基于 WinPcap 库的通用程序设计模型

General Program System Design Model Based 0n WinPcap

简清明 （四川理工学院网管中心 643000）

摘要：大多数的 Unix 系统都提供一组称为 Libpcap 的系统调用，为用户空间的数据包捕获提供便利. WinPcap 将这些功能移植到 Win32 平台下并增加了一些新的特性。本文描述 WinPcap 体系结构的细节并给出基于 WinPcap 程序的一般设计。

关键词：BSD 数据包过滤器（BPF） WinPcap 库

1 WinPcap 概述

WinPcap 是 Politecnico di Torino 的 NetGroup 小组开发的基于 Win32 平台的包捕获和网络分析的基础构架，由 UNIX 下的 libpcap 库移植而来，用于用户层次的数据包截获工作。它为底层网络监控编程提供了一个易于移植的应用框架。WinPcap 库和 Libpcap 一样，采用内核过滤机制，并且只支持 BPF（Berkeley 分帧过滤器，WinPcap 下称为 NPF）接口的内核过滤。如果主机上没有 BPF 机制. 则所有的效据包都必须读取到用户空间后，再在 WinPcap 库中进行过滤处理，这样就会增加额外的处理负担，导致性能的下降。

2 WinPcap 网络数据捕获体系

WinPcap 的数据捕获体系称为 NPF，源于著名的 BPF。BPF 产生于 1992 年，由两个组件组成：BPF 网络阀和数据包过滤器 Filter。通常在数据报到达网络适配器时，设备驱动程序会将数据报传递给协议栈的其它部分。网络阀实际上是个回调函数，从网络设备驱动程序处收集数据，并将它们传递给正在监听的应用程序。收集的数据报若满足 Filter 的条件，那么将它们保存在缓冲区内，在应用程序读取时，将它们拷贝到用户层的缓冲区中。BPF 经使用证明是一个功能强大的和性能稳定的体系结构，WinPcap 保留了 BPF 绝大部分的重要模块：一个过滤器，两个缓冲区（核心和用户）以及用户空间的两个函数库，其基本结构如图 1 所示. 但是，WinPcap 在结构上和捕获堆栈的方式上与 BPF 也有一些不同之处。一个重要不同是 NPF 采用循环缓冲区作为核心缓冲区，可以一次复制一组数据包。数据复制也不在使用固定长度（libpcap 的核心和用户层使用同样的缓冲区大小 32KB）. 这种实现允许所有的存储空间都可用于存储数据，大大提高了缓冲区的利用率。

Winpcap 为应用程序提供了与数据包截获有关的两个不同层次的编程接口：底层的 packet. dll，高层的 wpcap. dll 以及一个虚拟设备驱动程序 npf. vxd，应用程序可以自由选用。Npf. vxd 工作在内核级，与操作系统和硬件是相关的；packet. dll 和 wpcap. dll 工作于用户级，与具体的硬件无关，其中 packet. dll 属于底层的 API，与操作系统是相关的，在 Win9x（95，98，me）和 WinNT（NT4，2000，XP）系列中有所不同。Wpcap. dll 是高层的 API，他是与操作系统无关的。因此基于 packet. dll 和 wpcap. dll 编写的应用程序各有其不同的优点，前者能够直接访问到驱动程序，便于实现一些底层的功能，而后者与 Unix 系统中使用的 libpcap 兼容，基于它所编写的程序在 windows 系统和 Unix 系统中都可以运行。

3 WinPcap 主要编程接口

WinPcap 为用户编程提供了一系列功能强大的函数调用，其中几个重要函数如下：

（1）typedef void（ * pcap_handler ）（u_char * user，const struct pcap_pkthdr * pkt_header，const u_char * pkt_data）. 这是回调函数的原型. 用户可以通过 pcap_loop 或 pcapf_dispatch 定义自己的回调函数。

两者之间的唯一的差别是处理超时的方式不同：pcap_loop 将忽略超时参数而 pcap_dispatch 在到设定时间时将产生读超时的错误。

（7）u_char * pcap_next(pcap_t * p, struct pcap_pkthdr * h)

