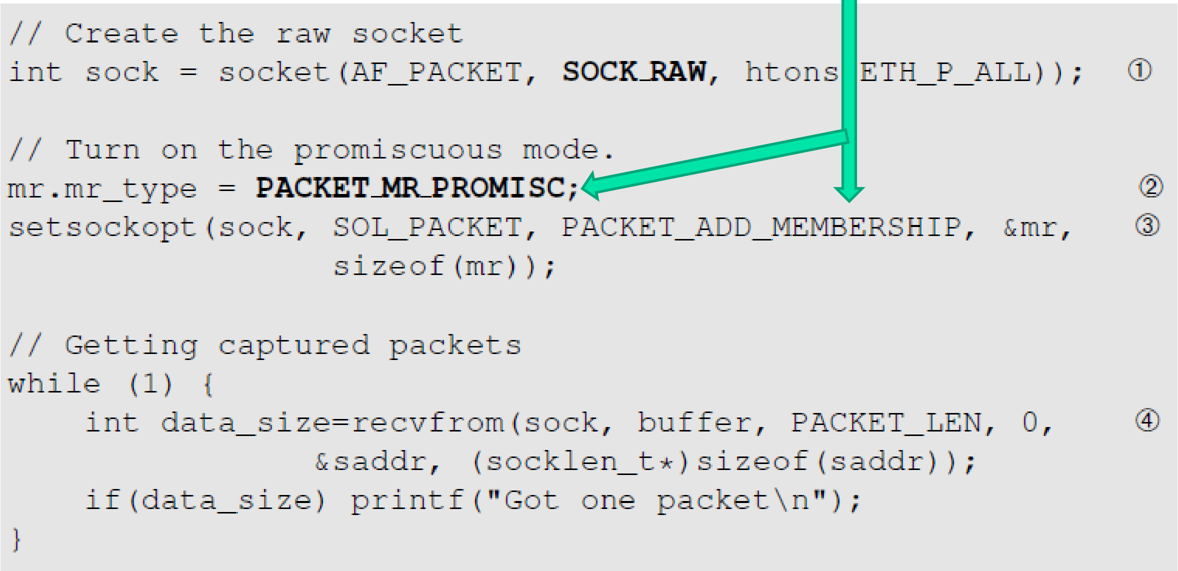
```
// Create the raw socket
int sock = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_ALL));   ①

// Turn on the promiscuous mode.
mr.mr_type = PACKET_MR_PROMISC;                              ②
setsockopt(sock, SOL_PACKET, PACKET_ADD_MEMBERSHIP, &mr,     ③
                 sizeof(mr));

// Getting captured packets
while (1) {
    int data_size=recvfrom(sock, buffer, PACKET_LEN, 0,      ④
                 &saddr, (socklen_t*)sizeof(saddr));
    if(data_size) printf("Got one packet\n");
}
```

- get all packets coming to computer
- but if packets are not destined for us
  - cannot be captured
- turn on *promiscuous mode*
  - let in all packets on network
  - once they are in, we can get copy

Packet Sniffing Using Raw Sockets

- Packet capture using ***raw socket***
  - other than setting up raw socket, the rest of the program is similar to normal socket program

enable ***promiscuous mode***



```
// Create the raw socket
int sock = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_ALL));    ①

// Turn on the promiscuous mode.
mr.mr_type = PACKET_MR_PROMISC;                               ②
setsockopt(sock, SOL_PACKET, PACKET_ADD_MEMBERSHIP, &mr,      ③
                 sizeof(mr));

// Getting captured packets
while (1) {
    int data_size=recvfrom(sock, buffer, PACKET_LEN, 0,       ④
                 &saddr, (socklen_t*)sizeof(saddr));
    if(data_size) printf("Got one packet\n");
}
```

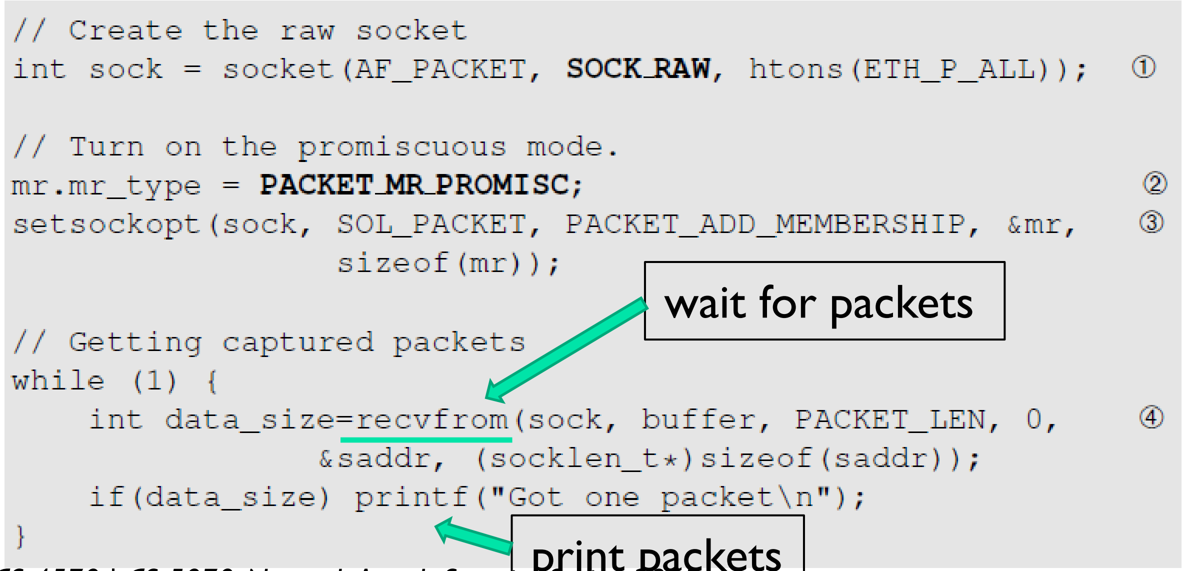
- PACKET_MR_PROMISC
  - enables receiving all packets on a shared medium (often known as "promiscuous mode")
- PACKET_ADD_MEMBERSHIP
  - to receive all frames, regardless of destination

