基于 Winsock 技术的数据包解析研究

熊安萍
(重庆邮电学院计算机科学与技术学院　重庆 400065)

摘　要　数据包解析技术是数据包过滤的基础。对数据包进行解析，是基于数据包过滤的防火墙要解决的核心问题，构造数据包的协议有很多种，要根据构造数据包的协议对该包进行处理，要正确理解在网络中传输的单元，进而才能很好地控制网络单元的传输，实现数据包的过滤。Winsock 的服务提供者编程接口的编程技术，打破了底层网络服务提供者的透明性，提供了修改系统 SPI 接口服务的可能性，利用这项技术能比较容易地完成数据包过滤功能，具体地说就是能增加一些自定义的功能函数，来实现数据包通信的控制，比如截获、转发、丢弃数据包等功能，也就是所说的防火墙实现的功能。当然也可以在这个基础上延伸下去，从而可以完成诸如传输质量控制、扩展 TCP/IP 协议栈、URL 过滤及网络安全控制等功能。

关键词　防火墙，包过滤，包解析，Winsock 服务提供者编程接口

Research on Packet Analyzing Based on Winsock Technology

XIONG An-Ping
(College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065)

Abstract　Abstract Packet analyzing is the basis of packer filtering. Analyzing packets is the most important issue which must be resolved by packet filtering firewall. There are many types of protocols to construct packets. It should manipulate packets according to the protocol used. It has to comprehend exactly packets, control them, and implement packet filtering at last.

Winsock SPI breaks the transparence of services at the bottom, and provides the possibility to modify the SPI service connection. It can implement packer filtering easily. Concretely, we can append self-defined functions to control communications of packets. For example, it can intercept, transmit and discard packets. Of course, it can also accomplish QoS, expand the TCP/IP protocol stack, filter URL and defend the network security.

Keywords　Firewall, Packet filter, Packet analyzing, Winsock SPI

1　引言

随着计算机网络技术的飞速发展，网络安全问题显得尤其突出。互联网已经发展成熟，网络通信带宽增加，利用网络进行办公、进行商务活动已经大众化。网络的运用得到提高和扩展。网上交易以及网上银行不知不觉走入人们的生活，网络的硬件也正在飞速发展，已经缓解了硬件需求上的压力，而网络的传输和通信安全问题却成了制约网络进一步应用的瓶颈。

Internet 互联网是一种高信息量和即时性信息网络，它这样的特点就要求这个网络有极高的安全性，保证网络信息的可靠性。当然安全问题就成了 Internet 诸多技术中的一个至关重要的环节[1~3]。人们努力用 Internet 来推动社会的发展，服务社会，然而任何事物都有它的负面影响，尤其是这种新兴技术，如果没有解决好网络安全问题，那它就会给人们致命的伤害。

2　Winsock 2 标准网络接口

Winsock 2[4,5]是一个为上层应用程序提供的一种标准网络接口，而不是协议，所以它可以用于发现和使用任意数量的底层传输协议所提供的通信能力。上层应用程序不用关心 Winsock 实现的细节，它为上层应用程序提供透明的服务。最开始的 Winsock 是围绕着 TCP/IP 协议运行的，但是在 Winsock 2.0 中却增加了对更多传输协议的支持。Winsock 2.0 不仅提供了一个供应用程序访问网络服务的 Windows socket 应用程序编程接口(API：Application Programming Interface)，还包含了由传输服务提供者和名字解析服务提供者实现的 Winsock 服务提供者接口(SPI：Service Provider Interface)和 ws2_32.dll。

Winsock 2.0 引入的这个新功能(SPI 技术)打破了服务提供者的透明，让开发者可以编写自己的服务提供者程序。SPI 以动态链接库(DLL)的形式存在，它工作在应用层，为上层 API 调用提供接口函数。

