第16卷 第5期
2006年5月

计算机技术与发展
COMPUTER TECHNOLOGY AND DEVELOPMENT

Vol.16 No.5
May 2006

基于SSL/TLS的安全文件传输系统

孟 彦,侯整风
(合肥工业大学 计算机与信息学院,安徽 合肥 230009)

摘 要:为了保证Internet上数据的安全传输,加密传输得到越来越多的应用。主要阐述了如何通过JSSE实现基于SSL/TLS协议的安全文件传输系统,给出了使用JSSE创建安全套接字连接的具体方法。系统具有跨平台、传输性能高效等特点。在此基础上分析了基于SSL/TLS协议的文件传输系统的安全性。由于数据加密往往是很耗时的,会在一定程度上影响传输效率,所以还通过将本系统与一般的文件传输系统相比较,分析了基于SSL/TLS安全文件传输系统的传输效率。

关键词: SSL; 安全文件传输; JSSE
中图分类号:TP309 **文献标识码**:A **文章编号**:1673-629X(2006)05-0118-03

Secure File Transport System Based on SSL/TLS Protocol

MENG Yan, HOU Zheng-feng
(School of Computer and Information, Hefei University of Technology, Hefei 230009, China)

Abstract: Nowadays encryption has become a key to protect the data which are transferred on Internet. This thesis mainly discussed how to achieve the secure file transport system by JSSE, and the method which can make the secure socket connection by JSSE. The system is efficient and can run on several platforms with no rebuilding. On other hand this thesis analyzed the security about the file transport system based on SSL/TLS protocol and the transport efficiency by comparing with common file transport system.

Key words: SSL; secure file transport; JSSE

1 系统概述

随着网络与电子商务、电子政务的广泛普及和发展,网上数据传输的安全性一直是一个极其重要的问题。明文传输重要的文件、合同和一些私密性很强的文件是很危险的,很容易被第三方截获并篡改。为了保护网上传输的文件,出现了很多加密的文件传输协议和方法,如安全文件传输协议(SFTP),以及各种其他建立在底层协议上的加密传输方法。

文中所设计的安全文件传输器是建立在SSL协议基础上的文件传输器,充分利用了SSL协议优秀的安全架构,在SSL协议的基础上实现了安全的高效的点到点的文件传输。相对于传统的其他协议或其他方法实现的安全文件传输,使用SSL协议实现安全文件传输由于SSL协议本身的结构严谨,不需要自己设计加密层的传输协议,从而使编程开发变得更加简洁高效[1]。在SSL协议中,应用对称加密体制加密被传输的数据,应用非对称加密体制验证实体身份和交换密钥。这样达到了安全和加密速度的均衡,使加密更安全,使传输更快捷[2]。

随着Linux,Unix等其他操作系统在企业和政府中的广泛应用,在现有的文件传输过程中,在不同的平台之间传输文件往往是不方便的。然而文中设计的安全文件传输器由Java语言实现,采用JSSE安全套接字扩展实现的SSL协议,Java Swing包开发界面,使得系统可以在多平台上实现文件传输,并且拥有统一的用户界面,大大提高了易用性。此外由于Java语言的优秀特性,文件传输器可以做为一个子系统无缝集成到其他大型应用的内部。

系统的开发环境为:

开发操作系统:Windows XP,Linux(Fedora Core 3);

Java JDK版本:Java JDK1.5;

开发平台:Eclipse 3.1。

2 系统的实现

文中设计的安全文件传输系统是一个基于C/S模式的点到点的文件传输系统。系统完全由Java语言编写,基于Java 2 version 1.5类库,界面由Java Swing包实现。系统基于C/S模式,每个单独的传输器均包含服务器端和客户端。为了使客户机与服务器通信更加安全,连接使用SSL协议中的验证模式一,即对客户机与服务器都验证。

由于建立安全套接字连接需要发送证书验证身份,所以在建立连接前需要分别创建服务器端和客户端的证书。JSSE中本地的证书、私钥等信息都是存放在keystore中,然后供程序运行时读取[3]。

