OpenSSL 在网络安全传输中的应用研究

霍 英[1,2]
(1.韶关学院 信息工程学院,广东 韶关 512005;2.中南大学 信息科学与工程学院,湖南 长沙 410083)

摘 要:文章针对每一种攻击的安全服务概念,介绍了相应的 OpenSSL 实现的方法,并将其在基于 Internet 的多层次分布式网络考试系统中进行了具体的应用,重点针对 Web 服务器之间传输数据的加密和解密进行了设计,并给出了相应的实现算法。

关键词:OpenSSL;网络安全;传输机制;加密
中图分类号:TP393.08 文献标识码:A 文章编号:1006-8937(2006)08-0009-03

The application research of OpenSSL in network security transmission

HUO Ying[1,2]
(1.College of Information Engineering, Shaoguan University, Shaoguan, Guangdong 512005, China;
2. College of Information Science and Engineering, Central South University, Changsha, Hunan 410083, China)

Abstract:The implements of OpenSSL being dead against each conception of secure service were introduced and the application of OpenSSL in a multi-layer distributed exam system based on Internet was presented, the design of encrypting and decrypting scheme of data being transferred between web servers was emphasized, at last the relevant realization arithmetic was presented.
Keywords: OpenSSL; network security; transfers mechanism; encrypting

网络传输中安全的实现通常是对传输数据进行加密。目前密码编码算法分 3 大类:对称密码算法、公开密钥算法、消息摘要算法,要真正去实现这些算法以及相应的一些协议,工作量是非常巨大的,因此作为工程实现,通常采用现存的一些密码算法软件包来实现二次开发。其中 OpenSSL 是由 C 语言开发的软件包,开放源码,支持到 SSL3.0 和 TLS,并且支持 Linux、Windows、BSD、Mac、VMS 等平台,这使得 OpenSSL 具有广泛的适用性。本文将以 OpenSSL 在一个具体系统中的应用为例说明其在网络安全传输中的应用。

1 OpenSSL 软件包简介

OpenSSL 整个软件包大致可以分成 3 个主要的功能部分:密码算法库、SSL 协议库以及应用程序。OpenSSL 的根目录下有不少文件,这些文件包含 OpenSSL 各个平台下编译安装的说明文档、编译安装的配置文件以及 OpenSSL 本身版本变化的一些说明文档。表 1 简要列出了每个目录与其对应的功能说明。

表 1 OpenSSL 部分目录的功能

目 录	功 能
Crypto	存放 OpenSSL 所有加密算法源码文件和相关标注,是 OpenSSL 中最重要的目录,包含了 OpenSSL 密码算法库的所有内容。
SSL	存放 OpenSSL 中 SSL 协议各个版本和 TLS 1.0 协议源码文件,包含了 OpenSSL 协议库的所有内容。
Apps	存放 OpenSSL 中所有应用程序源码文件,如 CA、X509 等应用程序的源文件就存放在这里。
Doc	存放了 OpenSSL 中所有的使用说明文档,包含 3 个部分:应用程序说明文档、加密算法库 API 说明文档以及 SSL 协议 API 说明文档。
Demos	存放了一些基于 OpenSSL 的应用程序例子。
Include	存放了使用 OpenSSL 的库时需要的头文件。
Test	存放了 OpenSSL 自身功能测试程序的源码文件。

2 安全服务的概念及相应的 OpenSSL 实现

2.1 机密性及 OpenSSL 实现

机密性是对截获攻击的防范措施,通常通过对

收稿日期:2006-04-27
作者简介:霍英(1975—),女,陕西绥德人,博士生,讲师,主要研究方向:网络计算与分布式处理。
基金资助:本课题得到国家教育部博士点基金(20040533036)资助

通信数据进行加密与解密来实现。如果数据量大，一般采用对称加密算法。OpenSSL 使用 EVP 封装了所有的对称加密算法，使得各种对称加密算法能够使用统一的 API 接口 EVP_Encrypt 和 EVP_Decrypt 进行数据的加密和解密，大大提高了代码的可重用性。OpenSSL 中与对称密钥加密相关的函数可在 evp.h 文件中找到，用到的主要函数有：

EVP_CipherInit_ex (&ctx, cipher, NULL, key, iv, type)); //初始化；type 为 1，则加密；如果 type 为 0，则解密；如果 type 是 - 1，则不改变数据；cipher 是加密算法指针；NULL 表示使用默认的算法

