# 第 3 章
# 数据库安全

数据库技术研究和管理的对象是数据。数据库技术所涉及的具体内容：通过对数据的统一组织和管理，按照指定的结构建立相应的数据库和数据仓库；利用数据库管理系统和数据挖掘系统设计出能够实现对数据库中的数据进行添加、修改、删除、处理、分析、理解、报表和打印等多种功能的数据管理和数据挖掘应用系统；利用应用管理系统最终实现对数据的处理、分析和理解。

## 3.1 数据库技术

数据库技术是信息系统的一个核心技术，是一种计算机辅助管理数据的方法，它研究如何组织和存储数据，如何高效地获取和处理数据。它是通过研究数据库的结构、存储、设计、管理及应用的基本理论和实现方法，并利用这些理论来实现对数据库中的数据进行处理、分析和理解的技术，即数据库技术是研究、管理和应用数据库的一门软件科学。

数据库技术是现代信息科学与技术的重要组成部分，是计算机数据处理与信息管理系统的核心。数据库技术研究和解决了计算机信息处理过程中大量数据有效地组织和存储的问题，在数据库系统中减少数据存储冗余、实现数据共享、保障数据安全，以及高效地检索数据和处理数据。

### 3.1.1 数据库基础知识

数据库技术产生于 20 世纪 60 年代末 70 年代初，其主要目的是有效地管理和存取大量的数据资源。数据库技术主要研究如何存储、使用和管理数据。数年来，数据库技术和计算机网络技术的发展相互渗透，相互促进，已成为当今计算机领域发展迅速、应用广泛的两大领域。数据库技术不仅应用于事务处理，并且进一步应用到情报检索、人工智能、专家系统

和计算机辅助设计等领域。

数据管理技术是对数据进行分类、组织、编码、输入、存储、检索、维护和输出的技术。它的发展大致经过了三个阶段：人工管理阶段，文件系统阶段，数据库系统阶段。

数据模型是现实世界在数据库中的抽象，也是数据库系统的核心和基础。数据模型通常包括以下三个要素。

(1) 数据结构：数据结构主要用于描述数据的静态特征，包括数据的结构和数据间的联系。

(2) 数据操作：数据操作是指在数据库中能够进行的查询、修改、删除现有数据或增加新数据的各种数据访问方式，并且包括数据访问相关的规则。

(3) 数据完整性约束：数据完整性约束由一组完整性规则组成。

数据库理论领域中最常见的数据模型主要有以下三种。

(1) 层次模型(hierarchical model)：层次模型使用树形结构来表示数据及数据之间的联系。

(2) 网状模型(network model)：网状模型使用网状结构表示数据及数据之间的联系。

(3) 关系模型(relational model)：关系模型是一种理论最成熟、应用最广泛的数据模型。在关系模型中，数据存放在一种称为二维表的逻辑单元中，整个数据库又是由若干个相互关联的二维表组成的。

数据库技术发展的趋势：数据库与学科技术的结合将会建立一系列新数据库，如分布式数据库、并行数据库、知识库和多媒体数据库等，这将是数据库技术重要的发展方向。其中，许多研究者都将多媒体数据库作为研究的重点，并认为多媒体技术和可视化技术引入多媒体数据库将是未来数据库技术发展的热点和难点。

许多研究者从实践的角度对数据库技术进行研究，提出了适合应用领域的数据库技术，如工程数据库、统计数据库、科学数据库、空间数据库和地理数据库等。这类数据库在原理上没有多大的变化，但是它们却与一定的应用相结合，从而加强了系统对有关应用的支撑能力，尤其是在数据模型、语言和查询方面。部分研究者认为，随着研究工作的继续深入和数据库技术在实践工作中的应用，数据库技术将会更多朝着专门应用领域发展。

## 3.1.2　数据库典型攻击方法

### 1. 密码入侵

以前的 Oracle 数据库有一个默认的用户名 Scott，以及默认的密码 tiger；其他数据库系统也大多有默认密码。这些默认的登录对于黑客来说尤其简单，借此他们可以轻松地进

入数据库。Oracle 和其他主要的数据库厂商在其新版本的产品中对其进行弥补，不再让用户保持默认的和空的用户名及密码。但即使是唯一的、非默认的数据库密码也是不安全的，通过暴力破解就可以找到弱密码。

**2. 特权提升**

特权提升通常与管理员的配置错误有关，如一个用户被误授予超过其实际需要的访问权限。此外，拥有一定访问权限的用户可以轻松地从一个应用程序跳转到数据库，即使他并没有这个数据库的相关访问权限。黑客只需要得到少量特权的用户密码，就可以进入数据库系统，然后访问读取数据库内的任何表，包括信用卡信息和个人信息。

**3. 漏洞入侵**

当前，正在运行的多数 Oracle 数据库中，有 10～20 个已知的漏洞，黑客们可以利用这些漏洞攻击进入数据库。虽然 Oracle 和其他的数据库都为其漏洞做了补丁，但是很多用户并没有给他们的系统漏洞打补丁，因此这些漏洞常常成为黑客入侵的途径。其他数据库系统也一样，不同的版本有不同的已知的安全漏洞。

