



## 信息安全原理设计报告

### Project 1: L2TP 协议实验与设计

学 号： 2053182

姓 名： 王润霖

专 业： 信息安全

二〇二三年三月

### 小组分工

任务点	内容			完成情况
1	基于 windows 或 linux 抓取 12tp 协议流量。			√
2	针对每一个目标用户，实时监控其 12tp 请求。			√
3	分析、还原、呈现目标用户连接和业务载荷。			√
4	支持对特定目标的静载荷替换。			√
学号	姓名	主要任务点	贡献率	
2052338	鲍宇轩	1、2、3	50%	
2053182	王润霖	1、2、4	50%	

我们在各自的电脑和手机上分别搭建了能够抓取 12tp over IPsec 协议流量的环境（任务点 1），并编写了能够针对每一个目标用户实时监控 12tp 请求的基本代码（任务点 2），并验证了其正确性。

鲍宇轩编写代码，拆解并分析了 12tp 请求数据，同时拆解并分析了更高层次的以太网协议、Ipv4 协议和 UDP 协议，分析、还原、呈现目标用户连接和业务载荷（任务点 3）。

王润霖在抓取到 12tp over IPsec 协议流量之外，配置了能够抓取未加密的 12tp 数据包的环境，翻译了 12tp 流量中的 data 数据，对特定目标进行了静载荷替换（任务点 4）。

**截至 2023 年 3 月 17 日：**

基于两个电脑（windows 系统）和一个云主机，实现了 12tp 协议流量的抓取、分析、还原、呈现和静载荷替换的工作。

这时，我们使用其中一个电脑替代了手机。

这是因为我们的云主机能够提供 IPsec 加密服务，而通过电脑强制禁用 IPsec 加密的方法，获得了未经加密的 12tp 报文。而手机无法禁用 IPsec，只能拿到加密的报文。

因此，为了能够拆解 12tp 报文，我们用电脑替代了手机。

**更新 2023 年 4 月 1 日：**

我们在云主机成功搭建不使用 IPsec 的 12tp vpn！

这使得手机能够连入 12tp vpn，通过电脑抓取手机的 12tp 协议流量！

## 第一部分 实验内容

### 1.1 实验题目

基于 C++设计并实现一个简单系统，实现：

- (1) 基于 windows 或 linux 抓取 l2tp 协议流量；
- (2) 针对每一个目标用户，实时监控其 l2tp 请求，并分析、还原、呈现其连接和业务载荷；

- (3) 支持对特定目标的静载荷替换。

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。

### 1.2 实验方案与结果概述

#### (1) 基于 windows 或 linux 抓取 l2tp 协议流量

我们基于 windows10 系统和阿里云服务器（CentOS），搭建了能够抓取 l2tp 协议流量的服务端和电脑环境。在服务端，使用 l2tp over IpSec 协议，开放 1701 端口和 4500 端口，在 windows 系统，通过修改注册表和管理策略，禁用 IpSec 功能，使得协议流量均为未加密的 l2tp 协议流量，便于后续实验分析。编写 C++程序，并配置使用 WinPcap4.1.3，抓取协议流量。

同时，通过观察对比实验，验证了第一问的正确性。

```
time:17:13:00,244209 len:94
3e 58 c2 b8 2e a5 8c c8 4b d4 e0 99 08 00 45 00
00 50 46 e5 00 00 80 11 77 53 c0 a8 89 69 8b c4
a6 8e 06 a5 06 a5 00 3c 67 59 40 02 00 34 b7 b3
fa 36 ff 03 00 21 45 00 00 28 da f8 40 00 80 06
3d 1a c0 a8 12 05 08 08 08 08 de 2a 00 35 a3 0c
13 80 32 07 52 23 50 10 02 03 b1 fd 00 00
```

（图 1：实验结果-基于 windows 或 linux 抓取 l2tp 协议流量）

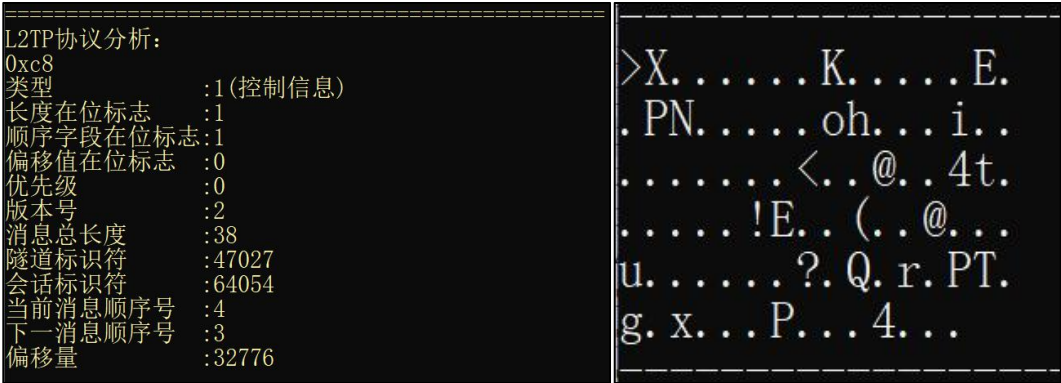
#### (2) 针对每一个目标用户，实时监控其 l2tp 请求，并分析、还原、呈现其连接和业务载荷

首先，我们实时监控了手机的 l2tp 请求。由于手机无法禁用 IpSec 功能，因此通过 l2tp over IpSec 协议与服务端进行数据交换，通过抓取其数据包，使用的端口为 4500，证明了其为加密过的报文。

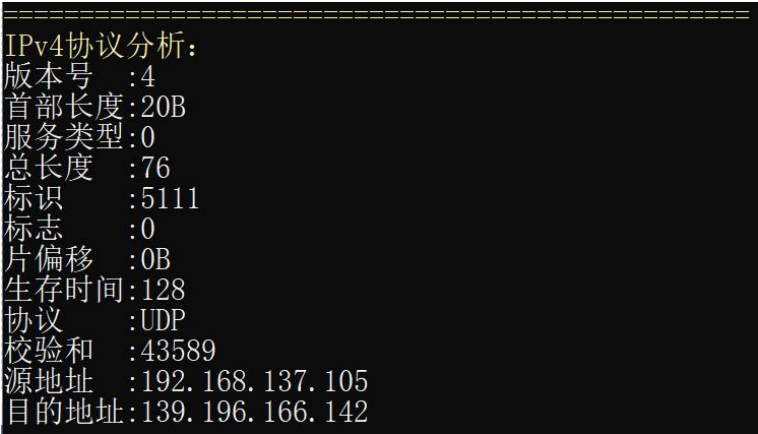
为了进一步拆解 l2tp 请求数据，我们使用另一台电脑模拟手机环境，在该电脑同样禁用 IpSec，这两台电脑通过热点相互连接，并使用 l2tp vpn，使得能够对目标用户的 l2tp 请求拆解，分析、还原并呈现其连接和业务载荷，包括类型、长度在位标志、顺序字段在位标志、偏移值在位标志、优先级、版本号、消息总长度、隧道标识符、会话标识符、当前消息顺序号、下一消息顺序号、偏移量等信息。

