Libpcap 数据包捕获机制剖析与研究

■ 江苏信息职业技术学院 平震宇

【摘要】数据包截获与分析是网路管理中的基本技术，本文深入研究了国外优秀的数据包捕获函数库Libpcap。介绍了用于用户层数据包截获的系统独立的api接口Libpcap库，给出了利用BPF和Libpcap设计基于包捕获的应用程序的应用框架。
【关键词】Libpcap BPF 数据包捕获

Dissecting and Researching of Packet Capture Mechanism Libpcap

Ping Zhen - yu
(JiangSu College Of Information Technology Engineering Department of The Computer JiangSu, 214101)

Abstract: Packet capture and packet analysis is one of the basic technologies used in network management.This paper fully studies the structure and principle of the packet capture library libpcap. It then introduced libpcap library - a system-independent interface for user-level packet capture and described the general model of packet capture applications based on BPF and libpcap
Keywords: Libpcap; BPF; packet capture

1 引言

Libpcap 的英文意思是 Packet Capture library，即数据包捕获函数库。该库提供的 C 函数接口可用于需要捕获经过网络接口（通过将网卡设置为混杂模式，可以捕获所有经过该接口的数据报，目标地址不一定为本机）数据包的系统开发上。Libpcap 提供的接口函数主要实现和封装了与数据包截获有关的过程。这个库为不同的平台提供了一致的编程接口，在安装了 Libpcap 的平台上，以 Libpcap 为接口写的程序，能够自由的跨平台使用。在 Linux 系统下，Libpcap 可以使用 BPF（Berkeley Packet Filter）分组捕获机制来获得很高的性能。这个库为不同的平台提供了一致的编程接口，在安装了 Libpcap 的平台上，以 Libpcap 为接口写的程序、应用，能够自由的跨平台使用。

操作系统所提供的分组捕获机制主要有三种：BPF（Berkeley Packet Filter），DLPI（Data Link Provider Interface），及 Linux下的 SOCK_PACKET 类型套接口。基于 BSD 的系统使用 BPF，基于 SVR4 的系统一般使用 DLPI。从文献上看 BPF 比 DLPI 性能好很多，而 SOCK_PACKET 更弱。

Libpcap 主要由两部分组成：Network Tap、Packet Filter。Network Tap 从网络设备驱动程序中收集数据拷贝，Packet Filter 决定是否接受该数据包。目前很多优秀的网络数据包捕获软件都是以 Libpcap 为基础，如著名的 Tcpdump、Ethereal 等。

2 数据包捕获机制

由于在安全程序中通常需要对网络通讯的细节（如连接双方地址 / 端口、服务类型、传输控制等）进行检查、处理或控制，象数据包截获、数据包头分析、数据包重写、甚至截断连接等，都几乎在每个网络安全程序中必须实现。为了简化网络安全程序的编写过程，提高网络安全程序的性能和健壮性，同时使代码更易重用与移植，最好的方法就是将最常用和最繁复的过程函数封装起来，以 API library 的方式提供给开发人员使用。

在众多的 API library 中，对于类 Unix 系统平台上的网络安全工具开发而言，目前最为流行的 C API library 有 libnet、libpcap、libnids 和 libicmp 等。它们分别从不同层次和角度提供了不同的功能函数。使网络开发人员能够忽略网络底层细节的实现，从而专注于程序本身具体功能的设计与开发。

在网卡缺省的工作模式下，只能收到广播的数据包和目的地址是自己的数据包。所以在进行数据包捕获时，我们首先要将网卡设置为混杂模式，这样就能捕获到流经该网卡的所有数据包。值得注意的是，捕获到的仅仅是数据包的一份拷贝，不影响数据包的正常传输。正常情况下当网络数据包到达网卡时，它常规的传输路径是依次经过网卡、设备驱动器、数据链路层、IP 层、传输层、最后到达用户层。Libpcap 包捕获机制是在数据链路层增加一个旁路处理。当一个数据包到达网络接口时，Libpcap 首先利用已经创建的 Socket 从链路层驱动程序获得该数据包的拷贝，再通过 Tap 函数将数据包发送给 BPF 过滤器。BPF 过滤器收到数据包后，根据用户已经

定义好的过滤规则对数据包进行逐一的匹配，符合过滤规则的数据包就是我们需要的，将它放入内核缓冲器，并传递给用户层缓冲器，等待应用程序对其进行处理。不符合过滤规则的数据包就被丢弃。 如果没有设定过滤规则，所有的数据包都将被放入内核缓冲器。

