原始套接字是一种套接字底层技术，它工作在网络层。利用原始套接字可以完成如下功能。

- 设置网卡为混杂模式，嗅探当前网路流经本网卡的所有数据包。
- 构造各种数据包（IP，ICMP，TCP，UDP等），并进行发送。
- 进行新协议的验证。

原始套接字可用于木马中的通信模块，伪造IP地址，拒绝服务攻击，数据包嗅探。

原始套接字的创建：

```
int rawsock=socket(AF_INET, SOCK_RAW, htons(ETH_P_IP));
//可以获取IP层的所有数据报文
```

htons参数的可选值及其意义

协议码	协议名
IPPROTO_ICMP	ICMP协议
ETH_P_IP	IP协议
IPPROTO_TCP	TCP协议
IPPROTO_UDP	UDP协议
IPPROTO_IPV6	IPv6协议
IPPROTO_EGP	EGP协议

数据发送：

在原始套接字中，执行数据发送前要条用setsocketopt函数进行套接字的首部设定：

```
int opt;
setsockopt(sockfd,IPPROTO_IP, IP_HDRINCL, &opt, sizeof(opt));
```

例子：

```
//利用原始套接字实现一个简单的采集网络数据包，并进行反向解析IP，MAC地址
#include <stdio.h>
#include <sys/socket.h>
#include <unistd.h>
#include <sys/types.h>
#include <linux/if_ether.h>
#include <linux/in.h>

#define BUFFER_MAX 2048
```

```
int main(int argc, char **argv)
{
    int rawsock;
    char buffer[BUFFER_MAX];
    char *ethhead;
    char *iphead;
    char *phead;

    //创建原始套接字
    if((rawsock=socket(PF_PACKET,SOCK_RAW,htons(ETH_P_IP)))<0){
        printf("error:create raw socket!\n");
        exit(0);
    }

    long framecount =0;

    while(1){
        int readnum = recvfrom(rawsock,buffer,2048,0,NULL,NULL);

        if(readnum<42){
            printf("error:header is incomplete!\n");
            exit(0);
        }

        ethhead=(char*)buffer;
        phead=ethhead;
        int ethernetmask=0XFF;
        framecount++;

        printf("---------------AnalysisiPacket[%d]---------------
\n",framecount);
        printf("MAC:");
        int i=6;
        for(;i<=11;i++)
            printf("%.2X:",phead[i]&ethernetmask);
        printf("------->");
        for(i=0;i<=5;i++)
            printf("%.2X:",phead[i]&ethernetmask);
        printf("\n");
```

```
        iphead=ethhead+14;
        phead=iphead+12;

        printf("IP:");
        for(i=0;i<=3;i++){
            printf("%d",phead[i]&ethernetmask);
            if(i!=3)
                printf(".");
        }
        printf("------->");
        for(i=4;i<=7;i++){
            printf("%d",phead[i]&ethernetmask);
            if(i!=7)
                printf(".");
        }
        printf("\n");

        int prototype=(iphead+9)[0];
        phead=iphead+20;

        printf("Protocol:");
        switch(prototype){
        case IPPROTO_ICMP:
            printf("ICMP\n");
            break;
        case IPPROTO_IGMP:
            printf("IGMP\n");
            break;
        case IPPROTO_IPIP:
            printf("IP");
            break;
        case IPPROTO_TCP:
            printf("TCP|source port: %u |",(phead[0]<<8)&0XFF00|phead[1]&0XFF);
            printf("destport: %u\n",(phead[2]<<8)&0XFF00|phead[3]&0XFF);
            break;
        case IPPROTO_UDP:
            printf("UDP|source port: %u |",(phead[0]<<8)&0XFF00|phead[1]&0XFF);
            printf("destport: %u\n",(phead[2]<<8)&0XFF00|phead[3]&0XFF);
            break;
        case IPPROTO_RAW:
            printf("RAW\n");
```

```
            break;
        default:
            printf("Unkown\n");
        }
        printf("-----------------end--------------------");
    }

    return 0;
}
```