

# I2tp协议分析实验与设计

2151140 王谦 信息安全

## 一、实验目标

基于c++设计并实现一个简单系统，实现

1. 基于windows或linux抓取I2tp协议流量

2. 针对每一个目标用户，实时监控其I2tp请求，并分析、还原、呈现其连接和业务载荷

3. 支持对特定目标的净载荷替换

构建一个demo：用笔记本电脑实现对手机上网对象的以上功能

## 二、目标分析

1. 搭建能够抓取 I2tp协议流量的环境

2. 抓取I2tp协议流量

3. 针对特定目标用户抓取I2tp协议流量，实时监控I2tp请求

4. 并分析、还原、呈现其连接和业务载荷

5. 支持对特定目标的静载荷替换

## 三、功能实现展示

### 1.搭建能够抓取 I2tp协议流量的环境

借助阿里云服务器和开源脚本，成功搭建了能够抓取 I2tp协议流量的环境。

<input type="checkbox"/>	实例ID/名称	执行状态	IP地址	执行时间	ExitCode
<input type="checkbox"/>	i-bp1fhwpinnsdb178cxlm iZbp1fhwpinnsdb178cxlmZ	成功执行	47.98.179.198 (公) 172.24.65.91 (私有)	22秒	0

更新时间：2024年3月24日 21:39:23

262

If there is no [FAILED] above, you can connect to your L2TP

263

VPN Server with the default Username/Password is below:

264

265

Server IP: 47.98.179.198

266

PSK : teddysun.com

267

Username : teddysun

268

Password : 5jFhqKyJAz

269

270

If you want to modify user settings, please use below command(s):

271

l2tp -a (Add a user)

272

l2tp -d (Delete a user)

273

l2tp -l (List all users)

274

l2tp -m (Modify a user password)

275

276

Welcome to visit our website: <https://teddysun.com/448.html>



## 2. 抓取I2tp协议流量

借助winpcap，使用c++代码编程，对ip不设限制，可以实现对I2tp协议流量的抓取。

```
=====
网卡1信息:
网卡名      : rpcap://\Device\NPF_{2AF860F4-1406-4E23-B9C1-F07A44420BF9}
网卡描述    : Network adapter 'TAP-Windows Adapter V9' on local host
回环接口    : 否
=====
网卡2信息:
网卡名      : rpcap://\Device\NPF_{B4A3EB86-3B24-4DE9-963E-888F2D274076}
网卡描述    : Network adapter 'Microsoft' on local host
回环接口    : 否
IP地址类型  : AF_INET
IP地址      : 100.80.58.57
掩码        : 255.254.0.0
广播地址    : 255.255.255.255
=====
网卡3信息:
网卡名      : rpcap://\Device\NPF_{C6EDC623-C96F-4A1B-B2F0-D6ABA628C983}
网卡描述    : Network adapter 'Microsoft' on local host
回环接口    : 否
=====
网卡4信息:
网卡名      : rpcap://\Device\NPF_{D7CBA4C6-A59E-4A02-BC94-5E5A2FB2412D}
网卡描述    : Network adapter 'ZeroTier Virtual Port' on local host
回环接口    : 否
IP地址类型  : AF_INET
IP地址      : 10.65.129.77
掩码        : 255.255.255.0
广播地址    : 255.255.255.255
=====
网卡5信息:
网卡名      : rpcap://\Device\NPF_{96BCA3F7-6F29-47A4-B86A-85097AC6926E}
网卡描述    : Network adapter 'Microsoft' on local host
回环接口    : 否
Enter the interface number (1-5): 2
```

## 3. 针对特定目标用户抓取I2tp协议流量，实时监控I2tp请求

借助winpcap，使用c++代码编程，通过对ip进行筛选限制，可以实现针对特定目标用户抓取I2tp协议流量，实时监控I2tp请求。



```
-----  
以太网协议分析:  
类型      : IPv4  
源地址    : 9c:54:c2:0d:50:02  
目的地址  : 64:bc:58:ff:ee:61  
-----
```

```
IPv4协议分析:  
版本号   : 4  
首部长度: 20 bytes  
服务类型 : 20  
总长度   : 294 bytes  
标识     : 35146  
生存时间 : 49  
协议     : UDP  
源地址   : 47.98.179.198  
目的地址 : 100.80.58.57  
-----
```

```
UDP协议分析:  
源端口   : 1701  
目的端口 : 1701  
数据长度 : 274  
校验和   : 0  
-----
```

```
L2TP协议分析:  
0x0  
类型      : 0 (数据信息)  
长度在位标志 : 0  
顺序字段在位标志 : 0  
偏移值在位标志 : 0  
优先级     : 0  
版本号     : 2  
隧道标识符 : 1  
会话标识符 : 65283  
-----
```

## 5.支持对特定目标的静载荷替换

借助winpcap，使用c++代码编程，能够实现对特定目标的静载荷替换。

```
Time: 16:48:11.283272, Length: 108  
64 bc 58 ff ee 61 9c 54 c2 0d 50 02 08 00 45 14  
00 5e 8a 11 00 00 31 11 7d b8 2f 62 b3 c6 64 50  
3a 39 06 a5 06 a5 00 4a 00 00 00 02 00 10 00 01  
ff 03 00 21 45 14 00 38 6c 68 00 00 6f 11 86 03  
7b 70 0b 1b c0 a8 12 02 3c 7d 27 09 00 24 88 97  
af f2 3d 3f e3 69 5e 44 61 c3 7c b8 e3 b7 2a 6c  
10 47 18 b7 c6 f2 11 f3 ef 49 ad a4  
-----
```

```
d.X..a.T..P...E.  
^....1.}./b..dP  
:9.....J.....  
...!E..8lh..o...  
{p.....<}'..$...  
..=?..i^Da.|.....  
.....  
-----
```

```
d.X..a.T..P...E.  
^....1.}./b..dP  
:9.....J.....  
...!E..8lh..o...  
{p.....<}'..$...  
..=?..i^Da.|.Hell  
o-World-TJ.  
-----
```

## 四、实验过程

### 1.搭建能够抓取 l2tp协议流量的环境

#### 一、配置使用阿里云服务器（CentOS 7.9 64位），设置安全组规则

← iZbp1fhwinn5dbl78cxlmZ 运行中

实例详情

监控

安全组

云盘

快照一致性组

快照

弹性网卡

定时与自动化任务

操作记录

健康诊断

事件

基本信息

实例ID

i-bp1fhwinn5dbl78cxlm

[远程连接](#) | [重置密码](#)

实例名称

iZbp1fhwinn5dbl78cxlmZ

[实例状态](#)

所在可用区

杭州 可用区K

付费类型

按量 按量付费转包年包月

健康状态

正常

[实例问题排查](#) | [实例问题排查历史](#)

实例状态

运行中 [停止](#) | [重启](#)

自动释放时间

2024年4月19日 20:52:00

[释放](#)

← sg-bp127s8x2jtsqv8anp9x

安全组详情

实例列表

辅助网卡

基本信息

安全组ID

sg-bp127s8x2jtsqv8anp9x

安全组名称

sg-bp127s8x2jtsqv8anp9x

网络

vpc-bp1m87ys3qvnf6hr6z7

组内连通策略

组内互通 [修改组内网络连通策略](#)

安全组类型

普通安全组

创建时间

2024年3月19日 19:42:43

描述

System created security group.