我们也拆解并分析了更高层次的以太网协议、Ipv4 协议和 UDP 协议，详细呈现了其连接和业务载荷。

同时，通过 wireshark 抓取数据包的对比，验证了第二问的正确性。



(图 2：实验结果-针对每一个目标用户，实时监控其 l2tp 请求，分析、还原、呈现其连接和业务载荷-l2tp 协议)

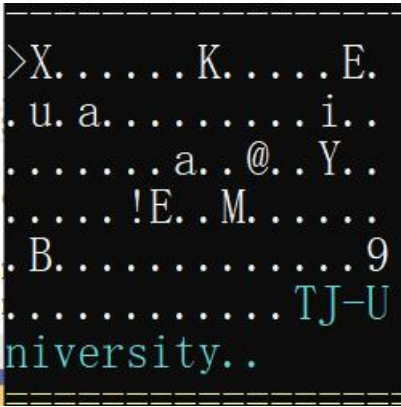


(图 3：实验结果-针对每一个目标用户，实时监控其 l2tp 请求，分析、还原、呈现其连接和业务载荷-IPv4 协议)

(3) 支持对特定目标的静载荷替换

我们通过对 l2tp 报文封装结构的分析，找到数据包的静载荷部分，并将特定数据包的静载荷替换为自己的 data 数据，替换完成后，再按照 l2tp 报文结构进行封装，产生一个新的数据包。

同时，使用第 2 问的拆解算法，对新的数据包重新拆解，得到了我们替换的新静载荷。



(图 4：支持对特定目标的静载荷替换)

第二部分 实验过程

2.1 基于 windows 或 linux 抓取 12tp 协议流量

2.1.1 搭建实验服务器环境

我们对服务器的搭建进行了多次实验，得到了两种可行的方法。

(1) 方案一：使用腾讯云 Windows Server 系统

参考网络教程，配置腾讯云服务器，该过程较为繁琐。

(2) 方案二：使用阿里云 CentOS 系统

参考网络教程，配置阿里云服务器。

根据实验要求，需要实时抓取用户的 12tp 协议流量，因此，我们搭建提供 12tp vpn 的服务器，并在电脑与服务端进行连接。



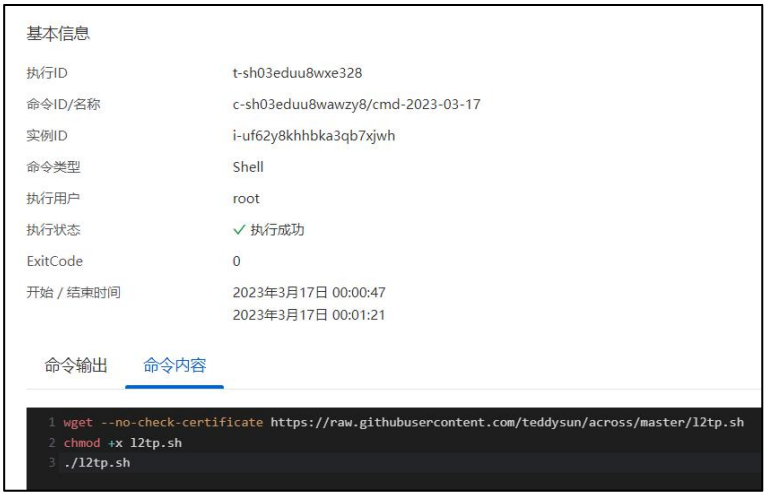
(图 5：阿里云服务器实例)

为了使得 12tp 协议流量正常，需要配置安全组出入站规则。我们分别对手机和电脑的协议流量进行了监控，因此，12tp 端口 1701、IpSec 端口 4500、500 端口均要打开。



(图 6：服务端安全组规则)

使用开源脚本，对服务端进行配置，记录 ip 地址，并设置用户名、密码以及 IpSec 的预共享密钥。



(图 7：配置服务端基本信息)

### 2.1.2 配置 Windows 系统环境，连接 vpn

#### (1) 修改注册表和管理策略，禁用 IpSec

修改注册表信息如下表所示。

(表 1：修改注册表信息)

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RasMan\Parameters			
注册表项	值	类型	含义
AllowL2TPWeakCrypto	1	DWORD	允许 L2TP 弱密码
AllowPPTPWeakCrypto	1	DWORD	允许 PPTP 弱密码
ProhibitIpSec	1	DWORD	禁用 IpSec
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\PolicyAgent			
注册表项	值	类型	含义
AssumeUDPEncapsulation ContextOnSendRule	2	DWORD	表示无论客户端还是 LNS 端位于 NAT 网络之后均可建立 VPN 连接

修改管理策略：

Routing and Remote Acces 自动；

Remote Access Connection Manager 自动；

IPsec Policy Agent 自动。

#### (2) 电脑连接 12tp vpn

在电脑设置适配器属性和 vpn 属性，根据用户名、密码和预共享密钥，连接 12tp vpn，过程不再赘述。



(图 8：VPN 连接成功)

### 2.1.3 通过 WinPcap 抓取 12tp 协议流量

配置使用 WinPcap，编写程序捕获数据包，算法原理如下：

(1) 打开网卡，并设为混杂模式。

(2) 回调函数 Network Tap 在得到监听命令后，从网络设备驱动程序处收集数据包把

监听到的数据包负责传送给过滤程序。

(3) 当 Packet filter 监听到有数据包到达时, NDIS 中间驱动程序首先调用分组驱动程序, 该程序将数据传递给每一个参与进程的分组过滤程序。

(4) 然后由 Packet filter 过滤程序决定哪些数据包应该丢弃, 哪些数据包应该接收, 是否需要将接收到的数据拷贝到相应的应用程序。

(5) 通过分组过滤器后, 将数据未过滤掉的数据包提交给核心缓冲区。然后等待系统缓冲区满后, 再将数据包拷贝到用户缓冲区。监听程序可以直接从用户缓冲区中读取捕获的数据包。

