Packet Sniffing & Spoofing

Outline

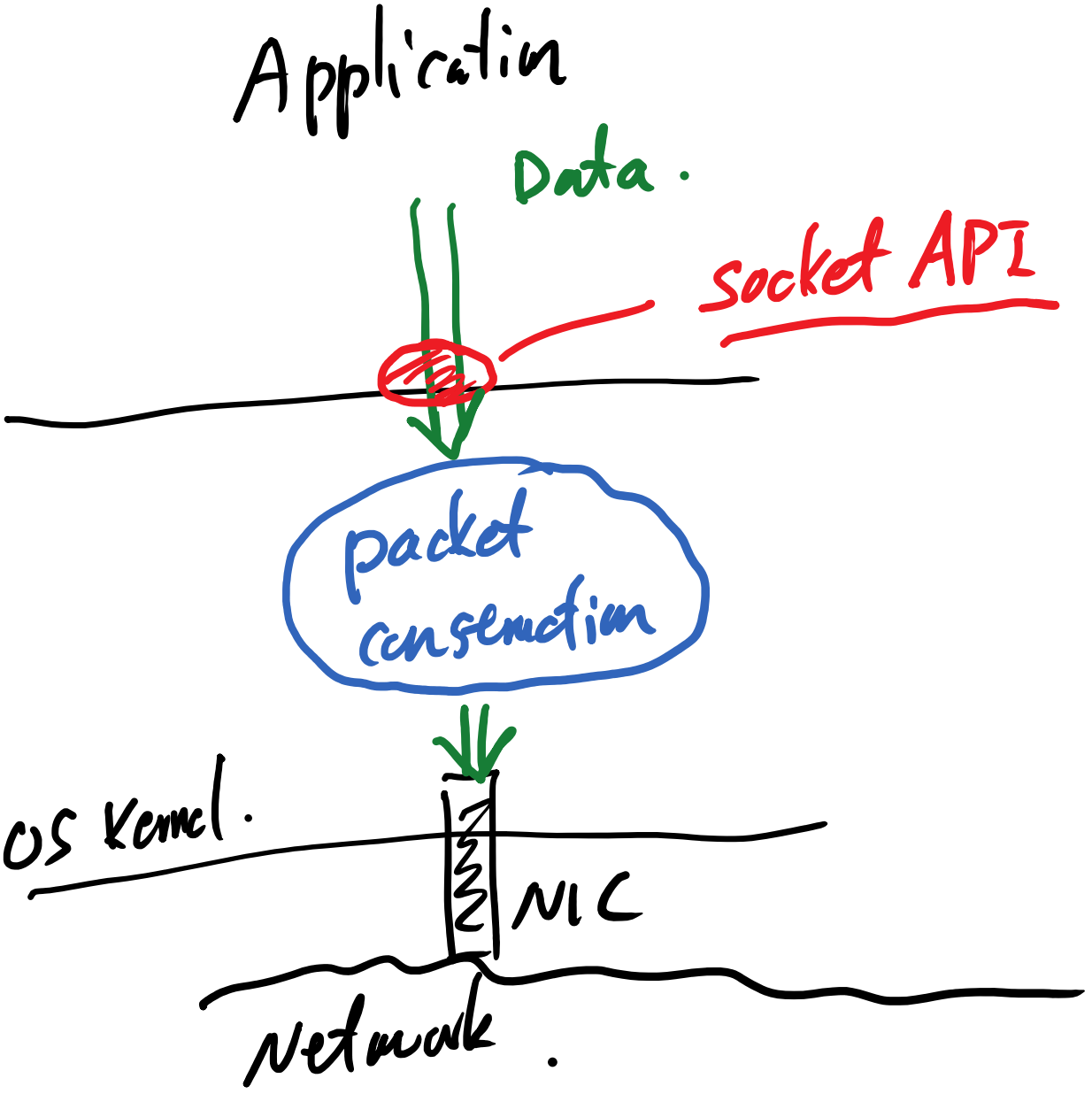
- Sending/Receiving packets
- Sniffing packets
- Spoofing packets
- Scapy vs C
- Byte order and checksum

- **Reading:** Chapter 15
- **Lab: Packet Sniffing and Spoofing Lab**

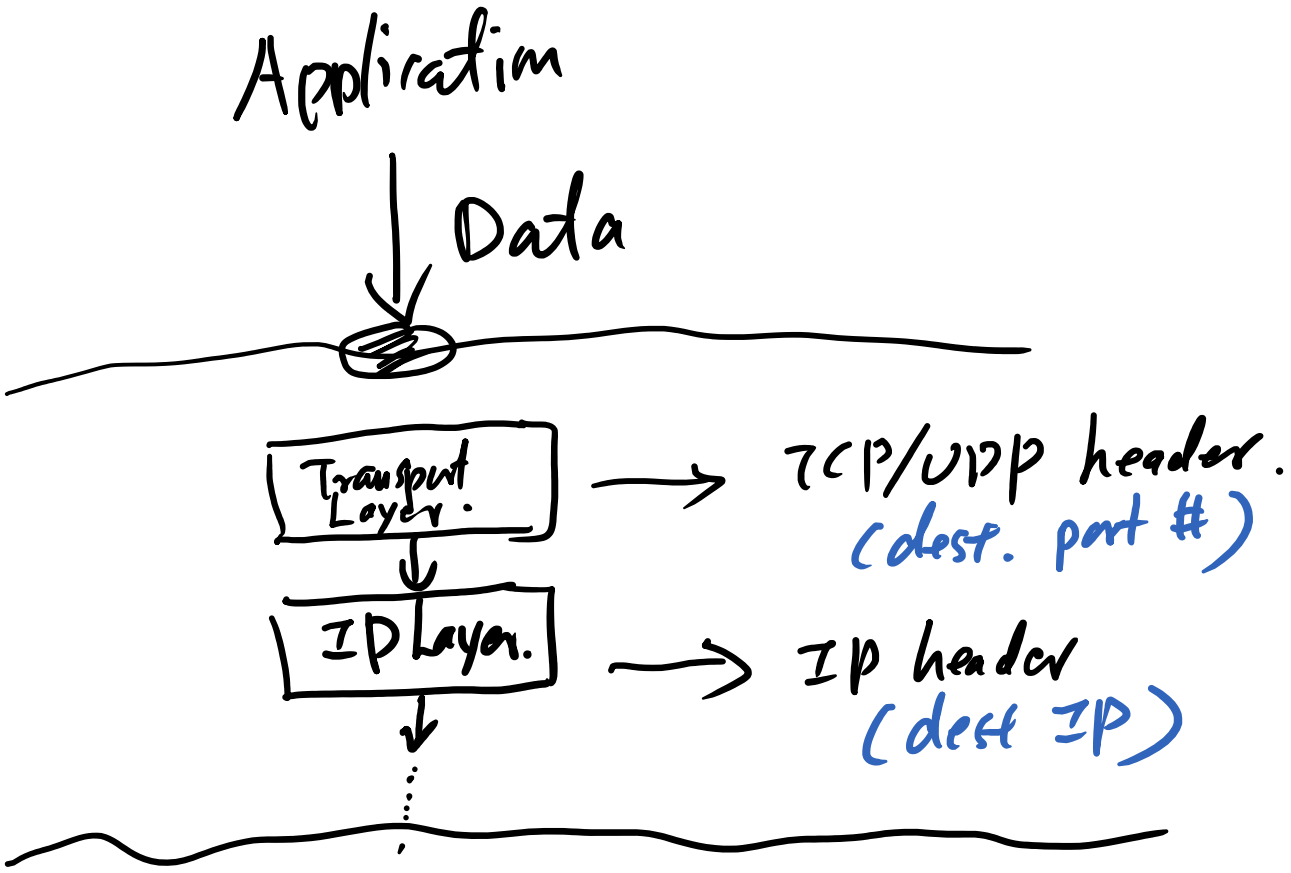
End

Socket Programming: Sending/Receiving Packets

Socket APIs


Application
Data.
socket API
packet
consumption
OS Kernel.
NIC
Network.

