

数据库系统安全性测试技术研究

周 薇

(江苏自动化研究所,江苏 连云港 222061)

摘要:数据库系统安全性测试是软件测试中的一个重要领域。传统的数据库系统测试往往侧重于数据库功能测试,不能有效地发现深层次的安全问题。文中通过对数据库系统的安全性机理的研究,结合数据库应用软件的特点,提出了数据库管理系统安全性测试方法,具体包括数据的完整性、健壮性、防范攻击性、备份恢复、安全性访问控制。以工程项目为例详细阐述了数据库系统安全性测试方法的应用。通过使用有效的测试方法,保证了对数据库系统安全性的合理验证,从而提高了数据库系统的软件质量。

关键词:安全性测试;数据库系统;软件测试

中图分类号:TP311

文献标识码:A

文章编号:1673-629X(2014)02-0140-05

doi:10.3969/j.issn.1673-629X.2014.02.034

Research on Security Testing Technique for Database System

ZHOU Wei

(Jiangsu Automation Research Institute, Lianyungang 222061, China)

Abstract: Security testing for database system is an important field in software testing. Traditional database system testing tends to focus on the database function testing, cannot find deep-seated problems effectively. In this paper, through research on database system security theory and characteristics of database application software, give out testing methods, including data integrity, robustness, attacks preventing, backup and recovery, security access control and other aspects. Take the project as an example to elaborate the application of database system security testing method. Through the use of effective testing methods, ensure reasonable verification of database system security, so improving the quality of database system software.

Key words: security testing; database system; software testing

0 引言

无论是基于B/S还是C/S结构开发的软件系统,都离不开数据库系统开发。大量的数据信息一般都采用大型数据库来管理,在这种情况下,数据库安全性测试可以帮助数据库系统发现安全漏洞,减少泄密的风险^[1]。长期以来,由于缺乏有效的测试技术及方法,数据库系统安全性测试难于开展。文中重点描述数据库系统安全性测试要测试什么以及怎样去测试的问题,提出实用可行的测试方法,使用综合有效的测试策略,保证对数据库系统安全性的合理验证。

1 数据库系统安全性测试概述

数据库系统安全性测试(Security Testing)是数据库软件测试中的一种测试类型,其含义是检验数据库

系统对非法入侵的防范能力^[2]。数据库系统安全性测试是侧重于只有预期的主角才可以访问测试对象和数据(或系统)的测试,用来验证集成在数据库系统内的保护机制是否能够在实际使用中保护数据库系统不受非法侵入。

数据库系统安全性测试过程中,包含三个方面内容,分别是资源、风险、安全性控制。

a)资源。

这里的资源指功能或数据。应列出所有应被保护的功能和数据,确定各种功能及数据对于合法用户和潜在的侵入者具有怎样不同的价值,并分析有哪些非法方法可以利用这些功能和数据。

b)风险。

风险是可能导致损失或伤害的事件。安全性测试中应列出所有可能的风险,将意外风险、自然风险与人

为风险区别开来。一方面应分析所有风险发生的概率,另一方面应了解风险一旦变成现实将导致的严重后果,并密切关注导致最严重后果的风险。

c) 安全性控制。

安全性控制是针对风险的保护措施。安全性测试中应针对所有的风险,给出保护性方案,其中重点关注人为风险和偶然的系统误操作。

对于数据库系统的安全测试策略,可以从正向和反向两个层面来考虑。正向是从系统的需求分析、概要设计、详细设计、编码这几个方面来发现可能出现安全隐患的地方,并以此作为测试空间来进行测试。而反向测试过程是从已知的缺陷空间出发,在软件中寻找可能的缺陷,建立缺陷威胁模型,通过威胁模型来寻找入侵点,对入侵点进行已知漏洞的扫描测试。对安全性要求较低的数据库系统,一般按反向测试过程来测试即可;对于安全性要求较高的数据库系统,应以正向测试过程为主,反向测试过程为辅^[3-4]。