资源组

[新增](#)

标签

未绑定标签

访问规则

[导入安全组规则](#) | [导出](#) | [健康检查](#)

入方向

出方向

手动添加

快速添加

不合并

授权策略	优先级	协议类型	端口范围	授权对象	描述	创建时间	操作
<input type="checkbox"/> 允许	100	自定义 UDP	目的: 4500/4500	源: 0.0.0.0/0		2024年3月19日 19:50:25	<a href="#">编辑</a>   <a href="#">复制</a>   <a href="#">删除</a>
<input type="checkbox"/> 允许	100	自定义 UDP	目的: 1701/1701	源: 0.0.0.0/0		2024年3月19日 19:50:03	<a href="#">编辑</a>   <a href="#">复制</a>   <a href="#">删除</a>
<input type="checkbox"/> 允许	100	自定义 UDP	目的: 500/500	源: 0.0.0.0/0		2024年3月19日 19:49:35	<a href="#">编辑</a>   <a href="#">复制</a>   <a href="#">删除</a>

二、配置l2tp vpn

首先对服务器进行配置，执行命令进行搭建

×

t-hz04fg0nk7c02dc

实例列表

执行信息

命令名称

cmd\_2024-03-19\_20-05-06

命令ID

c-hz04fg0nk6zikn4

执行模式

立即执行

执行用户

root

命令类型

Shell

超时时间

60秒

命令内容

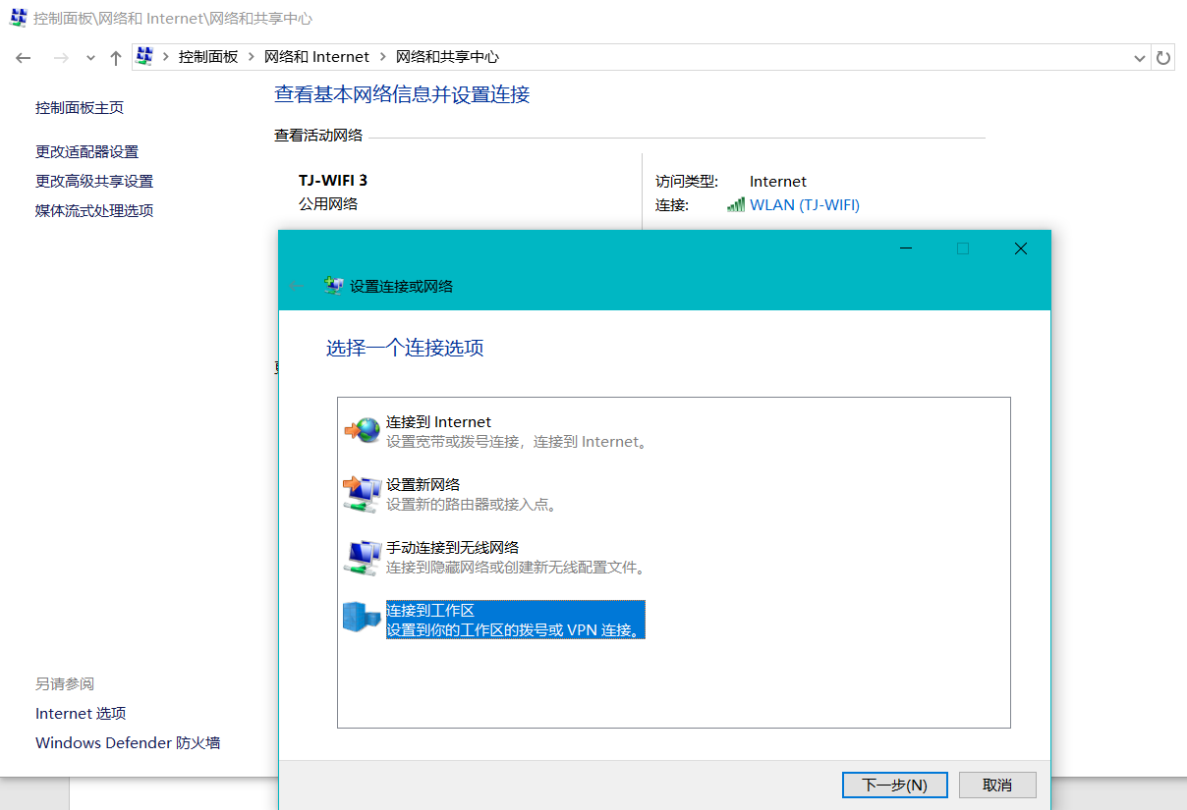
```
1 # 下载脚本
2 wget --no-check-certificate https://raw.githubusercontent.com/teddysun/across/master/l2tp.sh
3
4 # 赋予权限
5 chmod +x l2tp.sh
6 ./l2tp.sh
```

<input type="checkbox"/>	实例ID/名称	执行状态	IP地址	执行时间	ExitCode
<input type="checkbox"/>	i-bp1fhwpinnsdb178cxlm iZbp1fhwpinnsdb178cxlmZ	成功执行	47.98.179.198 (公) 172.24.65.91 (私有)	22秒	0

更新时间: 2024年3月24日 21:39:23

```
262 If there is no [FAILED] above, you can connect to your L2TP
263 VPN Server with the default Username/Password is below:
264
265 Server IP: 47.98.179.198
266 PSK      : teddysun.com
267 Username : teddysun
268 Password : 5jFhqKyJAz
269
270 If you want to modify user settings, please use below command(s):
271 l2tp -a (Add a user)
272 l2tp -d (Delete a user)
273 l2tp -l (List all users)
274 l2tp -m (Modify a user password)
275
276 Welcome to visit our website: https://teddysun.com/448.html
```

### 三、调整windows主机设置，连接vpn





键入要连接的 Internet 地址


网络管理员可提供此地址。

Internet 地址(I):

47.98.179.198

目标名称(E):

VPN

- ☐ 使用智能卡(S)
- ☒ 记住我的凭据(R)
-  ☒ 允许其他人使用此连接(A)  
这个选项允许可以访问这台计算机的人使用此连接。

创建(C)

取消

← 设置

主页

查找设置

网络和 Internet

状态

WLAN

拨号

VPN

飞行模式

移动热点

代理

状态

网络状态

TJ-WIFI 公用网络

你已连接到 Internet

如果你的流量套餐有限制，则你可以将此网络设置为按流量计费的连接，或者更改其他属性。

WLAN (TJ-WIFI)