**4. SQL 注入**

SQL 注入攻击是黑客对数据库进行攻击的常用手段之一。随着 B/S(browser/server)模式应用开发的发展，使用这种模式编写应用程序的程序员也越来越多。但是由于程序员的水平及经验参差不齐，许多程序员在编写代码时，没有对用户输入数据的合法性进行判断，使应用程序存在安全隐患。用户可以提交一段数据库查询代码，根据程序返回的结果，获得某些他想得知的数据，这就是 SQL Injection(SQL 注入)。SQL 注入是从正常的 WWW 端口访问，而且表面看起来和一般的 Web 页面访问没有区别，所以市面上的防火墙都不会对 SQL 注入发出警报。如果管理员没有查看 IIS 日志的习惯，可能被入侵很长时间都不会发觉。SQL 注入的手法相当灵活，在注入的时候会碰到很多意外的情况，需要构造巧妙的 SQL 语句，从而成功获取想要的数据。

**5. 窃取备份**

如果备份硬盘在运输或仓储过程中被窃取，而这些磁带上的数据库数据又没有加密，黑客根本不需要接触网络就可以实施破坏了。通过窃取备份实施的攻击主要是由于管理员对备份的介质疏于跟踪和记录。

**6. DDoS 攻击**

数据库多数表现为 CPU 长期处于 100%的状态，连接数比较多，同时服务器一般可以登录。DDoS(distributed denial of service，分布式拒绝服务)是黑客用傀儡机对数据库进行攻击的一种模式。

## 3.2　常见的数据库系统

目前常见的数据库系统有 Oracle、DB2、SQL Server、MySQL、PostgreSQL 和 SQLite 等。

### 3.2.1　Oracle

20 世纪 70 年代，Ampex 软件公司为美国中央情报局设计了一套名为 Oracle 的数据库，埃里森是程序员之一。

1977 年，埃里森与同事 Robert Miner 创立“软件开发实验室”(Software Development Labs)，当时 IBM 发表《关系数据库》的论文，埃里森以此造出新数据库，名为甲骨文。

1978 年，公司迁往硅谷，更名为“关系式软件公司”(RSI)。RSI 在 1979 年的夏季发布了可用于 DEC 公司的 PDP-11 计算机上的商用 ORACLE 产品，这个数据库产品整合了比较完整的 SQL 实现，其中包括查询、连接及其他特性。美国中央情报局想买一套这样的软件来满足他们的需求，但在咨询 IBM 公司之后发现 IBM 没有可用的商用产品，他们联系了 RSI，于是 RSI 有了第一个客户。1982 年再更名为甲骨文(Oracle)。

Oracle 是甲骨文公司的一款关系数据库管理系统(relational database management system，RDBMS)Oracle 经典图标如图 3.1 所示。


**图 3.1　Oracle 经典图标**

Oracle 安全加固包括以下 11 个方面。

1) 安全补丁的更新

及时更新数据库的安全补丁，减少数据库系统可能受到的安全攻击面。参考 Oracle 厂商建议，仅对已发现的特定漏洞或缺陷安装相应补丁。

2）$ORACLE_HOME/bin 目录权限保护

确保对 $ORACLE_HOME/bin 目录的访问权限尽可能少，运行命令：

```
chown -R oracle:dba $ORACLE_HOME/bin
```

验证：ls-l $ORACLE_HOME/bin

确保该目录下的文件属主为 Oracle 用户，且其他用户没有写权限。

3）Oracle 数据字典的保护

设置保护后，可防止其他用户（具有'ANY'system privileges）使用数据字典时，具有相同的'ANY'权限。使用文本方式，打开数据库配置文件 init < sid >. ora；更改以下参数 O7_DICTIONARY_ACCESSIBILITY=。

（1）Oracle 9i、10g：默认值是 False。

（2）Oracle 8i：默认值是 True，需要改成 False。

（3）如果用户必须需要该权限，赋予其权限 SELECT ANY DICTIONARY。

验证：SQL>show parameter O7_DICTIONARY_ACCESSIBILITY

```
NAME TYPE VALUE
------------------------------------------------------------------
O7_DICTIONARY_ACCESSIBILITY boolean FALSE
```

4）加强访问控制

设置正确识别客户端用户，并限制操作系统用户数量（包括管理员权限、root 权限和普通用户权限等）。

（1）使用文本方式，打开数据库配置文件 init < sid >. ora；设置参数 REMOTE_OS_AUTHENT 值为 False（SAP 系统不可设置为 False）。

（2）在数据库的账户管理中删除不必要的操作系统账号。

设置（需重启数据库）：alter system set remote_os_authent=false scope=spfile；

验证：SQL>show parameter remote_os_authent

```
NAME TYPE VALUE
------------------------------------------------------------------
remote_os_authent boolean FALSE
```

5）密码文件管理

配置密码文件的使用方式，使用文本方式，打开数据库配置文件 init < sid >. ora；设置参数 REMOTE_LOGIN_PASSWORD_FILE=NONE。

None：使得 Oracle 不使用密码文件，只能使用 OS 认证，不允许通过不安全网络进行远程管理。

Exclusive：可以使用唯一的密码文件，但只限一个数据库。密码文件中可以包括除了SYS用户的其他用户。

Shared：可以在多个数据库上使用共享的密码文件，但是密码文件中只能包含SYS用户。

设置：（需重启数据库）alter system set remote_login_passwordfile＝none scope＝spfile；

验证：SQL＞show parameter remote_login_passwordfile

```
NAME TYPE VALUE
------------------------------------------------------------
remote_login_passwordfile string NONE
```