Packet Sniffing Using Raw Sockets

- Packet capture using ***raw socket***
  - other than setting up raw socket, the rest of the program is similar to normal socket program

setsocketopt(): set up filters

- SO_ATTACH_FILTER option (for BPF)



```
// Create the raw socket
int sock = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_ALL));   ①

// Turn on the promiscuous mode.
mr.mr_type = PACKET_MR_PROMISC;                               ②
setsockopt(sock, SOL_PACKET, PACKET_ADD_MEMBERSHIP, &mr,      ③
           sizeof(mr));

// Getting captured packets
while (1) {
    int data_size=recvfrom(sock, buffer, PACKET_LEN, 0,       ④
                 &saddr, (socklen_t*)sizeof(saddr));
    if(data_size) printf("Got one packet\n");
}
```

wait for packets

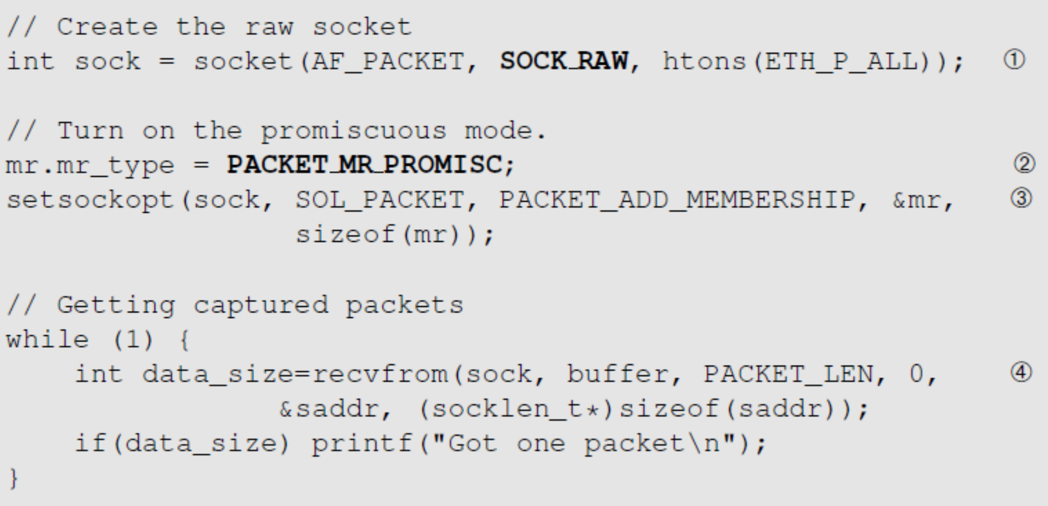
print packets

- setsocketopt():
  - set options for a socket
  - allows you to control various socket behaviors

```
int setsockopt(
   int s, // socket descriptor
   int level, // socket level
   int optname,
   char *optval,     } // socket option
   int optlen
)
```

Packet Sniffing Using Raw Sockets (sniff_raw.c)

- Summary: four major steps
  1. creating raw socket
  2. choose protocol
  3. enable promiscuous mode
  4. wait for packets



```
// Create the raw socket
int sock = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_ALL));   ①

// Turn on the promiscuous mode.
mr.mr_type = PACKET_MR_PROMISC;                              ②
setsockopt(sock, SOL_PACKET, PACKET_ADD_MEMBERSHIP, &mr,     ③
              sizeof(mr));

// Getting captured packets
while (1) {
    int data_size=recvfrom(sock, buffer, PACKET_LEN, 0,      ④
                   &saddr, (socklen_t*)sizeof(saddr));
    if(data_size) printf("Got one packet\n");
}
```

Note:

- First, create a socket using a special socket type called SCOK_RAW.
- Second, specify what type of packet (i.e., ETH_P_ALL, ETH_P_IP) to receive.
- Third, turn on the promiscuous mode (i.e., PACKET_ADD_MEMBERSHIP, PACKET_MR_PROMISC) on the network interface card to let in all packets on the network.
- Fourth, wait for packets to arrive using recvfrom(…)