最近 30 天内

71.61 GB

属性

数据使用量

显示可用网络

查看周围的连接选项。

高级网络设置

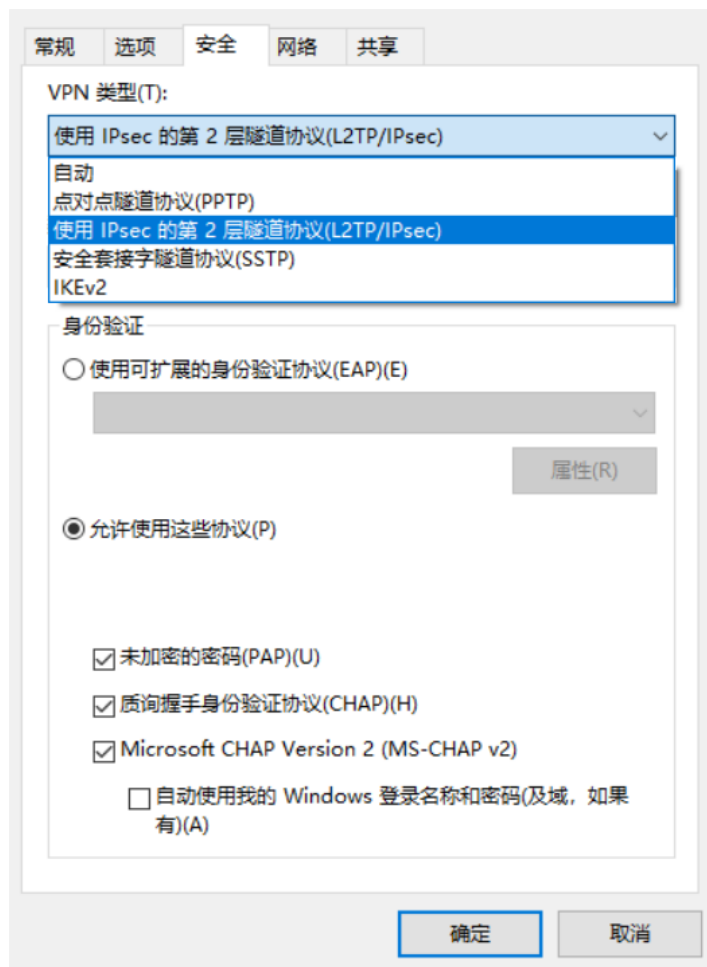
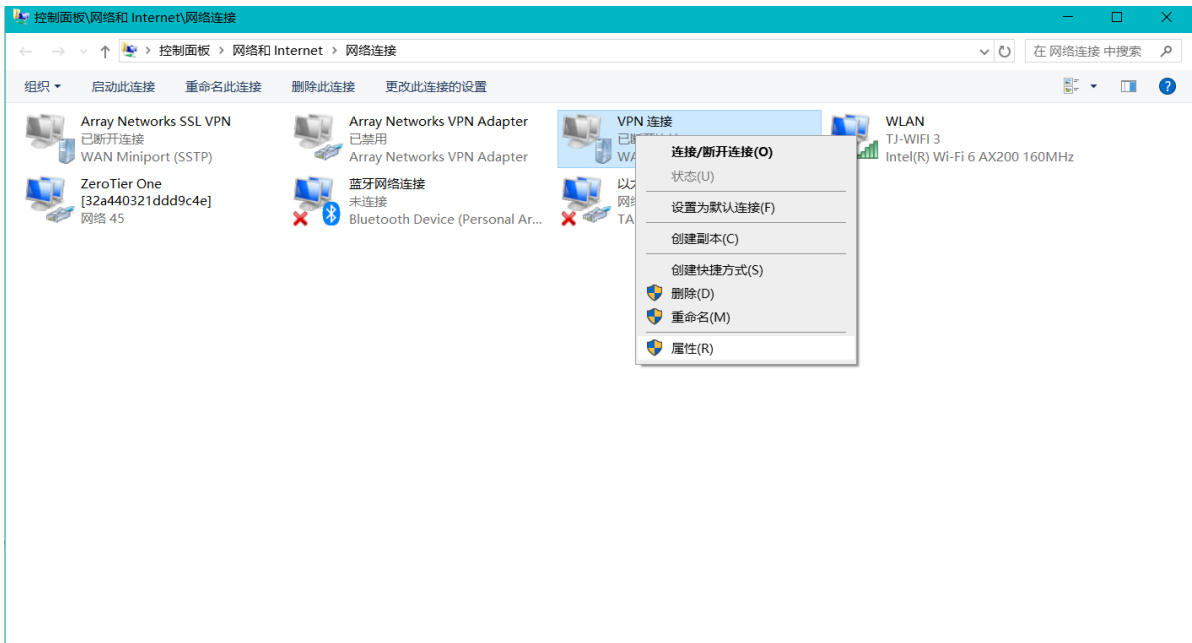
更改适配器选项

查看网络适配器并更改连接设置。

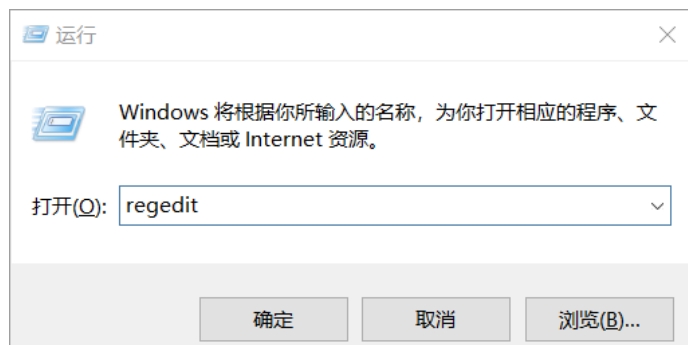
网络和共享中心

根据已连接到的网络 决定要共享的内容

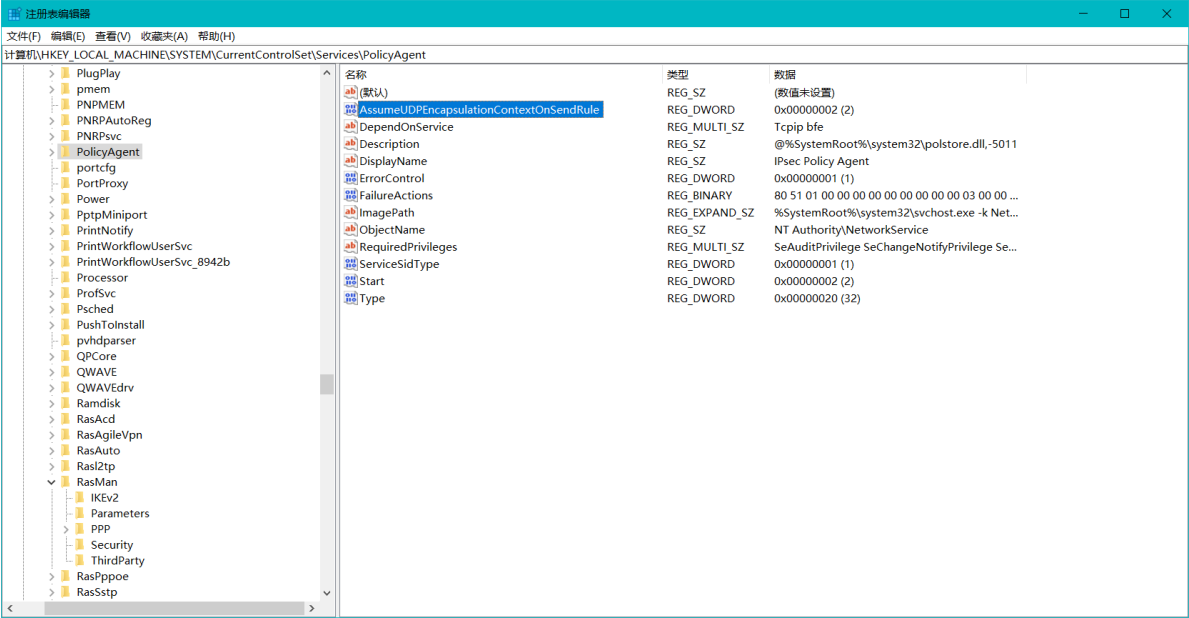
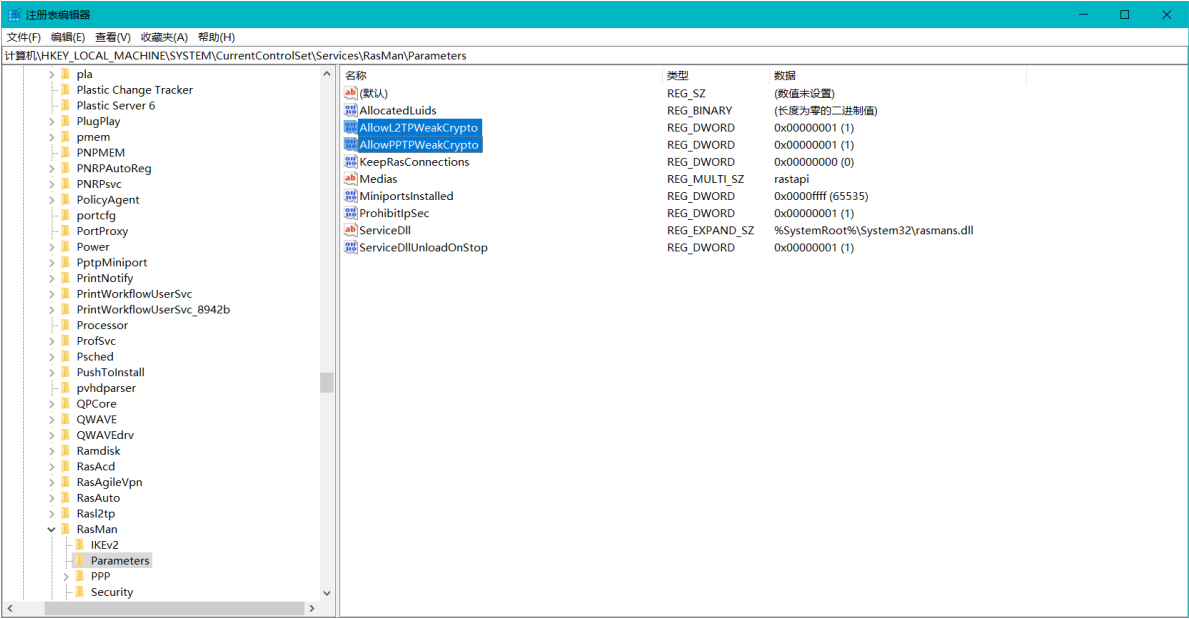