(6) 关闭网卡。

```
// 打开设备
if ((adhandle = pcap_open(d->name,           // 设备名
    65536,                                     // 要捕捉的数据包的部分, 65535 保证能捕获到不同数据链
    PCAP_OPENFLAG_PROMISCUOUS,              // 混杂模式
    1000,                                     // 读取超时时间
    NULL,                                     // 远程机器验证
    errbuf                                     // 错误缓冲池
)) == NULL) {
    fprintf(stderr, "\nUnable to open the adapter. %s is not supported by
WinPcap\n", d->name);
    pcap_freealldevs(alldevs);
    return -1;
}
// 检测链接层
if (pcap_datalink(adhandle) != DLT_EN10MB) {
    fprintf(stderr, "\n 此程序只能运行在以太网上.\n");
    pcap_freealldevs(alldevs);
    return -1;
}
if (d->addresses != NULL) {
    netmask = ((struct
sockaddr_in*)(d->addresses->netmask))->sin_addr.S_un.S_addr; // 获取接口第一个地
址的掩码
}
else {
    netmask = 0xffffffff; // 如果这个接口没有地址, 那么我们假设这个接口在 C 类
网络中
}
if (pcap_compile(adhandle, &fcode, packet_filter, 1, netmask) >= 0) {
    // 设置过滤器
    if (pcap_setfilter(adhandle, &fcode) < 0) {
        fprintf(stderr, "\nError setting the filter.\n");
        pcap_freealldevs(alldevs);
        return -1;
    }
}
```

```
    }  
}  
else {  
    fprintf(stderr, "\nError 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, 0, packet_handler, NULL);
```

抓取 12tp 协议流量如图所示：

```
time:17:41:19,212367 len:836  
3e 58 c2 b8 2e a5 8c c8 4b d4 e0 99 08 00 45 00  
03 36 14 8e 00 00 80 11 a6 c4 c0 a8 89 69 8b c4  
a6 8e 06 a5 06 a5 03 22 3e 6d 40 02 03 1a c7 32  
50 91 ff 03 00 21 45 00 03 0e f0 6d 40 00 80 06  
e2 05 c0 a8 12 04 67 1e eb ab f1 ee 01 bb 1d 20  
30 4d b4 0f 75 8d 50 18 02 00 69 59 00 00 17 03  
03 02 e1 00 00 00 00 00 00 00 02 40 2e 76 1a e7  
8d 5a 39 48 7f 7c 7a 3b 4d 03 81 4e 5e 59 05 36  
3b b3 0a 8a 6a b5 a9 b8 ff 69 cf 0c 2c 36 8b 94  
b1 c0 ff 68 e3 7a c6 83 15 36 8c 3b 14 97 f9 b1  
57 1b a1 23 fa 88 56 7e 71 f1 83 49 75 74 1c f6  
aa db ee 56 f3 4c ca 6d 03 32 17 6c ee 3b c2 a4
```

(图 9： 12tp 协议流量数据包)

通过协议流量时间的连续性，且实时输出，证明抓取的实时性。

```
time:17:41:19,060313 len:94  
time:17:41:19,134563 len:94  
time:17:41:19,136297 len:220  
time:17:41:19,212367 len:268
```

(图 10： 实时 12tp 协议流量抓取时间的连续性)

## 2.2 针对每一个目标用户，实时监控其 12tp 请求，并分析、还原、呈现其连接和业务载荷

### 2.2.1 以太网协议分析

以太网常用帧格式有两种，一种是 Ethernet II，另一种是 IEEE 802.3 格式。这两种格式区别是：Ethernet II 中包含一个 Type 字段。而 IEEE 802.3 格式中，此位置是长度字段。



(图 11： 两种以太网封装格式)

```
// 定义以太网首部格式
typedef struct ethernet_header
{
    mac_address des_mac_addr;
    mac_address src_mac_addr;
    u_short type;
}ethernet_header;
```

```
=====
以太网协议分析:
类型      :0x800(IPV4)
源地址    :140:200:75:212:224:153
目的地址  :62:88:194:184:46:165
```

(图 12: 以太网协议分析结果)

### 2.2.2 IPv4 协议分析

#### IPv4 主报头解读:

**第一个字段 Version (版本)** 描述是 IPV4。

**第二个字段 Header Length (报头长度)** 描述你有多少个字节, 因为长度可变你每次发送的数据是多少字节不能确定, 通过这个字段来告诉接收者你的 IP 报头有没有添加可选项, 没有添加就是 20 字节, 添加就是大于 20 字节。

**第三个字段 Type Of Service (服务类别)** 简称 TOS。用来实现 IP QOS (服务质量) 的。现在网络环境中 有上网玩游戏的冲浪流量、办公业务流量、上传下载大批量文件的大块数据流量、交互式视频会议流量、Voip 语音流量、动态路由选择协议流量这些不同类别的流量 发送品质的要求要差异化的处理这些流量, 要让网络设备根据流量的特征来做一个区分叫做 流量分类。

**第四个字段 Total Length (总长度)** 指包括 IP 报头在内完整的数据包有多少字节, 不包含帧头帧尾。

**第五, 六, 七个字段分别是标识符、标记、分段偏移量** 主要用来做三层的数据分片的。通常做数据切片的时传输层, 但三层也具备数据切片的能力一般是使用 IPV4。

**标识符:** 用来标识属于同一个数据包的所有分片打上一个标记值并且不同数据包分片属于不同标记值来做一个彼此区分。

**标记:** 它只有 3bit

**分段偏移量:** 每一个分片的分段偏移量就是描述我这个分片相对于这个完整数据包是从第几 bit 到第几 bit。根据每一个数据包的分片所属的字节区间接收者收到若干分片之后就 算乱序, 我能跟据你的序列来做一个重新的排序, 保证能把这个分片后的数据包给封装。(四 层切片是四层协议的本质属性, 这个切片对 CPU, 对内存消耗都不大。但三层切片导致网络 品质降低, 不建议使用三层切片)。