How to Send Packets



Sending Packets (Python)

❖ UDP Client Example

```
#!/usr/bin/python3

import socket

IP   = "127.0.0.1"
PORT = 9090
data = b'Hello, World!'

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.sendto(data, (IP, PORT))
```

❖ Execution result

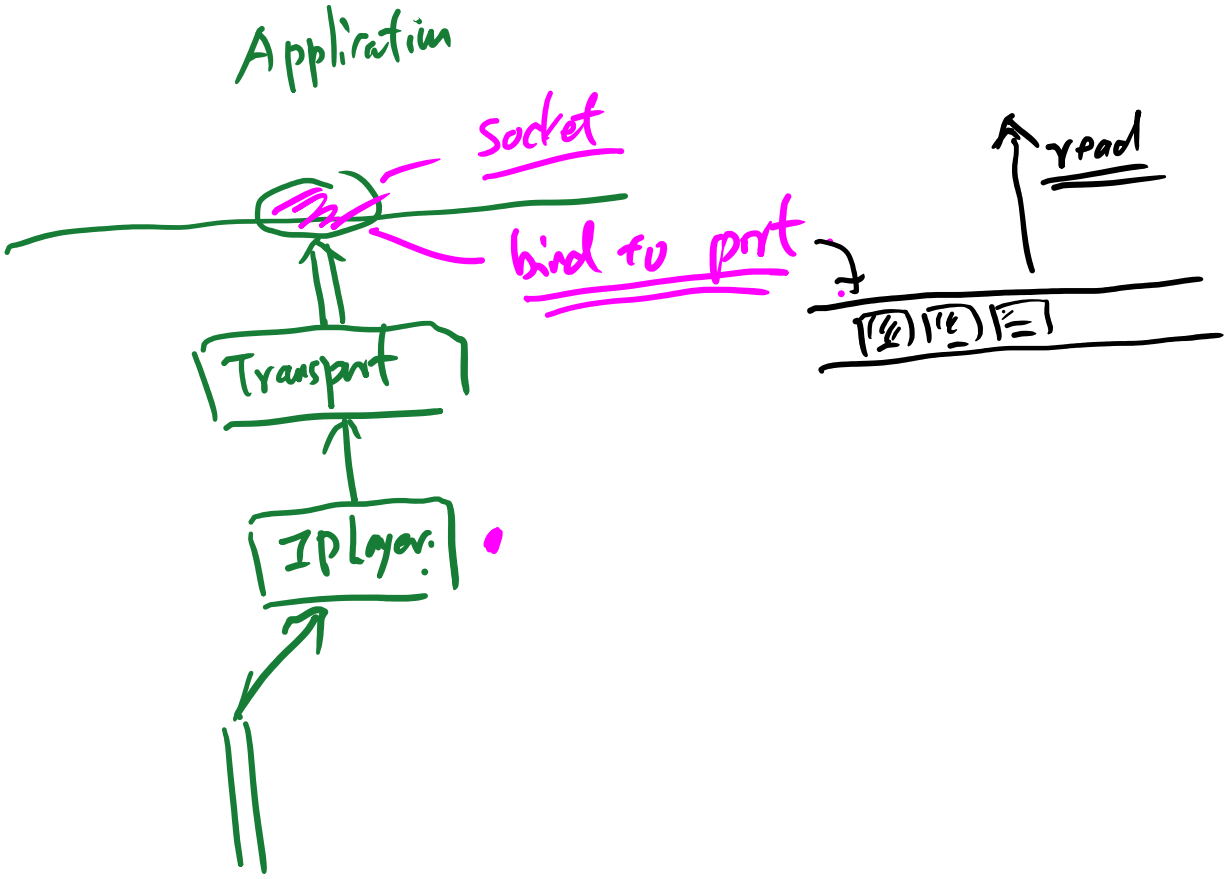
```
$ nc -luv 9090
Listening on [0.0.0.0] (family 0, port 9090)
Hello, World!
```

Packet Sending (C)

```c
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/ip.h>

void main()
{
    struct sockaddr_in dest_info;
    char *data = "Hello Server.\n";

    // Create a network socket.
    int sock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);

    // Provide needed information about destination.
    memset((char *) &dest_info, 0, sizeof(dest_info));
    dest_info.sin_family = AF_INET;
    dest_info.sin_addr.s_addr = inet_addr("10.0.2.5");
    dest_info.sin_port = htons(9090);

    // Send the packet out.
    sendto(sock, data, strlen(data), 0,
           (struct sockaddr *)&dest_info, sizeof(dest_info));
    close(sock);
}
```

How Packets Are Received

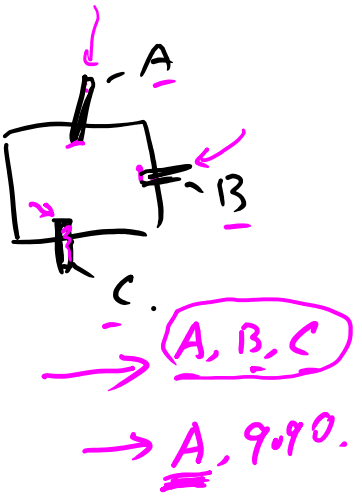
IP layer and above



Receiving Packets (Python)

❖ UDP Server Example

```
#!/usr/bin/python3

import socket

IP   = "0.0.0.0"
PORT = 9090

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((IP, PORT))

while True:
  data, (ip, port) = sock.recvfrom(1024)
  print("Sender: {} and Port: {}".format(ip, port))
  print("Received message: {}".format(data))
```



❖ Execution result

```
Terminal
seed@10.0.2.6:$ nc -u 10.0.2.7 9090
hello
hello again
```

```
Terminal
Server(10.0.2.7):$ udp_server.py
Sender: 10.0.2.6 and Port: 49112
Received message: b'hello\n'
Sender: 10.0.2.6 and Port: 49112
Received message: b'hello again\n'
```

Receiving Packets (C)