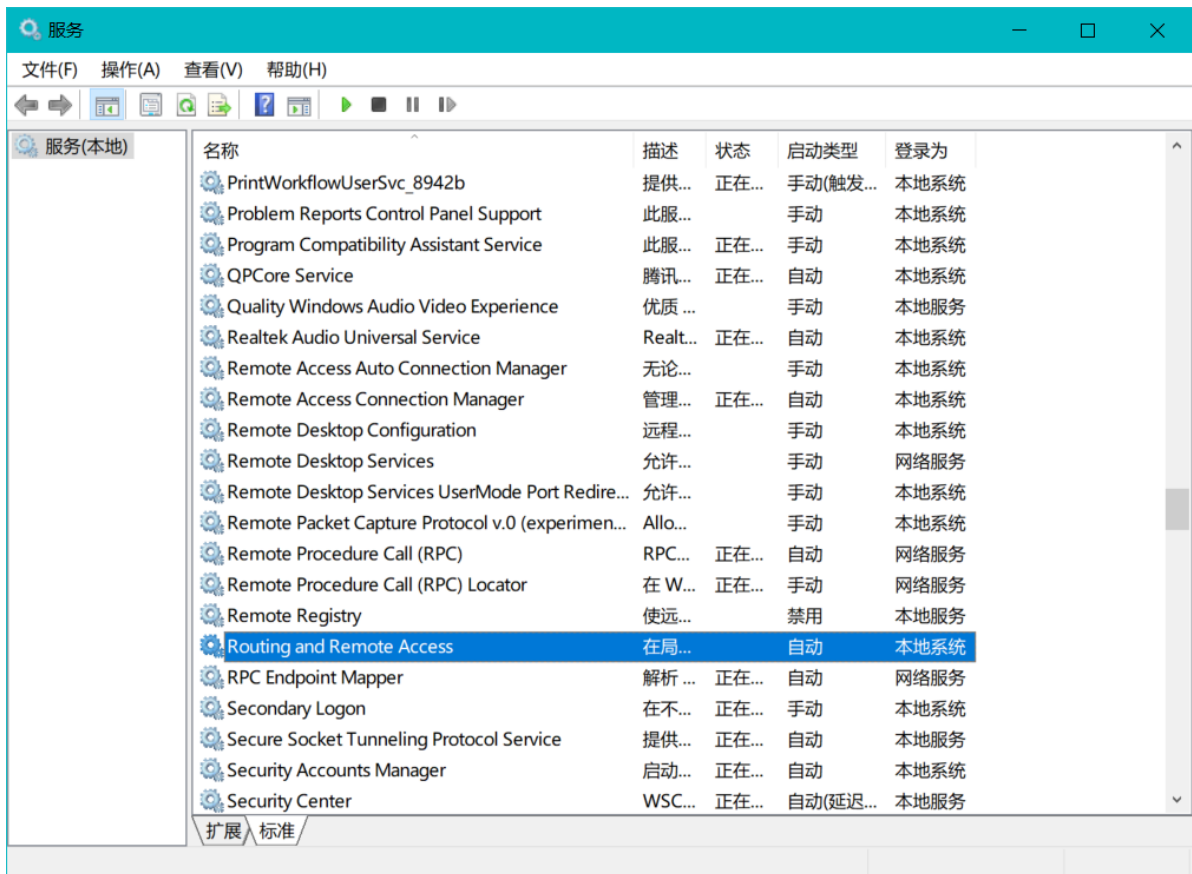
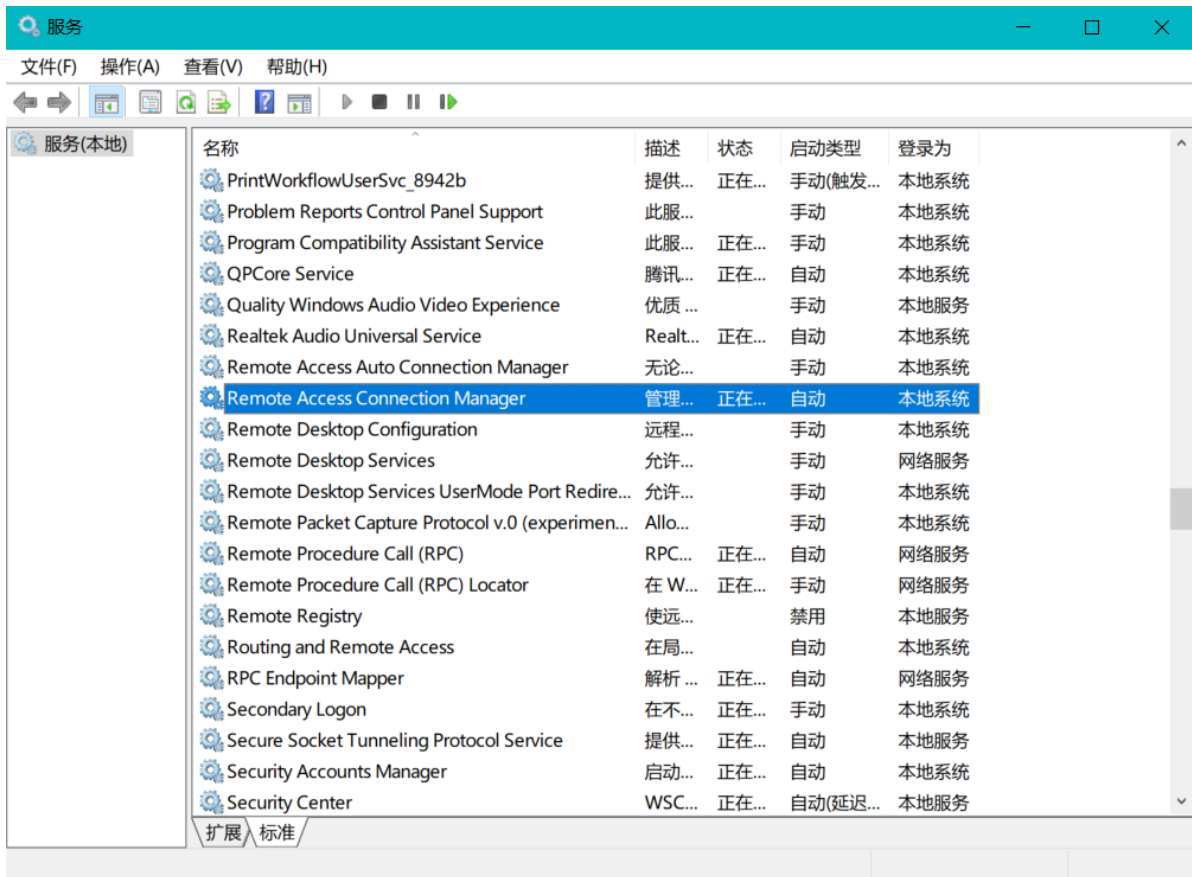
此时很可能仍然无法成功连接，继续对win10计算机进行设置：