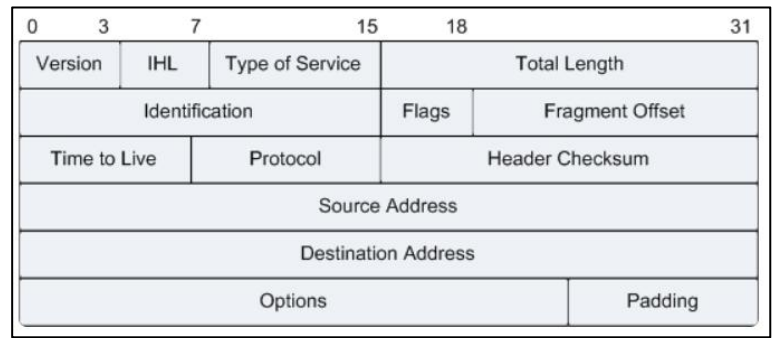
**第八个字段 Time to Live (存活时间)** 简称 TTL。描述一个数据包在网络中发送的时 候能传多长时间。用跳数来表示。一跳就是一台三层网络设备。跳数就是限制数据包在网络 中能传多少跳, 防止路由选择环路, 避免数据包在路径中永无止境的传递, 消耗链路带宽消 耗 CPU 资源, 给数据包设计一个 TTL 最多能传 255 跳, 传到 255 跳就被丢弃了, 因每台路由 器的硬件中都会包含一个分组改写引擎专门修改三层报头的字段的, 通过接收接口收到数据 包之后把 TTL 值-1, 减到 0 了就被丢包了。

**第九个字段 Protocol (上层协议)** 用来描述这个数据包传输层使用的什么协议。

**第十个字段 Header Checksum（报头校验和）**通过散列算法 CRC 对 IP 报头的头部字段做一个校验，把散列的算法值放到这个字段。接收方收到数据包之后会对整个三层报头在做一個校验得出一个乱码对比这个报头校验核得出的新的乱码看是否一致，一致就代表数据包的三层报头是没有被篡改的。

**第十一个字段 Source IP Address（源 IP 地址）**描述数据流量发送的发送接口。

**第十二个字段 Destination IP Address（目的 IP 地址）**描述数据流量接收的接口。三层地址具有端到端的一致性，排除公有地址私有地址，排除 PAT，所谓的端到端的三层一致性指的源发送流量去往目的地，不管数据包在哪个网络环境下发送他的源目地址都是保持不变的。



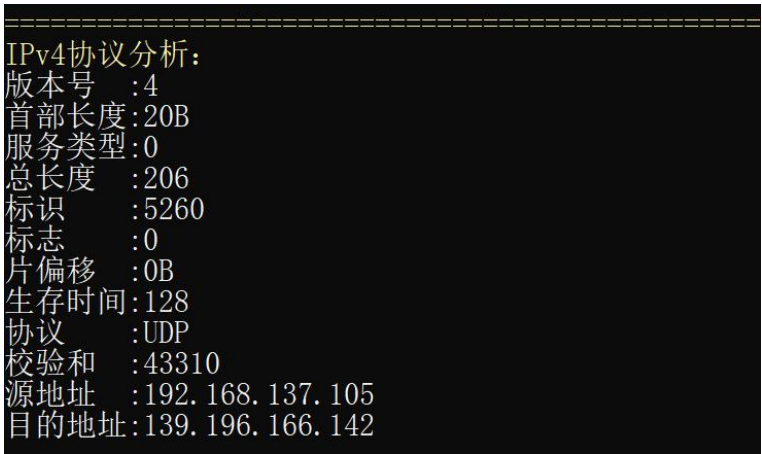
(图 13: IP 头封装格式)

```
// 定义 IPv4 地址结构
typedef struct ipv4_address
{
    u_char byte1;
    u_char byte2;
    u_char byte3;
    u_char byte4;
} ip_address;

// 定义 IP 首部格式
typedef struct ipv4_header
{
    u_char ver_ihl;           // 版本和首部长度
    u_char tos;               // 服务类型
    u_short tlen;             // 总长度
    u_short identification;   // 标识号
    u_short flags_fo;         // 段偏移量
    u_char ttl;               // 生存时间
    u_char proto;             // 协议
    u_short crc;              // 首部校验和
    ipv4_address saddr;       // 源 ip 地址
    ipv4_address daddr;       // 目的地址
    u_int op_pad;             // 选项和填补位
} ip_header;
```

字段	长度	含义
Version	4 比特	4: 表示为 IPV4;

字段	长度	含义						
		6: 表示为 IPV6。						
IHL	4 比特	首部长度, 如果不带 Option 字段, 则为 20, 最长为 60, 该值限制了记录路由选项。以 4 字节为一个单位。						
Type of Service	8 比特	服务类型。只有在有 QoS 差分服务要求时这个字段才起作用。						
Total Length	16 比特	总长度, 整个 IP 数据报的长度, 包括首部和数据之和, 单位为字节, 最长 65535, 总长度必须不超过最大传输单元 MTU。						
Identification	16 比特	标识, 主机每发一个报文, 加 1, 分片重组时会用到该字段。						
Flags	3 比特	标志位: IP Flag 字段格式 <table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>0</td><td>DF</td><td>MF</td></tr></table> Bit 0: 保留位, 必须为 0。 Bit 1: DF (Don't Fragment), 能否分片位, 0 表示可以分片, 1 表示不能分片。 Bit 2: MF (More Fragment), 表示是否该报文为最后一块, 0 表示最后一块, 1 代表后面还有。	0	1	2	0	DF	MF
0	1	2						
0	DF	MF						
Fragment Offset	12 比特	片偏移: 分片重组时会用到该字段。表示较长的分组在分片后, 某片在原分组中的相对位置。以 8 个字节为偏移单位。						
Time to Live	8 比特	生存时间: 可经过的最多路由数, 即数据包在网络中可通过的路由器数的最大值。						
Protocol	8 比特	协议: 下一层协议。指出此数据包携带的数据使用何种协议, 以便目的主机的 IP 层将数据部分上交给哪个进程处理。						
Header Checksum	16 比特	首部检验和, 只检验数据包的首部, 不检验数据部分。这里不采用 CRC 检验码, 而采用简单的计算方法。						
Source Address	32 比特	源 IP 地址。						
Destination Address	32 比特	目的 IP 地址。						
Options	可变	选项字段, 用来支持排错, 测量以及安全等措施, 内容丰富 (请参见下表)。选项字段长度可变, 从 1 字节到 40 字节不等, 取决于所选项的功能。						
Padding	可变	填充字段, 全填 0。						