Libpcap 主要包括三个部分：最底层的是针对硬件设备接口的数据包捕获机制，中间的是针对内核级的包过滤机制，第三层是针对用户程序的接口。Libpcap 通过几个主要的函数来捕获数据包。

2.1数据包捕获应用程序框架

下面列举一个最简单的程序框架：

⑴ 查找可以捕获数据包的设备

⑵ 创建捕获句柄，准备进行捕获

⑶ 设置过滤条件，编译和安装过滤代码

⑷ 进入（死）循环，反复捕获数据包

⑸ 处理捕获的数据包

⑹ 在程序结束时关闭捕获句柄

首先调用 pcap_lookupdev() 函数找到可用的网络接口设备，并返回该网络接口设备的名称。或者我们直接指定需要捕获数据包的接口。接下来调用 pcap_open_live() 函数，利用上面打开的接口设备名创建捕获句柄，准备捕获数据包。同时在该函数中，设置了捕获数据包的相关参数，包括每次捕获数据包的最大长度、等待超时的时间、接口设备的状态、分配错误信息缓冲区的大小等。然后可以设置过滤条件，把我们需要的数据包返回到我们的应用程序中。接着调用 pcap_loop 函数，设置回调函数，进入死循环。开始捕获数据包，当捕获数据包后，在回调函数中进一步处理。

2.2 示例代码

```
void CapLoopEth0(void *p)
{
        pd = pcap_open_live(dev,8196,g_SystemBaseSetup.
bGateWay==0?1:0,-1,ebuf);
        datalink = pcap_datalink(pd);
        // 设置过滤
         sprintf(filter,"not ether src %02X:%02X:%02X:%02X
:%02X:%02X",
                g_NetInterfaceSetup.uEth0MacAddr[0],
                 g_NetInterfaceSetup.uEth0MacAddr[1],
                 g_NetInterfaceSetup.uEth0MacAddr[2],
                g_NetInterfaceSetup.uEth0MacAddr[3],
                g_NetInterfaceSetup.uEth0MacAddr[4],
                g_NetInterfaceSetup.uEth0MacAddr[5]);
        pcap_compile(pd, &fcode, filter, 1, 0)；
        pcap_setfilter(pd, &fcode)；
        status = pcap_loop(pd,-1,ParseFromInPackets,ebuf);
        if (status == -1)
        {
                PrintErrLog("InCapLoopEth0: %s\n", ebuf);
        }
        if (status == -2)
        {
                PrintErrLog("InCapLoopEth0: We got
interrupted!\n");
        }
        pcap_close(pd);
        bRunFlagIn=0;
        return NULL;
}

void ParseFromInPackets(u_char *useless,const struct
pcap_pkthdr* pkthdr,const u_char* packet)
{
    struct ether_header *ep;
    ep = (struct ether_header *)packet;
    ether_type = ntohs(ep->ether_type);
    switch (ether_type)
    {
    case ETHERTYPE_IP:
    case ETHERTYPE_ARP:
    case ETHERTYPE_REVARP:
    default:
            break;
    }
    return;
}
```

3 数据包过滤机制

过滤器既可以放在用户空间执行也可以放在内核空间执行，但是由于数据从内核空间向用户空间拷贝要耗费大量的 CPU 周期。为了减少从内核空间向用户空间拷贝的数据包的数量，提高捕获数据包的效率，最好把过滤器放在内核级。这一点要特别注意，特别是用 Libpcap 开发网关类产品时，效率是首要考虑的问题。只有需要的数据才让他进入循环中进行处理。通过调用 pcap_compile 函数设置过滤条件，把无关

的数据都屏蔽。

大量的网络监控程序目的不同，期望的数据包类型也不同，但绝大多数情况都都只需要所有数据包的部分。包过滤机制的引入就是为了解决上述问题，用户程序只需简单的设置一系列过滤条件，最终便能获得满足条件的数据包。包过滤机制实际上是针对数据包的布尔值操作函数，如果函数最终返回 true，则通过过滤，反之则被丢弃。形式上包过滤由一个或多个谓词判断的并操作（AND）和或操作（OR）构成，每一个谓词判断基本上对应了数据包的协议类型或某个特定值，例如：只需要 TCP 类型且端口为 110 的数据包或 ARP 类型的数据包。包过滤机制在具体的实现上与数据包的协议类型并无多少关系，它只是把数据包简单的看成一个字节数组，而谓词判断会根据具体的协议映射到数组特定位置的值。如判断 ARP 类型数据包，只需要判断数组中第 13、14 个字节（以太头中的数据包类型）是否为 0X0806。从理论研究的意思上看，包过滤机制是一个数学问题，或者说是一个算法问题，其中心任务是如何使用最少的判断操作、最少的时间完成过滤处理，提高过滤效率。