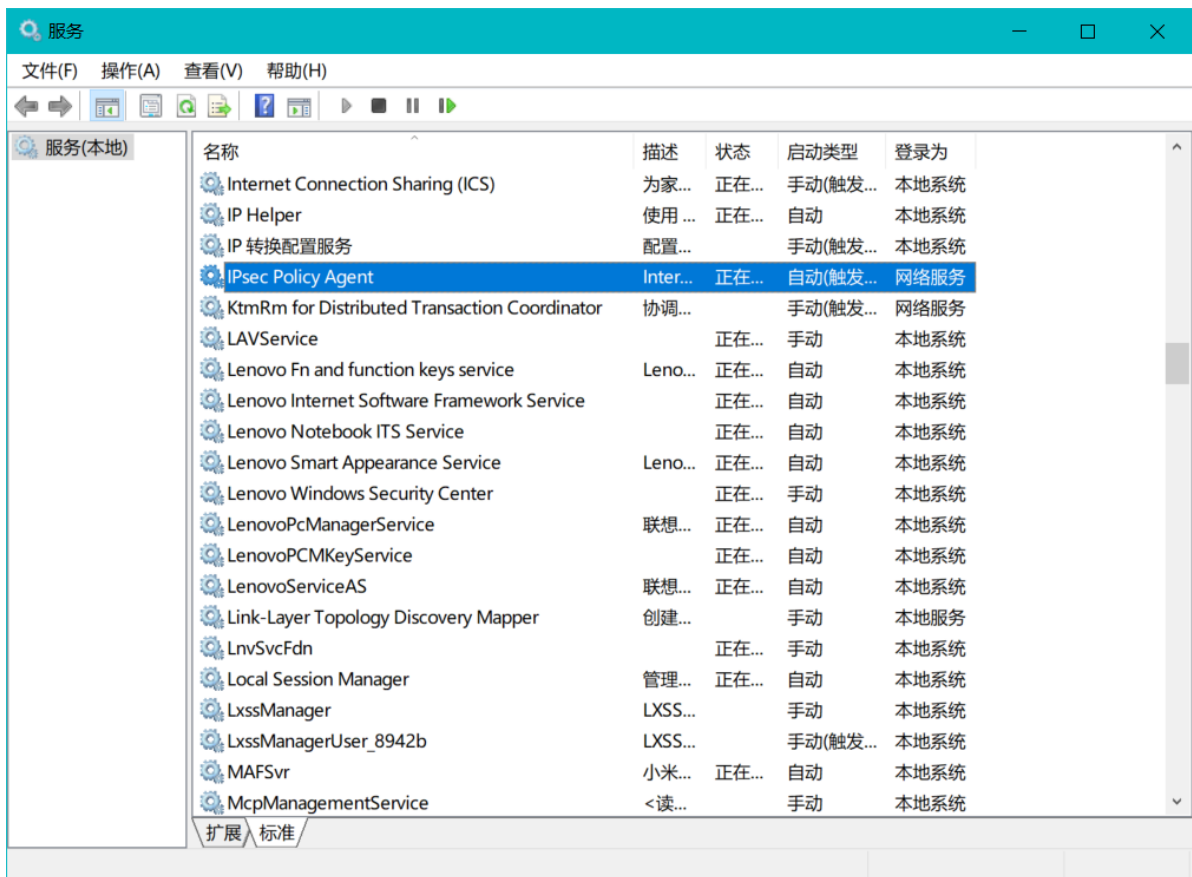


添加如下几项



并在服务中启用如下几项：





一般来说到此可以连接上vpn



## 2. 抓取I2tp协议流量

## 3. 针对特定目标用户抓取I2tp协议流量，实时监控I2tp请求

## 4. 并分析、还原、呈现其连接和业务载荷

## 5. 支持对特定目标的静载荷替换

借助winpcap，使用c++代码编程，能够实现以上功能。

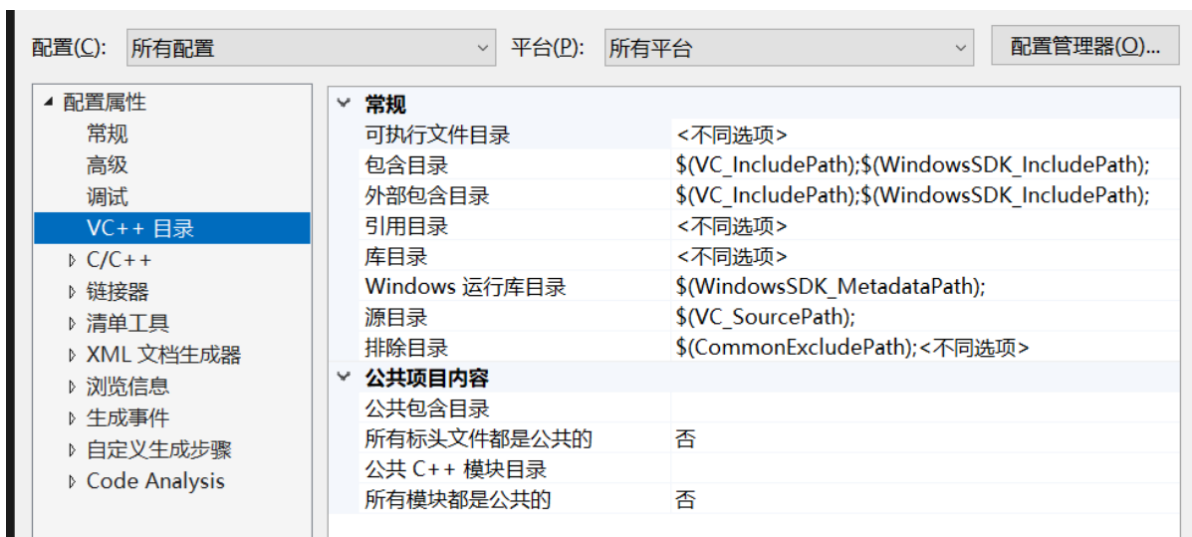
### 一、Visual Studio 2022配置winpcap

1. 首先下载安装 winpcap.exe, <http://www.winpcap.org/install/default.htm>

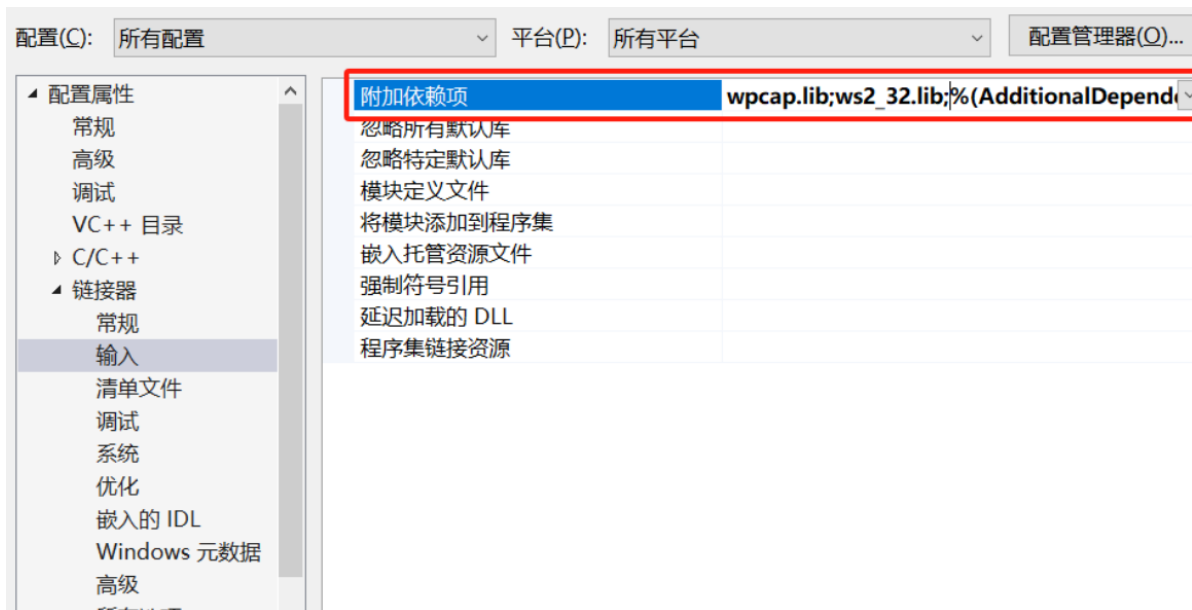
目的是安装相关驱动和 dll，安装完成之后基于 winpcap 的应用程序才能够正常运行，注意安装最后勾选上自动。

