第12章 安 全 性

本章主要讨论作为 MySQL 管理员的您，在维护 MySQL 安装的安全性和完整性方面能够做些什么。我们已经在第 11章 中略微谈到了一点安全性问题，如设置初始的 MySQL root 的口令的重要性以及怎样建立用户账号。这些内容是作为启动和运行安装过程的一部分被讨论的。在本章中，我们将更详细地讨论以下与安全性相关的问题：

- 为什么说安全性是重要的，应该警惕哪些攻击？
- 从服务器主机中的用户那里您将面临什么风险（内部安全性），能做什么？
- 从在网络上连接到服务器的客户机那里您将面临什么风险（外部安全性），能做什么？

MySQL 管理员有责任保护数据库内容的安全，使得记录只能由经过严格认证的那些用户访问。这包括内部安全性和外部安全性。

内部安全性关心文件系统级的问题，如保护 MySQL 数据目录免遭拥有运行服务器的机器账号的用户的攻击。但是，如果数据目录内容的文件许可权过分随意，有人可以将对应这些表的文件进行简单的替换的话，内部安全性就不能很好地确保适当建立对网络上客户机访问的授权表的控制。

外部安全性关心客户机从外部连接的问题，如防止 MySQL 服务器免遭通过网络进来的通过服务器的连接请求对数据库内容访问的攻击。要建立 MySQL 授权表使得它们不允许对服务器所管理的数据库的访问（除非提供了有效的名字和口令）。

本章提供了应该了解的有关问题的指导，并说明如何防止内部和外部级别中未认证的访问。

12.1 内部安全性：安全数据目录访问

MySQL 服务器提供了一个通过 mysql 数据库中的授权表来实现的灵活的权限系统。可以设置这些表的内容来允许或拒绝数据库对客户机的访问。这提供了关于未认证的网络访问数据的安全性。但是，如果服务器主机上的其他用户具有对该数据目录内容的直接访问权，则将不能对访问数据的网络建立良好的安全性。除非知道您是曾在运行 MySQL 服务器的机器上注册的惟一的一个人，否则需要关心在该机器上的其他用户获得对数据目录访问的可能性。

以下是您想要保护的内容：

- **数据库文件**。显然想要维护由服务器维护的数据库的保密性。数据库的所有者通常要考虑数据库内容的专有性。即使他们不考虑，也最多是使数据库的内容公共化，而不会使那些内容因数据库目录安全性低而被泄露。
- **日志文件。**常规和更新日志必须安全，因为它们包含了查询文本。这有相当的利害关系，因为具有日志文件访问的任何人都可以监控发生在数据库中的事务处理。

  与日志文件有关的更为特殊的安全性问题是，像 GRANT 和 SET PASSWORD 这样的查询被记录在日志中了。常规和更新日志文件包含敏感的查询文本，其中包括了口令（MySQL 使用口令加密，但这只适用于在口令设置之后的连接建立。设置口令的

过程包含在 GRANT、INSERT 或 SET PASSWORD 这样的查询中，但这些查询以纯文本的形式被记录。）如果一个攻击者具有对日志的读访问权，那他只需在日志中对 GRANT 或 PASSWORD 这样的词运行 grep 就能找到敏感信息。

显然，您不想让服务器主机上的其他用户拥有对数据目录文件的写访问权，因为那样的话，他们就可以在状态文件或数据库表上肆意践踏。但读访问也很危险。如果表文件可读取，那么窃取文件并使 MySQL 自己以纯文本的形式显示表的内容是微不足道的事。可按下列步骤进行：

1) 在服务器主机上安装您的MySQL服务器，但使用与正式服务器不同的端口、套接字和数据文件。

2) 运行 mysql_install_db 初始化您的数据目录。这将允许您作为 MySQL 的 root 用户访问服务器，因此您将具有完全控制服务器访问机制的权利。它还建立了一个 test 数据库。

3) 将您想窃取的表的相应文件拷贝到服务器数据目录下的 test 子目录中。

4) 启动作案服务器。您可以随意访问这些表。 SHOW TABLES FROM test 将显示您拥有一个被窃取表的备份， SELECT * 将显示任何这些表的全部内容。

5) 如果更坏一点，打开服务器的匿名用户账号的许可权，使任何人都能从任何地方连接到该服务器来访问您的 test 数据库。现在，您已经向全世界公布了这些被偷窃的表。

