PlainBox代码文档

## 概述

PlainBox Github：<https://github.com/DavyVan/PlainBox.git>

PlainBox本身既是客户端也是middlebox端，在客户端运行时需要某些应用程序的配合：Firefox浏览器、OpenSwan、OpenSSH，其中OpenSwan和OpenSSH经过修改，使其能将密钥相关信息输出到文件中：

OpenSwan Github：<https://github.com/DavyVan/OpenSwan_keyout.git>

OpenSSH Github：<https://github.com/DavyVan/OpenSSH_Keyout.git>

有关修改细节，参见各自Github中about文件，或者PlainBox/doc/Aboutxxx.txt

## 编译及运行

### 编译

运行环境：Ubuntu x64（其他Linux发行版未经测试），出客户端要求至少一块网卡，middlebox要求至少两块网卡。

1. 在接入互联网后，以root权限运行代码目录中install.sh即可自动安装所需依赖项；
2. 将代码从git或者其他介质中复制到计算机；
3. 进入代码根目录，执行make，即可编译，无安装步骤。

### 运行

在客户端运行时使用pcap嗅探模式，这将不会影响客户端的收发；在middlebox上运行时，使用nfqueue模式（-w 参数），这将强制所有数据包流经虚拟网卡并在处理后转发。为了使用ABE加密，还需指定公钥和私钥，最后的运行命令如下：

#./PlainBox –w –pub pub.key –prv mbox1\_key

若要使PlainBox能够解析HTTPS协议，需要配置Firefox浏览器将TLS Client Random和Master Secret输出到指定文件中，该文件路径使用环境变量SSLKEYLOGFILE定义，在运行PlainBox以及Firefox之前，设定环境变量，执行如下语句：

#export SSLKEYLOGFILE=/tmp/sslkey.log

若要使PlainBox能够解析IPSec协议，需要安装修改过的OpenSwan，安装方法参见Github中README，运行时尽量使用root权限。

若要使PlainBox能够解析SSH协议，需要安装修改过的OpenSSH，安装方法参见Github中OpenSSH的README，运行时尽量使用root权限。

## 重要文件及函数

### main.cpp

void got\_packet(u\_char \*args, const pcap\_pkthdr \*header, const u\_char \*packet)

数据报处理函数，每接收到一个数据报都会调用一次此函数。该函数会根据IP协议版本、传输层协议类型以及端口号等进行流的管理并调用上层处理函数。

void\* sslfile(void\* arg)

用于从SSLKEYLOGFILE指定的文件中读取TLS相关Client Random和Master Secret并存入内存待用。该函数被单独线程执行，不断监听文件的内容，如果有新内容添加进来，则立刻会被该函数读取并存入内存。

int main(int argc, char \*argv[])

主函数，主要工作就是解析参数，开启上述监听进程，初始化nfqueue或者pcap。

### 流管理相关

整个流管理模块由六个文件组成。

#### flowinfo.h/.cpp

定义了用于存储流基本信息的类，并提供相应信息返回函数。

int FlowInfo::handleTCPPacket(……)

根据IP地址确定数据流方向，之后调用TCP报文组装函数，进入传输层的处理。

#### flowkey.h/.cpp

定义了用于唯一标识流的类，IP1与IP2是源IP地址和目的IP地址，地址按照字节比较，较小的赋予IP1，较大的赋予IP2，端口跟随IP赋值。

#### flowmgr.h/.cpp

流管理模块的执行者，三个函数分别对应查找流、添加新流、删除流。流的信息以键值对的形式保存在一个map中，键是整数，由flowkey的函数map\_key\_gen()生成：IP1后两个字节+端口号+IP2后两个字节+端口号；值是flowinfo的boost指针。

### tcphandler.h/.cpp

定义了TCPHandler类，用于处理TCP报文。

int reAssemblePacket(……)

用于处理TCP报文重组，同时解决失序、重传等问题，有序的若干个TCP报文形成链表，并根据端口调用对应应用层协议处理器。

【注】用于发送密钥交换报文的三个函数也被定义在这两个文件中（但不属于TCPHandler类），其中一个是发送UDP数据报。

### esphandler.h/.cpp

定义了esphandler类，实现了ESP（IPSec）协议的解析与解密。

bool ESPHandler::parseAndDecrypt(……)

主要调用函数，用于解析ESP协议提取出SPI，然后利用SPI去查找对应的密钥，最后解密数据，支持协议嵌套（HTTPS over IPSec）。该类中的其他函数均服务于此函数。

KeyMaterial\_ESP\_Ptr ESPHandler::getKeys(unsigned int spi)

用于从文件中查找SPI对应的会话密钥，该文件的位置OpenSwan中添加的代码中描述的文件位置一致。密钥找到之后保存在内存中。

int ESPHandler::handleKeys(const uint8\_t \*payload, unsigned int length)