数据库系统安全性测试的重点是验证所有安全性需求都正确实现,确定软件的安全性薄弱环节,发现在极端条件及异常状态下产生导致安全问题的软件失效,为数据库系统安全性需求的完善提供依据。数据库系统安全性测试是对数据库系统常规测试的补充。

2 数据库管理系统安全性测试方法

一般来说,数据库系统分为数据库管理系统和数据库应用系统。数据库管理系统(Database Management System)是一个专门负责数据库管理和维护的计算机软件系统,是一种操纵和管理数据库的大型软件,用于建立、使用和维护数据库。它对数据库进行统一的管理和控制,以保证数据库的安全性和完整性^[5]。数据库应用系统则是通过数据库管理系统与数据库进行数据交互的应用软件,数据库应用系统安全性测试的重点是系统软件在异常条件下在极端条件及异常状态下产生导致数据安全性的问题,可以通过功能、性能、容量、压力测试进行验证。数据库系统的安全性很大程度上依赖于数据库管理系统,如果数据库管理系统的安全机制强大,则数据库系统的安全性就比较好。因此这里重点描述数据库管理系统安全性测试方法。数据库系统的安全性测试应包括四个部分内容,分别是数据完整性测试、健壮性及防范攻击性测试、备份与恢复测试、安全性访问控制测试^[6-8]。

a) 数据库数据完整性测试。

数据完整性是指数据库中的数据应保持一致性和可靠性,进而防止数据库中不符合语义规定的数据进入数据库。所谓数据库完整性测试,即在数据库表中发现不准确数据的过程。数据库完整性测试尽可能在

数据存储的方式中发现问题。影响数据库存储数据的方式有很多因素,比如数据类型和长度可能导致数据截断或失去精确性。通过文档及代码审查,检查数据库管理系统是否提供了实体完整性定义语句、域完整性定义语句、参照完整性定义语句、用户自定义完整性定义语句。检查数据库管理系统是否提供了事务机制,在 SQL 语句批处理执行过程中发生错误的时候,通过回滚事务,使数据库恢复到事务开始之前的状态,就像什么也没有发生过一样,从而有效地保证数据的完整性,并实现数据库的并发访问。

b) 数据库健壮性及防范攻击性测试。

数据库健壮性及防范攻击性测试主要测试是否能抵抗 SQL 注入攻击^[9-12]和缓冲区溢出攻击。

对于 SQL 注入攻击测试主要查看所有涉及 SQL 语句提交的地方,是否正确处理了用户输入的字符串或者使用 SQL 注入攻击工具对数据库管理系统进行攻击尝试,查验数据库管理系统的响应情况。

对于缓冲区溢出攻击测试主要查看是否向程序的缓冲区写超出其长度的内容造成缓冲区的溢出,从而破坏程序的堆栈,使程序转而执行其他指令或者使用数据库连接(符合被测数据库通信协议)攻击工具,向数据库服务端口发送构造的各类可能导致缓冲区溢出的畸形报文,查验数据库管理系统的响应情况。测试人员需要对每一个用户可能输入的地方尝试不同长度的数据输入,以验证程序在各种情况下正确地处理了用户的输入数据,而不会导致异常或溢出问题。

c) 备份与恢复测试。

数据库系统如果发生故障可能会导致数据的丢失,要恢复丢失的数据,必须对数据库系统进行备份测试。当出现意外情况如突然停电、硬件故障、网络断线时,导致数据意外丢失或破坏了数据的完整性,系统应提供恢复数据库功能,使系统有能力将数据库恢复到损坏以前的状态^[13],必须对数据库系统进行恢复测试。