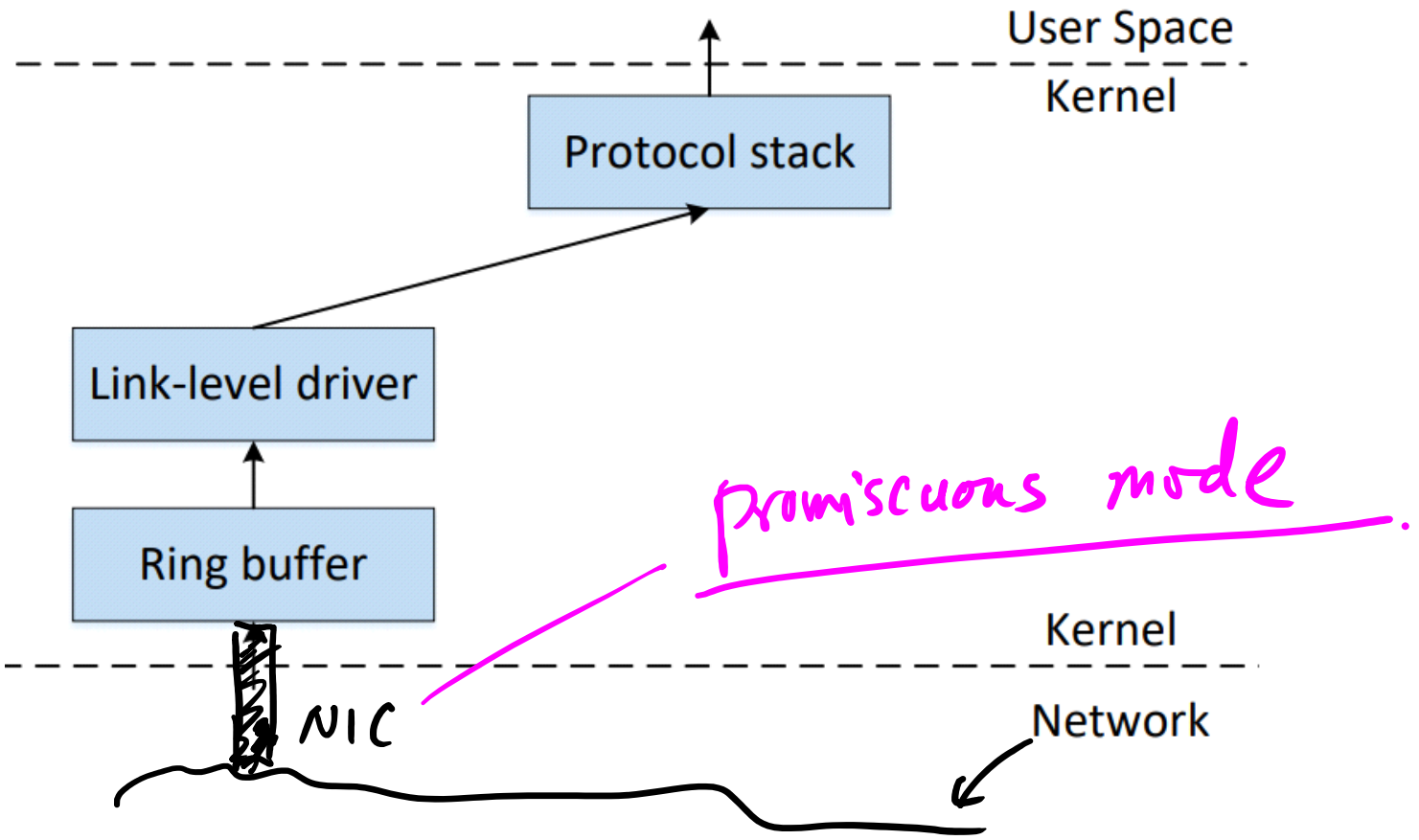
```c
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/ip.h>

void main()
{
    struct sockaddr_in server;
    struct sockaddr_in client;
    int clientlen;
    char buf[1500];

    int sock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);

    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");

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

End

Packet Sniffing

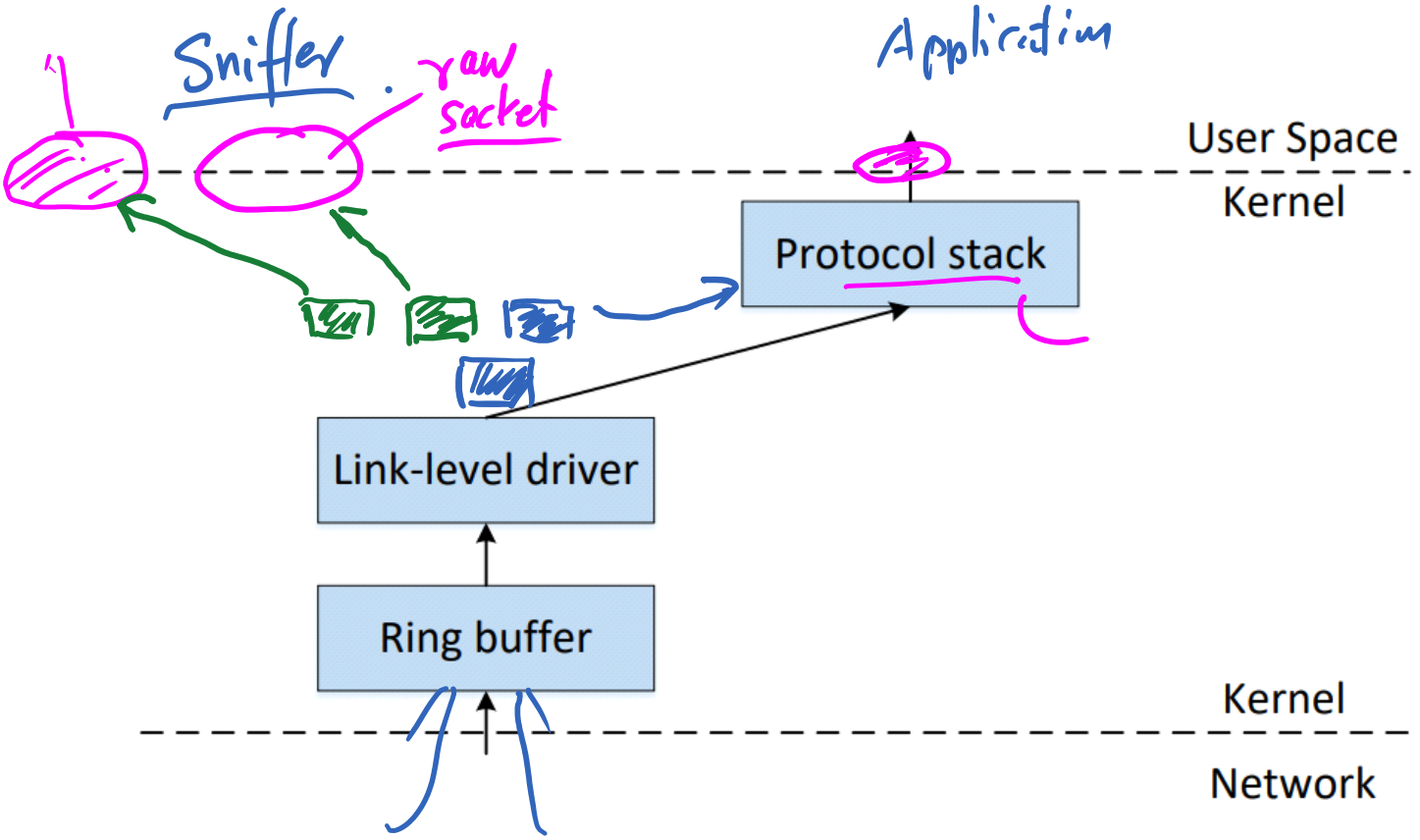
How Packets Are Received

Below the IP Layer



How to Get A Copy of Packet

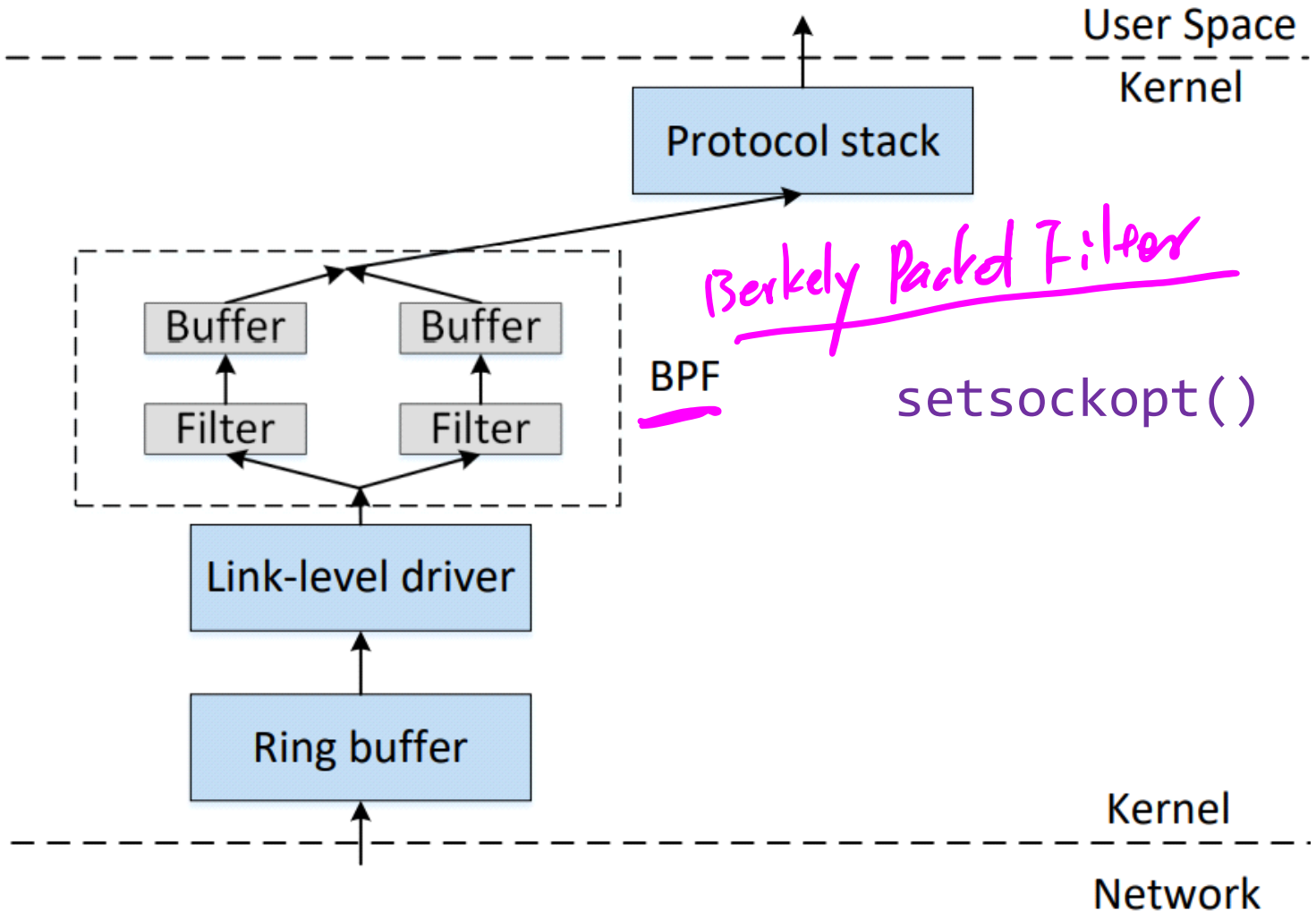
Raw Socket



Packet Capturing Using Raw Socket

```c
int main() {
    int PACKET_LEN = 512;
    char buffer[PACKET_LEN];
    struct sockaddr saddr;
    struct packet_mreq mr;

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

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

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

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

Filtering Out Unwanted Packets

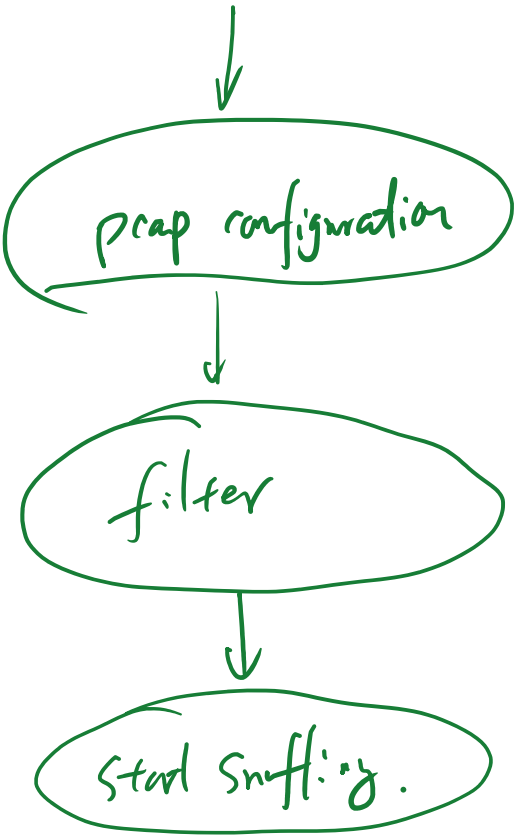


End

Packet Sniffing using PCAP

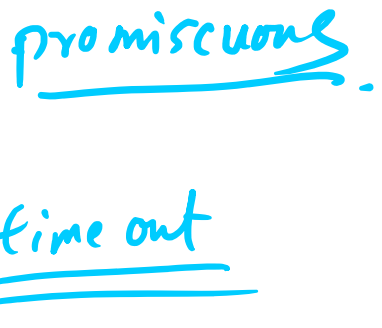
PCAP: Packet Capture API

- Originally coming from tcpdump
- Supported by multiple platforms:
  - Linux: libpcap
  - Windows: WinPcap and Npcap
- Written in C. Other language implement wrappers.
- Basis for many tools
  - Wireshark, tcpdump, Scapy, McAfee, Nmap, Snort,