2. 下载 winpcap 的开发包，头文件和库文件: <http://www.winpcap.org/devel.htm>，解压之后主要是头文件和库文件。

3. VS2022新建一个项目，在项目属性中点击VC++目录，修改“包含目录”和“库目录”，包含目录修改为: \WpdPack\Include;\$(IncludePath)，库目录修改为: \WpdPack\Lib\x64;\$(LibraryPath)（根据2中下载的位置）。

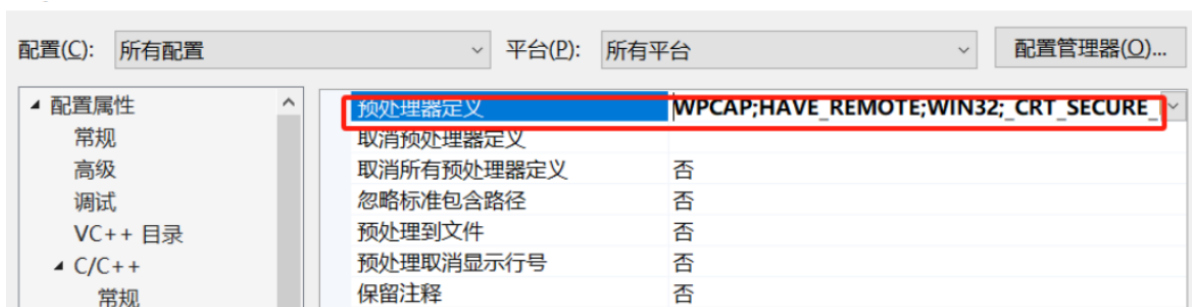


4.链接器 输入：wpcap.lib，ws2\_32.lib。



5.预处理器也适当修改为

“WPCAP;HAVE\_REMOTE;WIN32;\_CRT\_SECURE\_NO\_WARNINGS;\_WINSOCK\_DEPRECATED\_NO\_WARNINGS”。



此后可以开始写相关代码了。

## 二、协议分析

以太网：

以太网常用的帧格式有两种：Ethernet II 和 IEEE 802.3。它们之间的主要区别在于帧中包含的字段。

### 1. Ethernet II 格式：

- 在 Ethernet II 格式中，帧头中包含一个称为 Type 字段的字段。
- Type 字段的长度为 16 位，用于指示数据部分使用的上层协议，例如 IPv4、ARP、IPv6 等。
- 这种格式通常用于以太网上的 IP 数据包传输。

## 2. IEEE 802.3 格式：

- 在 IEEE 802.3 格式中，帧头中的相同位置用于存储长度字段。
- 长度字段指示了整个数据帧（包括帧头和数据部分）的长度，单位是字节。
- 这种格式通常用于以太网上的 IEEE 802.3 标准。

因此，区分这两种格式的关键是查看帧头中的特定字段位置：如果在帧头中存在一个 Type 字段，则是 Ethernet II 格式；如果在相同位置存在一个长度字段，则是 IEEE 802.3 格式。

IPv4：

1. **Version (版本)**：指示IP协议版本号，IPv4的版本号为4。
2. **Header Length (报头长度)**：描述IPv4报头的长度，以32位字（4字节）为单位。由于IPv4报头长度可变，这个字段指示了报头中有多少个32位字，通常是5（表示20字节），但如果报头中包含了选项字段，则会增加。
3. **Type Of Service (服务类别)**：简称TOS，用来实现IP QOS（服务质量）。它用于对不同类别的流量进行区分和处理，以满足不同流量的传输质量要求。
4. **Total Length (总长度)**：指示整个IP数据包的长度，包括IP报头和数据部分，以字节为单位。总长度不包括链路层帧头和帧尾。
5. **标识符、标记、分段偏移量**：这三个字段用于IPv4数据包的分片。当数据包太大而无法在网络上传输时，它会被分割成多个片段。这些字段用于标识和重新组装这些分段。标识符用于标识属于同一数据包的所有分段，标记字段用于指示是否还有后续分段，分段偏移量字段指示当前分段在原始数据包中的位置。
6. **Time to Live (存活时间)**：简称TTL，用于限制数据包在网络中传输的跳数。每经过一个路由器，TTL值会减一，当TTL值为0时，数据包会被丢弃。TTL的主要目的是防止数据包在网络中无限循环，消耗网络资源。
7. **Protocol (上层协议)**：指示数据包传输层所使用的协议，例如TCP、UDP、ICMP等。
8. **Header Checksum (报头校验和)**：通过对IP报头进行CRC校验得出的校验和，用于检测IP报头在传输过程中是否发生了错误或篡改。
9. **Source IP Address (源IP地址)**：描述数据包发送方的IP地址。
10. **Destination IP Address (目的IP地址)**：描述数据包接收方的IP地址。

UDP：

1. **源端口 (Source Port)**：占16位，用于标识发送端应用程序的端口号。接收端可以使用该端口号确定哪个应用程序应该接收数据。
2. **目的端口 (Destination Port)**：占16位，用于标识接收端应用程序的端口号。指示数据应该传递给哪个应用程序。
3. **长度 (Length)**：占16位，指示UDP数据报的总长度，包括UDP头部和数据部分。长度字段允许接收方区分UDP数据报的边界。
4. **校验和 (Checksum)**：占16位，用于检测UDP数据报在传输过程中是否发生了错误。校验和字段基于部分IP头信息、UDP头部和数据部分的内容计算得出，接收端会使用校验和来验证数据的完整性，以确保数据在传输过程中未被篡改。

L2TP：

### 1. 版本与协议类型字段：

- L2TP 协议头中的第一个字节包含了版本和协议类型字段。通过检查此字段，可以确定数据包的类型，即控制消息还是数据消息。

### 2. 标志字段解析：

- L2TP 头部中的标志字段（TLSSOOV字段）包含了控制消息和数据消息的各种标志位，用于指示消息的类型和属性。这些标志位包括：
  - 控制消息类型（T）
  - 长度在位标志（L）
  - 顺序字段在位标志（S）
  - 偏移值在位标志（O）
  - 优先级（P）

### 3. 版本号解析：

- L2TP 头部中的版本号字段（XXXXVER字段）表示 L2TP 协议的版本。

### 4. 长度字段：

- 如果长度在位标志位（L）为 1，则会在 L2TP 头部中包含消息的总长度。这个字段指示了整个消息（控制消息或数据消息）的长度。

### 5. 隧道标识符和会话标识符：

- 隧道标识符字段（Tunnel ID）和会话标识符字段（Session ID）用于标识 L2TP 隧道和会话。它们在建立 L2TP 隧道和会话时分配，并在后续的消息中用于识别不同的隧道和会话。