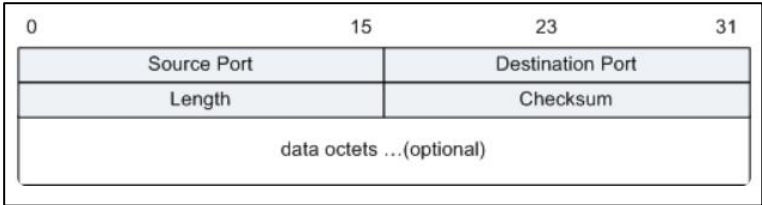
(图 14: IP 头分析结果)

2.2.3 UDP 协议分析

UDP 是为实现数据报 (Datagram) 模式的分组交换计算机网络通信而设计的。UDP 对应用程序提供了用最简化的机制向网络上的另一个应用程序发送消息的方法。UDP 提供无连接的、不可靠的数据报服务。

UDP 头包含以下字段：

- 1. 源端口 (Source Port)：16 位的源端口号，含义与 TCP 相同。
- 2. 目的端口 (Destination Port)：16 位的目的端口号，含义与 TCP 相同。
- 3. 长度 (Length)：16 位的长度字段，表明包括 UDP 头和数据在内的整个 UDP 数据报的长度，单位为字节。
- 4. 校验和 (Checksum)：16 位的错误检查字段，基于部分 IP 头信息、UDP 头和载荷数据的内容计算得到，用于检测传输过程中出现的错误。



(图 15: UDP 头封装格式)

```
// 定义 UDP 首部格式
typedef struct udp_header
{
    u_short sport; // 16bit 源端口
    u_short dport; // 16bit 目的端口
    u_short len;   // 16bit 长度
    u_short crc;   // 16bit 校验和
}udp_header;
```

字段	长度	描述
Source Port	2 字节	标识哪个应用程序发送（发送进程）。
Destination Port	2 字节	标识哪个应用程序接收（接收进程）。
Length	2 字节	UDP 首部加上 UDP 数据的字节数，最小为 8。

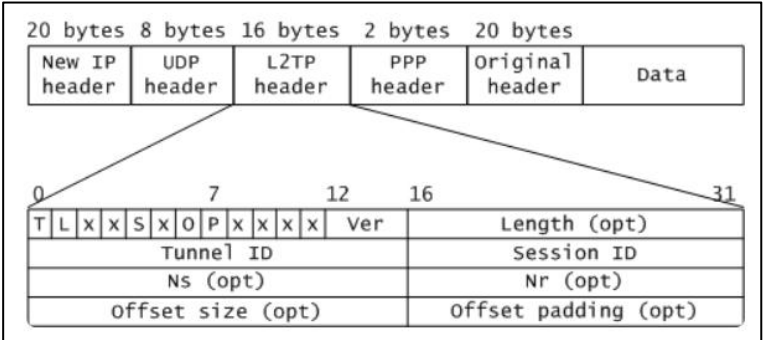
字段	长度	描述
Checksum	2 字节	覆盖 UDP 首部和 UDP 数据，是可选的。
data octets	变长	UDP 负载，可选的。

```
=====
UDP协议分析:
源端口   :1701
目的端口:1701
数据长度:186
校验和   :17213
=====
```

(图 16: UDP 协议分析结果)

2.2.4 L2TP 协议分析

L2TP 的控制消息和数据消息使用相同的报文头。



(图 15: L2TP 头封装格式)

```
// 定义 l2tp 首部格式
typedef struct l2tp_header
{
    u_char tlxxsxop;          // t 类型 (0 数据 1 控制)
                                // l 长度在位标志 (控制必为 1)
                                // s 顺序字段在位标志 (1 存在 ns nr 控制必为 1)
                                // o 偏移值在位标志
                                // p 优先级 (用于数据消息 控制必为 0)

    u_char xxxxver;           // 版本号
    u_short length;           // 消息总长度
    u_short tunnel_id;        // 隧道标识符 本地意义
    u_short session_id;       // 会话标识符 本地意义
    u_short ns;               // 当前消息顺序号
    u_short nr;               // 下一控制消息顺序号, 数据消息为保留字段
    u_short offset;           // 偏移值 指示载荷开始位置
    u_short offser_pading;    // 偏移量填充
} l2tp_header;

u_short t, l, s, o;
t = (pl2tpheader->tlxxsxop & 0x80) >> 7;
l = (pl2tpheader->tlxxsxop & 0x40) >> 6;
s = (pl2tpheader->tlxxsxop & 0x08) >> 3;
```

```
o = (pl2tpheader->tlxxsxop & 0x02) >> 1;
printf("L2TP 协议分析: \n");
printf("0x%x\n", pl2tpheader->tlxxsxop);
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", pl2tpheader->tlxxsxop & 0x01);
printf("版本号         :%d\n", pl2tpheader->xxxxver & 0x0f);
if (l == 1) { // 长度在位标志为1
    printf("消息总长度    :%d\n", ntohs(pl2tpheader->length));
}
printf("隧道标识符      :%d\n", ntohs(pl2tpheader->tunnel_id));
printf("会话标识符      :%d\n", ntohs(pl2tpheader->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(pl2tpheader->ns));
    if (t == 1) { // 控制信息 nr 才有意义
        printf("下一消息顺序号  :%d\n", ntohs(pl2tpheader->nr));
    }
}
if (o == 1) { // 偏移值在位标志为1
    printf("偏移量        :%d\n", ntohs(pl2tpheader->offset));
}
```

(表 2: L2TP 报文描述)

字段	长度	描述
T	1 比特	类型 (Type), 取值为“0”时表示数据消息, 取值为“1”时表示控制消息。
L	1 比特	长度在位标志, 取值为“1”时表示报文头中存在长度字段 Length。控制消息中必须为“1”。
x	1 比特	保留位
S	1 比特	顺序字段在位标志, 取值为“1”时表示报文头中存在 Ns 和 Nr 字段。控制消息中必须为“1”。
O	1 比特	取值为“1”时表示报文头中存在 offset size 字段。控制消息中必须为“0”。
P	1 比特	优先级 (Priority), 只用于数据消息。控制消息中必须为“0”。
Ver	4 比特	版本号, 对于 L2TPv2 协议取值为“2”。
Length	16 比特	消息的总长度, 单位为字节。
Tunnel ID	16 比特	隧道标识符, 只具有本地意义。Hello 控制消息具有全局性, 其 Tunnel ID 必须为 0。
Session ID	16 比特	会话标识符, 只具有本地意义。
Ns	16 比特	当前消息的顺序号。

