文章编号:1005-0523(2004)04-0142-05

基于JCE的网络数据库敏感信息加密技术与实现

左黎明[1],盛梅波[1],马 娟[2],刘二根[1]

(华东交通大学 基础科学学院,江西 南昌 330013;2.武汉科技大学 管理学院,湖北 武汉 430081)

摘要:通过分析基于JAVA技术的网络数据库敏感信息的特点与性质,提出了一套实用的基于JCE的数据库信息加密方案,同时介绍了JCE这种先进技术.

关 键 词:信息;JAVA;AES;RSA;JCE;网络数据库;

中图分类号:TP393.08 **文献标识码**:A

0 引 言

随着电子商务的兴起,数据的安全问题被提到前所未有的高度.一方面,企业需要对自己的关键数据进行有效的保护,另一方面,越来越多的企业由于业务的需要不得不把数据库平台从单机转向网络,在这种情况下,企业的数据存放在网络数据库服务器上,其安全性无法得到有效的保障.在这些数据中,有两类数据是最为敏感的:一类是数据库系统的用户信息,另一类是商业机密(包括业务信息、交易数据和商业情报等).这两类信息常常是黑客和竞争对手的最终想获得的东西.

2 数据库敏感信息安全分析与加密设计

2.1 数据库敏感信息安全分析

黑客攻击主要技术方法有以下几种:

1) 缓冲区溢出技术;

2) 木马技术;

3) 计算机病毒,主要是宏病毒和网络蠕虫;

4) DDOS攻击技术;

5) 暴力破解;

6) sniffer报文截获.

在大量的黑客技术文献和攻击日志中我们发现几乎没有多少攻击行为是针对于协议和密码学算法的,大部分攻击的主要方向是针对用户名和用户密码的,其中以木马对用户的数据安全威胁最大,尤其是基于三线程反弹端口式木马.

2.2 JCE简介和JDK 1.4中支持的密码算法

Java加密扩展即Java Cryptography Extension,简称JCE.它是Sun的加密服务软件,包含了加密和密匙生成功能.JCE是JCA(Java Cryptography Architecture)的一种扩展.

JCE没有规定具体的加密算法,但提供了一个框架,加密算法的具体实现可以作为服务提供者加入.除了JCE框架之外,JCE软件包还包含了SunJCE服务提供者,其中包括许多有用的加密算法,比如DES(Data Encryption Standard),在J2SDK1.4.2中还包含最新的高级加密标准(AES)算法.

2.3 加密方案设计

在数据库加密系统体系结构中,可以在应用程序与数据库服务器之间加入一套数据库加/脱密引擎,加解密引擎可捆绑在客户端,负责在客户端后台完成数据库信息的加/脱密处理,对应用开发人

收稿日期:2004-04-22

作者简介:左黎明(1981-),男,江西余江人,华东交通大学基础学院助教.

员和操作人员是透明的. 加密用的密钥可采用安全密钥托管方式或者用户自己采用安全的介质(如 U 盘)保存

通过分析，我们提出如图 3 的三层网络数据库加密系统体系结构，两层网络数据库加密系统体系结构与类似.

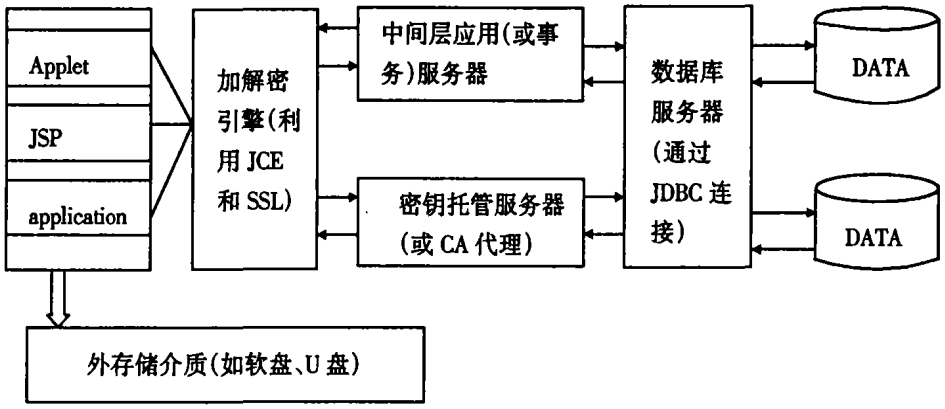

图 1　三层网络数据库加密体系结构

3　关键技术与实现

3.1　要点概述与性能分析

加密具体方案根据需数据库加密的具体要求可分为以下三个级别：

1) 字段数据级：针对某个敏感字段数据进行加密；

2) 记录级：针对数据表中的特定记录进行加密；

3) 数据表级：针对敏感的数据表进行加密；

4) 数据库级：对于整个数据库进行加密.