6）用户账号管理

为了安全考虑，应当锁定Oracle当中不需要的用户或改变默认用户的密码。锁定不需要的用户，使用SQL语句：ALTER USER user PASSWORD EXPIRE。

注意锁定MGMT_VIEW、DBSNMP、SYSMAN账号或修改密码（如果要使用DBConsole、DBSNMP、SYSMAN不能锁定，请修改密码）。

7）最小权限使用规则

（1）应该只提供最小权限给用户（包括SYSTEM和OBJECT权限）。

（2）从PUBLIC组中撤回不必要的权限或角色（如UTL_SMTP、UTL_TCP、UTL_HTTP、UTL_FILE、DBMS_RANDON、DBMS_SQL、DBMS_SYS_SQL、DBMS_BACKUP_RESTORE）。

撤销不需要的权限和角色，使用SQL语句：

```
REVOKE  EXECUTE ON SYS.UTL_HTTP FROM PUBLIC;
REVOKE  EXECUTE ON SYS.UTL_FILE FROM PUBLIC;
REVOKE  EXECUTE ON SYS.UTL_SMTP FROM PUBLIC;
REVOKE  SELECT ON ALL_USERS FROM PUBLIC;
```

8）SYS用户处理

Oracle数据库系统安装后，自动创建一个叫作SYS的数据库管理员用户，当该用户以sysdba方式连接数据库时，便具有全部系统权限，因而对它的保护尤为重要。

加固方法：更换SYS用户密码，符合密码复杂度要求；新建一个DBA用户，作为日常管理使用。

9）密码策略

在Oracle，可以通过修改用户概要文件来设置密码的安全策略，可以自定义密码的复杂度。以下参数和密码安全有关。

FAILED_LOGIN_ATTEMPTS：最大错误登录次数。

PASSWORD_GRACE_TIME：密码失效后锁定时间。

PASSWORD_LIFE_TIME：密码有效时间。

PASSWORD_LOCK_TIME：登录超过有效次数锁定时间。

PASSWORD_REUSE_MAX：密码历史记录保留次数。

PASSWORD_REUSE_TIME：密码历史记录保留时间。

PASSWORD_VERIFY_FUNCTION：密码复杂度审计函数。

10）数据库操作审计

Oracle 数据库具有对其内部所有发生的活动的审计能力，审计日志一般放在 sys.aud$ 表中，也可以写入操作系统的审计跟踪文件中。可审计的活动有三种类型：登录尝试、数据库活动和对象存取。默认情况下，数据库不启动审计，要求管理员配置数据库后才能启动审计。

使用文本方式，打开数据库配置文件 init < sid >.ora；更改以下参数配置 AUDIT_TRAIL＝True。

```
alter system set audit_trail = 'OS' scope = spfile;
alter system set Audit_sys_operations = true scope = spfile;
```

默认为 false，当设置为 true 时，所有 SYS 用户（包括以 sysdba、sysoper 身份登录的用户）的操作都会被记录。

验证：SQL＞show parameter audit

```
NAME TYPE VALUE
------------------------------------------------------------
audit_sys_operations boolean TRUE
audit_trail string OS
```

11）本地缓存区溢出防护

'oracle'程序存在本地缓冲区溢出。在传递命令行参数给'oracle'程序时缺少充分的边界缓冲区检查，可导致以'oracle'进程权限在系统上执行任意代码，需要进行有效加固。

以系统管理员权限登录操作系统，进入 Oracle 安装目录。

运行：chmod o-x oracle

加强对 Oracle 文件的可执行控制，这样非 Oracle 账号对该文件没有读取、运行权限。

### 3.2.2 DB2

IBM DB2 是美国 IBM 公司开发的一套关系型数据库管理系统，它主要的运行环境为

UNIX(包括 IBM 自家的 AIX)、Linux、IBM i(旧称 OS/400)和 z/OS,以及 Windows 服务器版本。

DB2 主要应用于大型应用系统,具有较好的可伸缩性,可支持从大型计算机到单用户环境。DB2 提供了高层次的数据利用性、完整性、安全性、可恢复性,以及小规模到大规模应用程序的执行能力,具有与平台无关的基本功能和 SQL 命令。DB2 采用了数据分级技术,能够使大型计算机数据很方便地下载到 LAN 数据库服务器,使得客户机/服务器用户和基于 LAN 的应用程序可以访问大型计算机数据,并使数据库本地化及远程连接透明化。DB2 以拥有一个非常完备的查询优化器而著称,其外部连接改善了查询性能,并支持多任务并行查询。DB2 具有很好的网络支持能力,每个子系统可以连接十几万个分布式用户,可同时激活上千个活动线程,对大型分布式应用系统尤为适用。

1970 年是数据库历史上划时代的一年,IBM 公司的研究员 E. F. Codd 发表了业界第一篇关于关系数据库理论的论文 *A Relational Model of Data for Large Shared Data Banks*,首次提出了关系模型的概念。这篇论文是计算机科学史上极重要的论文之一,奠定了 Codd 博士"关系数据库之父"的地位。

1973 年,IBM 研究中心启动了 System R 项目,研究多用户与大量数据下关系型数据库的可行性,它为 DB2 的诞生打下了良好基础。由此取得了一大批对数据库技术发展具有关键性作用的成果,该项目于 1988 年被授予 ACM 软件系统奖。