考虑一下刚才的情况，然后颠倒过来想。您希望有人对您这样做吗？当然不要。

通过在数据目录中执行 ls -l 可以确定数据目录中是否包含非安全的文件或目录。应查看具有以开启的“组”或“其他”许可权的文件或目录。以下是一个非安全数据目录的部分列表，是该数据目录中的一部分数据库目录：

```
% ls -l
total 10148
drwxrwxr-x  11 mysqladm wheel        1024 May  8 12:20 .
drwxr-xr-x  22 root     wheel         512 May  8 13:31 ..
drwx------   2 mysqladm mysqlgrp      512 Apr 16 15:57 menagerie
drwxrwxr-x   2 mysqladm wheel        512 Jan 25 20:43 mysql
drwxrwxr-x   7 mysqladm wheel         512 Aug 31  1998 sql-bench
drwxrwxr-x   2 mysqladm wheel        1536 May  6 06:11 test
drwx------   2 mysqladm mysqlgrp     1024 May  8 18:43 tmp
...
```

正如您所看到的，有些数据库目录有正确的许可权，而有些则不是这样。本例中的情况是由于时间引起的。较老的服务器创建了限制较少的许可权，且较老的服务器与较新的服务器相比，在设置许可权方面不严格（请注意，有更多限制的目录， menager 和 tmp，都有更为新的日期）。MySQL 当前的版本确保这些文件只对服务器运行的用户可读。

让我们来安排这些许可权，使得只有服务器的用户才能访问它们。主要的保护手段来自由 UNIX 文件系统本身提供的工具，这些工具可设置文件和目录的所有权及方式。操作步骤如下：

1) 定位到数据目录中：

```
% cd DATADIR
```

2) 设置该数据目录下所有文件的所有权为运行该服务器的账号所拥有（必须以 root 身份执行这一步）。在本书中，笔者对此账号的用户名和组名使用 mysqladm 和 mysqlgrp。可以用下列命令之一修改所有权：

```
# chown -R mysqladm.mysqlgrp .
# find . -follow -type d -print | xargs chown mysqladm.mysqlgrp
```

3) 修改数据目录和数据库目录的方式，使得它们仅对于 mysqladm 是可读的。这样防止了其他用户访问数据目录的内容。可以利用下列命令之一来进行，这些命令或者以 root 或者以 mysqladm 运行（后者更好，可以使作为 root 运行的命令数量最小化）：

```
% chmod -R go-rwx .
% find . -follow -type d -print | xargs chmod go-rwx
```

4) 对 mysqladm 用户设置数据目录内容的所有权和方式。现在，您应该确保总是以 mysqladm 运行，因为它现在是唯一拥有该数据目录访问权的用户。作为非 root 用户运行服务器的过程已在第 11 章中介绍。

在上述步骤之后，将拥有以下许可权：

```
% ls -l
total 10148
drwxrwx---  11 mysqladm mysqlgrp     1024 May  8 12:20 .
drwxr-xr-x  22 root     wheel         512 May  8 13:31 ..
drwx------   2 mysqladm mysqlgrp      512 Apr 16 15:57 menagerie
drwx------   2 mysqladm mysqlgrp      512 Jan 25 20:43 mysql
drwx------   7 mysqladm mysqlgrp      512 Aug 31  1998 sql-bench
drwx------   2 mysqladm mysqlgrp     1536 May  6 06:11 test
drwx------   2 mysqladm mysqlgrp     1024 May  8 18:43 tmp
...
```

12.2 外部安全性：安全网络访问

MySQL 安全性系统是灵活的。它允许以许多不同的方法设置用户访问权限。通常，可通过 GRANT 和 REVOKE 语句来进行，这些语句对控制客户机访问的授权表进行修改。但是，您拥有的可能是不支持这些语句的旧版本 MySQL（这些语句在 MySQL 3.22.11 以前的版本中没有使用），或者可能发觉用户的权限好像不是按希望地在工作。对于这样的情况，了解 MySQL 授权表的结构以及服务器怎样使用它们来决定访问许可权是有帮助的。您了解到这样一个程度，就可以通过直接修改授权表来增加、删除或修改用户的权限，还可以在检查表时诊断权限的问题。

笔者假定您已经阅读了第 11 章的“用户账号管理”一节，并理解 GRANT 和 REVOKE 语句是怎样工作的。GRANT 和 REVOKE 提供了建立 MySQL 账号和相关权限的便利方法，但是，它们只是一个前端。所有真正的操作都发生在 MySQL 授权表中。

12.2.1 MySQL 授权表的结构和内容

在网络上连接到服务器的客户机对 MySQL 数据库的访问是由授权表内容控制的。这些表定位在 mysql 数据库中，并在首次安装 MySQL 的过程中进行初始化（如附录 A 所描述的）。表12-1 和表12-2 示出列五个授权表，它们是 user、db、host、tables_priv 和 columns_priv。

