OpenSSL 程序设计

西安电子科技大学ISN国家重点实验室 袁素春 王育民
北京海信数码科技有限公司 李 纲

摘 要 本文介绍了用于提供安全通道的SSL协议。基于应用及其广泛的OpenSSL软件包详细给出了SSL协议实现的程序基本框架，并讲解了在实际系统设计中常见的几个问题及其解决方法。

关键词 SSL协议 OpenSSL Socket 安全通信

随着网络及其各种信息服务的迅猛发展，也随之引发了诸多的安全威胁，并且造成了巨大的损失。因此，为通信提供更强大的安全性成为必需之举。通信安全的中心内容就是保证信息在通道内的保密性、消息完整性、认证性。SSL（Secure Socket Layer）正是在两台机器之间提供安全通道的协议。

一、SSL 协议简介

SSL 协议采用对称加密或非对称加密技术、消息摘要技术、数字证书等密码技术实现通信安全的三个方面。

（1）SSL 实现的保密性 SSL 支持多种加密套件，这些加密套件指定一组不同强度供连接使用的算法。SSL协议通过握手过程完成服务器和客户端所使用加密套件的协商，随后通信中的数据传输均采用协商好的加密算法。

（2）SSL实现的消息完整性 利用Hash函数计算出所要发送信息的摘要值，保证对方可以验证所接受的数据是否被篡改。消息摘要算法也在握手过程中商定。

（3）SSL 实现的认证性 SSL 协议默认支持对服务器的认证，使用户可以确定连接另一端的身份，实现方法是通过检查数字证书链确定该服务器是否在自己的信任列表里，从而确定该服务器是否可信任。另外，SSL协议也支持对客户端的认证。

因此，SSL协议可以提供通道级的安全，即通过认证连接两端传输经过加密的数据，并且数据是完整没有被篡改的。

SSL协议应用在传输层和应用层之间。包括握手协议和记录层协议。其层次结构图如图1所示。其中SSL记录层协议用来将数据流分割成一系列的数据段，然后分别对每个数据段单独进行数据保护和传输。接受方，对每条记录单独进行解密和验证，这样的设计使数据一经准备好就可以从连接的一端传到另一端，而接受端也可以立即对数据处理。另外SSL协议支持4种内容类型：应用数据、告警、握手和交换的密文空间，其中传输的数据类型是应用数据，而剩下的三种内容类型用来对通信进行管理。

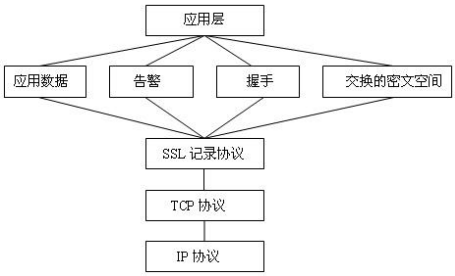

图1 SSL协议在协议栈中的位置

二、SSL 协议实现

SSL 连接类似于一个保密的 TCP 连接，并且 SSL 在语义上也模仿了 TCP 的语义，如图 2 所示：

	Socket	OpenSSL
客户端连接请求	connect()	SSL_connect()
服务器接受连接	accept()	SSL_accept()
读数据	read()	SSL_read()
写数据	write()	SSL_write()

图2 socket语义与tcp比较

1、协议实现

(1) 服务器端协议实现框架

1)生成SSL结构

```
SSLeay_add_ssl_algorithms ();
meth = SSLv23_server_method();          //依据所使用的SSL版本
ctx = SSL_CTX_new(meth);
```

2)建立Socket连接

```
listen_fd = socket ();
bind();
listen();
accept_fd = accept();
```

3)建立SSL连接

```
ssl = SSL_new(ctx);
```

```
SSL_set_fd(ssl, accept_fd);
SSL_accept(ssl); //握手过程
```

4)传送数据

```
SSL_write();
SSL_read();
```

(2) 客户端协议实现框架

1)生成SSL结构

```
SSLeay_add_ssl_algorithms();
meth = SSLv23_server_method(); //依据所使用的SSL版本
ctx = SSL_CTX_new(meth);
```