PCAP: Open Device for Capturing

```
pcapt_t *pcap_open_live(
            const char *device,
            int snaplen,
            int promisc,
            int to_ms,
            char *errbuf
);
```



PCAP: Compile and Set Filters

❖ Compile Filter

```
int pcap_compile(
            pcap_t *p,
            struct bpf_program *fp,
            const char *str,
            int optimize,
            bpf_u_int32 netmask
);
```

❖ Set Filter

```
int pcap_setfilter(
            pcap_t *p,
            struct bpf_program *fp
);
```

PCAP: Filter Examples

```
f1 = 'src net 10.0.2.0/24'
f2 = 'net 10.0.2.0/24'

f3 = 'dst host 10.0.2.8'
f4 = 'host 10.0.2.8'

f5 = 'udp and dst port 53'
f6 = 'udp and dst portrange 50-55'

f7 = 'icmp or tcp'

sniff(iface='enp0s3', filter=f7, prn=process packet)
```

PCAP: Start Capturing Packets

❖ Start Capturing

```
int pcap_loop(
        pcap_t *p,
        int cnt,
        pcap_handler callback,
        u_char *user
);
```


❖ The Callback Function

```
typedef void (*pcap_handler)(
        u_char *user,
        const struct pcap_pkthdr *header,
        const u_char *packet);
```

A Sniffer Program Using PCAP

❖ Set up the packet-capturing logic

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

        // Step 1: Open live pcap session on NIC with interface name
        handle = pcap_open_live("enp0s3", 8192, 1, 1000, errbuf);

        // Step 2: Compile filter_exp into BPF psuedo-code
        pcap_compile(handle, &fp, filter_exp, 0, net);
        if (pcap_setfilter(handle, &fp) !=0) {
             pcap_perror(handle, "Error:");
             exit(EXIT_FAILURE);
        }

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

        pcap_close(handle); //Close the handle
        return 0;
}
```

❖ Get a packet and process it

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

   if(ntohs(eth->ether_type) == 0x800) {
     struct ipheader *ip = (struct ipheader *) (packet +
                           sizeof(struct ethheader));
     printf("   From: %s\n", inet_ntoa(ip->iph_sourceip));
     printf("   To: %s\n",   inet_ntoa(ip->iph_destip));

     switch(ip->iph_protocol) {
       case IPPROTO_TCP:
              printf("        Protocol: TCP\n");
              break;
       case IPPROTO_UDP:
              printf("        Protocol: UDP\n");
              break;
       case IPPROTO_ICMP:
              printf("        Protocol: ICMP\n");
              break;
       default:
              printf("        Protocol: Others\n");
              break;
     }
  }
}
```

End

Packet Sniffing using Scapy or Wireshark

Python + Scapy

❖ What Is Scapy?

Powerful modules (or programs) for packet manipulation

- Packet parsing
- Packet sending and receiving
- Packet sniffing
- Packet spoofing
- Adding new protocols
- Many more applications …

❖ Installation

```
sudo pip3 install scapy
```

❖ Import Scapy Module in Python Program

```
from scapy.all import *
```

Sniffer Example 1

❖ Example: Sniff Packets

```
#!/usr/bin/python3

from scapy.all import *

pkt = sniff(iface='enp0s3',
            filter='icmp or udp',
            count=10)

pkt.summary()
```

Sniffer Example 2

❖ Invoke a Callback Function

```
#!/usr/bin/python3

from scapy.all import *

def process_packet(pkt):
      #hexdump(pkt)
      pkt.show()
      print("----------------------------------")

f = 'udp and dst portrange 50-55 or icmp'

sniff(iface='enp0s3', filter = f, prn=process_packet)
```

Different Ways to Display Packets

❖ Using hexdump()

```
>>> hexdump(pkt)
0000  52 54 00 12 35 00 08 00 27 77 2E C3 08 00 45 00  RT..5...'w....E.
0010  00 54 F2 29 40 00 40 01 2C 68 0A 00 02 08 08 08  .T.)@.@.,h......
0020  08 08 08 00 98 01 10 C7 00 02 B8 66 65 5E 3A 6D  ...........fe^:m
0030  0C 00 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15  ................
0040  16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25  .......... !"#$%
0050  26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35  &'()*+,-./012345
0060  36 37                                           67
```

❖ Using pkt.show()

```
>>> pkt.show()
###[ Ethernet ]###
  dst       = 52:54:00:12:35:00
  src       = 08:00:27:77:2e:c3
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     ...
     proto     = icmp
     chksum    = 0x3c9a
     src       = 10.0.2.8
     dst       = 8.8.8.8
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0x6905
        id        = 0x107a
        seq       = 0x2
###[ Raw ]###
           load      = '\x90ee^\x91\xb7\ ...'
```

Understanding Scapy's Layer Classes

❖ Iterate Through Layers

```
>>> pkt
<Ether  type=IPv4 |<IP  frag=0 proto=udp |<UDP  |<Raw  load='hello' |>>>>
```

payload

```
>>> pkt.payload
<IP  frag=0 proto=udp |<UDP  |<Raw  load='hello' |>>>
```

```
>>> pkt.payload.payload
<UDP  |<Raw  load='hello' |>>
```

```
>>> pkt.payload.payload.payload
<Raw  load='hello' |>
```

Accessing Layers

❖ Checking Layer Type

```
<Ether  type=IPv4 |<IP  frag=0 proto=udp |<UDP |<Raw  load='hello' |>>>>
```

```
>>> pkt.haslayer(UDP)
True
>>> pkt.haslayer(TCP)
0
>>> pkt.haslayer(Raw)
True
```

❖ Accessing Layers

```
>>> pkt[UDP]
<UDP  |<Raw  load='hello' |>>

>>> pkt.getlayer(UDP)
<UDP  |<Raw  load='hello' |>>

>>> pkt[Raw]
<Raw  load='hello' |>

>>> pkt[Raw].load
b'hello'
```

Sniffer Example 3

```
#!/usr/bin/python3

from scapy.all import *

def process_packet(pkt):
    if pkt.haslayer(IP):
        ip =  pkt[IP]
        print("IP: {} --> {}".format(ip.src, ip.dst))

    if pkt.haslayer(TCP):
        tcp = pkt[TCP]
        print("    TCP  port: {} --> {}".format(tcp.sport, tcp.dport))

    elif pkt.haslayer(UDP):
        udp = pkt[UDP]
        print("    UDP  port: {} --> {}".format(udp.sport, udp.dport))

    elif pkt.haslayer(ICMP):
        icmp = pkt[ICMP]
        print("    ICMP type: {}".format(icmp.type))

    else:
        print("    Other protocol")


sniff(iface='enp0s3', filter='ip', prn=process_packet)
```