EVP_CipherUpdate (&ctx, outbuf, &outlen, inbuf, inlen)); //加密的中间过程

EVP_CipherFinal_ex (&ctx, outbuf, &outlen); //加密的最终过程处理。

2.2 完整性及 OpenSSL 实现

完整性是对篡改攻击的防范措施，通常采用对消息摘要码的计算与对比来确认数据是否被篡改。OpenSSL 实现了 5 种消息摘要算法，分别是 MD2、MD5、MDC2、SHA(SHA1) 和 RIPEMD。OpenSSL 采用 EVP_Digest 接口作为消息摘要算法统一的 EVP 接口，对所有消息摘要算法进行了封装，提供了代码的重用性。OpenSSL 中与消息摘要相关的函数可在 evp.h 文件中找到，用到的主要函数有：

EVP_get_digestbyname (mdname); //获取算法名称，根据输入的消息摘要函数的名字得到相应的 EVP_MD 算法结构

EVP_DigestInit_ex(&md_ctx, md, impl); //消息摘要初始化，使用 md 的算法结构设置 md_ctx 结构，如果 impl 为 NULL，即使用缺省实现的算法(OpenSSL 本身提供的消息摘要算法)

EVP_DigestUpdate(&md_ctx, inbuf, len); //消息摘要算法的中间处理过程

EVP_DigestFinal_ex (&md_ctx, md_value,md_len); //消息摘要的最后处理过程，将完成的摘要信息存储在 md_value 里面，长度信息存储在 md_len 里面。

2.3 不可抵赖性及 OpenSSL 实现

不可抵赖性主要针对发送方对接收方接收到的信息的不可抵赖，通常采用的方法是对传输数据进行数字签名。OpenSSL 也使用 EVP 技术对不同功能的非对称加密算法进行封装，提供了统一的 API 接口。如果使用非对称加密算法进行密钥交换或者密钥加密，则使用 EVP_Seal 和 EVP_Open 进行加密和解密；如果使用非对称加密算法进行数字签名，则使用 EVP_Sign 和 EVP_Verify 进行签名和验证。

OpenSSL 中与证书和私钥读取相关的函数可在 x509.h 文件中找到，用到的主要函数有：

d2i_X509_bio (cert,NULL); //从 DER 格式的证书文件中读出 X509 格式的证书

d2i_PrivateKey_bio (bio, NULL); //从 DER 格式的私钥文件里读出 EVP_PKEY 结构的私钥

EVP_get_cipherbyname(cpname); //根据算法名称获得加密算法指针

X509_get_pubkey (x509); //从 X509 格式的证书中获取 EVP_PKEY 结构的公钥

EVP_PKEY_get1_RSA(pcert); //从 EVP_PKEY 格式的公钥内读出 RSA 结构的公钥信息

OpenSSL 中与签名、验证签名及数字信封相关的函数可在 evp.h 文件中找到，用到的主要函数有：

EVP_SignInit_ex (&md_ctx,md,NULL); //签名初始化

EVP_SignUpdate (&md_ctx, inbuf, len); //签名中间过程

EVP_SignFinal (&md_ctx, sig_buf, &sig_len, pkey); //签名最终过程

EVP_VerifyInit_ex (&md_ctx,md,NULL); //验证签名初始化

EVP_VerifyUpdate (&md_ctx, inbuf, len); //验证签名中间过程

EVP_VerifyFinal (&md_ctx, sig_buf, sig_len, pcert); //验证签名最终过程

EVP_SealUpdate (&ectx, ebuf, &ebuflen, buf, readlen); //数字信封的中间处理过程

EVP_SealFinal(&ectx, ebuf, &ebuflen); //数字信封的最后处理过程

EVP_OpenInit(&ectx,cipher, encryptKey,ekeylen, iv,pkey); //打开数字信封的初始化过程

EVP_OpenUpdate (&ectx, ebuf, &ebuflen, buf, readlen); //打开数字信封的中间处理过程

EVP_OpenFinal(&ectx, ebuf, &ebuflen); //打开数字信封的最后处理过程

3 OpenSSL 在网络安全传输中的应用案例

3.1 系统应用背景介绍

本文将 OpenSSL 在基于 Internet 的多层次分布式网络考试系统中进行了具体的应用。本网络考试

系统由一个考试中心和若干个考务中心和考点组成。考试中心负责考试基本信息管理、考题发布、考试系统参数设置、试题回收及相关管理等；考务中心负责考试数据的生成、审核、上报及相关管理等；考点负责组织实施具体的考试及相关管理。

