网络嗅探器的设计实现

严文励 202128015029011

一．实验目标

本实验在ubuntu linux环境通过libpcap使用C语言实现了网络嗅探器功能。包括：获取五层网络模型的全部流量，筛选出HTTP、TCP/UDP、TLS、ICMP等协议，并能追踪TCP流的内容。

二．实验环境和使用的工具

环境配置的注意事项：

下载libpcap后在ubuntu上解压安装。之后需要用sudo apt-get install下载libpcap-dev。注意在ubuntu环境下调用libpcap的相关函数，在编译是要加上-lpcap，否则会找不到函数。在运行时要加上sudo，否则没有抓包权限。

使用的libpcap的API有：

int pcap\_findalldevs(pcap\_if\_t \*alldevs, char\* errBuf)

获取当前所有网卡并储存在alldevs中。储存方式是一个链表，用alldev->name得到网络设备名。

pcap\_t \*pcap\_open\_live(char \*device, int snaplen, int promisc, int to\_ms, char \*errBuf)

获得网络数据包的捕获描述字。Device是数据名，snaplen是捕获的最大字节，设定为65536，promisc选择是否混杂模式，to\_ms是超时时间。

int pcap\_lookupnet(char \*device, bpf\_u\_int32 \*netp,bpf\_u\_int32 \*maskp, char \*errbuf)

从设备中获得网络号和掩码。

int pcap\_compile(pcap\_t \*p, struct bpf\_program \*fp, char \*str, int optimize, bpf\_u\_int32 netmask)

根据str指定的内容编译过滤程序，存放于fp中。过滤程序可以按照协议、方向、类型选择性抓包。

int pcap\_setfilter(pcap\_t \*p, struct bpf\_program \*fp)

设置过滤程序并与pcap\_open\_live得到的描述字绑定。

int pcap\_dispatch(pcap\_t \*p, int cnt, pcap\_handler callback, u\_char \*user)

调用自己编写的callabck函数处理数据包处理数据包。若cnt大于0，则会在处理完cnt个或超时情况下返回。

callback函数应有三个参数：一个由pcap\_dispatch或pcap\_loop传递来的u\_char指针，一个pcap\_pkthdr指针和一个表示数据包大小的u\_char指针。

int pcap\_loop(pcap\_t \*p, int cnt, pcap\_handler callback, u\_char \*user)

调用callback函数处理数据包，此函数不会超时，只会在处理完cnt个数据包或出错时中止。

u\_char \*pcap\_next(pcap\_t \*p, struct pcap\_pkthdr \*h)

返回下一个数据包的地址

三．实现过程

由于libpcap功能较全面，实现需要的效果并不困难。用C语言实现主要分为三个部分：

1. 创建筛选并在设备上抓包
2. 对抓到的数据包进行处理，分辨其协议类型、来源和去向的地址等信息。
3. 对其中的TCP包读取其TCP流。

如果对上述接口函数的熟练掌握和对TCP/IP协议结构熟悉，这三步不难完成。

四．实验结果

经验证，能够达成实验目标。具体可见演示视频。

Github地址见[119456ywl/SoftwareAndSystemSecurity (github.com)](https://github.com/119456ywl/SoftwareAndSystemSecurity)

五．实验的感想和收获

在这次实验中，虽然只实现了基本的功能，但是确实学到了有用的内容。最终是用libpcap实现的，但是在前期也尝试了课上提到的winPcap和Wireshark，主要的难点在于配置二次开发的环境。