利用 SPI 进行数据加密传输的研究①

The Research of Data Transmitting Encryption Based on SPI

刘翌南　（长沙理工大学计算机与通信工程学院 410076）

摘要：本文提出了基于 Winsock 2 SPI 网络封包截获技术的数据加密传输的模型并实现之，试验表明此模型可在不改变已有网络通信系统的基础上实现灵活的数据加密传输功能。

关键词：Winsock 2 SPI　对称加密算法　散列函数

1　引言

随着计算机网络技术的日益发展和普及，远程网络数据传输的安全性显得越来越重要。一般情况下，若要安全地实现远程数据传输，可采用 Winsock 技术编写专用的数据加密传输系统，也就是说首先需要将数据加密，然后采用 Winsock 技术将数据包传送出去，在接收方接收到数据之后，再将数据包解密。本文实现的文件加密传输系统不同于上述实现方法。本文实现了一种基于 SPI 网络封包截获技术的文件加密传输方法，此方法具有更好的通用性及灵活性。

2　传输模型

2.1　Winsock 2 SPI 剖析

Winsock 2 引入了一种新的编程接口，称为服务提供者接口（Service Provider Interface，SPI）。Winsock 2 SPI 除了有完成网络传输的传输服务提供者，还有提供友好名字服务的名字空间服务提供者。其中，传输服务提供者能够提供建立通信、传输数据、流量控制和错误控制等服务。本文仅对传输服务提供者进行讨论。Winsock 2 提供的服务其结构如图 1 所示[1]。

各种服务提供者是 Windows 支持的动态链接库（DLL），挂靠在 Winsock 2 的 Ws2_32. dll 模块下。对于应用程序使用的许多 Winsock 2 API 函数来说，这些服务提供者都提供了与它们对应的函数（例 如，API 函数 WSASend 有相对应的 SPI 函数 WSPSend）。多数情况下，一个应用程序在调用 Winsock 2 API 函数时，Ws2_32. dll 会调用相应的 Winsock 2 SPI 函数，利用特定的服务提供者执行所请求的服务。

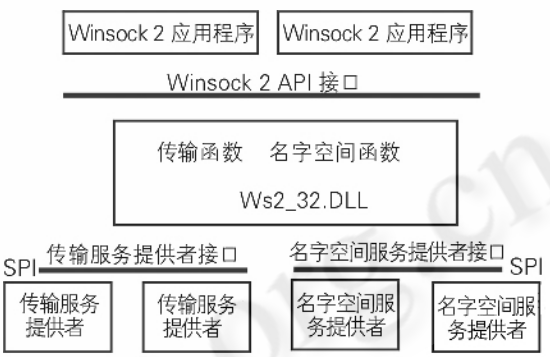

图 1　Winsock 2 SPI 结构

SPI 以动态链接库的形式出现，工作在 TCP/IP 协议的应用层，为上层 API 调用提供接口函数。由于 SPI 工作在 TCP/IP 协议的应用层，因此对基于应用层的数据包 SPI 都可以截获[2]。

2.2　传输模型

基于 SPI 的文件加密传输系统的工作模型如图 2 所示。

在发送方，用户层通信程序发送的网络封包被自定义的 SPI 程序所截获，SPI 程序将数据包的 IP 地址、端口等信息提取出来，经过规则判断函数判断之后，如果需要加密，则调用加密函数完成加密工作，并在封包

① 基金项目：湖南省教育厅基金资助项目（03C078）

中设置加密标志。数据接收方在 windows 核心层将接收的网络封包上传给用户层接收程序之前，自定义的 SPI 程序又将此数据封包截获，规则判断函数首先检查网络封包中的加密标志，若数据包是加密的数据包，则调用解密函数进行解密，最终将解密了的数据包向上传送给用户层的接收程序。