2)建立Socket连接

```
fd = socket();
connect();
```

3)建立SSL连接

```
ssl = SSL_new(ctx);
SSL_set_fd(ssl, fd);
SSL_connect();
```

4)传送数据

```
SSL_write();
SSL_read();
```

2、实际应用中常见的几个问题

(1) 多线程问题

对于一个服务器而言，一般情况下连接的客户端不止一个,这就要求服务器可以同时响应处理多个不同的客户端。解决这个问题的一个常用且有效的方法就是多线程技术。

首先,上述基本的程序框架只能实现一个服务器连接一个客户端,为了使服务器可以接受多个客户端的连接请求,采用开辟多线程的方法,为每个连接的客户端生成一个新的线程,随后针对该客户的操作都在这个线程上进行。主要针对Socket连接建立阶段，实现方法如图3所示。

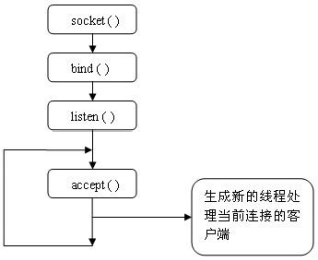

图3 改进的socket连接

其次，对服务器与某个客户端的连接，OpenSSL默认SSL_read为阻塞模式,该模式对于系统性能来说有利有弊。一方面，线程阻塞在SSL_read处，这样就减轻了系统反复检测Socket句柄是否就绪的负荷,使系统性能不致降低,并且一旦有数据到达线程可以立即做出响应。另一方面,阻塞模式使线程阻塞在SSL_read处，这样会影响线程随时地处理其他的命令。

实际中有多种方法可以将Socket设定为非阻塞模式，常用的有以下两种。

1) 对socket直接操作。在初始化时设定Socket属性的超时时间,如果在超时时间内没有检测到Socket正常连接就会跳过、重新连接而不再阻塞。设定Socket属性的函数为Setsockopt()，属性选项为SO_RCVTIMEO和SO_SNDTIMEO，分别代表接收和发送超时。

2) 使用Select函数。该函数可依次检测设定范围的Socket句柄是否可读或可写,一旦发现有句柄就绪立即执行相应的读或写操作。这种方式在Socket连接较多的情况下可以获得不错的系统性能。

(2) 同一个连接的同时写操作问题

在利用多线程技术实现了前一个问题之后,服务器和客户端可以实现实时的通信,但是这又会引发另一个问题,即连接两端可能会同时向这一个连接发送数据,或者一端的多个线程同时向该连接中发送数据，这样的情况下就会出现竞争和混乱，从而使整个SSL连接出错毁掉。为此，我们必须保证当前只有一个线程在对连接执行写操作。对于同一个进程里的不同线程,我们可以通过线程间添加互斥锁的办法,迫使在同一时刻只有一个线程对连接写入数据。对于服务器和客户端两端之间的这种错误,一个保守的办法可能也是最有效的办法就是分别在一个Socket连接上建立多个平行的SSL连接，从而避免了两端的冲突。

(3) OpenSSL与Socket写函数的区别

经过大量的实验，在Socket通信中，不同的线程分别写入发送Buffer里的数据并不能在接收端正确的区别开来,而是两个发送的数据出现了混淆、夹杂在一起。这是因为发送Buffer是定时刷新的,如果在这段时间内同时写入了不同线程的数据，这样一次发送的东西就不再是我们所期望的。但是OpenSSL就成功地解决了这个问题,线程的互斥即使在刷新时间内也可以正确区分不同的发送。

OpenSSL软件包给实现安全通信提供了非常方便而且强大的接口函数,目前已经广泛应用于各种网络通信中。在实际应用中除了本文提到的问题,OpenSSL地应用还存在很多针对不同需求而有待解决的问题。

参考文献

[1] [美]ERIC Rescorla著. 崔凯译. SSL与TSL,中国电力出版社，2002.

[2] 张斌,高波等著. Linux 网络编程,清华大学出版社,2000.

[3] 王育民,刘建伟. 通信网的安全——理论与技术,西安电子科技大学出版社，1999.