1992 年,第一届 IDUG(The International DB2 Users Group)欧洲大会在瑞士日内瓦召开,这标志着 DB2 应用的全球化,DB2 经典图标如图 3.2 所示。


**图 3.2　DB2 经典图标**

DB2 数据库安全加固包括以下方面。

1) 最小化权限设置

在数据库权限配置功能中,根据用户的业务需要配置所需的最低权限,防止滥用数据库权限,降低安全风险。

从用户 Roy 撤销 staff 表上的 alter 特权,语句如下:

```
REVOKE ALTER ON TABLE staff FROM USER roy
```

从 Jen 撤销 staff 表上的所有特权，语句如下：

```
REVOKE ALL PRIVILEGES ON TABLE staff FROM USER roy
```

2）启用日志记录，设置为存档日志模式

实现在线备份和恢复，日志的默认模式是循环日志。默认情况下，只能实现数据库的离线备份和恢复。

```
db2 update db cfg for using logretain on
```

**注意**：更改为 on 后，当查看数据库配置参数 logretain 的值时，实际显示是 recovery。更改此参数后，再次连接到数据库将显示数据库处于备份挂起状态。此时，需要对数据库进行离线备份，以使数据库状态正常。

**注意**：确保将内存中仍然缓冲的所有审计记录写入磁盘：db2audit flush。

3）用户身份验证失败锁定

对于采用静态密码认证技术的数据库，应在用户连续认证失败次数超过 6 次（不含 6 次）时配置。

修改/etc/login.defs，设置如下：

```
vi /etc/ login.defs
LOGIN_ RETRIES 6
```

### 3.2.3 SQL Server

SQL Server 是 Microsoft 公司推出的关系型数据库管理系统，具有使用方便、可伸缩性好与相关软件集成程度高等优点，可跨越从运行 Microsoft Windows 98 的微型计算机到运行 Microsoft Windows 2012 的大型多处理器的服务器等多种平台使用。

Microsoft SQL Server 是一个全面的数据库平台，使用集成的商业智能（business intelligence，BI）工具提供了企业级的数据管理。Microsoft SQL Server 数据库引擎为关系型数据和结构化数据提供了更安全可靠的存储功能，可以构建和管理用于业务的高可用和高性能的数据应用程序。

SQL Server 只在 Windows 上运行，Microsoft 这种专有策略的目标是将客户锁定到 Windows 环境中，限制客户通过选择一个开放的基于标准的解决方案来获取革新和价格竞争带来的好处。此外，Windows 平台本身的可靠性、安全性和可伸缩性也是有限的，SQL Server 经典图标如图 3.3 所示。


**图 3.3　SQL Server 经典图标**

SQL Server 安全加固包括以下方面。

1）安装安全补丁

在安装补丁之前应先对数据库进行备份，停止 SQL Server 服务，然后在 Microsoft SQL Server Download Web Site 下载补丁进行安装。

2）禁用不必要的服务

在 SQL Server 默认安装时，有 MSSQLSERVER、QLSERVERAGENT、SSQLServerADHelper 和 Microsoft Search 这四个服务，除 MSSQLSERVER 外，其他的如果不需要，建议禁用。

3）限制 SQL Server 使用的协议

在 Microsoft SQL Server 程序组，运行服务网络实用工具，建议只使用 TCP/IP 协议，禁用其他协议。

TCP/IP 协议栈的增强主要是一些注册表项的修改，主要包括以下内容：

```
HKLM\System\CurrentControlSet\Services\Tcpip\Parameters\DisableIPSourceRouting
```

**注意**：密钥值应设置为 2，以防止源路由欺骗攻击。

```
HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\enableimpredirect
```

描述：ICMP 重定向的键值应设置为 0。

```
HKLM\System\CurrentControlSet\Services\Tcpip\Parameters\SynAttackProtect
```

**注意**：密钥值应设置为 2，以防止 SYN FLOOD 攻击。

### 3.2.4　MySQL

MySQL 是一个关系型数据库管理系统，是瑞典的 MySQL AB 公司开发的。

关系型数据库管理系统将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就加快了速度并提高了灵活性。

MySQL 所使用的 SQL 语言是用于访问数据库的最常用标准化语言。MySQL 软件采用了双授权政策，分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是开

图 3.4　MySQL 经典图标

放源码这一特点，一般中小型网站的开发会选择 MySQL 作为网站数据库，MySQL 经典图标如图 3.4 所示。

与其他的大型数据库（如 Oracle、DB2、SQL Server 等）相比，MySQL 有它的不足之处，但对于一般的个人使用者和中小型企业来说，MySQL 提供的功能已经足够，而且由于 MySQL 是开放源码软件，因此可以大大降低总体拥有成本。

Linux 作为操作系统，Apache 或 Nginx 作为 Web 服务器，MySQL 作为数据库，PHP/Perl/Python 作为服务端脚本解释器。由于这四个软件都是免费或开放源码软件，因此使用这种方式就可以建立起一个稳定、免费的网站系统，被业界称为 LAMP 或 LNMP 组合。

MySQL 数据库安全加固包括以下方面。

1）安装完 MySQL 后需要做的工作

安装 mysql-client，运行 mysql_secure_installation 会执行几个设置：

```
[root@localhost ~]# mysql_secure_installation
```