Get Information of Protocol Classes

❖ Get Attribute Names

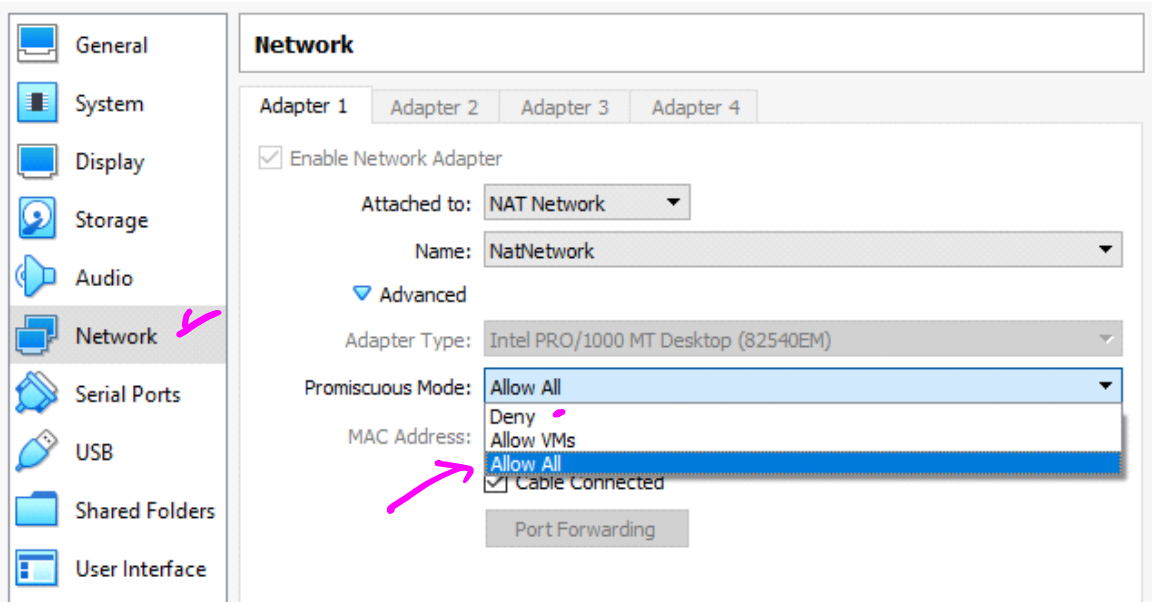
```
>>> ls(IP)
version    : BitField (4 bits)                   = (4)
ihl        : BitField (4 bits)                   = (None)
tos        : XByteField                          = (0)
len        : ShortField                          = (None)
id         : ShortField                          = (1)
flags      : FlagsField (3 bits)                 = (<Flag 0 ()>)
frag       : BitField (13 bits)                  = (0)
ttl        : ByteField                           = (64)
proto      : ByteEnumField                       = (0)
chksum     : XShortField                         = (None)
src        : SourceIPField                       = (None)
dst        : DestIPField                         = (None)
options    : PacketListField                     = ([])
```

❖ Get Method Names

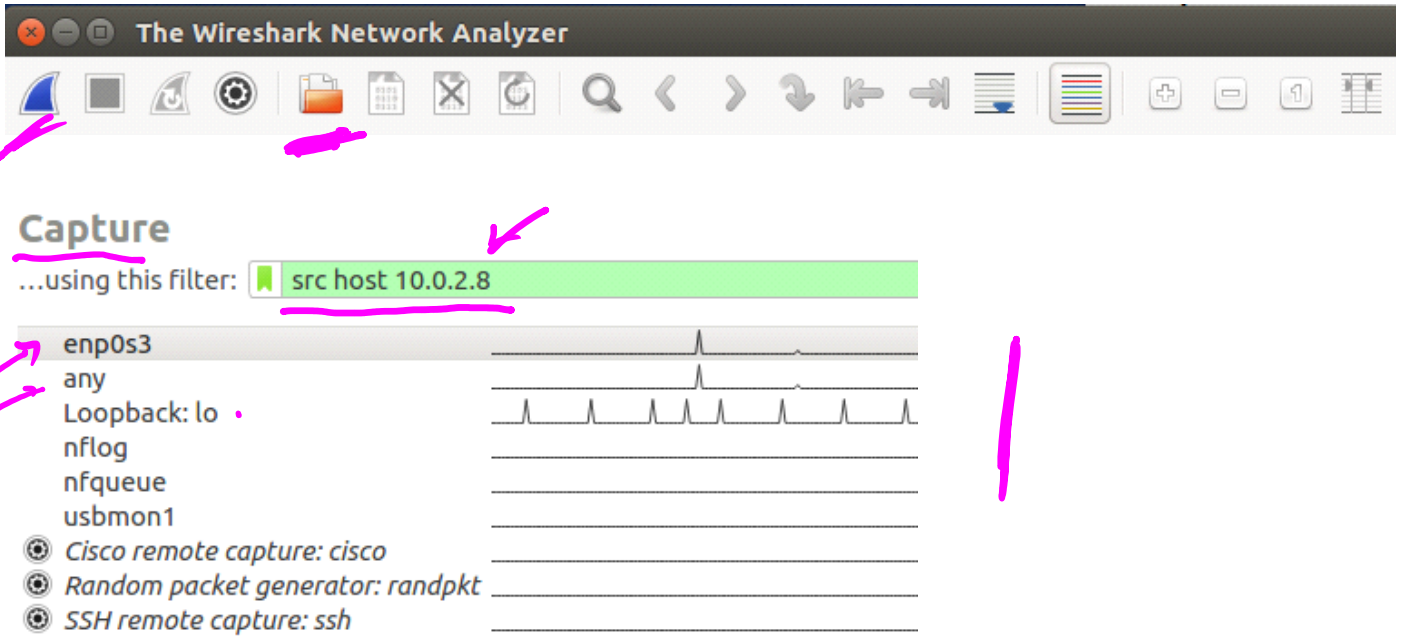
```
>>> help(IP)

Help on class IP in module scapy.layers.inet:
class IP(scapy.packet.Packet, IPTools)
 |
 |  Method resolution order:
 |      IP
 |      scapy.packet.Packet
 |      ...
 |      builtins.object
 |
 |  Methods defined here:
 |     ...
 |
 |  Methods inherited from scapy.packet.Packet:
 |     ...
```

Wireshark

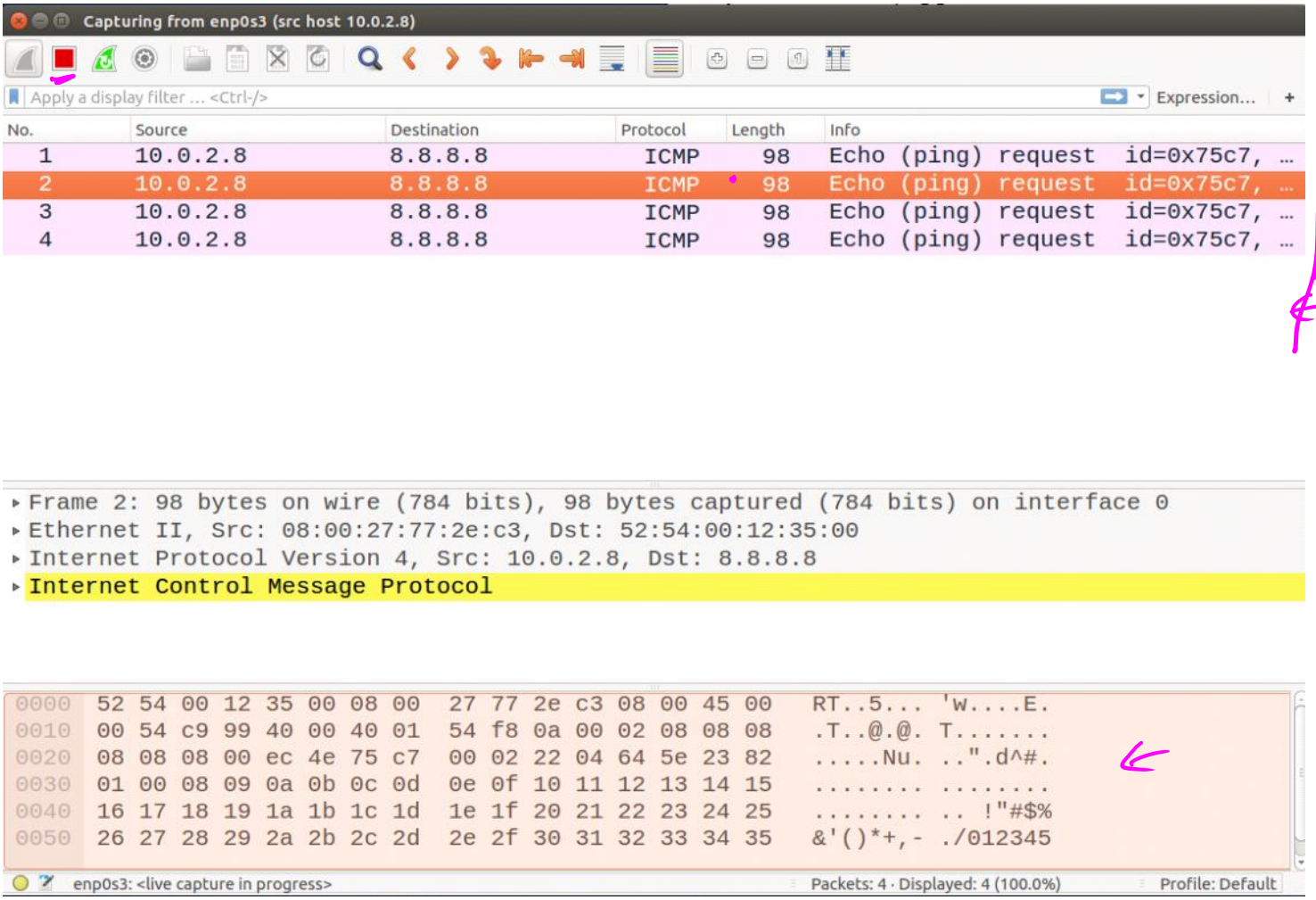
❖ VM Setting



❖ Start Capturing



Wireshark


Capturing from enp0s3 (src host 10.0.2.8)
Apply a display filter ... <Ctrl-/>
Expression...
No. Source Destination Protocol Length Info
1 10.0.2.8 8.8.8.8 ICMP 98 Echo (ping) request id=0x75c7, ...
2 10.0.2.8 8.8.8.8 ICMP 98 Echo (ping) request id=0x75c7, ...
3 10.0.2.8 8.8.8.8 ICMP 98 Echo (ping) request id=0x75c7, ...
4 10.0.2.8 8.8.8.8 ICMP 98 Echo (ping) request id=0x75c7, ...
Frame 2: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
Ethernet II, Src: 08:00:27:77:2e:c3, Dst: 52:54:00:12:35:00
Internet Protocol Version 4, Src: 10.0.2.8, Dst: 8.8.8.8
Internet Control Message Protocol
0000 52 54 00 12 35 00 08 00 27 77 2e c3 08 00 45 00 RT..5... 'w....E.
0010 00 54 c9 99 40 00 40 01 54 f8 0a 00 02 08 08 08 .T..@.@. T.......
0020 08 08 08 00 ec 4e 75 c7 00 02 22 04 64 5e 23 82 .....Nu. ..".d^#.
0030 01 00 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 ....
0040 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 .... !"#$%
0050 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 &'()*+,- ./012345
enp0s3: <live capture in progress>
Packets: 4 · Displayed: 4 (100.0%)
Profile: Default