字段	长度	描述
Nr	16 比特	希望接收的下一条控制消息的序号。数据消息中是保留字段。
Offset size	16 比特	偏移值，指示载荷数据开始的位置。
Offset	16 比特	填充位。
padding		

```
=====
L2TP协议分析:
0xc8
类型                :1(控制信息)
长度在位标志        :1
顺序字段在位标志    :1
偏移值在位标志      :0
优先级              :0
版本号              :2
消息总长度          :108
隧道标识符          :0
会话标识符          :0
当前消息序号        :0
下一消息序号        :0
偏移量              :32776
=====
```

(图 16: L2TP 协议分析结果-控制信息)

```
=====
L2TP协议分析:
0x40
类型                :0(数据信息)
长度在位标志        :1
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
消息总长度          :41
隧道标识符          :48490
会话标识符          :53728
偏移量              :729
=====
```

(图 17: L2TP 协议分析结果-数据信息)

2.3 支持对特定目标的静载荷替换

通过对 12tp 报文封装结构的分析，找到数据包的静载荷部分，并将特定数据包的静载荷替换为自己的 data 数据，替换完成后，再按照 12tp 报文结构进行封装，产生一个新的数据包。使用第 2 问的拆解算法，对新的数据包重新拆解，得到了我们替换的新静载荷。

```
=====
>X.....K.....E.
.u.a.....i..
.....a..@..Y..
.....!E..M.....
.B.....9
.....TJ-U
niversity..
=====
```

(图 18: 支持对特定目标的静载荷替换)

### 3. 更新：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！

#### 3.1 写在前面

我们在第一版本的作业中，实现了：

“构建一个 demo：用笔记本电脑实现对**笔记本电脑**上网对象的以上功能。”

在这一版本中，我们通过搭建一个手机能够连接的 l2tp vpn，实现：

“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。”

#### 3.2 实现方法

##### （1）安装 l2tpd

###### ①安装 EPEL 源

```
yum install -y epel-release
```

```
[root@iZuf62y8khbk3qb7xjwhZ ~]# yum install -y epel-release
Loaded plugins: fastestmirror
Determining fastest mirrors
base
epel
extras
updates
```

###### ②安装 xl2tpd

```
yum install xl2tpd
```

```
[root@iZuf62y8khbk3qb7xjwhZ ~]# yum install xl2tpd
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
Resolving Dependencies
--> Running transaction check
--> Package xl2tpd.x86_64 0:1.3.15-1.el7 will be installed
--> Processing Dependency: ppp >= 2.4.5-18 for package: xl2tpd-1.3.15-1.el7.x86_64
--> Running transaction check
--> Package ppp.x86_64 0:2.4.5-34.el7_7 will be installed
--> Finished Dependency Resolution
```

###### ③安装 libreswan

```
yum install -y libreswan
```

```
[root@iZuf62y8khbk3qb7xjwhZ ~]# yum install -y libreswan
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
Resolving Dependencies
--> Running transaction check
--> Package libreswan.x86_64 0:3.25-9.1.el7_8 will be installed
```

##### （2）修改 IPsec 配置文件

```
sudo vim /etc/ipsec.conf
```

```
config setup
# Normally, pluto logs via syslog.
#logfile=/var/log/pluto.log
#
# Do not enable debug options to debug configuration issues!
#
# plutodebug="control parsing"
# plutodebug="all crypt"
plutodebug=none
#
# NAT-TRAVERSAL support
# exclude networks used on server side by adding %v4:1a.b.c.0/24
# It seems that T-Mobile in the US and Rogers/Fido in Canada are
# using 25.0 as "private" address space on their wireless networks.
# this range has never been announced via BGP (at least up to 2015)
nat_traversal=yes
virtual_private=%v4:10.0.0.0/8,%v4:192.168.0.0/16,%v4:172.16.0.0/12,%v4:25.0.0.0/8,%v4:100.64.0.0/10,%v6:fd00::/8,%v6:fe80::/10
```

### (3) 创建 IPsec 与 L2TP 服务关联的配置文件

vim /etc/ipsec.d/l2tp\_psk.conf

```
✘ 华东2(上海)i-uf62y8khhbka3qb7xjwh iZuf62y8khhbka3qb7xjwhZ root@139.196.166.142
> 5. root@iZuf62y8khhbka3qb7xjwhZ:~ ✘
conn L2TP-PSK-NAT
    rightsubnet=vhost:%priv
    also=L2TP-PSK-noNAT
conn L2TP-PSK-noNAT
    authby=secret
    pfs=no
    auto=add
    keyingtries=3
    dpddelay=30
    dpdtimeout=120
    dpdaction=clear
    rekey=no
    ikelifetime=8h
    keylife=1h
    type=transport
    left=139.196.166.142 # 网卡IP
    leftprotoport=17/1701
    right=%any
    rightprotoport=17/%any
```

### (4) 创建保存预共享密钥的文件

vim /etc/ipsec.d/ipsec.secrets

```
[root@iZuf62y8khhbka3qb7xjwhZ ~]# vim /etc/ipsec.d/ipsec.secrets
```

: PSK "2053182"

```
✘ 华东2(上海)i-uf62y8khhbka3qb7xjwh iZuf62y8khhbka3q
> 5. root@iZuf62y8khhbka3qb7xjwhZ:~ ✘
: PSK "2053182"
```

### (5) 设置 l2tp 的账号等信息

vim /etc/ppp/chap-secrets

```
>_ 5. root@iZuf62y8khhbka3qb7xjwhZ:~ ×  
# Secrets for authentication using CHAP  
# client      server  secret      IP addresses  
vpnuser      *      2053182  
~
```