（1）为 root 用户设置密码。

（2）删除匿名账号。

（3）取消 root 用户远程登录。

（4）删除 test 库和对 test 库的访问权限。

（5）刷新授权表使修改生效。

通过这几项的设置能够提高 MySQL 库的安全。

2）禁止远程连接数据库

在命令行 netstat -ant 下看到默认的 3306 端口是打开的，此时打开了 mysqld 的网络监听，允许用户远程通过账号和密码连接本地数据库，默认情况是允许远程连接数据库的。为了禁止该功能，启动 skip-networking，不监听 SQL 的任何 TCP/IP 的连接，切断远程访问的权利，保证安全性。若需要远程管理数据库，可通过安装 PhpMyadmin 来实现。若确实需要远程连接数据库，至少修改默认的监听端口，同时添加防火墙规则，只允许可信任的网络的 MySQL 监听端口的数据通过。

```
# vi /etc/my.cf
```

将 # skip-networking 注释去掉。

```
# /usr/local/mysql/bin/mysqladmin -u root -p shutdown    //停止数据库
#/usr/local/mysql/bin/mysqld_safe -user=mysql &         //后台用 mysql 用户启动 mysql
```

3）限制连接用户的数量

数据库的某用户多次远程连接会导致性能下降和影响其他用户的操作，有必要对其进行限制。可以通过限制单个账户允许的连接数量来实现，即设置 my.cnf 文件的 mysqld 中的 max_user_connections 变量来完成。GRANT 语句也可以支持资源控制选项来限制服务器对一个账户允许的使用范围。

```
# vi /etc/my.cnf
[mysqld]
max_user_connections = 2
```

4）用户目录权限限制

默认的 MySQL 安装在/usr/local/mysql，而对应的数据库文件在/usr/local/mysql/var 目录下，因此，必须保证该目录不允许未经授权的用户访问后将数据库打包复制，所以要限制对该目录的访问。确保 mysqld 运行时，只使用对数据库目录具有读或写权限的 Linux 用户来运行。

```
# chown -R root /usr/local/mysql/                    //mysql 主目录给 root 用户
# chown -R mysql.mysql /usr/local/mysql/var          //确保数据库目录权限所属 mysql 用户
```

### 3.2.5　PostgreSQL

PostgreSQL 是一种特性非常齐全的自由软件的对象-关系型数据库管理系统（object relational database management system，ORDBMS），是加州大学伯克利分校的计算机系开发的。PostgreSQL 支持大部分的 SQL 标准且提供了很多其他现代特性，如复杂查询、外键、触发器、视图、事务完整性和多版本并发控制等。同样，PostgreSQL 也可以用许多方法扩展，例如，通过增加新的数据类型、函数、操作符、聚集函数、索引方法和过程语言等。此外，因为许可证的灵活，任何人都可以以任何目的免费使用、修改和分发 PostgreSQL。

PostgreSQL 最初设想于 1986 年，当时被叫作 Berkley Postgres Project。该项目一直到 1994 年都处于演进和修改中，直到开发人员 Andrew Yu 和 Jolly Chen 在 Postgres 中添加了一个 SQL（structured query language，结构化查询语言）翻译程序，该版本叫作 Postgres 95，在开放源代码社区发放。PostgreSQL 经典图标如图 3.5 所示。


**图 3.5　PostgreSQL 经典图标**

1996 年，开发人员再次对 Postgres 95 做了较大的

改动，并将其作为 PostgresSQL 6.0 版发布。该版本的 Postgres 提高了后端的速度，包括增强型 SQL92 标准及重要的后端特性(包括选择、默认值、约束和触发器)。

PostgreSQL 数据库安全加固包括以下方面。

(1) PostgreSQL 支持丰富的认证方法：信任认证、密码认证、PAM 认证等。PostgreSQL 默认配置只监听本地端口，无法通过远程 TCP/IP 连接数据库，需要修改 postgresql.conf 中的 listen_address 字段修改监听端口，使其支持远程访问。例如，listen_addresses ='*' 表示监听所有端口。

(2) 线上重要数据库禁止使用 trust 方式进行认证，必须使用 md5 方式。

(3) 重命名数据库超级管理员账户为 pgsqlsuper，此账号由 DBA 负责人保管，禁止共用。

(4) 配置数据库客户端支持 SSL 连接的配置。客户端认证是由一个配置文件控制的，存放在数据库集群的数据目录里。

(5) 用 openssl 命令生成密钥对，创建一个自签名的服务器密钥(server.key)和证书(server.crt)。

(6) 开启 TCP/IP 连接：将 postgresql.conf 参数 tcpip_socket 设置为 true。

(7) 开启 SSL：将 postgresql.conf 参数 ssl 设置为 true。

(8) 根据最小权限要求给用户配置角色和权限。

```
postgres=# select * from pg_authid; //查看用户具有的角色
```

为了保护数据安全，在用户对某个数据库对象进行操作之前，必须检查用户在对象上的操作权限。访问控制列表(access control lists，ACL)是对象权限管理和权限检查的基础，在 PostgreSQL 通过操作 ACL 实现对象的访问控制管理。