数据库系统进行备份恢复测试主要考察是否提供了数据备份方式:冷备份、热备份和逻辑备份,是完全备份还是增量备份或累积备份;数据库管理系统是否提供了恢复技术:单纯以备份为基础的恢复技术、以备份和运行日志为基础的恢复技术和基于多备份的恢复技术。测试时还需要注意数据库服务器硬盘或客户端被访问和备份时是否防止恶意拷贝,另外数据库版本更新后,原数据库中的数据是否已经备份,备份在导入导出过程中是否遗漏或不兼容。

d) 安全性访问控制。

数据库管理系统安全性访问控制测试主要包括访问控制测试、用户安全测试和数据保密性测试。

访问控制测试主要是阻止非授权用户进入数据库

管理系统,通过对授权用户存取系统敏感信息时进行安全性检查。包括尝试使用不同权限登录系统、管理员用户能否修改同级用户的权限、删除正在进行操作的权限、系统管理员修改数据库数据时是否影响普通用户登录等等。

用户安全测试主要是对数据库系统用户的身份进行认证,系统对输入的用户名与合法用户名对照,鉴别此用户是否为合法用户。若是,则可以进入下一步的核实;否则,不能使用系统。

数据保密性测试需要从数据文件加密与解密、数据信息密码显示方面进行测试,查看加密文件是否以密文、密码代码的形式显示,加密文件解密后是否以原文显示,以及数据库中的机密信息是否以密文或密码的形式显示。

3 数据库管理系统安全性测试方法应用

某军事政务管理系统主要包括人事管理、电子公文管理、日常信息管理、技术培训管理功能。系统采用 B/S 结构,应用软件采用 ASP.NET 技术开发,数据库采用 Oracle 关系型数据库。数据库系统安全性测试按照前文所述,设计四个测试项,包括数据完整性测试、健壮性及防范攻击性测试、备份与恢复测试、安全性访问控制测试。

3.1 数据完整性测试

数据完整性测试具体设计如下:

a) 检查该数据库是否提供了定义与数据完整性有关的语句。

用 CREATE RULE 语句创建规则;

用 DROP RULE 删除规则;

用 CREATE DEFAULT 语句创建默认;

用 DROP DEFAULT 删除默认;

用 UPDATE 使用默认值;

用存储过程 SP_BINDRULE 绑定规则/默认;

用存储过程 SP_UNBINDRULE 解除规则/默认与字段的绑定;

用 CREATE TABLE 语句在建立新表的同时定义约束;

用 ALTER TABLE 语句向已存在表中添加约束。

b) 检查该数据库是否提供了定义与数据完整性有关的控制策略。

用 BEGIN TRANSACTION 语句,启动事务的开始;

用 COMMIT TRANSACTION 语句,提交事务;

用 ROLLBACK TRANSACTION 语句,回滚事务;

提供排它锁,不允许别的事务读取或修改已被锁定的资源;

提供共享锁,允许别的事务读取已经被锁定的资

源,但不允许对锁定的资源进行修改操作;

用存储过程 sp_lock 检索系统中锁的有关信息;

用 SETLOCK_timeout 语句设置事务请求锁定的最长等待时间。

c) 编写事务机制测试程序验证数据完整性。

事务测试程序如下:

```
BEGIN TRANSACTION
INSERT s(sno,sgrade,sname,email) VALUES('1008','士兵','黄黄','hh@126.com')
INSERT s(sno,sgrade,sname,email) VALUES('1009','排长','周周','zz@126.com')
COMMIT TRANSACTION
BEGIN TRANSACTION
INSERT s(sno,sgrade,sname,email) VALUES('1008','士兵','王王','ww@126.com')
INSERT s(sno,sgrade,sname,email) VALUES('1009','排长','张张','zz@126.com')
ROLLBACK TRANSACTION
```

运行结束,查看数据库中的记录,前两条记录被成功插入,后两条记录没有被成功插入。测试通过,该数据库管理系统提供的事务机制可保证数据完整性。

3.2 健壮性及防范攻击性测试

进行防范攻击性测试的具体过程如下:

a) 代码审查,是否存在 SQL 注入式漏洞。

传入用户输入的管理员用户名和密码,安全验证代码如下:

```
public bool Login(string strUserID,string strPassword)
{
    string strSQLText = string.Format("select * from [AdminGroup] where [UserID]='{0}' and [Password]='{0}'",strUserID, strPassword);
    SqlCommand cmd = new SqlCommand(cmdText, conn);
    .....
}
```