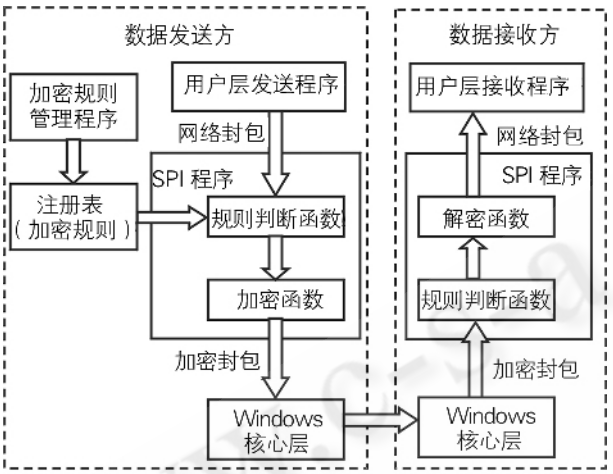

图 2　基于 SPI 的文件加密传输模型

为了更灵活的对加密的规则进行管理，系统需要采用加密规则管理程序对加密规则进行处理，加密规则存储于注册表中。

由于在网络通信中，每一方都需要发送和接收数据，因此，必须在通信双方都运行此系统才能实现正确的数据加密传送。

3　实现的关键技术

3.1　封包截获

用户在进行数据封包截获时，需将自己编写好的 SPI 程序安装到系统上，这时，所有的 Winsock 请求都会首先发送到这个预先编好的 SPI 程序中。在这个程序中，用户可以加上自己的处理程序段，如提取封包中的 IP 地址和端口，进行数据加密/解密等功能，然后调用系统函数，由系统函数完成网络传输功能。

具体实现方法是：传输服务提供者作为一个标准 Windows 平台下的 DLL，它对外只有一个引出函数——WSPStartup，这个函数是 Windows Socket 2 应用程序调用 SPI 程序的初始化函数，也就是入口函数。WSPStartup 的参数 LPWSAPROTOCOL_INFOW 指针提供应用程序所期望的协议信息，然后通过这个结构指针我们可以从注册表中查到所保存的系统服务提供者的 DLL 名称和路径，加载系统服务提供者后，查找到系统 SPI 程序的 WSPStartup 函数的指针，通过这个指针，我们就可以将自己服务提供者的 WSPStartup 函数和系统 SPI 程序的 WSPStartup 函数相关联，从而得到调用系统的 30 个服务提供者的函数指针。由于每个 Winsock 的 API 函数都与 SPI 函数之间具有对应关系，因此，只要将这些 SPI 函数指针赋值为自己函数的地址（也就是自己编写的 SPI 函数），当相应的 Winsock 函数请求发生时就会首先由自己编写的 SPI 函数来处理。

例如，当一个应用程序在调用 Winsock 2 API 函数时（如调用 WSASend 函数），Ws2_32. dll 会调用与这个 API 函数对应的 SPI 函数（WSPSend），由于经过上面的处理之后，基础服务提供者的 WSPSend 函数指针已经被用户自己编写的 WSPSend 函数所替代，因此，就会先执行用户自己编写的程序段，然后用户程序再调用基础服务提供者的 WSPSend 函数，由这个系统函数完成最终的网络传输功能。

3.2　规则管理

加密规则包含以下信息：

（1）端口范围；

（2）IP 地址范围；

（3）是否需要加密的标志；

（4）加密算法标志；

这些加密规则被存储在注册表中。当发送方的网络封包被自定义的 SPI 程序截获之后，规则判断程序读取注册表中存储的规则，若网络封包的 IP 地址和端口范围与加密规则中的相匹配，则读取是否需要加密的标志，若需要加密，则根据加密算法标志决定采用哪种加密算法进行数据加密。

加密规则管理程序用于对加密规则进行管理，加密规则管理程序既可以对需要加密的规则进行管理，也可以对不需要加密的规则进行管理。这样就提供了灵活的设置手段，防止系统对一些不需要加密的标准网络服务进行加密处理。加密规则管理程序采用口令保护方式，只有知道访问密码的人才能对加密规则进行设置。