(9) 审计是指记录用户的登录退出及登录后在数据库里的行为操作，可以根据安全等级的不同设置不一样级别的审计。默认需设置如下的安全配置参数：

logging_collector：是否开启日志收集开关，默认 off，开启要重启 DB。

log_destination：日志记录类型，默认是 stderr，只记录错误输出。

log_directory：日志路径，默认是 $PGDATA/pg_log。

log_filename：日志名称，默认是 postgresql-%Y-%m-%d_%H%M%S.log。

log_connections：用户 session 登录时是否写入日志，默认 off。

log_disconnections：用户 session 退出时是否写入日志，默认 off。

log_rotation_age：保留单个文件的最大时长，默认是 1d。

log_rotation_size：保留单个文件的最大尺寸，默认是 10MB。

(10) 严格控制数据库安装目录的权限，除了数据文件目录，其他文件和目录属主都改

为 root。及时更新数据库 bug 和安全补丁。

### 3.2.6　SQLite

SQLite 是一款轻型的数据库，是关系型数据库管理系统。它包含在一个相对小的 C 库中，是 D. RichardHipp 建立的公有领域项目。它的设计目标是嵌入式的，占用的资源非常低，在嵌入式设备中，可能只需要几百千字节（KB）的内存就够了。它能够支持 Windows/Linux/UNIX 等主流的操作系统，同时能够和很多程序语言相结合，如 TCL、C＃、PHP 和 Java 等，还有 ODBC 接口。与 MySQL 和 PostgreSQL 这两款开源的世界著名数据库管理系统比，它的处理速度更快。SQLite 第一个 Alpha 版本诞生于 2000 年 5 月。

不像常见的客户-服务器范例，SQLite 引擎不是程序与之通信的独立进程，而是连接到程序中成为它的一个主要部分。所以主要的通信协议是在编程语言内的直接 API 调用。这在消耗总量、延迟时间和整体简单性上有积极的作用。整个数据库（定义、表、索引和数据本身）都在宿主主机上存储在一个单一的文件中。它的简单的设计是通过在开始一个事务时锁定整个数据文件而完成的。SQLite 经典图标如图 3.6 所示。


**图 3.6　SQLite 经典图标**

SQLite 安全加固包括以下方面。

1）SQLite 数据库加密（SQLCipher）

检查 SQLite 是否使用了 SQLCipher 开源库。SQLCipher 是对整个数据库文件进行加密。注意，该检测项不是警告用户有风险，而是提醒用户采用了 SQLite 对数据库进行了加密。

检测方法：SQLCipher 开源库会产生 Lnet/sqlcipher/database/SQLiteDatabase 包路径，只需在包路径中查找是否存在该路径的包名即可。

2）SQLit 使用 SQLite Encryption Extension（SEE）插件

SEE 是一个数据库加密扩展插件，允许 App 读取和写入加密的数据库文件，是 SQLite 的加密版本（收费版），提供的加密方式有 RC4、AES-128 in OFB mode、AES-128 in CCM mode、AES-256 in OFB mode。

检测方法：SEE 插件会产生 Lorg/sqlite/database/sqlite/SQLiteDatabase 包路径，只需在包路径中查找是否存在该路径的包名即可。

3) SQLite sql 注入漏洞防护

SQLite 作为 Android 平台的数据库，对于数据库查询，如果开发者采用字符串链接方式构造 SQL 语句，就会产生 SQL 注入。

建议：

(1) Provider 不需要导出，将 export 属性设置为 false；

(2) 若导出仅为内部通信使用，则设置 protectionLevel=signature；

(3) 不直接使用传入的查询语句用于 projection 和 selection，使用由 query 绑定的参数 selectionArgs；

(4) 完备的 SQL 注入语句检测逻辑。

4) Databases 任意读写漏洞防护

APP 在使用 openOrCreateDatabase 创建数据库时，将数据库设置了全局的可读权限，攻击者恶意读取数据库内容，获取敏感信息。在设置数据库属性时如果设置全局可写，攻击者可能会篡改、伪造内容，可能会进行诈骗等行为，造成用户财产损失。

建议：

(1) 用 MODE_PRIVATE 模式创建数据库；

(2) 使用 sqlcipher 等工具加密数据库；

(3) 避免在数据库中存储明文和敏感信息。

## 3.3 数据库技术新动态

当前数据库技术的新动态有键值对存储 Redis、列存储 HBase 和文档数据库存储 MongoDB 等。

### 3.3.1 键值对存储 Redis

Redis(remote dictionary server，远程字典服务)，是一个开源的使用 ANSI C 语言编写、支持网络、可基于内存亦可持久化的日志型、Key-Value 数据库，并提供多种语言的 API。从 2010 年 3 月 15 日起，Redis 的开发工作由 VMware 主持。从 2013 年 5 月开始，Redis 的开发由 Pivotal 赞助。

Redis 是一个 Key-Value 存储系统，和 Memcached 类似。它支持存储的 value 类型相对更多，包括 string(字符串)、list(链表)、set(集合)、zset(sorted set 有序集合)和 hash(哈希类型)。这些数据类型都支持 push/pop、add/remove 及取交集、并集和差集，以及更丰富

的操作，而且这些操作都是原子性的。在此基础上，Redis 支持各种不同方式的排序。与 Memcached 一样，为了保证效率，数据都是缓存在内存中。区别是 Redis 会周期性地将更新的数据写入磁盘或将修改操作写入追加的记录文件，并且在此基础上实现 master-slave（主从）同步。