收稿日期:2005-09-11

作者简介:孟 彦(1981-),男,安徽合肥人,硕士研究生,研究方向为网络与信息安全;侯整风,教授,硕士生导师,研究方向为计算机网络与信息安全。

2.1 服务器端与客户端的安全套接字创建

服务器端的关键部分是建立SSL安全套接字。在客户端和服务器端建立连接握手时需要读取本地的证书并发送,然后再读取本地可信证书库来验证对方的身份。SSL安全套接字的建立代码如下:

```
//建立本地证书和可信证书的 keystore
KeyStore ks = KeyStore.getInstance(KeyStore.getDefaultType());
KeyStore tks = KeyStore.getInstance(KeyStore.getDefaultType());
//设置 keystore 的密码
char[] password = "123456".toCharArray();
//建立读取证书的文件流并读取证书
FileInputStream fis = new java.io.FileInputStream("./usekey");
FileInputStream tfis = new java.io.FileInputStream("./usetrust");
tks.load(tfis,password);
ks.load(fis, password);
fis.close();
tfis.close();
kmf = KeyManagerFactory.getInstance("SunX509");
tmf = TrustManagerFactory.getInstance("SunX509");
kmf.init(ks,password);
tmf.init(tks);
//建立 SSL 连接环境 SSLContext 类
sc = SSLContext.getInstance("TLS");
sc.init(kmf.getKeyManagers(), tmf.getTrustManagers(), null);
//由 SSLContext 类创建 SSLServerSocket
sServersf = (SSLServerSocketFactory) sc.getServerSocketFactory();
sServers = (SSLServerSocket) sServersf.createServerSocket(port);
sServers.setNeedClientAuth(true);
```

客户端的安全套接字创建与服务器端的基本相同,只是最后由SSLContext类所创建的是SSLSocket类。

2.2 keystore和证书的生成

在建立安全套接字之前,首先需要生成连接所需的本地keystore、证书、证书库和可信证书库[3]。Java语言提供了可以生成keystore和导出导入证书的工具keytool.exe。利用keytool工具生成所需的证书。

首先生成客户端和服务器端的自签发证书,在命令行方式下运行以下命令生成一个标识为sslfiledog、用RSA算法加密、存放在程序根目录下文件名为usekey的keystore文件:

```
keytool -genkey -alias sslfiledog -keyalg RSA -keystore ./usekey
```

然后按照提示输入相关的证书信息即可创建一个存放证书的keystore。接着由生成的keystore导出证书到sslfiledog.cer文件:

```
keytool -export -alias sslfiledog -keystore usekey -file sslfiledog.cer
```

证书导出后就可以将导出的证书加入生成的可信证书库:

```
keytool -import -alias sslfiledog -file sslfiledog.cer -keystore usetrust
```

这样就由证书sslfiledog.cer生成的可信证书keystore,并将证书sslfiledog.cer添加到了可信证书的keystore:usetrust。

完成了以上步骤以后就等于配置好了程序所需的证书和证书库。这样SSL协议握手时所需的所有信息都提供完全了,握手也就可以顺利完成了。

3 安全性分析

由于文中所述的安全文件传输系统建立在SSL协议的基础上,所以其安全性很大部分依赖于SSL协议的安全性。SSL/TLS协议是一种设计精良、结构严谨的安全协议,协议的自身结构设计就具有很好的安全性。下面列出系统所具有的安全性[4]:

(1)系统通过SSL协议对主体证书的检查实现了对主体的验证。验证方得到被验证方的证书后,对证书进行各种检查。CA证书的理论基础保证了通过证书的方式验证主体的有效性。

(2)迄今为止只发现了对SSL握手协议验证模式的前两种模式的有效中间人攻击,而本系统采用了SSL握手协议中三种验证模式中的第三种,即客户端和服务器端都被验证。采用这种验证方式,完全杜绝了中间人攻击。

(3)系统传输文件时,SSL协议通过对称加密算法保证数据的保密性,可以防止被动窃听和主动攻击。

a.防止被动窃听:使用对称加密算法和临时生成的短期会话密钥对所有的应用层数据加密。在美国密码产品出口法律允许的情况下,允许使用强加密算法。

b.防止主动攻击:不同的连接采用不同的密钥;即使是同一连接,两个方向上的密钥也各不相同。因而不可能通过组合不同连接的数据包,或同一连接的不同方向上的数据包实现主动攻击,并且记录层对所有的数据包都采用HMAC强验证,从而阻止了对数据的修改。

(4)系统传输文件时,SSL协议通过加密的MAC保证数据的完整性。

a.防止对MAC的穷尽攻击:无论SSL协议是否是出口版本,均采用真实强度为28位的MAC秘密,且同一连接的两个方向采用相互独立的MAC秘密。从而,有效地防止了针对MAC的穷尽攻击。

b.防止对MAC算法的密码分析攻击:SSL协议记录层采用HMAC方法,计算消息的MAC。HMAC是基于密钥的散列算法,能够防止密码分析攻击,具有理论上可证明的强安全性。

c. 防止重访攻击:SSL 协议记录层为每一个上层消息做 MAC 验证时,加入了消息序列号,这不仅可以防止对记录层消息的重放攻击,还可以使消息免遭延迟、重排和删除。消息序列号为 64 位,不会溢出。每个连接上的两个方向上的序列号被独立保存。