系统中网络安全传输包括两部分，一是各类用户的浏览器与其对应的 Web 服务器之间的安全传输(B- Web 传输)，主要是确保网页传输、消息传输、消息响应、表单提交、控件下载等会话事务的安全；另一类是各 Web 服务器之间的安全传输 (Web-Web 传输)，是指在上下级 Web 服务器之间提供安全的文件下载、数据回收、参数设置等批量数据传输。

3.2　加密解密设计

加密算法主要针对传输中的机密性、完整性、不可抵赖性而设计，加密设计如图 1 所示。

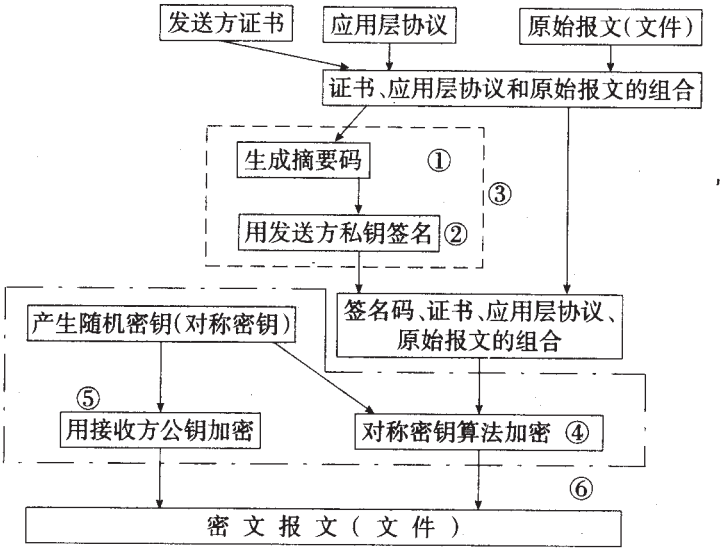

图 1 加密流程图

图 1 中：

①生成摘要码是为了保证数据的完整性，用 EVP_Digest 系列函数实现。

②用发送方私钥签名是为了实现发送方的不可抵赖，用 EVP_ private_decrypt 函数实现。

③在 OpenSSL 中可以用 EVP_Sign 系列函数集成实现 (在 EVP_Sign 系列函数中包含了生成摘要码和私钥签名的功能实现，为我们的设计提供了方便) 。很多时候我们是把这个合成步骤统称为数字签名。

④对称密钥算法加密是为了保证数据的机密性，用 EVP_Encrypt 接口实现。

⑤用接收方公钥加密。在标准的 SSL 实现中是为了交换对话密钥，实现一次会话过程的多次报文的往返传输安全，因此它在实现上是把交换对话密钥与报文加密分成两个步骤的，但是在 Web 服务器之间的传输类型不是以会话为主，而是以批量数据的传输为主，因此我们在实现上把它与传输报文绑定，这样可以简化传输过程。用 RSA_public_encrypt 函数实现。

⑥在 OpenSSL 中可以用 EVP_Seal 系列函数集成实现 (在 EVP_Seal 系列函数中包含了公钥加密和对称加密两个步骤的功能实现，为我们的设计提供了方便) 。

整个加密过程用算法表示如下：

```
bool Sf_To_Df(void &DestPackage,char *ScertFile,char *SDatabase,char *STable,char *DHost,char *DDatabase,char *DTable,char *OperateType)
{生成应用层协议;
组合证书、应用层协议和源报文=》NewPackage1;
EVP_Sign(NewPackage1,Private_Key)=》NewPackage2;
EVP_Seal(NewPackage2,Public_Cert)=》DestPackage;
}
```

4　结 语

网络安全传输是一个不断发展的研究课题，加密解密算法和理论也在不断发展，OpenSSL 作为一个内容丰富的软件包，其在网络工程中的应用和需求也在不断变化之中，这些都是我们未来需要不断探索的内容。

参考文献：

[1] 邵兵，鲁东明.一个数据加密传输系统的研究与实现[J].计算机应用，2000，20(6) .

[2] 吴凯，陈晓苏，肖道举.网络传输层安全协议 SSL 的安全研究[J].计算机系统应用，2003，(1) :48- 50.

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