正常情况下,用户传入正确的用户名和密码进行验证,如传入“AdminName”和“AdminPassword”进行验证,得到的 SQL 语句将是:

```
Select * from [AdminGroup] where {UserID} = 'AdminName' and [Password] = 'AdminPassword'
```

这个 SQL 语句很正常。但是,通过代码审查发现,如果分别传入“AdminName' or 1 = 1 --”,“AdminPassword'”,得到的 SQL 语句将是:

```
Select * from [AdminGroup] where [UserID] = 'AdminName' or 1 = 1 --' and [Password] = 'AdminPassword'
```

在用户名“AdminName' or 1 = 1 --”中,第一个单引号结束了原有字符串中第一个单引号的配对,而“or”后面的“1 = 1”会导致不管前面的验证结果如何,都

会返回真(True)值,而随后的“--”将把其后的 SQL 语句注释掉。现在问题出现了,不管使用什么用户名和密码,都能验证通过。在这个存在漏洞的用户验证页面,如果注入 JOIN 语句,就能获取数据库里的所有数据,显示在页面上,如获取用户名、密码等;而注入 UPDATE/INSERT/DELETE 语句将改变数据,如添加新的管理员账号等。这样,数据库将不再安全。

```

    以下是改进后的安全验证方法:
    public bool Login(string strUserID,string strPassword)
    {
        string strSQLText = "select * from [ AdminGroup ] where
[ UserID ]=@ UserID and [ Password ]=@ Password";
        SqlParameter sqlUID = new SqlParameter( "@ UserID",SqlDbType.
VarChar);
        sqlUID. Value= strUserID;
        SqlParameter sqlPWD = new SqlParameter( "@ Password",
SqlDbType. VarChar);
        sqlPWD. Value= strPassword;
        SqlCommand cmd=new SqlCommand( cmdText,conn);
        cmd. Parameters. Add( sqlUID);
        cmd. Parameters. Add( sqlPWD);
        .....
    }
```

```

    b)编写下列测试脚本,测试 SQL 注入式漏洞。
    require 'tests/setup'
    require 'watir'
    //定义 Web 应用程序的主页面
    $ APP_HOME = ' http://localhost:8080/department '
    //定义用户名变量
    $ USERNAME = ' user1 '
    //定义密码变量
    $ PASSWORD = ' pwd1 '
    //用于注入测试的字符串变量
    $ SQL_CONCAT_USERNAME = ' user\ ' +\' 1 '
    //将单元测试案例引入到 SQL_Injection_Test 类中
    class SQL_Injection_Test < Test::Unit::TestCase
    //包含 Watir 工具
    include Watir
    //定义 test_SQL_Blind_Injection() 方法
    def test_SQL_Blind_Injection()
    //转向主页面
    $ ie. goto( $ APP_HOME)
    //根据链接找到 loginForm. do 并执行点击动作
    $ ie. link( :url, /loginForm. do/). click
    //在文本域中 $ USERNAME 属性后面追加 ' OR 1=1 --'
字符串
    $ ie. text_field( :name, ' username ' ). set( $ USERNAME+
\' OR 1=1--' )
    //执行并提交
    $ ie. form( :action, " /department/login. do " ). submit
```

```

    assert( $ ie. contains_text( ' Login failed' ));
    end
    def
    //定义 SQL_Injection_String_Concat() 方法
    test_SQL_Injection_String_Concat()
    //转向主页面
    $ ie. goto( $ APP_HOME)
    //根据链接找到 loginForm. do 并执行点击动作
    $ ie. link( :url, /loginForm. do/). click
    //将文本域的 username 属性值设置为 $ SQL_CONCAT_
    USERNAM 的值
    $ ie. text_field( :name, ' username' ). set( $ SQL_CONCAT_
    USERNAM)
    //将文本域的 password 属性值设置 为 $ PASSWOR 的值
    $ ie. text_field( :name, ' password ' ). set( $ PASSWOR)
    $ ie. form( :action, "/department/login. do " ). submit
    assert( $ ie. contains_text( ' Login failed' ));
    end
    end
```