End

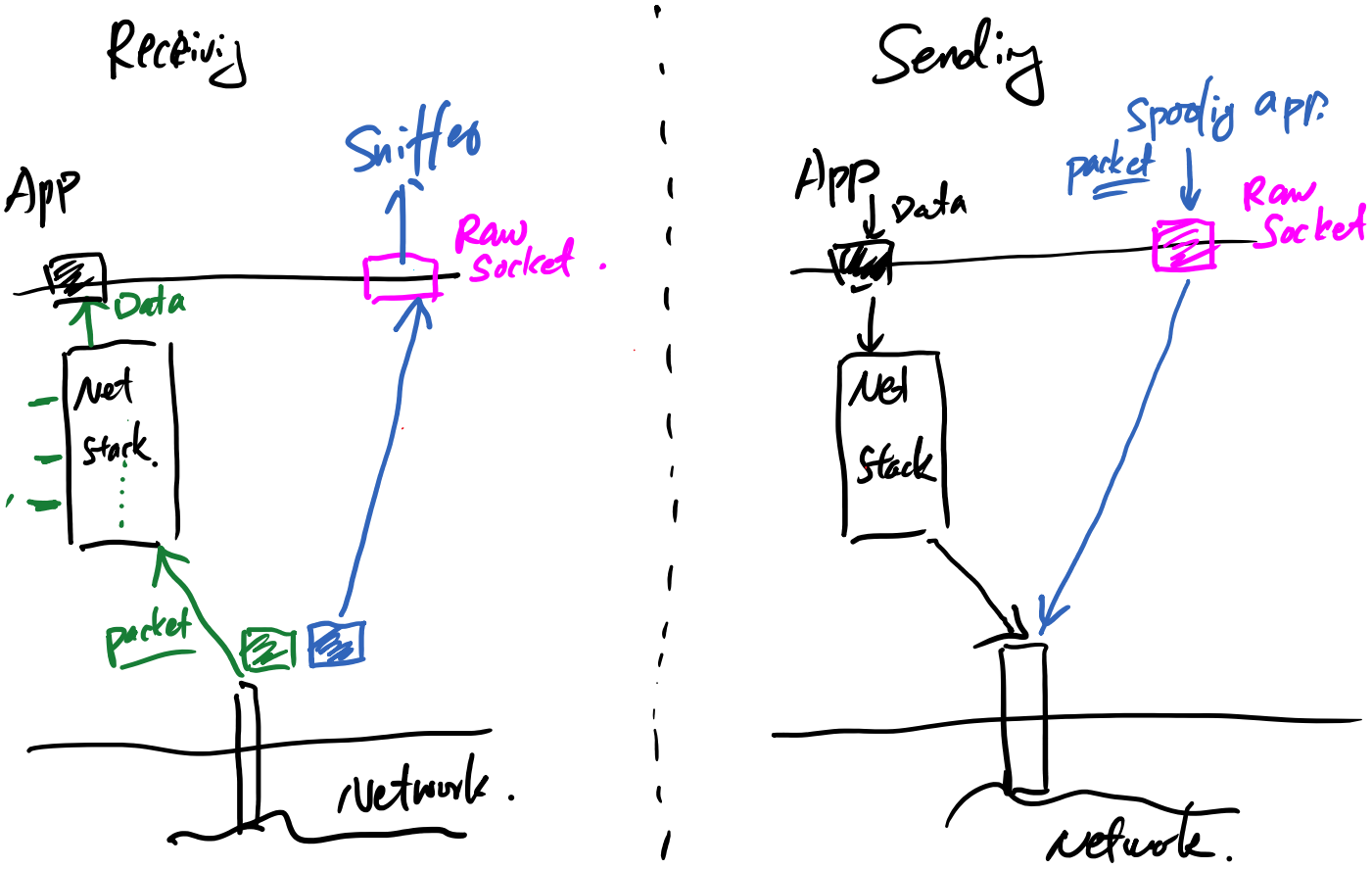
Packet Spoofing

Sending Packets (Python)

❖ UDP Client Example

```
#!/usr/bin/python3

import socket

IP   = "127.0.0.1"
PORT = 9090
data = b'Hello, World!'

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.sendto(data, (IP, PORT))
```

Spoofing Packet Using Raw Socket



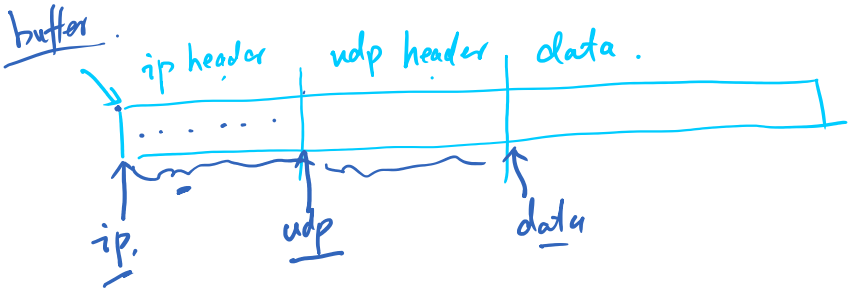
Spoofing IP Packet: C Code

```c
void send_raw_ip_packet(struct ipheader* ip)
{
  struct sockaddr_in dest_info;
  int enable =1;

  // Create a raw socket
  int sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);

  setsockopt(sock, IPPROTO_IP, IP_HDRINCL, &enable, sizeof(enable));

  //Destination info
  dest_info.sin_family = AF_INET;
  dest_info.sin_addr   = ip->iph_destip;

  //Send the packet out
  printf("Sending spoofed IP packet...\n");
  if(sendto(sock, ip, ntohs(ip->iph_len), 0,
     (struct sockaddr *)&dest_info, sizeof(dest_info)) < 0){
        perror("PACKET NOT SENT\n");
        return;
  }
  close(sock);
}
```

Constructing Raw Packets

❖ Type casting

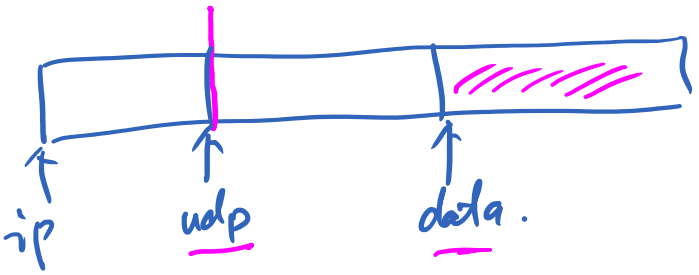
```
char buffer[PACKET_LEN];
memset(buffer, 0, PACKET_LEN);

// Find the starting point of each layer
struct ipheader *ip = (struct ipheader *)buffer;
struct udpheader *udp = (struct udpheader *)
                        (buffer + sizeof(struct ipheader));
char   *data = buffer + sizeof(struct ipheader)
                      + sizeof(struct udpheader);
```



Spoofing UDP Packet

❖ The code

```
char buffer[PACKET_LEN];
memset(buffer, 0, PACKET_LEN);

// Find the starting point of each layer
struct ipheader *ip = (struct ipheader *)buffer;
struct udpheader *udp = (struct udpheader *)
                        (buffer + sizeof(struct ipheader));
char   *data = buffer + sizeof(struct ipheader)
                      + sizeof(struct udpheader);
```




```
// Add UDP data
char *msg="Hello Server.\n";
int data_len=strlen(msg);
strncpy(data, msg, data_len);
```

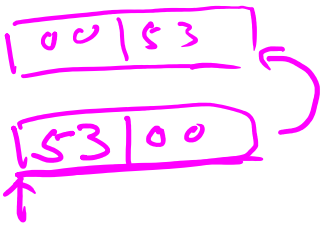
udp

UDP Datagram Header Format

Bit #	0 – 7	8 – 15	16 – 23	24 – 31
0	Source Port		Destination Port	
32	Length		Header and Data Checksum	

or 0053

```
// Construct UDP Header
udp->udp_dport = htons(DST_PORT);
udp->udp_sport = htons(9999);
udp->udp_ulen  = htons(sizeof(struct udpheader) + data_len);
udp->udp_sum   = 0;
```