### 6. 顺序字段：

- 如果顺序字段在位标志位（S）为 1，则消息中可能包含顺序字段，用于指示消息的顺序号和下一消息的顺序号。这些字段通常在控制消息中使用。

### 7. 偏移值字段：

- 如果偏移值在位标志位（O）为 1，则消息中可能包含偏移值字段，用于指示数据的偏移量。这个字段通常在控制消息中使用。

### 8. 其他信息：

- 根据 L2TP 头部中不同标志位的值，可以解析出其他相关信息，如消息的优先级等。

## 三、代码展示

```
#define WIN32
#define _CRT_SECURE_NO_WARNINGS
#define _WINSOCK_DEPRECATED_NO_WARNINGS
#define MAX_PROTO_TEXT_LEN 16 // 子协议名称最大长度
#define MAX_PROTO_NUM 12 // 子协议数量
#define IPSEC_PORT 4500
#define L2TP_PORT 1701
#pragma comment(lib,"wpcap.lib")
#pragma comment(lib,"ws2_32.lib")
using namespace std;

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pcap.h>
#include "l2tp_test.h"

char a[15] = "Hello-world-TJ";
```

```

#define IPTOSBUFFERS 12

// 将unsigned long型的IP转换为字符串类型的IP
char* iptos(u_long in) {
    static char output[IPTOSBUFFERS][3 * 4 + 3 + 1];
    static short which = 0;
    u_char* p = (u_char*)&in;
    which = (which + 1 == IPTOSBUFFERS ? 0 : which + 1);

    // 格式化IP地址字符串
    sprintf(output[which], "%d.%d.%d.%d", p[0], p[1], p[2], p[3]);
    return output[which];
}

// 输出网络接口信息
void print_interface_info(pcap_if_t* device, int num) {
    pcap_addr_t* a;
    printf("\n===== \n");
    printf("网卡%d信息: \n", num);
    printf("网卡名      : %s\n", device->name);
    printf("网卡描述      : %s\n", device->description ? device->description : "No description available");
    printf("回环接口      : %s\n", (device->flags & PCAP_IF_LOOPBACK) ? "是" : "否");
    // IP地址
    for (a = device->addresses; a; a = a->next) {
        switch (a->addr->sa_family) {
            case AF_INET:
                printf("IP地址类型  : AF_INET\n");
                if (a->addr)
                    printf("IP地址      : %s\n", iptos(((struct sockaddr_in*)a->addr)->sin_addr.s_addr));
                if (a->netmask)
                    printf("掩码        : %s\n", iptos(((struct sockaddr_in*)a->netmask)->sin_addr.s_addr));
                if (a->broadaddr)
                    printf("广播地址    : %s\n", iptos(((struct sockaddr_in*)a->broadaddr)->sin_addr.s_addr));
                if (a->dstaddr)
                    printf("目标地址    : %s\n", iptos(((struct sockaddr_in*)a->dstaddr)->sin_addr.s_addr));
                break;
            default:
                //printf("Address Family Name: Unkown\n");
                break;
        }
    }
}

// 输出UDP协议分析结果
void print_udp_info(udp_header* udphheader) {
    printf("\n===== \n");
    printf("UDP协议分析: \n");
    printf("源端口   : %d\n", ntohs(udphheader->sport));
    printf("目的端口 : %d\n", ntohs(udphheader->dport));
}

```



```

    printf("数据长度: %d\n", ntohs(udpheader->len));
    printf("校验和 : %d\n", ntohs(udpheader->crc));
}

// 输出IPv4协议分析结果
void print_ipv4_info(ip_header* ipheader) {
    printf("\n=====n");
    printf("IPv4协议分析: \n");
    printf("版本号 : %d\n", (ipheader->ver_ihl & 0xf0) >> 4);
    printf("首部长度: %d bytes\n", (ipheader->ver_ihl & 0xf) * 4);
    printf("服务类型: %d\n", ipheader->tos);
    printf("总长度 : %d bytes\n", ntohs(ipheader->tlen));
    printf("标识 : %d\n", ntohs(ipheader->identification));
    printf("生存时间: %d\n", ipheader->ttl);
    printf("协议 : %s\n", ipheader->proto == IPPROTO_UDP ? "UDP" : "*");
    printf("源地址 : %d.%d.%d.%d\n", ipheader->saddr.byte1, ipheader->saddr.byte2,
ipheader->saddr.byte3, ipheader->saddr.byte4);
    printf("目的地址: %d.%d.%d.%d\n", ipheader->daddr.byte1, ipheader->daddr.byte2,
ipheader->daddr.byte3, ipheader->daddr.byte4);
}

// 输出以太网协议分析结果
void print_ethernet_info(ethernet_header* ethheader) {
    printf("\n=====n");
    printf("以太网协议分析: \n");
    printf("类型 : %s\n", ntohs(ethheader->type) == 0x0800 ? "IPv4" :
"Other");
    printf("源地址 : %02x:%02x:%02x:%02x:%02x:%02x\n",
ethheader->src_mac_addr.byte1, ethheader->src_mac_addr.byte2,
ethheader->src_mac_addr.byte3,
ethheader->src_mac_addr.byte4, ethheader->src_mac_addr.byte5,
ethheader->src_mac_addr.byte6);
    printf("目的地址: %02x:%02x:%02x:%02x:%02x:%02x\n",
ethheader->des_mac_addr.byte1, ethheader->des_mac_addr.byte2,
ethheader->des_mac_addr.byte3,
ethheader->des_mac_addr.byte4, ethheader->des_mac_addr.byte5,
ethheader->des_mac_addr.byte6);
}

// 解析L2TP协议
int decode_l2tp(char* l2tpbuf)
{
    struct l2tp_header* pl2tpheader;
    pl2tpheader = (l2tp_header*)l2tpbuf;
    u_short t, l, s, o;
    t = (pl2tpheader->t1xxsxp & 0x80) >> 7;
    l = (pl2tpheader->t1xxsxp & 0x40) >> 6;
    s = (pl2tpheader->t1xxsxp & 0x08) >> 3;
    o = (pl2tpheader->t1xxsxp & 0x02) >> 1;

    printf("\n=====n");
    printf("L2TP协议分析: \n");
    printf("0x%x\n", pl2tpheader->t1xxsxp);
    printf("类型 : %s\n", t ? "1(控制信息)" : "0(数据信息)");
    printf("长度在位标志 : %d\n", l);
}

```

```

printf("顺序字段在位标志:%d\n", s);
printf("偏移值在位标志  :%d\n", o);
printf("优先级           :%d\n", p12tpheader->tlxxsxop & 0x01);
printf("版本号           :%d\n", p12tpheader->xxxxver & 0x0f);
if (l == 1) { // 长度在位标志为1
    printf("消息总长度       :%d\n", ntohs(p12tpheader->length));
}
printf("隧道标识符         :%d\n", ntohs(p12tpheader->tunnel_id));
printf("会话标识符         :%d\n", ntohs(p12tpheader->session_id));
if (s == 1) { // 顺序字段在位标志为1
    printf("当前消息顺序号   :%d\n", ntohs(p12tpheader->ns));
    if (t == 1) { // 控制信息nr才有意义
        printf("下一消息顺序号 :%d\n", ntohs(p12tpheader->nr));
    }
}
if (l == 1) { // 偏移值在位标志为1
    printf("偏移量           :%d\n\n", ntohs(p12tpheader->offset));
}
return true;
}

// 解析UDP协议
void decode_udp(char* udpbuf) {
    udp_header* pudpheader = (udp_header*)udpbuf;
    print_udp_info(pudpheader);

    // 判断是否为L2TP协议，如果是则进一步解析
    if (ntohs(pudpheader->sport) == 1701 && ntohs(pudpheader->dport) == 1701) {
        decode_l2tp(udpbuf + sizeof(udp_header));
    }
}

// 解析IPv4协议
void decode_ipv4(char* ipbuf) {
    ip_header* ipheader = (ip_header*)ipbuf;
    print_ipv4_info(ipheader);

    // 如果是UDP协议，则进一步解析
    if (ipheader->proto == IPPROTO_UDP) {
        decode_udp(ipbuf + ((ipheader->ver_ihl & 0xf) * 4)); // 移动到UDP头部的位置
    }
}

// 解析以太网帧
void decode_ethernet(char* etherbuf) {
    ethernet_header* ethheader = (ethernet_header*)etherbuf;
    print_ethernet_info(ethheader);

    // 如果是IPv4协议，则进一步解析
    if (ntohs(ethheader->type) == 0x0800) {
        decode_ipv4(etherbuf + sizeof(ethernet_header));
    }
}

// 包处理回调函数，对于每个嗅探到的数据包