pcap_next( ) 返回一个指向 pcap_pkthdr 结构的

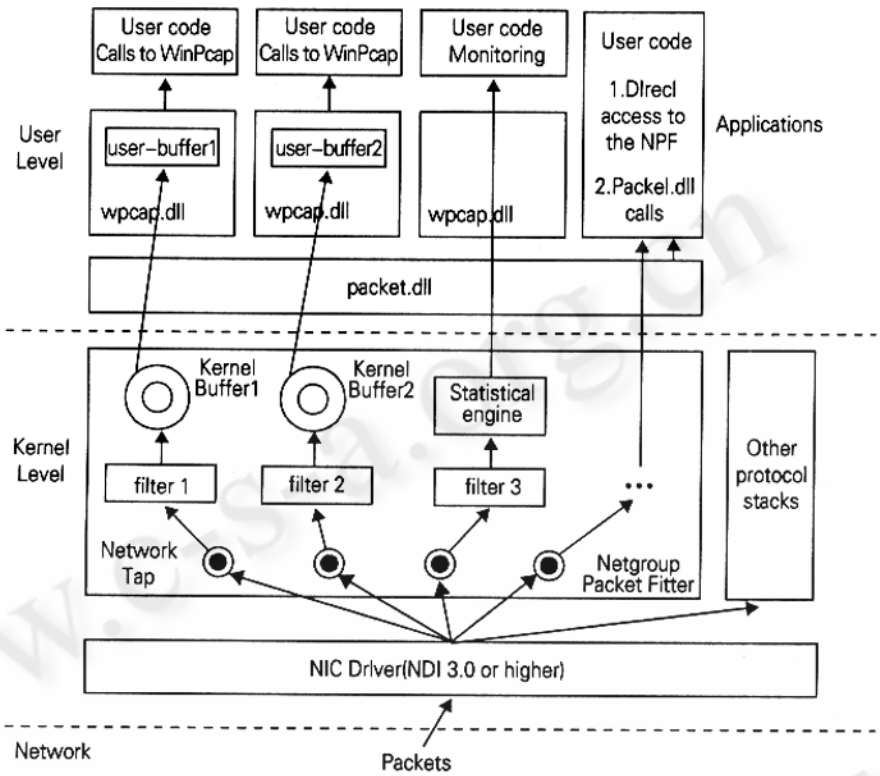

图 1　WinPcap 数据捕获体系

（2）pcap_open_live ( ). 用来获得一数据截获描述符。

（3）pcap_dispatch ( ). 该函数捕获网络数据包，然后调用参数中指定的 pcap_handler 函数对数据包进行处理，并根据设定的条件返回. 另一个相似的命令是 pcap_loop( )。一般在简单的应用中 pcap_loop( )就已足够，而在一些复杂的程序里往往用 pcap_dispatch( )。

（4）pcap_compile( )：编译一个过滤设备，它通过一个高层的 boolean 型变量和字符串产生一系列能够被底层驱动所解释的二进制编码。

（5）pcap_setfilter( )：用来关联一个基于内核驱动的过滤器。这时所有网络数据包都将流经过滤器，并拷贝到应用程序中。函数 pcap_compile( ) 和 pcap_setfilter( ) 用来实现数据过滤的功能。

（6）void pcap_close (pcap_t * p). 关闭相关的文件并释放对应资源。

指针。另一个具有相似功能但更加强大的函数是 pcap_next_ex( )，不过它只在 Win32 环境下运行，因为它不是原始 libpcap API 中的一部分。

4　基于 WinPcap 的程序流程

WinPcap 的功能非常强大，基于 WinPcap 的应用程序虽然各不相同，但都有相似之处。一个建立在 WinPcap 之上的应用程序一般的处理流程如图 2 所示。

5　应用实例

现在我们以一个实际的例子来演示基于 WinPcap 的程序的开发过程。该程序是一个简单的 sniffer，嗅探所有流经端口 23 网络流量，并打印包的长度。源代码如下：

```
#include <pcap.h>
#include <stdio.h>
```

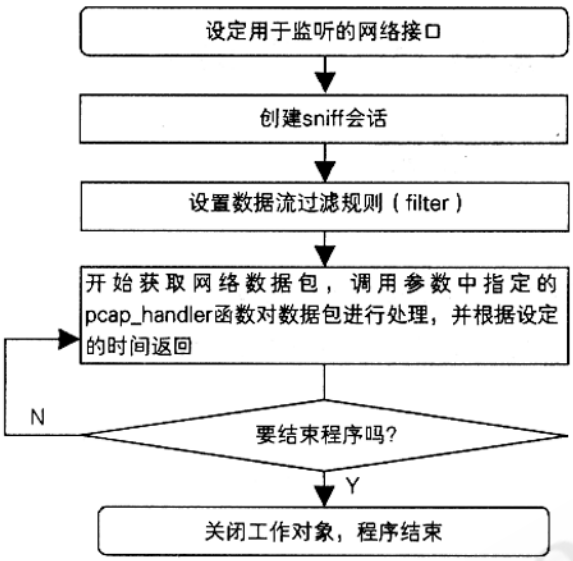

图 2　基于 WinPcap

```
int main( )
{
  pcap_t * handle; /* 会话句柄 */
  char * dev; /* 嗅探的网络设备 */
  char errbuf[ PCAP_ERRBUF_SIZE ]; /* 错误字符串 */
  struct bpf_program filter; /* 已编译的 filter */
  char filter_app[ ] = " port 23" ; /* filter 表达式 */
  bpf_u_int32 mask; /* 本机 IP 地址掩码 */
  bpf_u_int32 net; /* 本机 IP */
  struct pcap_pkthdr header; /* The header that pcap gives us */
  const u_char * packet; /* 实际数据包 */
  /* 设定监听设备 */
  dev = pcap_lookupdev( errbuf) ;
  pcap_lookupnet( dev, &net, &mask, errbuf) ;
  /* 在混杂模式下打开会话 */
  handle = pcap_open_live( dev, BUFSIZ, 1, 0, errbuf) ;
  /* 编译和应用 filter */
  pcap_compile ( handle, &filter, filter_app, 0, net) ;
  pcap_setfilter( handle, &filter) ;
  /* 进入(死)循环,反复捕获数据包捕获数据包 */
  for( ; ; )
    {
    while( ( ptr = ( char * ) ( pcap_next( handle , &header ) ) ) = = NULL) ;
      {/* 打印包的长度,这里可插入其它用户操作 */
      printf( " Jacked a packet with length of [ % d] \n" , header. len) ;
      }
    /* 这里可插入其它用户操作 */
    }
  /* 关闭会话 */
  pcap_close( handle) ;
  return( 0) ;
}
```

6　结束语

WinPcap 强大的数据捕获体系和应用编程接口，为 windows 平台下的网络安全应用提供了方便高效的开发手段，期望本文的探讨能给相关领域感兴趣的技术人员一点帮助。

参考文献

1 McCanne S, Jacobsml V The BSD Packet Filter: A New Architecture for User – level Packet Capture Proceedings of the 1993 Winter USENIX Technical Conference USENIX, 1993 – 01.

2 Fulvio Risso, Loris Degioanni, An Architecture for High Performance Network Analysis, Proceedings of the 6th IEEE Symposium on Computers and Communications (ISCC 2001), Hammamet, Tunisia, July 2001.

3 Tim Carstens, Programming with pcap, http://broker. dhs. org/pcap. html.