IPv4 Packet Header Format

Bit #	0	7	8	15	16	23	24	31
0	Version	IHL	DSCP	ECN	Total Length			
32	Identification				Flags	Fragment Offset		
64	Time to Live		Protocol		Header Checksum			
96	Source IP Address							
128	Destination IP Address							
160	Options (if IHL > 5)							

```
// Construct IP Header
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_UDP;
ip->iph_len = htons(sizeof(struct ipheader) +
                    sizeof(struct udpheader) + data_len);
ip->iph_chksum = 0; // Leave it to OS to set this field


// Send out the construct packet
send_raw_ip_packet(ip);
```

End

Packet Spoofing Using Scapy

Constructing Packets

```
>>> a = IP(src='1.2.3.4', dst='10.20.30.40')
>>> b = UDP(sport = 1234, dport = 1020)
>>> c = 'Hello World'
>>> pkt = a/b/c

>>> pkt.show()
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = udp
  chksum    = None
  src       = 1.2.3.4
  dst       = 10.20.30.40
  \options   \
###[ UDP ]###
     sport     = 1234
     dport     = 1020
     len       = None
     chksum    = None
###[ Raw ]###
        load      = 'Hello World'
```

send( )

Layer Stacking

Packet.

```
>>> a = IP()/UDP()/Raw("hello")
>>> a
<IP  frag=0 proto=udp |<UDP  |<Raw  load='hello' |>>>
```

payload.

❖ Operator Overloading

```
def __div__(self, other):
    if isinstance(other, Packet):
        cloneA = self.copy()
        cloneB = other.copy()
        cloneA.add_payload(cloneB)
        return cloneA
    elif isinstance(other, (bytes, str)):
        return self / conf.raw_layer(load=other)
    else:
        return other.__rdiv__(self)
__truediv__ = __div__
```

Packet Spoofing using Scapy

❖ Spoofing ICMP packet

```
#!/usr/bin/python3
from scapy.all import *

print("SENDING SPOOFED ICMP PACKET.........")
ip = IP(src="1.2.3.4", dst="93.184.216.34")
icmp = ICMP()
pkt = ip/icmp
pkt.show()
send(pkt,verbose=0)
```

❖ Spoofing UDP packet

```
#!/usr/bin/python3
from scapy.all import *

print("SENDING SPOOFED UDP PACKET.........")
ip = IP(src="1.2.3.4", dst="10.0.2.69") # IP Layer
udp = UDP(sport=8888, dport=9090)       # UDP Layer
data = "Hello UDP!\n"                   # Payload
pkt = ip/udp/data
pkt.show()
send(pkt,verbose=0)
```

Sniff Request and Spoof Reply

Sniff-and-Spoof Example

```
def spoof_pkt(pkt):
  if ICMP in pkt and pkt[ICMP].type == 8:
     print("Original Packet.........")
     print("Source IP : ", pkt[IP].src)
     print("Destination IP :", pkt[IP].dst)

     ip = IP(src=pkt[IP].dst, dst=pkt[IP].src, ihl=pkt[IP].ihl)
     ip.ttl = 99
     icmp = ICMP(type=0, id=pkt[ICMP].id, seq=pkt[ICMP].seq)

     if pkt.haslayer(Raw):
        data = pkt[Raw].load
        newpkt = ip/icmp/data
     else:
        newpkt = ip/icmp

     print("Spoofed Packet.........")
     print("Source IP : ", newpkt[IP].src)
     print("Destination IP :", newpkt[IP].dst)

     send(newpkt,verbose=0)

sniff(filter='icmp and src host 10.0.2.7',prn=spoof_pkt)
```

Demo

- Spoof UDP
- Sniff and Spoof ICMP

End

Scapy v.s. C

Comparing Scapy and C

- Default values
- Packets (byte array) to Objects/Structures
- Performance (experiment)

Scapy: **106** packets per second
C: **4000** packets per second

Scapy: None and Default Value

❖ None and Default value

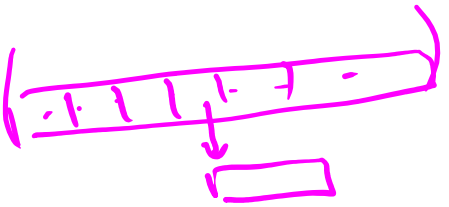
```
seed@10.0.2.6:$ python3
Python 3.5.2 (default, Nov 17 2016, 17:05:23)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from scapy.all import *
>>> ls(IP)
version    : BitField (4 bits)                   = (4)
ihl        : BitField (4 bits)                   = (None)
tos        : XByteField                          = (0)
len        : ShortField                          = (None)
id         : ShortField                          = (1)
flags      : FlagsField (3 bits)                 = (<Flag 0 ()>)
frag       : BitField (13 bits)                  = (0)
ttl        : ByteField                           = (64)
proto      : ByteEnumField                       = (0)
chksum     : XShortField                         = (None)
src        : SourceIPField                       = (None)
dst        : DestIPField                         = (None)
options    : PacketListField                     = ([])
```

Parsing Application-Layer Protocol

Using Scapy to Parse Protocol Header

❖ Example: A Simple DNS Server

```
#!/usr/bin/python3
from scapy.all import *
from socket import AF_INET, SOCK_DGRAM, socket

sock = socket(AF_INET, SOCK_DGRAM)
sock.bind(('0.0.0.0', 1053))

while True:
  request, addr = sock.recvfrom(4096)
  DNSreq = DNS(request)
  query = DNSreq.qd.qname
  print(query.decode('ascii'))

  Anssec = DNSRR(rrname=DNSreq.qd.qname, type='A',
                 rdata='1.1.1.1', ttl=259200)
  NSsec1 = DNSRR(rrname="example.com", type='NS',
                 rdata='ns1.example.com', ttl=259200)
  NSsec2 = DNSRR(rrname="example.com", type='NS',
                 rdata='ns2.example.com', ttl=259200)
  Addsec1 = DNSRR(rrname='ns1.example.com', type='A',
                 rdata='1.2.3.4', ttl=259200)
  Addsec2 = DNSRR(rrname='ns2.example.com', type='A',
                 rdata='1.2.3.5', ttl=259200)
  DNSpkt = DNS(id=DNSreq.id, aa=1, rd=0, qr=1,
               qdcount=1, ancount=1, nscount=2, arcount=2,
               qd=DNSreq.qd, an=Anssec,
               ns=NSsec1/NSsec2, ar=Addsec1/Addsec2)
  print(repr(DNSpkt))
  sock.sendto(bytes(DNSpkt), addr)
```



Hybrid Approach: Python + C

Hybrid Approach: Example

- **Example: Construct DNS response using Scapy**

```
# Construct the IP and UPD header
IPpkt = IP(dst=dstIP, src=srcIP)
UDPpkt = UDP(dport=33333, sport=53, chksum=0)

# Construct the DNS header and records
Qdsec = DNSQR(qname=targetName)
Anssec = DNSRR(rrname=targetName, type='A', rdata='1.1.1.1', ttl=259200)
NSsec = DNSRR(rrname=targetDomain, type='NS', rdata=attackerNS, ttl=259200)
DNSpkt = DNS(id=0xAAAA, aa=1, rd=1, qr=1,
             qdcount=1, ancount=1, nscount=1, arcount=0,
             qd=Qdsec, an=Anssec, ns=NSsec)

# Stack all the layers together and save the result to a file.
Replypkt = IPpkt/UDPpkt/DNSpkt
with open('ip_resp.bin', 'wb') as f:
    f.write(bytes(Replypkt))
```

- **Modify the DNS packet using C**

```c
void send_dns_response(unsigned char *ip_resp, int n_resp,
            unsigned char *src_ip, char * name,
            unsigned short transaction_id)
{
    // Modify the src IP in the IP header (offset=12)
    int ip = (int) inet_addr(src_ip);
    memcpy(ip_resp+12, (void *) &ip, 4);

    // Modify the name in the question field (offset=41)
    memcpy(ip_resp+41, name, 5);

    // Modify the name in the answer field (offset=64)
    memcpy(ip_resp+64, name, 5);

    // Modify the transaction ID field (offset=28)
    unsigned short id[2];
    *id = htons(transaction_id);
    memcpy(ip_resp+28, (void *) id, 2);

    // Send the IP packet out
    send_raw_packet(ip_resp, n_resp);
}
```