Libpcap 重点使用 BPF（BSD Packet Filter）包过滤机制，BPF 于 1992 年被设计出来，其设计目的主要是解决当时已存在的过滤机制效率低下的问题。BPF 的工作步骤如下：当一个数据包到达网络接口时，数据链路层的驱动会把它向系统的协议栈传送。但如果 BPF 监听接口，驱动首先调用 BPF。BPF 首先进行过滤操作，然后把数据包存放在过滤器相关的缓冲区中，最后设备驱动再次获得控制。注意到 BPF 是先对数据包过滤再缓冲，避免了类似 sun 的 NIT 过滤机制先缓冲每个数据包直到用户读数据时再过滤所造成的效率问题。

BPF 对 CFG 算法的代码实现非常复杂，它使用伪机器方式。BPF 伪机器是一个轻量级的，高效的状态机，对 BPF 过滤代码进行解释处理。由用户来写过滤代码太过复杂，因此 libpcap 允许用户书写高层的、容易理解的过滤字符串，然后将其编译为 BPF 代码。

Libpcap 调用 pcap_compile() 函数将其编译成 BPF 代码，然后利用 pcap_setfilter() 函数把 BPF 代码安装到内核中。这样 BPF 就可以在内核中根据用户定义好的过滤规则，对捕获到的数据包进行过滤。通过 BPF 过滤器的数据包会被放入内核缓冲器，没有通过的数据包直接被丢弃。Libpcap 在内核中使用两个缓冲器：分别是存储缓冲器和保持缓冲器。存储缓冲器用来保存通过过滤器的数据包，而保持缓冲器则用来向用户提供数据。当数据包通过过滤器后，被不断的放入存储缓冲器，同时用户程序调用函数不断的从保持缓存器中取数据。当存储缓冲器满并且保持缓冲器为空时，Libpcap 将它们进行交换，继续重复上面的动作。但是如果存储缓冲器装满了，而用户程序还没有取完保持缓冲器中的数据，那么再通过过滤器的数据包就不得不被丢弃，直到保持缓冲器变成空为止。Libpcap 在内核和用户层使用的缓冲器的大小是固定的，它们都是 32KB。这样就存在一定的局限性，每次调用系统对它进行读写时，只能存取 32KB 的数据。在高速网络中，调用系统读写的次数会显著增多，相应的丢包率也会增高。

作者简介：平震宇（1975-），男，江苏信息职业技术学院 计算机系，工程师，研究方向：网络安全与管理领域、HFC 网络管理。

4 结论

目前众多的网络安全程序、工具和软件都是基于 socket 设计和开发的。由于在安全程序中通常需要对网络通讯的细节（如连接双方地址 / 端口、服务类型、传输控制等）进行检查、处理或控制，象数据包截获、数据包头分析、数据包重写、甚至截断连接等，都几乎在每个网络安全程序中必须实现。为了简化网络安全程序的编写过程，提高网络安全程序的性能和健壮性，同时使代码更易重用与移植，最好的方法就是将最常用和最繁复的过程函数。

Libpcap 最主要的优点在于平台无关性，用户程序几乎不需做任何改动就可移植到其它平台上；其次，Libpcap 也能适应各种过滤机制，特别对 BPF 的支持最好。在开发网络安全程序是选用 libpcap 是一个很好的选择。 （责编 张岩）

参考文献：

[1]Loris WindumpManul. http://www.tcpdump.org,2002

[2]StevensW R.TC P/IPI llustrated(VoL1,2).A ddison-Wesley,1995

Blue Coat 公司完成对 Packeteer 公司的收购

不久前，广域网络应用传输与安全网络网关设备厂商 Blue Coat 系统公司完成了对提供包括应用程序分类和性能管理等尖端 WAN 流量优化技术的 Packeteer 公司的收购。收购在网络流量优先化和整形网络方面保持领先的 Packeteer，不仅扩展了 Blue Coat 的市场领先地位，同时能够为广大用户群提供更加广泛的 WAN 应用交付解决方案。合并后，Blue Coat 在全球 26 个国家和地区的 1400 名员工可以为 1 万 5 千多 Blue Coat 用户带来最先进的综合的新一代可视性和控制性智能网络，预计公司年收入将达加到 5 亿美金。（编辑 马华）