测试发现,修改前的程序能够登录成功,说明该程序对 SQL 注入不免疫,SQL 注入式测试是失败的;修改后的程序登录失败,说明该程序对 SQL 注入攻击免疫,SQL 注入式测试成功。

3.3 备份与恢复测试

- 备份与恢复测试的具体过程如下:
 - 鉴于该系统的数据量很大,该数据库管理系统提供了增量备份策略。测试人员设计下列测试用例进行备份与恢复测试。
 - 步骤 1:进行数据备份,验证数据备份过程本身能够正常进行;
 - 步骤 2:在原始机上进行数据恢复,验证数据恢复过程本身能够正常进行;
 - 步骤 3:在非原始机上进行数据恢复,验证数据恢复过程本身能够正常进行;
 - 步骤 4:将备份数据和恢复的数据进行比较,检查是否存在记录丢失的现象,是否出现设置内容不正确的现象;
 - 步骤 5:数据恢复后,进行业务流程测试,验证数据恢复的正确性。

测试通过,该数据库可进行数据的备份与恢复。

3.4 安全性访问控制测试

- 安全性访问控制测试的具体过程如下:
 - a)分析用户的身份,测试身份认证和访问控制。
 - 虽然该软件的数据关系比较复杂但是数据属于同一类,属于“录入数据并对数据进行查询与统计”的模式,没有数据的再加工。可将操作员分为三种角色:
 - (1)查询者角色:数据库 db_datareader 权限,只能对数据进行查询与统计。

(2) 录入者角色: 数据库 db_datawriter 权限, 除查询和统计外, 还可对数据进行添加、修改及删除操作。

(3) 管理员角色: dbcreators 或 sysadmin 权限, 除上述两项权利外, 还可对数据进行备份、恢复, 增加、删除操作人员, 为操作人员设置、更改口令等系统管理操作, 并且能够创建系统数据库。

软件运行时会出现一个登录窗口, 在此输入操作员的名称及口令, 若操作员合法且口令正确则进入软件主窗体, 主窗体上的菜单根据操作员的角色而不同。

b) 进行数据保密性测试。

打开存储密码的文件 MIMA.dat, 密码不是显示明文, 而是经过加密保存。对 MIMA.dat 文件中的密码内容删除后, 输入明文密码, 将文件保存, 并以明文密码登录, 数据库系统提示密码错误。

4 结束语

文中对数据库系统安全性测试技术进行具体分析和研究, 将数据库安全性测试分为数据完整性测试、健壮性及防范攻击性测试、备份与恢复测试、安全性访问控制测试, 并分别介绍四个测试项在某政务信息系统中的应用, 为数据库安全性测试提供了策略和依据。

参考文献:

[1] 郑雷雷, 宋丽华, 郭锐, 等. B/S 架构软件的安全性测试

(上接第 139 页)

4 结束语

针对航天测控系统关键数据存储备份的现状, 研究了远程数据容灾技术, 并基于 Oracle 数据库的远程数据复制技术设计了容灾方案。为验证容灾方案的有效性, 搭建了实验验证环境。实验测试结果表明, 设计的容灾方案可以实现 $RTO < 10 \text{ min}$ 及 $RPO < 5 \text{ min}$ 的容灾能力, 达到第 5 级的容灾要求。在未来的工作中, 将进一步评估航天测控系统数据容灾的可行性, 建设航天测控数据容灾系统, 实现对航天测控关键数据的持续数据保护^[11-12]。

参考文献:

- [1] 于志坚. 我国航天测控系统的现状与发展[J]. 中国工程科学, 2006, 8(10): 42-46.
- [2] 黄瑜华, 周彬. 容灾技术在军事航天指控中心设计中的应用[J]. 测控技术, 2007, 26(6): 23-24.
- [3] Fallara P. Disaster recovery planning[J]. IEEE potentials, 2004, 22(5): 42-44.
- [4] Damoulakis J. Continuous protections [J]. Storage, 2004, 3

研究[J]. 计算机技术与发展, 2012, 22(1): 211-224.