为了比较性能和选择具体加密方案，我们对于不同的加密级别在单机平台做了相应的实验，比较加密与普通不加密情况下的性能差别.

数据库应用系统性能，如下表 1：

表 1　加密数据应用系统性能

	AES		RSA	
	50000 条记录	100000 条记录	50000 条记录	100000 条记录
字段数据级	各项操作速度与普通无明显差别	查询差别明显，删除、添加等操作缓慢	各项操作时间严重延迟	失败，各项时间均过长
记录级	各项操作速度缓慢	各项操作速度缓慢	各项操作时间严重延迟，数据库出错	失败，各项时间均过长
数据表级	失败，各项时间均过长	失败，各项时间均过长	失败，各项时间均过长	失败，各项时间均过长
数据库级	失败，各项时间均过长	失败，各项时间均过长	失败，各项时间均过长	失败，各项时间均过长

从实验结果来看，采用对称加密系统 AES 比采用公钥系统 RSA 效果要好，字段数据级效果比记录级、数据表级和数据库级要好，对于大型数据集加解密，时间代价太大，目前还没有找到合适的解决方法. 因此，我们将加解密处理定位在字段数据级上，加解密算法采用 AES. 由于加密方案是建立在字段级上的，因此程序设计时不会影响到正常的 SQL 操作，也不会影响到中间层. 要开发一套三层网络数据库加密系统，中间层和数据库服务器只要注意字段数据类型必须能够兼容加密后的密文即可，相关的程序设计与普通的三层结构没有差别，关键的问题在于客户端如何进行加解密以及密码算法的选择.

3.2　数据库中敏感信息字段的设计

由于我们要对某些敏感信息进行加密，加密后得到的密文可能是一些不可识别的符号，因此在数据库表的设计过程中，对应于这些敏感信息字段的字段数据类型必须能够兼容加密后的密文，一般我们都将其设置为 Byte(或 String)类型.

3.3　客户端信息加解密与数据发送接收

表 3 数据库加解密过程原理

信息加密与发送数据的原理过程	数据接收与信息解密的原理过程
(1) 用户输入敏感信息;	(1)创建 Socket 对象建立与中间层服务器的连接;
(2) 创建密码算法实例,获得加密密钥,并保存密钥;	(2)用该 Socket 对象创建输入、输出流;
(3) 按字段级进行加密;	(3)读取输入流中的信息,分离加密信息,读取相关密钥;
(4)创建 Socket 对象建立与中间层服务器的连接;	(4)按字段级进行解密;
(5)用该 Socket 对象创建输入、输出流;	(5)解密后明文输出到用户界面;
(6)将密文和其它信息写入输出流;	(6) 交互完毕,关闭输入、输出流与 Socket.
(7)与中间层服务器交互,发送数据;	
(8) 交互完毕,关闭输入、输出流与 Socket.	

加解密过程的相关代码和说明如下:

1) 引入加密算法框架 JCE 和算法类

```
import java.security.*;
import javax.crypto.*;
import javax.crypto.spec.*;
import java.io.*;
```

2) 为保存显示方便,可以将 2 进制转化为 16 进制或者字符串:

```
public static String asHex (byte buf[]) //2 进制转化为 16 进制
{StringBuffer strbuf = new StringBuffer(buf.length * 2);
int i;
for (i = 0; i < buf.length; i++) {
if (((int) buf[i] & 0xff) < 0x10)
strbuf.append("0");
strbuf.append(Long.toString((int) buf[i]] & 0xff, 16));
}
return strbuf.toString();
}
public String byte2hex(byte[] b) //二行制转字符串
{
String hs="";
String stmp="";
for (int n=0;n<b.length;n++)
{stmp=(java.lang.Integer.toHexString(b[n] & 0XFF));
if (stmp.length()==1) hs=hs+"0"+stmp;
else hs=hs+stmp;
if (n<b.length-1) hs=hs+":";}
return hs.toUpperCase();
}
```

3) 在加密或解密任何数据之前需要有一个密匙,生成一个安全密匙,并完成所有初始化工作

```
KeyGenerator kgen = KeyGenerator.getInstance("AES");
// 为我们选择的 AES 算法生成一个 KeyGenerator 对象
kgen.init(128);//初始化 KeyGenerator 对象
SecretKey skey = kgen.generateKey();// 生成密匙
byte[] raw = skey.getEncoded();// 获取密匙数据
SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
// 从原始密匙数据创建 SecretKeySpec 对象
Cipher cipher = Cipher.getInstance("AES");// Cipher 对象实际完成加密操作
```