为了防止加密规则在注册表中被篡改,这些加密规则还需要通过单向散列函数如 MD5、SHA－1 等进行处理[3],加密规则防止篡改的工作原理如图 3 所示。

散列函数以注册表中的加密规则 R 作为输入,计算产生一个固定长度的散列码 H(R)作为输出,H(R)也称为"指纹"或"报文摘要"。可将散列码 H(R)存储在注册表中。每次对规则进行访问的时候,再次对加密规则进行相同的计算,将得到的摘要与存储的摘要进行比对,若相等,则表示加密规则没有被篡改。

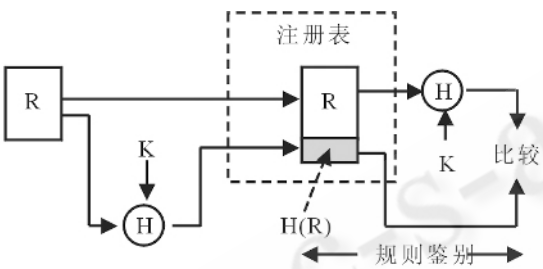

图 3　散列码用于加密规则鉴别的基本原理

3.3 数据加密解密

数据加密采用对称加密方式,常用的对称加密算法包括三重 DES、RC5、Blowfish、IDEA 等。对称加密的

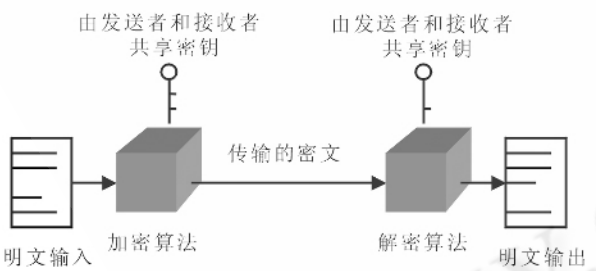

图 4　对称加密的简化模型

基本模型如图 4 所示[3]。明文输入通过加密算法加密产生了密文,密文能够用于数据的安全传输,接收方通过解密算法将密文解密,得到明文输出。发送方和接收方共享相同的密钥,只有知道密钥才能得到正确的解密结果。

在加密算法实现的过程中,由于不同加密算法对明文的长度都有不同的规定,因此在加密之前需要将加密缓冲区中的数据根据加密算法类型进行分段、对于不满足加密长度的数据还要进行字节填充。

本文实现的系统可由发送方的加密规则决定选择哪种加密算法,因此需要在加密的数据包的前面增加一个包头信息,其中包含版本号、加密算法类型标志、加密算法对应的最大加密长度、填充字节数和包含填充字节的发送的总字节数。在接收方可根据包头信息的情况对数据包进行解密。

3.4 系统实现

对网络封包进行截获需要对注册表进行修改,因此需要编写自定义的系统安装函数 InstallModule 和系统卸载函数 RemoveModule。InstallModule 函数完成的功能是在注册表中修改与 SPI 注册项相关的信息。首先保存基础服务提供者的路径信息,然后将注册项中指向基础提供服务者的路径改为用户自己所编写的 DLL 动态链接库的路径。这样,当系统调用系统 SPI 的时候,就会调用用户自定义的 DLL 程序。RemoveModule 函数完成的功能是把注册表中修改过的信息复原。

用户自定义的 SPI 程序是以 DLL 的形式存在的,我们需要对它的入口函数 WSPStartup 进行编写,通过这个函数,可以得到调用系统的 30 个服务提供者的函数指针,并将这些 SPI 函数指针赋值为自己编写的 SPI 函数的地址。这时,当相应的 Winsock 函数请求发生时就会首先调用自己编写的 SPI 函数。

在 SPI 用户程序的实现过程中,并不需要将 30 个服务函数全部拦截,只需要拦截 WSPRecvFrom、WSPRecv、WSPSendTo、WSPSend 函数即可。对于这些函数,实现时都需要编写自定义的代码实现封包截获功能。下面以面向连接的接收函数和发送函数为例,说明系统的实现。

对于发送方,需要编写面向连接的 WSPSend 函数,其算法如下:

(1) 调用数据包分解函数提取数据包中的 IP 地址和端口等信息;