3　解析数据包的类 CProtocolInfo

CProtocolInfo 类是用来分析数据包的模块。计划将 HTTP 协议的域名、FTP 协议上传和下载的文件名等信息分析出来作为备注字段保存在日志信息中。

由于这些信息按协议格式包含在发送和接收的数据包中，因此在处理发送和接收的数据数据包时利用这个模块可将需要的信息分离出来。

CProtocolInfo 类的成员函数表如表 1 所示。

熊安萍　副教授，主要研究方向：操作系统、计算机网络等。

表 1 CProtocolInfo 类成员函数

成员函数	说明
GetProtocolInfo	从数据包缓冲区中解析出自己感兴趣的数据
GetFromSend	解析发送的数据包
GetFromRecv	解析接收的数据包
GetFtp	解析 FTP 协议，解析出登录服务器的用户名、密码和传输的文件信息
GetHttp	解析 HTTP 协议，从包头中找出主机的信息

表 2 参数表

参数	说明
session	指向需要填充的备注字段的网络封包结构的指针
buf	指向要进行数据解析的网络数据缓冲区指针
nBufLenth	缓冲区的数据长度
IsSend	TRUE：表示解析发送数据； FALSE：表示解析接收数据

3.1 GetProtocolInfo 总体控制函数

GetProtocolInfo 用来解析协议信息的总体控制函数，它是供外部调用的公共函数。它根据当前工作状态是发送还是接收数据来判断是调用 GetFromSend 函数还是调用 GetFromRecv 函数来解析协议信息。

3.2 解析(发送的)数据包

GetFromSend 函数是解析发送的数据包的总体控制函数。它根据数据包的不同类型分别调用不同的协议解析函数进行数据包的解析采样工作。

3.3 解析(接收的)数据包

GetFromRecv 是解析接收的数据包的总体控制函数。根据接收数据包的不同协议类型分别调用不同的协议解析函数进行数据包的解析采样工作。这里也只对 http 和 ftp 协议进行数据包的解析。

3.4 Ftp 协议数据包的解析

GetFtp 函数从 FTP 协议里分离出用户名、密码及上传/下载的文件名信息，并把需要的信息保存到 Session 的备注字段里。

3.5 Http 协议数据包的解析

GetHttp 函数从 HTTP 协议里分离域名信息，并把它保存在 Session 的备注字段里。

4 数据包解析的实例

4.1 HTTP 协议包头实例

下面是一个访问某网站主页时的一个 HTTP 协议包头实例：

```
GET/HTTP/1.1
Accept: */*
Accept-Language: zh-cn
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Host: www.xx.xx.cn
Connection: Keep-Alive
```

实例中可以看到一个 Host 栏位连接的域名，用一种简单的算法就可以从这些数据中提取出 Host 的内容。仔细分析数据包的结构可以发现，域名包含在“Host:”关键字和回车切换符之间，利用这个结构特点可以从数据包结构中把域名信息分离出来。Gethttp()函数就是根据这个来编写的。

4.2 FTP 下载/上传文件数据包实例

下面是一个 FTP 下载文件数据包实例：

```
USER test
PASS test
PORT 192,168,10,1,4,8
RETR a.bat
```

FTP 上传文件数据包如下：

```
PORT 192,168,10,1,4,9
STOR b.bat
```

在利用 FTP 进行文件传输时必须首先登录 FTP 服务器，在登录时需要用户认证，“USER”和“PASS”关键字就是分别登录服务器的用户名和密码。“PETR”是标志下载文件名的关键字，“STOR”是表示上传文件名的关键字。知道了数据包的结构特点后就可以解析出需要的信息。

5 封包的处理

封包是一种结构体，记录了一次完整的通信信息，通过封包来记录和控制网络中的通信，它与数据包不同，数据包是指传输过程中对传输的数据进行打包，加入各层的信息。封包是自己定义的结构体，要能理解这点。

5.1 网络封包结构

网络封包的信息都存在下面这个 SESSION 结构里。

```
typedef struct _SESSION
{
SOCKET s; //Socket 连接标志
DWORD ulRemoteIP; //目的 IP 地址
Ctime tStartTime; //连接开始时间
BYTE bDirection; //连接方向
BYTE bProtocol; //协议类型
UINT uiPort; //远端端口
BYTE bAction; //管制动作
UINT uiLocalPort; //本地端口
DWORD ulLocalIP; //本地 IP 地址
DWORD ulSendData; //发送数据流量
DWORD ulRecvData; //接收数据流量
TCHAR sPathName[MAX_PATH];
  //发生连接的应用程序的路径
TCHAR sMemo[MAX_PATH]; //备注信息
Ctime tEndTime; //连接结束时间
} SESSION, *PSESSION;
```