4) 加密过程

```
byte data[] //获取得到字段数据
cipher.init(Cipher.ENCRYPT-MODE, skeySpec); // 用密匙初始化 Cipher 对象
byte encryptedData[] = cipher.doFinal(data);//执行加密
String encryptedString = new String(asHex(encryptedData));
doSomething(encryptedData); // 进一步对解密后的数据进行处理
```

5) 保存密钥,可以放入优盘或者存入密钥管理器

```
FileOutputStream fos = new FileOutputStream(ENCRYPT-KEY-FILE);
fos.write(raw);
fos.close();
doSomething(ENCRYPT-KEY-FILE);//进一步
```

处理密钥文件

(6)解密过程

```
byte raw [] / * 获取的原始密匙数据 * /;
SecretKeySpec skeySpec = new SecretKeySpec (raw, "AES");
// 从原始密匙数据创建 SecretKeySpec 对象
Cipher cipher = Cipher. getInstance ("AES");// Cipher 对象实际完成解密操作
byte encrypteddata[] //获取加密后的字段数据
cipher. init (Cipher. DECRYPT-MODE, skeySpec);
// 用密匙初始化 Cipher 对象
byte Data[] = cipher. doFinal (encrypteddata);// 执行解密
String dataString = new String(asHex (Data));
doSomething(Data ); // 进一步对解密后的数据进行处理
```

3.4 用户验证方法设计

为了对付木马和 sniffer 报文截获,有必要改变传统的用户名+密码的验证方式.其主要原理是本地用户合法性验证与远程数据库用户验证相结合的方法,本地验证是采用了经过数字水印技术处理过的用户身份 ID 图片(其技术原理比较复杂,可参考文献[7]),通过用户名和密码计算数字水印信息的序列号,另外通过数字水印提取算法从图片提取数字水印,也计算出数字水印序列号,如果发现两个序列号完全相等,则允许连接中间层,并提交给服务器进行验证,验证成功则提示登陆成功,否则登陆失败.采用这种算法的好处是数字水印的提取和计算是不通过键盘输入的,而且含水印的图片 ID 可以放在 U 盘上,用户只要客户端有 USB 接口就可操作,这样即使黑客利用木马和 sniffer 报文截获得到用户名和密码,都无法完成数据库的登陆操作,从而保证数据库系统的安全.

4　总结

在用 JAVA 开发三层网络系统应用时,可以采用的开发环境方案很多,通常用的方案是 apache+tomcat+j2sdk 或者 Jbuilder8.0+weblogic,后台数据库通常为 SQLSERVER2000、MYSQL 等,如何实现三层网络系统应用有很多成熟的代码和方法.加解密是针对字段级的,因此加解密过程相对比较简单可靠.密钥托管方案是一种不错的选择,但是配置与保护密钥服务器比较复杂,成本较高.随着 U 盘的普及,利用 U 盘代替密码卡,可以省去读卡器等相关设备,降低成本且携带方便,可以充分保证密钥安全.

参考文献:

[1](美)Bruce Eckel 著,候捷译.Java 编程思想(第 2 版)[M].北京:机械工业出版社 2002

[2]http://java.sun.com/j2ee/

[3](美)John Bell Tony Loton. Java Servlets 2.3 编程指南[M].北京: 电子工业出版社,2002.

[4](美)Jess Garms 著,庞南,等,译. Java 安全性编程指南[M].北京:电子工业出版社,2002.

[5](美)Joseph J.Bambara 等,著,刘　等,译. J2EE 技术内幕[M].北京:机械工业出版社 2002.

[6](美)Li Gong 著.JAVA 2 平台安全技术——结构、API 设计和实现[M].北京: 机械工业出版社 2000

[7]左黎明,刘二根,曾伟.一种基于方形码和 DCT 的数字水印技术与实现.[J]:华东交通大学学报 2004,(1):129～132

Sensitive Information Encryption of Network Database Base on JCE

ZUO li-ming, SHENG mei-bo, MA Juan, LIU er-geng

(1 School of Natural Science, East China Jiaotong Uni., Nanchang 330013; 2 School of Management, Wuhan University of Science and Technology, Wuhan 430081)

Abstract: Through analysing the characteristic and nature of sensitive information of network database based on JAVA technology, proposed that a set of scheme of sensitive information encryption based on JCE, and introduced JCE such advanced technology at the same time.

Key words: information; JAVA; AES; RSA; JCE; network database