Packet Sniffing Using pcap API

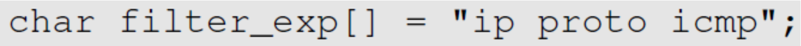
- ***pcap*** (**pa**cket **cap**ture) API provides *platform-independent interface* for accessing packets
  - specify *filtering rules* using Boolean expr.
  - translate Boolean expr. to BPF pseudo-code (used by kernel)
  - pcap API is implemented in
    - Unix as libpcap
    - Windows as WinPcap
    - Linux using raw sockets
  - *ref.*: https://www.tcpdump.org/pcap.html

Packet Sniffing Using pcap API (cont.)

ICMP (Internet Control Message Protocol) packets:

- used for network diagnostics and error-reporting

- E.g., capturing all ICMP packets using pcap
  - *goal*: print out "Got a packet" msg. once a packet is captured

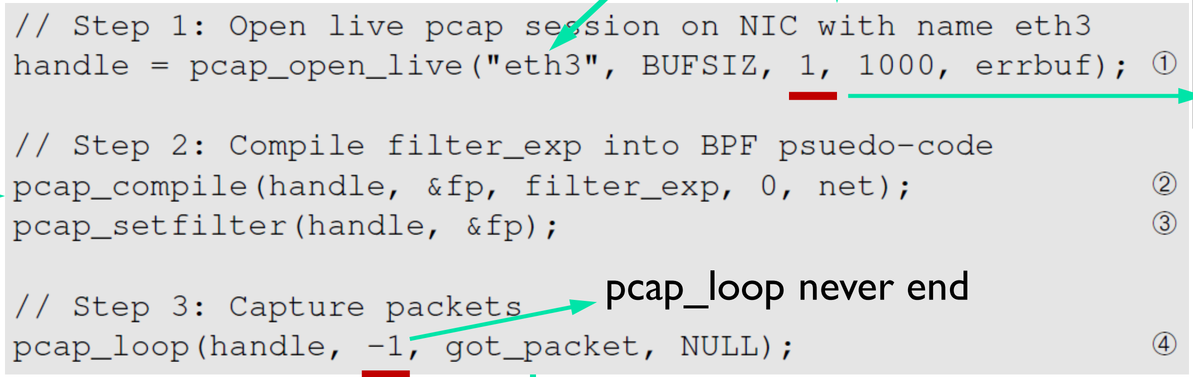


```
char filter_exp[] = "ip proto icmp";
```

network device name (*ifconfig* command to find out)

initialize raw socket, set NIC into promiscuous mode.

filter

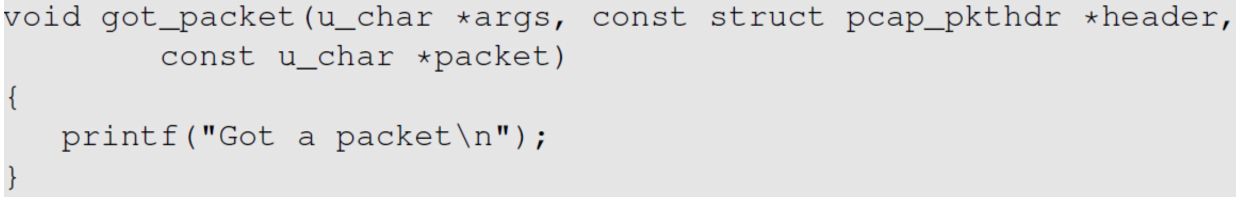


```
// Step 1: Open live pcap session on NIC with name eth3
handle = pcap_open_live("eth3", BUFSIZ, 1, 1000, errbuf);  ①

// Step 2: Compile filter_exp into BPF psuedo-code
pcap_compile(handle, &fp, filter_exp, 0, net);             ②
pcap_setfilter(handle, &fp);                               ③

// Step 3: Capture packets
pcap_loop(handle, -1, got_packet, NULL);                   ④
```

pcap_loop never end

char filter_exp[] = "ip proto icmp";

invoke this function for every captured packet



```
void got_packet(u_char *args, const struct pcap_pkthdr *header,
        const u_char *packet)
{
   printf("Got a packet\n");
}
```

Packet Sniffing Using pcap API (cont.)

- E.g., capturing all ICMP packets using pcap
  - *goal*: print out "Got a packet" msg. once a packet is captured
  - three steps:
    1. Open live pcap session: pcap_open_live(…)
       - initialize raw socket; set network device (e.g., 'enp0s3') into promiscuous mode (i.e., '1'); binds socket to the card using setsockopt(…)
    2. Set the filter: pcap_compile(…) and pcap_setfilter(…)
       - compile the specified filter exp.; set the BPF filter on the socket
    3. Capture packets: pcap_loop()
       - enter the main execution loop of pcap session
  - Compilation: $ gcc -o sniff sniff.c **-lpcap**

all letters