```

```

void packet_handler(u_char* param, const struct pcap_pkthdr* header, const
u_char* pkt_data) {
    // 输出时间戳
    struct tm* ltime;
    char timestr[16];
    time_t local_tv_sec = header->ts.tv_sec;
    ltime = localtime(&local_tv_sec);
    strftime(timestr, sizeof timestr, "%H:%M:%S", ltime);
    printf("Time: %s.%.6d, Length: %d\n", timestr, header->ts.tv_usec, header-
>len);
    char mypkt[1000];

    // 打印数据包内容
    int len = header->caplen;
    for (int i = 0; i < len; i++) {
        if(i<=93)
            mypkt[i] = pkt_data[i];
        printf("%.2x ", pkt_data[i]);
        if ((i + 1) % 16 == 0) printf("\n");
    }
    printf("\n");

    printf("\n-----\n");

    for (int i = 1; i < len; i++)
    {
        char temp = mypkt[i - 1];
        if (temp >= 32 && temp <= 126) {
            printf("%c", temp);
        }
        else {
            printf(".");
        }
        if ((i % 16) == 0)
            printf("\n");
    }
    printf("\n-----\n");

    for (int i = 93; i < 93 + 15; i++) {
        mypkt[i - 1] = a[i - 93];
    }

    for (int i = 1; i < 93 + 15; i++)
    {
        if (i == 93)
            SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
FOREGROUND_INTENSITY | FOREGROUND_GREEN | FOREGROUND_BLUE);
        char temp = mypkt[i - 1];
        if (temp >= 32 && temp <= 126) {
            printf("%c", temp);
        }
        else {
            printf(".");
        }
        if ((i % 16) == 0)

```

```

        printf("\n");
    }
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
        FOREGROUND_INTENSITY | FOREGROUND_GREEN | FOREGROUND_BLUE | FOREGROUND_RED);

    decode_ethernet((char*)pkt_data);

    printf("\n-----\n");
    printf("\n\n");

    // 解析以太网帧
    decode_ethernet((char*)pkt_data);
}

int main(int argc, const char* argv[]) {
    pcap_if_t* alldevs;
    pcap_if_t* d;
    int inum;
    char errbuf[PCAP_ERRBUF_SIZE];
    u_int netmask;

    // 获取本机设备列表
    if (pcap_findalldevs_ex((char*)PCAP_SRC_IF_STRING, NULL, &alldevs, errbuf)
        == -1) {
        fprintf(stderr, "Error in pcap_findalldevs: %s\n", errbuf);
        return -1;
    }

    // 打印设备列表，并让用户选择一个接口
    int i = 0;
    for (d = alldevs, i = 0; d; d = d->next, i++) {
        print_interface_info(d, i + 1);
    }
    if (i == 0) {
        printf("No interfaces found! Make sure winPcap is installed.\n");
        return -1;
    }

    printf("Enter the interface number (1-%d): ", i);
    scanf("%d", &inum);

    if (inum < 1 || inum > i) {
        printf("Interface number out of range.\n");
        pcap_freealldevs(alldevs);
        return -1;
    }

    // 跳转到已选中的适配器
    for (d = alldevs, i = 0; i < inum - 1; d = d->next, i++);

    // 打开设备
    pcap_t* adhandle = pcap_open(d->name, 65536, PCAP_OPENFLAG_PROMISCUOUS,
        1000, NULL, errbuf);
    if (adhandle == NULL) {

```

```

        fprintf(stderr, "Unable to open the adapter. %s is not supported by
winPcap\n", d->name);
        pcap_freealldevs(alldevs);
        return -1;
    }

    // 检查链接层
    if (pcap_datalink(adhandle) != DLT_EN10MB) {
        fprintf(stderr, "This program works only on Ethernet networks.\n");
        pcap_freealldevs(alldevs);
        return -1;
    }

    // 获取网络接口的掩码
    if (d->addresses != NULL) {
        netmask = ((struct sockaddr_in*)(d->addresses->netmask))-
>sin_addr.s_un.s_addr;
    } else {
        netmask = 0xffffffff; // 默认C类网络
    }

    // 设置过滤器
    // 筛选出发送的数据(src) 和 到达的数据(dst)
    char packet_filter[] = "src host 47.98.179.198 and src port 1701 and dst
port 1701";
    struct bpf_program fcode;
    if (pcap_compile(adhandle, &fcode, packet_filter, 1, netmask) < 0 ||
pcap_setfilter(adhandle, &fcode) < 0) {
        fprintf(stderr, "Error setting the filter.\n");
        pcap_freealldevs(alldevs);
        return -1;
    }

    printf("\nListening on %s...\n", d->description);
    pcap_freealldevs(alldevs);

    // 开始嗅探
    pcap_loop(adhandle, 5, packet_handler, NULL);
    pcap_close(adhandle);
    return 0;
}

```

## 五、实验评价

对我来说几乎是一窍不通，举步维艰，我在极大程度地参考了github上学长的内容。

参照学长的流程，又在网上查找搭建l2tp协议vpn的方法，解决win10无法连接vpn的问题，查找使用winpcap的方法，尝试和修改代码。

由于安卓13不再支持l2tp协议vpn的连接，因此换用笔记本来进行测试。

另外，提供的可执行文件正常情况下应该不能直接运行，需要在代码中的过滤器处进行设置

```
// 设置过滤器
// 筛选出发送的数据(src) 和 到达的数据(dst)
char packet_filter[] = "src host 47.98.179.198 and src port 1701 and dst port 1701";
struct bpf_program fcode;
if (pcap_compile(adhandle, &fcode, packet_filter, 1, netmask) < 0 || pcap_setfilter(adhandle, &fcode) < 0) {
    fprintf(stderr, "Error setting the filter.\n");
    pcap_freealldevs(alldevs);
    return -1;
}
```

## 六、参考内容

- [1] ChestnutSilver. 同济大学信息安全原理课程作业[EB/OL]. 2023[2024.3.20]. <https://github.com/ChestnutSilver/l2tp-analysis>
- [2] [使用CentOS 7实例配置PPTP VPN服务端到客户端的连接-阿里云帮助中心 \(aliyun.com\)](#)
- [3] [\[Win10、Win11系统创建VPN连接以及需要进行的相关设置\\_prohibitipsec-CSDN博客\] \(https://blog.csdn.net/qq1507171150/article/details/131890722\)](#)
- [4] [VS2022配置WinPcap开发 vs winpcap-CSDN博客](#)
- [5] [【好奇心驱动力】内网搭建L2TP服务器及抓包数据分析 免费l2tp服务器地址-CSDN博客](#)
- [6] chatgpt <https://chat.openai.com/>
- [7] <https://community.ui.com/questions/L2TP-unsecure-update-to-IKEv2-VPN-recommended/353e37b3-30bf-427c-a4fd-33dbcd8badddc>