4 系统演示及效率分析

4.1 实际运行演示

系统的运行界面如图 1 所示。

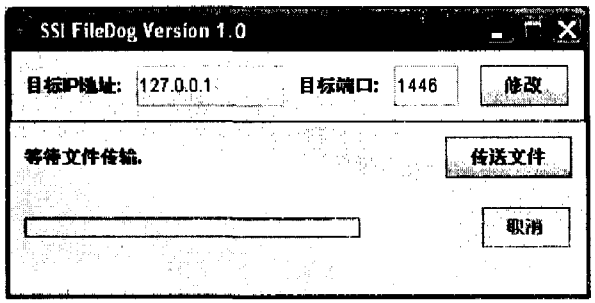

图 1 系统界面

传输文件前设置好目标地址和端口,然后点击传送文件即可选择要传输的文件。当有文件需要接受时,系统会自动弹出信息窗口,显示文件的源地址和文件名,并询问是否接收。用户的使用过程中数据的加密解密等对用户来说是透明的,用户无需关心数据如何加密、何时解密等底层问题。

4.2 加密传输效率分析

采用了对比的方法分析 SSL 安全套接字扩展的实际传输效率。通过与未加密的文件传输器对比传输速度来分析实际应用中的加密效率。为了保证对比的结果具有可比性,普通的文件传输器除了使用普通的 socket 传输外,其他代码和加密的版本一样。

测试中使用的测试文件大小从 6kB~72MB 不等递增,每个文件均传输五次,取五次传输速度的平均值(测量单位:ms),结果如表 1。

通过对比表 1 发现:

a. 加密传输所需的加密时间随着传输文件大小的增加而增加;

b. 随着文件大小的增加,加密传输时间增加的速度和无加密传输时间增加的速度基本一致。

通过以上分析可以知道使用 SSL 加密传输在效率上是可行的。实际应用中在传输大小小于 10MB 的文件时,加密传输所需加密时间是很小的,对传输没有很大的影响。而文中设计的文件传输器正是为了传输加密的公文、电子合同等商务政务方面的小文件。所以传输效率方面是完全可以满足应用需求的。

表 1 效率比较

文件大小	无加密	SSL 加密
6kB	2ms	3.2ms
117kB	15.6ms	24.8ms
777kB	156.4ms	200ms
1.6MB	249.8ms	368.6ms
6.74MB	756.2ms	1406.2ms
72MB	8668.6ms	11299.8ms

5 结束语

主要介绍了基于 SSL 协议的安全文件传输器的实现原理,以及对于 Java JSSE 的应用。基于 SSL 协议的文件传输器直接应用了 SSL 协议这种结构严谨、安全的协议,大大简化了安全文件传输的底层设计难度。此外体现出了 Java JSSE 包对于 SSL 协议很好的封装性,对于底层复杂的加密签名等算法进行了很好的包装隐藏,使得开发者能够轻松应用其 API 实现基于 SSL 协议的数据加密传输。在提供了很好的易用性的同时,JSSE 还提供了强大的可扩展性,使得人们可以轻松地定制各种加密签名的细节。但是由于美国对出口密码产品的管制和编程中某些 SSL/TLS 协议不当的用法,系统还是存在面临攻击的危险[3],如穷尽搜索 40 位 RC4 密钥攻击、利用 RSA PKCS#1 编码方法的脆弱性获得 pre-master-secret 的攻击等危险、拒绝服务攻击等[5]。此外可以看出 SSL/TLS 协议的传输效率是完全可以满足 10MB 以下小文件传输的。

参考文献:

[1] Apostolopoulos G, Peris V, Saha D. Transport Layer Security How much does it really cost[A]. INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings[C]. [s. l.]: IEEE, 1999.717-725.

[2] Rescorla E. SSL 与 TTL[M]. 北京:中国电力出版社,2002.

[3] Gong L, Ellison G, Dageforde M. 深入 Java 2 平台安全——体系架构、API 设计和实现(第 2 版)[M]. 北京:电子工业出版社,2004.

[4] 卿斯汉. 安全协议[M]. 北京:清华大学出版社,2005.

[5] 王克苑,张维勇,王建新. SSL 安全性分析研究[J]. 合肥工业大学学报(自然科学版),2004,27(1):87-91.

(上接第 117 页)

[5] Hamzaoui R, Ganz B, Saupe D. Quadtree based variable rate oriented mean shapegain vector quantization[A]. Storer J A, Cohn M. in: Proceedings DCC'97 Data Compression Conference[C]. [s. l.]: IEEE Comp Soc Press, 1997.327-336.

[6] Lin T W. Compressed quadtree representations for storing similar images[J]. Image Vision Computing, 1997, 15(11): 833-843.

[7] 魏为民. 基于彩色静止数字图像的信息隐藏技术研究[J]. 计算机应用与软件,2002(11):52-54.