- [2] 张岩. 数据库安全性测试研究[J]. 计算机安全, 2012(11): 33-36.
- [3] 周伟明. 软件测试实践[M]. 北京: 电子工业出版社, 2008.
- [4] 何鑫, 郑军, 刘畅. 软件安全性测试研究综述[J]. 计算机测量与控制, 2011, 19(3): 493-496.
- [5] 魏祖宽. 数据库系统及应用[M]. 北京: 电子工业出版社, 2008.
- [6] 贺红, 徐宝文, 袁胜忠. 对应用软件进行安全测试的对手模式及其应用[J]. 计算机科学, 2006, 33(9): 266-269.
- [7] 张敏, 徐霞, 冯登国. 数据库安全[M]. 北京: 科学出版社, 2005.
- [8] 王爱平. 软件测试[M]. 北京: 清华大学出版社, 2008.
- [9] Huang Y W, Huang S K, Lin T P, et al. Web application security assessment by fault injection and behavior monitoring [C]//Proceedings of the twelfth international world wide web conference. Budapest, Hungary: [s. n.], 2003: 21-25.
- [10] 施寅生, 邓世伟, 谷天阳. Web 服务安全性测试技术研究[J]. 计算机工程与科学, 2007, 29(10): 11-13.
- [11] Microsoft Developer Network. The trustworthy computing security development lifecycle [EB/OL]. 2005-05-23. <http://msdn.microsoft.com/en-au/library/ms995349.aspx>.
- [12] Wikipedia. Software testing [EB/OL]. 2008-11-28. http://en.wikipedia.org/wiki/Software_testing.
- [13] 荀珂. 数据库系统安全性浅析[J]. 电脑知识与技术, 2011, 7(18): 4286-4288.

(4): 33-39.

- [5] 蒋秀凤, 何凤英. Oracle 9i 数据库管理教程[M]. 北京: 清华大学出版社, 2005.
- [6] 王靖, 刘丽洁. Oracle DataGuard 容灾监控方案探讨[J]. 信息通信, 2012(6): 201-203.
- [7] 徐西波. Oracle GoldenGate 技术在港口的应用浅介[J]. 山东通信技术, 2012(3): 42-47.
- [8] 张云帆. Oracle 数据库备份与恢复策略[J]. 计算机工程, 2009, 35(15): 85-87.
- [9] 李峰, 刘晓洁, 林翰翔. 基于 Oracle 数据库的容灾系统[J]. 计算机工程与设计, 2011, 32(11): 3573-3577.
- [10] 厉剑, 廉国斌, 黄栋. 数据容灾系统与 CDP 技术[J]. 计算机技术与发展, 2009, 19(1): 168-171.
- [11] 叶嘉酩, 胡晓勤, 王喆. 多数据库容灾系统的设计与实现[J]. 计算机工程与设计, 2012, 33(12): 4541-4545.
- [12] Han Hoonglin, Li Lin, Zhu Dehai. Research and implementation on remote disaster recovery system [C]//Proceedings of 2012 international conference on computer science and service system. [s. l.]: IEEE Press, 2012: 875-879.

数据库系统安全性测试技术研究

作者：

周薇， ZHOU Wei

作者单位：

江苏自动化研究所, 江苏 连云港, 222061

刊名：

计算机技术与发展

ISTIC

英文刊名：

Computer Technology and Development

年，卷(期)：

2014(2)

本文链接：http://d.wanfangdata.com.cn/Periodical_wjfz201402035.aspx