## (6) 修改内核参数

### ①修改 sysctl 的内核支持文件

```
vim /etc/sysctl.conf  
华东2(上海)i-uf62y8khhbka3qb7xjwh iZuf62y8khhbka3qb7xjwhZ root@139.196.166.142  
>_ 5. root@iZuf62y8khhbka3qb7xjwhZ:~ ×  
vm.swappiness = 0  
kernel.sysrq = 1  
  
net.ipv4.neigh.default.gc_stale_time = 120  
  
# see details in https://help.aliyun.com/knowledge_detail/39428.html  
net.ipv4.conf.all.rp_filter = 0  
net.ipv4.conf.default.rp_filter = 0  
net.ipv4.conf.default.arp_announce = 2  
net.ipv4.conf.lo.arp_announce = 2  
net.ipv4.conf.all.arp_announce = 2  
  
# see details in https://help.aliyun.com/knowledge_detail/41334.html  
net.ipv4.tcp_max_tw_buckets = 5000  
net.ipv4.tcp_syncookies = 1  
net.ipv4.tcp_max_syn_backlog = 1024  
net.ipv4.tcp_synack_retries = 2  
  
net.ipv4.ip_forward = 1  
net.ipv4.conf.all.accept_redirects = 0  
net.ipv4.conf.all.rp_filter = 0  
net.ipv4.conf.all.send_redirects = 0  
net.ipv4.conf.default.accept_redirects = 0  
net.ipv4.conf.default.rp_filter = 0  
net.ipv4.conf.default.send_redirects = 0  
net.ipv4.conf.lo.accept_redirects = 0  
net.ipv4.conf.lo.rp_filter = 0  
net.ipv4.conf.lo.send_redirects = 0[]
```

### sysctl -p

```
[root@iZuf62y8khhbka3qb7xjwhZ ~]# sysctl -p  
vm.swappiness = 0  
kernel.sysrq = 1  
net.ipv4.neigh.default.gc_stale_time = 120  
net.ipv4.conf.all.rp_filter = 0  
net.ipv4.conf.default.rp_filter = 0  
net.ipv4.conf.default.arp_announce = 2  
net.ipv4.conf.lo.arp_announce = 2  
net.ipv4.conf.all.arp_announce = 2  
net.ipv4.tcp_max_tw_buckets = 5000  
net.ipv4.tcp_syncookies = 1  
net.ipv4.tcp_max_syn_backlog = 1024  
net.ipv4.tcp_synack_retries = 2  
net.ipv4.ip_forward = 1  
net.ipv4.conf.all.accept_redirects = 0  
net.ipv4.conf.all.rp_filter = 0  
net.ipv4.conf.all.send_redirects = 0  
net.ipv4.conf.default.accept_redirects = 0  
net.ipv4.conf.default.rp_filter = 0  
net.ipv4.conf.default.send_redirects = 0  
net.ipv4.conf.lo.accept_redirects = 0  
net.ipv4.conf.lo.rp_filter = 0  
net.ipv4.conf.lo.send_redirects = 0
```

## ②检验 IPsec 服务配置

ipsec verify

```
[root@iZuf62y8khhbka3qb7xjwhZ ~]# ipsec verify
Verifying installed system and configuration files

Version check and ipsec on-path [OK]
Libreswan 3.25 (netkey) on 3.10.0-1127.19.1.el7.x86_64
Checking for IPsec support in kernel [OK]
NETKEY: Testing XFRM related proc values
    ICMP default/send_redirects [OK]
    ICMP default/accept_redirects [OK]
    XFRM larval drop [OK]
Pluto ipsec.conf syntax [OK]
Two or more interfaces found, checking IP forwarding [OK]
Checking rp_filter [OK]
Checking that pluto is running [OK]
Pluto listening for IKE on udp 500 [OK]
Pluto listening for IKE/NAT-T on udp 4500 [OK]
Pluto ipsec.secret syntax [OBSOLETE]
    003 WARNING: using a weak secret (PSK)
Checking 'ip' command [OK]
Checking 'iptables' command [OK]
Checking 'prelink' command does not interfere with FIPS [OK]
Checking for obsolete ipsec.conf options [OBSOLETE KEYWORD]
Warning: ignored obsolete keyword 'nat_traversal'
```

## ③重启 IPsec 服务

systemctl restart ipsec

```
[root@iZuf62y8khhbka3qb7xjwhZ ~]# systemctl restart ipsec
```

## (7) 修改 l2tp 配置文件

vim /etc/xl2tpd/xl2tpd.conf

```
[global]
listen-addr = 192.168.1.98
;
; requires openswan-2.5.18 or higher - Also does not yet work in combination
; with kernel mode l2tp as present in linux 2.6.23+
ipsec saref = yes
; Use refinfo of 22 if using an SAREf kernel patch based on openswan 2.6.35 or
; when using any of the SAREf kernel patches for kernels up to 2.6.35.
; saref refinfo = 30
;
; force userspace = yes
;
; debug tunnel = yes

[lns default]
ip range = 192.168.1.128-192.168.1.254
local ip = 192.168.1.99
```

## (8) 修改安全配置

vim /etc/ppp/options.xl2tpd

```
require-mschap-v2
ipcp-accept-local
ipcp-accept-remote
ms-dns 8.8.8.8
ms-dns 114.114.114.114
```

## (9) 安装和配置防火墙规则

### ①停止 firewalld 服务并禁用

```
systemctl stop firewalld  
systemctl mask firewalld
```

```
[root@iZuf62y8khhbka3qb7xjwhZ ~]# systemctl stop firewalld  
[root@iZuf62y8khhbka3qb7xjwhZ ~]# systemctl mask firewalld  
Created symlink from /etc/systemd/system/firewalld.service to /dev/null.
```

## ②安装 iptables 服务

```
yum install iptables-services
```

```
[root@iZuf62y8khhbka3qb7xjwhZ ~]# yum install iptables-services  
Loaded plugins: fastestmirror  
Loading mirror speeds from cached hostfile  
Resolving Dependencies  
--> Running transaction check  
--> Package iptables-services.x86_64 0:1.4.21-35.el7 will be installed  
--> Processing Dependency: iptables = 1.4.21-35.el7 for package: iptables-services-1.4.21-35.el7.x86_64  
--> Running transaction check  
--> Package iptables.x86_64 0:1.4.21-34.el7 will be updated  
--> Package iptables.x86_64 0:1.4.21-35.el7 will be an update  
--> Finished Dependency Resolution
```