(2) 调用加密规则判断函数,根据 IP 地址、端口等信息判断是否需要加密:

若要加密,则读取加密算法标志,进入步骤(3);

若不要加密,则直接进入步骤(7);

(3) 将所有的发送缓冲合并成一个大的缓冲区;

(4) 根据加密算法确定最大加密长度值,根据此长度值对缓冲区进行分段,对于最后的分段,需要在其

尾部填充随机数据。然后调用加密函数对数据进行分段加密。

(5) 构造包头信息;

(6) 将包头信息和加密信息合并形成新的数据;

(7) 调用系统的 WSPSend 函数,将数据发送出去。

对于接收方,需要编写面向连接的 WSPRecv 函数。由于 Winsock 采用非阻塞发送方式,因此接收方接收数据的情况比较复杂,可能需要多次接收才能完成,并且多次接收的数据可能与多个发送的 socket 有关,因此,需要设置一个公用的结构体数组变量(以下称之为列表),用来记录当前活跃的 socket 数据包接收的情况。在接收过程中,一个 socket 的数据可能需要在多次调用 WSPRecv 函数之后才能获得。当处理加密的数据包时,程序必需对接收到的数据情况进行判断,只有当所有的数据到来之后,才能对其进行解密。WSPSend 的算法如下:

① 如果是 I/O 重叠操作,则设置自定义回调函数;

② 调用系统的 WSPRecv 函数,实现数据包的接收;

③ 若接收数据包时有错误发生,或发生 I/O 重叠操作,则直接返回。

若在此函数中接收到了完整的数据,则进入步骤④。

④ 将所有的接收缓冲区合并成一个缓冲区;

⑤ 判断接收的数据包是否包含包头信息:

若不包含包头信息,表示数据包没有被加密,则不需进行后续处理,直接返回即可;

若包含包头信息,说明数据包被加密,继续步骤⑥;

⑥ 判断此函数调用者的套接字(socket)是否已在列表内,若在列表中,则得到其在列表中的对应序号,将这个序号作为当前的访问序号;若不在列表中,则加入列表,得到当前访问序号;

⑦ 设置当前 socket 的数据包分包标志的初始值为 false;

⑧ 判断数据包分包标志;

若为 false,说明是发送时的第一个数据包,则此数据包中包含包头信息。从包头中取得数据包的总长度,以及加密算法标志等信息,并根据数据包的总长度为其分配缓冲区。然后判断是否分包(当前接收的数据长度是否小于数据包总长度),若分了包,则设置数据包分包标志为 true,那么下次调用时再继续接收后续的数据包内容。

若为 true,说明是后继数据包,则将后继数据包的信息与先前接收的信息进行合并。并判断是否是最后一个数据包,如果是,则设置数据包分包标志为 false(表示接收结束)。

⑨ 再次判断数据包分包标志;

若为 false,表示数据包已经没有后继数据包,即数据包在此次调用中已全部接收完成。此时可根据缓冲区中接收的数据和加密算法标志、填充字节数等信息对数据进行解密。

⑩ 将解密后的缓冲区数据分解到多个接收缓冲区中(与步骤④的动作相反)。

4 结论

本文通过 Winsock 2 SPI 封包截获技术对各种基于应用层的网络封包进行截获,然后根据加密规则的设置情况决定数据包是否加密。采用这种实现方式,可在不改变已有通信系统的情况下实现灵活的数据安全传输功能。系统采用 Visual C++ 6.0 开发工具实现,在 windows 98、windows 2000 和 windows XP 操作系统环境下均测试通过,经测试证明,系统具有配置方式灵活,加解密效率高等特点。

今后还可对系统进行进一步扩展,并结合 CA 认证体系实现会话密钥的灵活分发,从而使系统具有更好的安全性。

参考文献

1 Authony Jones、Jim Ohlund, Windows 网络编程技术[M],机械工业出版社,2001。

2 朱雁辉, Windows 防火墙与网络封包截获技术[M],电子工业出版社,2002。

3 William Stallings,密码编码学与网络安全:原理与实践(第二版)[M],电子工业出版社,2001。