Redis 的出现，很大程度补偿了 Memcached 这类 key/value 存储的不足，在部分场合可以对关系数据库起到很好的补充作用。它提供了 Java、C/C++、C#、PHP、JavaScript、Perl、Object-C、Python、Ruby 和 Erlang 等客户端，使用很方便。Redis 经典图标如图 3.7 所示。


**图 3.7　Redis 经典图标**

Redis 安全加固包括以下方面。

1）网络加固

如果仅为本地通信，要确保 Redis 监听在本地。具体设置：/etc/redis/redis.conf 中配置如 bind 127.0.0.1。

2）防火墙设置

如果需要其他机器访问，或者设置了 Master-Slave 模式，需添加防火墙设置，具体参考如/sbin/iptables -A INPUT -s x.x.x.x -p tcp --dport 6379 -j ACCEPT。

3）添加认证

默认情况下，Redis 未开启密码认证。开启认证模式，具体参考如下配置：

打开 /etc/redis/redis.conf，找到 requirepass 参数，设置密码，保存 redis.conf，最后重启 Redis 服务，/etc/init.d/redis-server restart。

4）设置单独账户

可设置一个单独的 Redis 账户。创建 Redis 账户，通过该账户启动 Redis 服务，具体操作如 setsid sudo -u redis /usr/bin/redis-serer /etc/redis/redis.conf。

5）限制 Redis 文件目录访问权限

设置 Redis 的主目录权限为 700；如果 Redis 配置文件独立于 Redis 主目录，将权限修改为 600，因为 Redis 密码明文存储在配置文件中。

```
$ chmod 700 /var/lib/redis              #Redis 目录
$ chmod 600 /etc/redis/redis.conf       #Redis 配置文件
```

## 3.3.2　列存储 HBase

HBase 是一个分布式的、面向列的开源数据库，该技术源于 Fay Chang 所撰写的

Google 论文《Bigtable：一个结构化数据的分布式存储系统》。就像 Bigtable 利用了 Google 文件系统(file system)所提供的分布式数据存储一样，HBase 在 Hadoop 之上提供了类似于 Bigtable 的能力。HBase 是 Apache 的 Hadoop 项目的子项目。HBase 不同于一般的关系数据库，它是一个适合非结构化数据存储的数据库。另一个不同的是 HBase 基于列的而不是基于行的模式。

HBase-Hadoop Database 是一个高可靠性、高性能、面向列、可伸缩的分布式存储系统，利用 HBase 技术可在廉价 PC Server 上搭建起大规模结构化存储集群。


**图 3.8　HBase 经典图标**

与商用大数据产品不同，HBase 是 Google Bigtable 的开源实现，类似 Google Bigtable 利用 GFS 作为其文件存储系统，HBase 利用 Hadoop HDFS 作为其文件存储系统。Google 运行 MapReduce 来处理 Bigtable 中的海量数据，HBase 同样利用 Hadoop MapReduce 来处理 HBase 中的海量数据；Google Bigtable 利用 Chubby 作为协同服务，HBase 利用 Zookeeper 作为对应。HBase 经典图标如图 3.8 所示。

HBase 安全加固包括以下方面。

1) 集群的模式配置

集群的模式分为分布式和单机模式，如果设置成 false，HBase 进程和 Zookeeper 进程在同一个 JVM 进程。

建议：线上配置为 true，默认值为 false。

```
${hbase.tmp.dir}/hbase
hbase.cluster.distributed
```

2) HBase 认证

HBase 集群安全认证机制，目前的版本只支持 kerberos 安全认证。

建议：线上配置为 kerberos，默认值为空。

将以下内容添加到每个客户端上的 hbase-site.xml 文件中。

```
<property>
    <name>hbase.security.authentication</name>
    <value>kerberos</value>
</property>
```

3）HBase 授权

HBase 是否开启安全授权机制。

建议：线上配置为 true，默认值为 false。

```
<property>
    <name>hbase.security.authorization</name>
    <value>true</value>
</property>
```

### 3.3.3　文档数据库存储 MongoDB

MongoDB 是一个基于分布式文件存储的数据库，由 C++语言编写，旨在为 Web 应用提供可扩展的高性能数据存储解决方案。

MongoDB 是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富且最像关系数据库的。它支持的数据结构非常松散，是类似 json 的 bson 格式，因此可以存储比较复杂的数据类型。MongoDB 最大的特点是它支持的查询语言非常强大，其语法有些类似于面向对象的查询语言，可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引。

在 MongoDB 中，文档是数据的基本单位，类似于关系数据库中的行（但是比行复杂）。多个键及其关联的值有序地放在一起就构成了文档。MongoDB 经典图标如图 3.9 所示。


**图 3.9　MongoDB 经典图标**

MongoDB 安全加固包括以下方面。

1）启用 auth

在可信赖网络中部署 MongoDB 服务器时启用 auth 是项好的安全实践。当网络受到攻击时，它能够提供“深层防御”。编辑配置文件来启用 auth。代码为 auth＝true。

2）不要将开发环境的数据库暴露在 Internet 上

限制对数据库的物理访问是安全性的非常重要的一个措施。如果没有必要，不要将开发环境的数据库暴露在 Internet 上。如果攻击者不能物理连接到 MongoDB 服务器，这种情形大打折扣，那么数据就不会比现在更安全。如果服务部署在亚马逊 Web 服务（AWS）