用于从密钥交换报文（已经过ABE解密）中提取密钥，提取出来的密钥保存在内存中。

【注】其他类中的handleKeys函数作用一样，不再赘述。

void ESPHandler::decrypt(……)

用于解密ESP报文。

### sshhandler.h/.cpp

定义了处理SSH协议数据报的类，继承自AppLayerHandler。定义了SSH Record、SSH链路状态、会话密钥与认证密钥以及用于重组SSH Record的各种数据结构。函数功能与esphandler类似，只是将esphandler中的parseAndDecrypt函数一分为二：

void\* parse(TCPDataNode \*head, TCPDataDirection direction, FlowKey\* flowkey)

用于从TCP Payload中解析出一个或者多个SSH Record，并能够处理跨越多个TCP报文的超长SSH Record。

void process(void \*record, TCPDataDirection direction, FlowKey\* flowkey)

每当解析出来一个完整的SSH Record时便调用此函数，用于解析SSH Record中的信息，包括解密数据。

### tlshandler.h/.cpp

定义了TLS协议的处理器类TLSHandler，其结构与SSHHandler类似，也需要先解析TLS Record然后在进行数据处理。额外地，TLS实现了计算MAC校验码的功能，以便能够修改TLS报文。

### ABE相关

ABE相关代码有abe文件夹及abe.h/abe.cpp。在解析密钥交换报文的时候主要用到abe.h中定义的三个函数：

void abe\_init(char \*pub\_key\_file, char \*prv\_key\_file)

用于初始化abe，参数为PlainBox运行时参数指定的公钥和私钥两个文件。

ABEFile abe\_encrypt(unsigned char\* input\_file, int len, char \*policy\_)

用于abe加密，policy参数为字符串形式，类似布尔表达式，例如“ChinaUnicom and TLS”。

ABEFile abe\_decrypt(const unsigned char\* input\_file)

用于abe解密，解密所使用的密钥必须和加密时的policy相符合。

### 其他

|  |  |
| --- | --- |
| 文件 | 用途 |
| applayerhandler.h/.cpp | 规定了应用层协议处理器的基类，包含必需的纯虚函数，不可实例化。 |
| ip4hdr.h/.cpp | 定义IPv4协议头类，用于保存IP报头各种信息，并提供相应get函数。 |
| ip6hdr.h/.cpp | 定义IPv6协议头类，同IPv4。 |
| ipaddr.h/.cpp | 定义IPAddr基类；定义IPv4Addr与IPv6Addr，均继承自IPAddr，用于保存IP地址，提供返回字符串或字节类型的函数，并定义流输出运算符和小于号 |
| tcpdatastructure.h | 定义了保存TCP报文的结构体以及标记TCP流方向的枚举变量。 |
| tcphdr.h/.cpp | 定义TCPHdr类，用于保存TCP头相关信息，并提供各种get函数。 |
| udpher.h/.cpp | 定义了UDP报文头部类，是对linux内核udphdr的简单封装，提供了几个get函数 |
| tls.h/.cpp | 定义了TLS密钥结构体，定义了将从SSLKEYLOGFILE中读取的密钥信息存入vector数据结构中的函数，以及定义了查找密钥的函数 |
| nfqueue.h/.cpp | 定义了nfqueue相关函数，包括初始化、数据报处理、关闭等 |
| install.sh | 用于简化PlainBox部署的脚本，负责安装所有的依赖项。 |

## 仍存在问题

* 内存泄露严重，如果在大流量情况下长时间运行的话会出现上百兆的内存泄露，主要原因是new出来的数组没有及时delete掉。
* SSH可能会出现超长报文，跨越多个TCP报文，但只有一个头部，而且头部被加密（可能，具体细节记不清了，肯定的是获取长度上有问题），无法获得准确的报文长度，使用TCP来计算的临时方法会出现间歇性崩溃的情况，还需要深究一下SSH的实现细节才能根本解决。
* 现在所有的密钥都是先输出到文件，然后我们再读取，这样在文件很长的情况下会降低性能，是否可以通过应用程序间的通讯来完成？

## 所缺功能

### 手机端

最重要的是确定手机端对应TLS、SSH、IPSec的应用程序，找到传递密钥的方法

其他的跟PC上的功能差不多，包括获取密钥、传递密钥，监听流量、流的管理。这是我暂时的想法，我还会继续想一想（总感觉少点儿什么……）

初步想法是实现成一个系统服务之类的，才能在后台一直监听，前台app的话可能被从内存中清出去。

在后台拦截数据可能会带来权限问题，这方面需要一些调研

开发时如何测试

### Middlebox端

暂时没想到针对手机端需要做什么修改，只是之前有一些没有来得及实现的功能，比如报文修改