5.2 网络封包的初始化

网络封包处理函数定义为 BOOL InitializeSession(SESSION * session)，它对网络封包记录进行初始化。参数 Session 是结构 SESSION 的结构指针，指向用来初始化的 SESSION 结构数据缓冲区，它永远返回 TRUE。具体工作为设置 Session 各个成员变量的初始值。

5.3 创建网络封包记录

创建新的网络封包记录函数 CreateSession，定义为 int CreateSession(SOCKET s, int nProtocol)。输入的参数 s 是 Socket 标志，输入的参数 nProtocol 为 Socket 连接的协议信息。根据参数 s 和 nProtocol 得出协议信息，并将时间和应用程序信息一起写进新建的网络封包记录里。

(下转第 134 页)

Kececioglu 与 Ravi 首先对基因组反转和移位操作并存的问题进行研究，对有符号基因组给出了近似度为 1.5 的多项式时间算法，同时对无符号基因组给出了近似度为 2 的多项式时间算法，这两个算法的时间复杂度都是 $O(n^2)$。Hannenhalli 和 Pevzner 设计了有符号基因组反转和移位操作并存的的多项式算法[25]，并给出了排序距离的计算公式。

总结 随着快速测序技术的发展，对大规模 DNA 分子的研究与其中的基因相对次序有关。基因组重组排序为重构生物进化的相对关系提供了较好的估计值，并已经被大量的生物学实验数据所验证。本文讨论了几种典型的基因组重组操作及其研究进展。

参 考 文 献

1 Bafna V, Pevzner P. Genome rearrangements and sorting by reversals. FOCS, 1993. 148～157
2 Hannenhalli S, Pevzner P A. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). STOC, 1995. 178～189
3 Kaplan H, Shamir R, Tarjan R E. Faster and simpler algorithm for sorting signed permutations by reversals. SIAM Journal of Computing, 2000, 29(3): 880～892
4 Bader D A, Moret B M E, Yan Mi. A Linear-Time Algorithm for Computing Inversion Distance Between Signed Permutations with an Experimental Study. Journal of Computational Biology, 2001, 8(5): 483～491
5 Kececioglu J, Sankoff D. Exact and approximation algorithms for the inversion distance between two permutations. CPM, 1993. 87～105
6 Hannenhalli S, Pevzner P A. To cut … or not to cut (applications of comparative physical maps in molecular evolution). SODA, 1996. 304～313
7 Caprara A. Sorting by reversals is difficult. RECOMB, 1997. 75～83
8 Berman P, Karpinski M. On some tighter inapproximability results. Electronic Colloquium on Computational Complexity (ECCC), 1998, 5(29)
9 Christie D A. A 3/2 Approximation Algorithm for Sorting by Reversals. SODA, 1998. 244～252
10 Berman B, Hannenhalli S, Karpinski M. 1.375-approximation algorithm for sorting by reversals. ESA, 2002. 200～210
11 Hannenhalli S. Polynomial-time Algorithm for Computing Translocation Distance Between Genomes. CPM, 1995. 162～176
12 Zhu Daming, Ma Shaohan. Improved polynomial-time algorithm for computing translocation distance between genomes. The Chinese Journal of Computers (in Chinese), 2002, 25(2): 189～196
13 Wang Lusheng, Zhu Daming, Liu Xiaowen, An $O(n^2)$ algorithm for signed translocation. APBC, 2005. 349～358
14 Li G, Qi X, Wang X, et al. A linear-time algorithm for computing translocation distance between signed genomes. CPM, 2004
15 Kececioglu J, Ravi R. Of mice and men: Algorithms for evolutionary distances between genomes with translocation. SODA, 1995. 604～613
16 Zhu Daming, Ma Shaohan, Wang Lusheng. Sorting unsigned genome by translocation is NP-hard. Theor Comput Sci, 2006, 352(1-3): 322～328
17 Cui Yun, Wang Lusheng, Zhu Daming. A 1.75-Approximation Algorithm for Unsigned Translocation Distance. ISAAC, 2005. 392～401
18 Bafna V, Pevzner P A. Sorting by transpositions. SIAM Journal on Discrete Mathematics, 1998, 11(2): 224～240
19 Christie D A, Irving R W. Sorting Strings by Reversals and by Transpositions. SIAM J Discrete Math, 2001, 14(2): 193～206
20 Hartman T. A Simpler 1.5-Approximation Algorithm for Sorting by Transpositions. CPM, 2003. 156～169
21 Elias I, Hartman T. A 1.375-Approximation Algorithm for Sorting by Transpositions. WABI, 2005
22 Walter M E T, Dias Z, Meidanis J. Reversal and Transposition Distance of Linear Chromosomes. SPIRE, 1998. 96～102
23 Gu Qian-Ping, Peng Shietung, Sudborough I H. A 2-Approximation Algorithm for Genome Rearrangements by Reversals and Transpositions. Theor Comput Sci, 1999, 210(2): 327～339
24 Hartman T, Sharan R. A 1.5-approximation algorithm for sorting by transpositions and transreversals. J Comput Syst Sci, 2005, 70(3): 300～320
25 Hanennhalli S, Pevzner P A. Transforming Men into Mice (Polynomial Algorithm for Genomic Distance Problem). FOCS, 1995. 581～592