## ③配置 iptables 规则

```
sudo iptables -t nat -A POSTROUTING -s 192.168.1.99/24 -o ens33 -j MASQUERADE  
sudo iptables -I FORWARD -s 192.168.1.99/24 -j ACCEPT  
sudo iptables -I FORWARD -d 192.168.1.99/24 -j ACCEPT  
sudo iptables -A INPUT -p udp -m policy --dir in --pol ipsec -m udp --dport 1701 -j ACCEPT  
sudo iptables -A INPUT -p udp -m udp --dport 1701 -j ACCEPT  
sudo iptables -A INPUT -p udp -m udp --dport 500 -j ACCEPT  
sudo iptables -A INPUT -p udp -m udp --dport 4500 -j ACCEPT  
sudo iptables -A INPUT -p esp -j ACCEPT  
sudo iptables -A INPUT -m policy --dir in --pol ipsec -j ACCEPT  
sudo iptables -A FORWARD -i ppp+ -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT  
sudo iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
```

```
[root@iZuf62y8khhbka3qb7xjwhZ ~]# sudo iptables -t nat -A POSTROUTING -s 192.168.1.99/24 -o ens33 -j MASQUERADE  
[root@iZuf62y8khhbka3qb7xjwhZ ~]# sudo iptables -I FORWARD -s 192.168.1.99/24 -j ACCEPT  
[root@iZuf62y8khhbka3qb7xjwhZ ~]# sudo iptables -I FORWARD -d 192.168.1.99/24 -j ACCEPT  
[root@iZuf62y8khhbka3qb7xjwhZ ~]# sudo iptables -A INPUT -p udp -m policy --dir in --pol ipsec -m udp --dport 1701 -j ACCEPT  
[root@iZuf62y8khhbka3qb7xjwhZ ~]# sudo iptables -A INPUT -p udp -m udp --dport 1701 -j ACCEPT  
[root@iZuf62y8khhbka3qb7xjwhZ ~]# sudo iptables -A INPUT -p udp -m udp --dport 500 -j ACCEPT  
[root@iZuf62y8khhbka3qb7xjwhZ ~]# sudo iptables -A INPUT -p udp -m udp --dport 4500 -j ACCEPT  
[root@iZuf62y8khhbka3qb7xjwhZ ~]# sudo iptables -A INPUT -p esp -j ACCEPT  
[root@iZuf62y8khhbka3qb7xjwhZ ~]# sudo iptables -A INPUT -m policy --dir in --pol ipsec -j ACCEPT  
[root@iZuf62y8khhbka3qb7xjwhZ ~]# sudo iptables -A FORWARD -i ppp+ -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT  
[root@iZuf62y8khhbka3qb7xjwhZ ~]# sudo iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
```

## ④保存 iptables 规则

```
service iptables save
```

```
[root@iZuf62y8khhbka3qb7xjwhZ ~]# service iptables save  
iptables: Saving firewall rules to /etc/sysconfig/iptables:[ OK ]
```

## ⑤重启防火墙

```
systemctl restart iptables
```

```
[root@iZuf62y8khhbka3qb7xjwhZ ~]# systemctl restart iptables
```

## (10) 检查 iptables、ipsec 正常运行

### ①检查 iptables 服务

```
systemctl status iptables
```

```
[root@izuf62y8khhbka3qb7xjwhZ ~]# systemctl status iptables
● iptables.service - IPv4 firewall with iptables
   Loaded: loaded (/usr/lib/systemd/system/iptables.service; disabled; vendor preset: disabled)
   Active: active (exited) since Sun 2023-04-02 00:43:24 CST; 32s ago
     Process: 3547 ExecStart=/usr/libexec/iptables/iptables.init start (code=exited, status=0/SUCCESS)
    Main PID: 3547 (code=exited, status=0/SUCCESS)

Apr 02 00:43:24 izuf62y8khhbka3qb7xjwhZ systemd[1]: Starting IPv4 firewall with iptables...
Apr 02 00:43:24 izuf62y8khhbka3qb7xjwhZ iptables.init[3547]: iptables: Applying firewall rules: [ OK ]
Apr 02 00:43:24 izuf62y8khhbka3qb7xjwhZ systemd[1]: Started IPv4 firewall with iptables.
[root@izuf62y8khhbka3qb7xjwhZ ~]#
```

## ②检查 ipsec 服务

systemctl status ipsec

```
[root@izuf62y8khhbka3qb7xjwhZ ~]# systemctl status ipsec
● ipsec.service - Internet Key Exchange (IKE) Protocol Daemon for IPsec
   Loaded: loaded (/usr/lib/systemd/system/ipsec.service; disabled; vendor preset: disabled)
   Active: active (running) since Sun 2023-04-02 00:35:56 CST; 8min ago
     Docs: man:ipsec(8)
           man:pluto(8)
           man:ipsec.conf(5)
    Main PID: 3136 (pluto)
      Status: "Startup completed."
     CGroup: /system.slice/ipsec.service
            └─3136 /usr/libexec/ipsec/pluto --leak-detective --config /etc/ipsec.conf --nofork

Apr 02 00:35:56 izuf62y8khhbka3qb7xjwhZ pluto[3136]: | setup callback for interface lo:500 fd 17
Apr 02 00:35:56 izuf62y8khhbka3qb7xjwhZ pluto[3136]: | setup callback for interface eth0:4500 fd 16
Apr 02 00:35:56 izuf62y8khhbka3qb7xjwhZ pluto[3136]: | setup callback for interface eth0:500 fd 15
Apr 02 00:35:56 izuf62y8khhbka3qb7xjwhZ pluto[3136]: loading secrets from "/etc/ipsec.secrets"
Apr 02 00:35:56 izuf62y8khhbka3qb7xjwhZ pluto[3136]: loading secrets from "/etc/ipsec.d/ipsec.secrets"
Apr 02 00:35:56 izuf62y8khhbka3qb7xjwhZ pluto[3136]: WARNING: using a weak secret (PSK)
Apr 02 00:36:00 izuf62y8khhbka3qb7xjwhZ pluto[3136]: forgetting secrets
Apr 02 00:36:00 izuf62y8khhbka3qb7xjwhZ pluto[3136]: loading secrets from "/etc/ipsec.secrets"
Apr 02 00:36:00 izuf62y8khhbka3qb7xjwhZ pluto[3136]: loading secrets from "/etc/ipsec.d/ipsec.secrets"
Apr 02 00:36:00 izuf62y8khhbka3qb7xjwhZ pluto[3136]: WARNING: using a weak secret (PSK)
```

至此，搭建成功 l2tp vpn！能够使得手机也能连入该 vpn，从而在电脑上抓取手机协议流量。

l2tp 协议分析的内容及静载荷分析，和前述电脑抓取电脑的 l2tp 报文分析是一致的，在此不做赘述。