上，那么应当将数据库部署在虚拟私有云(virtual private cloud，VPC)的私有子网里。

3) 使用防火墙

防火墙的使用可以限制允许哪些实体连接 MongoDB 服务器。最佳的措施就是仅允许自己的应用服务器访问数据库。如果服务部署在不支持防火墙功能的提供商的主机上，那么可以使用 iptables 对服务器进行简单的配置。

4) 使用 key 文件建立复制服务器集群

指定共享的 key 文件，启用复制集群的 MongoDB 实例之间的通信。给配置文件中增加 keyfile 参数，复制集群里的所有机器上的这个文件的内容必须相同，代码如 keyFile＝/srv/mongodb/keyfile。

5) 禁止 REST 接口

在产品线环境下建议不要启用 MongoDB 的 REST 接口。这个接口不支持任何认证，默认情况下是关闭的。如果 rest 配置选项打开了这个接口，那么应该在产品线系统中关闭它，代码如 rest ＝ false。

6) 在 MongoDB 部署中使用 TLS/SSL

在 mongod 和 mongos 中包含以下配置选项：net.ssl 模式设置为 requireSSL。该设置限制每个服务器只能使用 TLS/SSL 加密连接。还可以指定值 allowSSL 或 preferSSL 来设置端口上混合 TLS/SSL 模式的使用。

## 3.4 近年数据库攻击披露

近年披露的部分数据库攻击如表 3.1 所示。

**表 3.1 近年数据库攻击披露**

| 漏 洞 号 | 影 响 产 品 | 漏 洞 描 述 |
| --- | --- | --- |
| CNVD-2020-29571 | Oracle MySQL Connectors<＝8.0.14<br>Oracle MySQL Connectors<＝5.1.48 | Oracle MySQL 中的 MySQL Connectors 8.0.14 及之前版本和 5.1.48 及之前版本的 Connector/J 组件存在安全漏洞。攻击者可利用该漏洞未授权读取、更新、插入或删除数据，影响数据的保密性和完整性 |
| CNVD-2020-29574 | Oracle Core RDBMS 11.2.0.4<br>Oracle Core RDBMS 12.1.0.2<br>Oracle Core RDBMS 12.2.0.1 | Oracle Database Server 中的 Core RDBMS 组件存在安全漏洞。攻击者可利用该漏洞控制 Core RDBMS，影响数据的可用性、保密性和完整性 |
| CNVD-2020-19263 | IBM DB2 V10.5<br>IBM DB2 V11.1<br>IBM DB2 V11.5 | 基于 Linux、UNIX 和 Windows 平台的 IBM DB2(包括 DB2 Connect Server)中存在缓冲区溢出漏洞。本地攻击者可利用该漏洞以 root 用户身份执行任意代码 |

续表

| 漏　洞　号 | 影 响 产 品 | 漏 洞 描 述 |
| --- | --- | --- |
| CNVD-2020-13176 | IBM DB2 V10.5<br>IBM DB2 V11.1<br>IBM DB2 V11.5 | IBM DB2 中存在安全漏洞。攻击者可通过发送特制的数据包利用该漏洞消耗大量内存并造成 DB2 异常终止 |
| CNVD-2020-27158 | MySQL AB<br>MySQL JDBC | MySQL AB 是由 MySQL 创始人和主要开发人创办的公司。<br>MySQL JDBC 驱动存在 XML 实体注入漏洞，攻击者可利用该漏洞获取服务器权限 |
| CNVD-2020-27461 | PostgreSQL JDBC driver v42.2.11 | PostgreSQL 是一个开源数据库系统。<br>PostgreSQL JDBC driver 存在命令执行漏洞，攻击者可利用该漏洞获取服务器权限 |
| CNVD-2020-02199 | PostgreSQL＜10.5<br>PostgreSQL＜9.6.10<br>PostgreSQL＜9.5.14 | PostgreSQL 中存在 SQL 注入漏洞。该漏洞源于基于数据库的应用缺少对外部输入 SQL 语句的验证。攻击者可利用该漏洞执行非法 SQL 命令 |
| CNVD-2020-22991 | Sqlite Sqlite＜＝3.31.1 | SQLite 3.31.1 及之前版本中存在安全漏洞，该漏洞源于程序未能正确处理 AggInfo 对象初始化。攻击者可利用该漏洞导致拒绝服务 |
| CNVD-2017-36161 | Redis Labs Redis＜3.2.7 | Redis 3.2.7 之前的版本中的 networking.c 文件存在跨站脚本漏洞。远程攻击者可利用该漏洞在浏览器中执行任意的脚本代码 |
| CNVD-2020-17183 | MongoDB Server＜4.0.11<br>MongoDB Server＜3.6.14<br>MongoDB Server＜3.4.22 | MongoDB Server 是美国 MongoDB 公司的一套开源 NoSQL 数据库。<br>MongoDB Server 存在权限许可和访问控制问题漏洞。攻击者可利用该漏洞执行其定义的代码 |

**说明**：如果想查看各个漏洞的细节，或者查看更多的同类型漏洞，可以访问国家信息安全漏洞共享平台：https://www.cnvd.org.cn/。

## 3.5　习题

1. 简述数据库技术出现的原因与发展。
2. 简述常见的数据库系统。
3. 简述数据库技术新动态。