（上接第 82 页）

5.4 删除网络封包记录

删除网络封包记录函数 DeleteSession 定义为 int DeleteSession(SOCKET s)，输入的参数 s 为 Socket 标志，用来区分不同的网络封包。它的主要功能是从已有的网络封包记录中删除一条，在删除之前要做的是调用 SendSessionToApp 函数，将删除的记录发给执行文件。

5.5 寻找功能

寻找函数 FindSession 定义为 int FindSession (SOCKET s)。输入的参数 s 为要查找的的网络封包记录的标志符。函数返回的是网络封包结构在数组中的索引，但如果返回的索引值大于网络数据封包总数，则表示没有找到。

5.6 设置功能

设置函数 SetSession 的定义为 int SetSession(SESSION * session, BYTE bDirection, UINT uiPort, DWORD ulRemoteIP)。输入和输出的参数 session 为指向要设置的网络封包数据结构，输入参数 bDirection 为连接的进出方向，输入参数 uiPort 为连接的目的端口，输入参数 ulRemoteIP 为连接的目的 IP 地址。它的主要功能是对一个已经存在的网络封包记录进行修改，修改的字段有：协议类型、进出方向、目的端口和目的 IP 地址。

5.7 修改记录动作

修改记录动作函数 SetSessionEx 定义为 int SetSessionEx (SESSION * session, BYTE bDirection, const TCHAR * pMemo, int ByteCount, BOOL isSend)。它与 SetSession 的不同是，其修改字段为：本地端口/IP、进出方向、备注信息及进出流量。输入参数 pMemo 是连接的备注信息，输入参数 isSend 标识是发送还是接收的流量，TRUE：ByteCount 表示出流量；FALSE：ByteCount 表示进流量。

结论 数据包的解析技术是从截获的数据包中收集比较重要的信息，定义了一个封包结构体来存储这些关键的信息，然后在利用该封包存储的信息去判别类似的数据包的行为，实现防火墙控制数据流通的功能：数据包的监视，实现可视化的数据包监视界面，提供清空监视列表，停止/开始监视及停止/开始滚动功能；控管规则的设置，包括自定义添加，修改及删除控管规则，设置网络 IP 地址段，设置访问时间等；系统参数配置，是否自动启动，是否声音报警，是否闪烁图标报警等。

参 考 文 献

1 Firewall is security device of choice. Australian Electronics Engineering, 2002, 000(M4): E123～E123
2 Wasti S. Hardware assisted packet filtering firewall. In: Proceedings of the 2000-2001 Grad Symposium, CS Dept, University of Saskatchewan, April 2001
3 Yamagaki N, Minami K. Packet Discarding Scheme Considering Both Instantaneous and Historical Use of Network Resources. IEICE Transactions on Communications, 2001, E84-B(8): 2115～2123
4 Li Xin. Stateful Inspection Firewall Session Table Roces Sing [J]. International, 2005, (2): 615～620
5 Law K L E, Leung R. A design and implementation of active network socket programming. Microprocessors and Microsystems, 2003, 27(5-6): 277～284