Other Uses of Scapy: Send and Receive

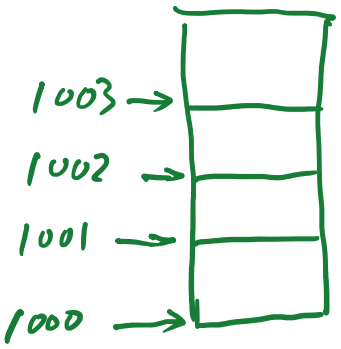
- `send()`: Send packets at Layer 3.
- `sendp()`: Send packets at Layer 2.
- `sr()`: Sends packets at Layer 3 and receiving answers.
- `srp()`: Sends packets at Layer 2 and receiving answers.
- `sr1()`: Sends packets at Layer 3 and waits for the first answer.
- `sr1p()`: Sends packets at Layer 2 and waits for the first answer.
- `srloop()`: Send a packet at Layer 3 in a loop and print the answer each time.
- `srploop()`: Send a packet at Layer 2 in a loop and print the answer each time.

```
#!/usr/bin/python3
from scapy.all import *

ip = IP(dst="8.8.8.8")
icmp = ICMP()
pkt = ip/icmp
reply = sr1(pkt)
print("ICMP reply .........")
print("Source IP : ", reply[IP].src)
print("Destination IP :", reply[IP].dst)
```

End

Byte Order (Endianess)

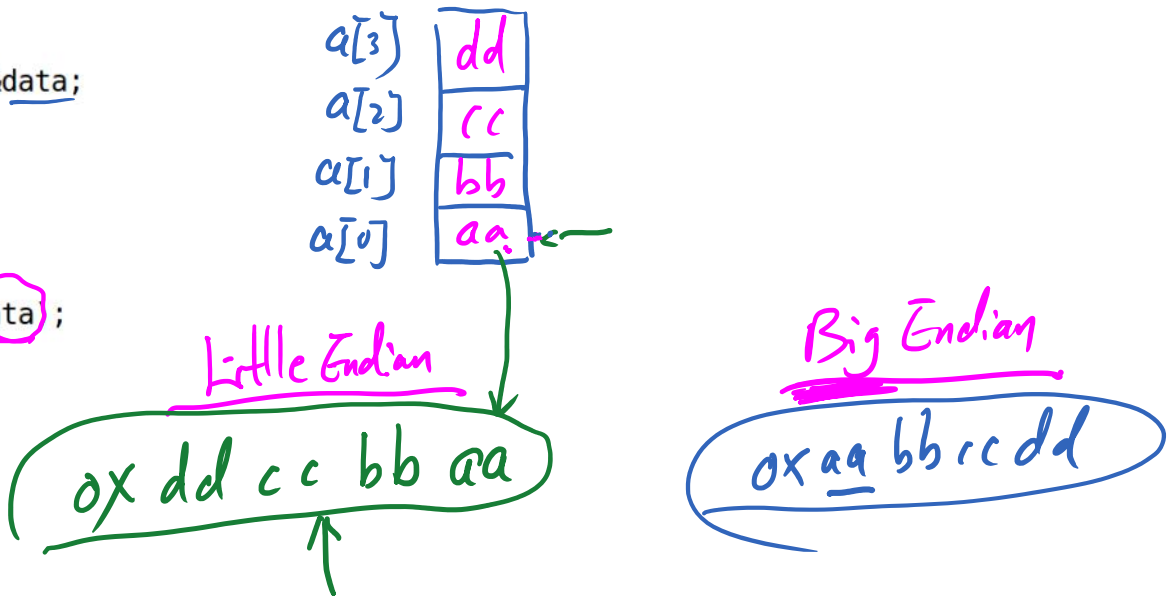
Byte Order (Endianess)



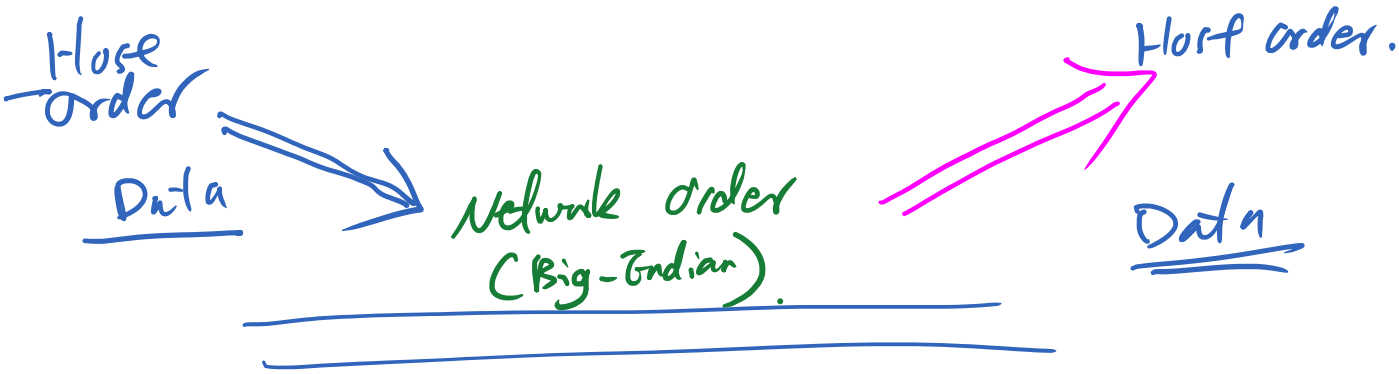
- Name: Jonathan Swift's 1726 novel, Gulliver's Travels
- Big-endian and Little-endian

Examples

```c
#include <stdio.h>
#include <arpa/inet.h>

void main()
{
  int data;
  char *a = (char *)&data;

  a[0] = 0xaa;
  a[1] = 0xbb;
  a[2] = 0xcc;
  a[3] = 0xdd;

  printf("0x%x\n", data);
}
```



Problem Caused by Endianess

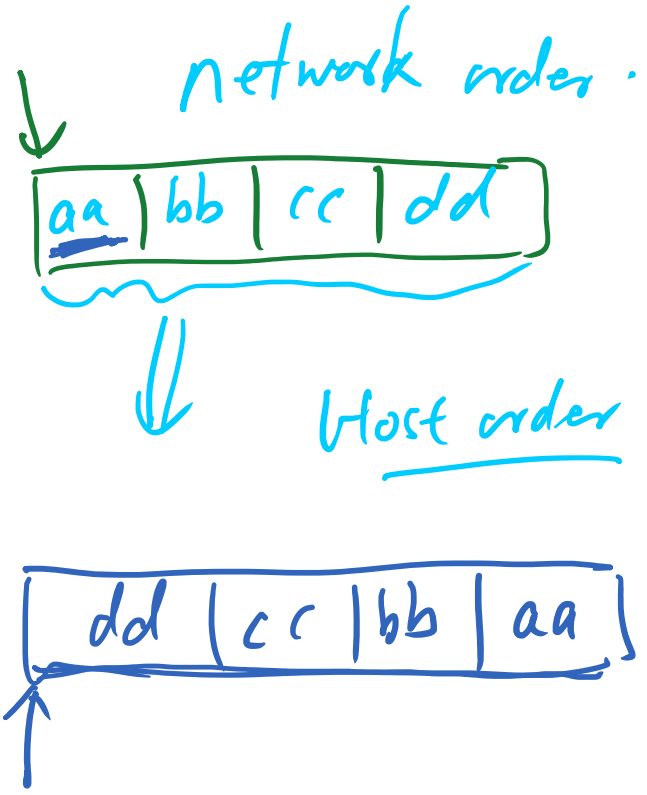


Byte-Order Conversion

Macro	Description
`htons()`	Convert unsigned short integer from host order to network order.
`htonl()`	Convert unsigned integer from host order to network order.
`ntohs()`	Convert unsigned short integer from network order to host order.
`ntohl()`	Convert unsigned integer from network order to host order.

Revised Examples

```c
#include <stdio.h>
#include <arpa/inet.h>

void main()
{
  int data;
  char *a = (char *)&data;

  a[0] = 0xaa;
  a[1] = 0xbb;
  a[2] = 0xcc;
  a[3] = 0xdd;

  printf("0x%x\n", ntohl(data));
}
```



0xaabbccdd

Exercise

Question: We need to save a 64-bit number 0XAABBCCDDEEFF1122 in a string, which is copied into a buffer by a program. This number will be used as an address by the program. In what order should we place this number in a string on a computer with an Intel Core i7 CPU?

End

Summary

- Socket programming
- Packet sniffing
  - How it works
  - How to write sniffer programs
- Packet spoofing
  - Using Scapy
  - Using C
- Spoffing various types of packets
- Byte order

Lab Exercise

Packet Sniffing and Spoofing Lab

SEED Lab: A Hands-on Lab for Security Education

Overview

Packet sniffing and spoofing are the two important concepts in network security; they are two major threats in network communication. Being able to understand these two threats is essential for understanding security measures in networking. There are many packet sniffing and spoofing tools, such as Wireshark, Tcpdump, Netwox, etc. Some of these tools are widely used by security experts, as well as by attackers. Being able to use these tools is important for students, but what is more important for students in a network security course is to understand how these tools work, i.e., how packet sniffing and spoofing are implemented in software.

The objective of this lab is for students to master the technologies underlying most of the sniffing and spoofing tools. Students will play with some simple sniffer and spoofing programs, read their source code, modify them, and eventually gain an in-depth understanding on the technical aspects of these programs. At the end of this lab, students should be able to write their own sniffing and spoofing programs.

Lab Tasks (Description)

PDF

- **VM version:** This lab has been tested on our pre-built `SEEDUbuntu16.04` VM.

End