表12-1 user、db 和 host 授权表的结构

访问列的活动范围		
user	**db**	**host**
Host	Host	Host

(续)

User	Db	Db
Password	User	
	数据库/表的权限列	
Alter_priv	Alter_priv	Alter_priv
Create_priv	Create_priv	Create_priv
Delete_priv	Delete_priv	Delete_priv
Drop_priv	Drop_priv	Drop_priv
Index_priv	Index_priv	Index_priv
Insert_priv	Insert_priv	Insert_priv
References_priv	References_priv	References_priv
Select_priv	Select_priv	Select_priv
Update_priv	Update_priv	Update_priv
	管理权限列	
user	**db**	**host**
File_priv	Grant_priv	Grant_priv
Grant_priv		
Process_priv		
Reload_priv		
Shutdown_priv		

授权表的内容按下列各项使用：

- user　user 表列出可连接到服务器和口令的用户，并指定用户拥有哪些全局（超级用户）权限（如果有的话）。任何在 user 表项中所允许的权限都是全局权限，并适用于所有的数据库。例如，如果在这里允许 DELETE 权限，则在该项中列出的用户可从任何表中删除记录。因此，在进行这项操作之前千万要小心，通常，最好在 user 表项中关闭所有的权限，而使用其他的、有更多限制的表来指定权限。对超级用户（如 root）则是一个例外，但一般很少有。
- db　db 表列出数据库以及哪些用户拥有访问这些数据库的权限。这里指定的权限适用于数据库中所有的表。
- host　host 表与 db表结合使用，在更细的级别上控制对特定主机的数据库访问权限。该表不受 GRANT 和 REVOKE 语句的影响，因此您会发现根本不会去使用它。
- tables_priv　tables_priv 表指定表级的权限。在这里所指定的权限适用于表中所有的列。
- columns_priv　columns_priv 表指定列级的权限。在这里所指定的权限适用于表中特定的列。

表12-2　table_priv 和 columns_priv 授权表的结构

访问列的活动范围	
tables_priv	**columns_priv**
Host	Host
Db	Db
User	User
Table_name	Table_name
Column_name	
权限列	
Table_priv	Column_priv

在12.2.4节“不用 GRANT 建立用户”中，我们将讨论 GRANT 语句怎样修改这些表以及怎样通过直接修改授权表来达到相同的结果。

tables_priv 和 columns_priv 表是在 MySQL 3.22.11 中引入的（与 GRANT 语句同时）。如

果您拥有 MySQL 的旧版本，则 mysql 数据库只有 user、db 和 host 表。如果已经从旧版本升级到 3.22.11 以上的版本，但仍没有 tables_priv 和 columns_priv 表的话，可运行 mysql_fix_privileges_table 脚本创建它们。

没有 rows_priv 表，因为 MySQL 不提供记录级的权限。例如，不能使用户只局限于某些列中包含某个值的表中的那些行。如果需要这个能力，必须编写应用程序。如果要想执行咨询记录级锁定（advisory record-level locking），可用附录 C 中介绍的 GET_LOCK() 函数进行。

授权表中包含两种类型的列：作用域列（ scope column）和权限列（privilege column），前者决定一个项何时可用，后者决定一个项可授予哪些权限（有些授权表中包含其他各式各样的列，但在这里与我们无关）。

1. 授权表的作用域列

授权表作用域列指定表项何时使用。每个授权表项中都包含 User 和 Host 列，以指明该项在特定的用户从特定的主机上连接时应用（ host 表是一个例外，它被用于一种特定的、我们还未接触过的方法中）。其他表中包含附加的作用域列。例如， db 表中包含一个 Db 列以指明该项适用于哪个数据库。同样， tables_priv 和 columns_priv 表中包含使该作用域对数据库中特定表或表中的列缩小的作用域字段。

2. 授权表权限列

授权表中还包含权限列。这些列指明了哪些权限被在作用域列中指定的用户所拥有。MySQL 支持的权限显示在以下列表中。该列表使用用于 GRANT 语句的权限名。对于大部分来说，在 user、db 和 host 表中权限列的名字与在第 11 章利用 GRANT 语句连接中所讨论的权限名显然是相同的。例如， Select_priv 列对应于 SELECT 权限。

3. 数据库和表的权限

以下权限适用于数据库和表的操作：

- ALTER 允许使用 ALTER TABLE 语句。这实际上是一个简单的一级权限，您必须根据打算用该表做什么以拥有其他的权限。
- CREATE 允许创建数据库和表。这个权限不允许您创建索引。
- DELETE 允许从表中删除已有记录。
- DROP 允许删除数据库和表。不允许删除索引。
- INDEX 允许创建或删除表的索引。
- INSERT 允许在表中插入新的记录。
- REFERENCES 未使用。
- SELECT 允许用 SELECT 语句从表中检索数据。该权限对于不含表的 SELECT 语句是不必要的，如 SELECT NOW() 或 SELECT 4/2。
- UPDATE 允许修改表中的已有记录。

4. 管理权限

下列权限适用于控制服务器操作或用户授权能力的管理操作：

- FILE 允许服务器在服务器主机上读写文件。该权限如果没有正当的原因则不应授予，如12.2.3节“ 授权表应避免的风险 ”中所述，它是危险的。服务器采取某些预防措施限定该权限在特定范围内使用。您只可以读世人都可读的文件，因此不需要考虑任何方

式的保护。正在进行写操作的文件不允许是已经存在的。这可防止服务器覆盖重要的文件（如 /etc/passwd）或属于其他用户的数据库文件。例如，如果不施加该约束，您可能会替换全部mysql 数据库中的授权表的内容。

如果您授予 FILE 权限，就不能作为 UNIX 的 root 用户运行服务器，因为 root 能在文件系统的任何地方创建新的文件。如果作为未授权用户运行服务器，则服务器只在该用户可访问到的目录中创建文件。

- GRANT 允许将您所拥有的权限授予其他用户，其中包括 GRANT 权限。
- PROCESS 允许通过使用 SHOW PROCESSLIST 语句或 mysqladmin processlist 命令查询有关正在服务器中运行的线程信息。该权限还允许利用 KILL 语句或 mysqladmin kill 命令取消线程。您始终可以查看或取消自己的线程。 PROCESS 权限授予您对任何线程进行这些操作的能力。
- RELOAD 允许执行多种服务器管理操作。可发布 FLUSH SQL 语句，还可执行 mysqladmin 的命令 reload、refresh、flush-hosts、flush-logs、flush-privileges 和 flush-tables。

  通常这不是有危险的权限，尽管它实际上是管理权限。
- SHUTDOWN 允许利用 mysqladmin shutdown 关闭服务器。

在 user、db 和 host 表中，每个权限都作为单独的列给出。这些列都是用 ENUM（“N”、“Y”）类型声明的，因此所有权限的缺省值为 “N” （off）。tables_priv 和 columns_priv 表的权限由 SET 语句表示，它允许权限以使用单个列的任何组合形式指定。这两个表比其他三个表新，这就是为什么它们使用了更有效的表示方法的原因（user、db 和 host 有可能在将来被改造，也通过 SET 来表示权限）。

table_priv 表的table_priv 列定义如下：

```
SET('Select','Insert','Update','Delete','Create','Drop',
    'Grant','References','Index','Alter')
```

columns_priv 表的 Column_priv 列定义如下：

```
SET('Select','Insert','Update','References')
```

列权限比表权限少得多，因为在列级中有意义的操作很少。例如，您可以创建一个表，但不能创建孤立的列。

user 表中包含对某些权限的列，这些权限不出现在任何其他授权表中，如 File_priv、Process_priv、Reload_priv 和 Shutdown_priv 表。这些权限适用于执行与任何特定数据库或与表无关的服务器的操作。例如，使用户根据当前数据库关闭服务器是无意义的。

12.2.2 服务器如何控制客户机的访问

在使用 MySQL 时，客户机访问控制有两个阶段。第一阶段发生在连接服务器时。服务器查找 user 表看看是否能够找到与您的名字、正在连接的主机以及所提供的口令相匹配的项。如果不匹配，则不能连接。若匹配，则建立连接，然后继续进行第二阶段。在此阶段中，对于您发布的每个查询，服务器都会检查授权表以查看您是否具有充足的权限来执行该查询。第二阶段继续，直到关于该服务器的会话结束为止。

本节详细讨论 MySQL 用来将授权表项与输入的客户机连接请求和与查询相匹配的规则。这包括在授权表作用域列中合法的值的类型、在授权表中权限信息组合的方法，以及表项检

查的次序。

1. 作用域列的内容

某些作用域列需要直接量，但大多数列允许通配符或其他特殊的值。

- Host　该列值可以是一个主机名或一个 IP 号。localhost 值的含义是本地主机，但仅当您实际使用 localhost 主机值而不是使用该主机名连接时才进行匹配。假定本地主机名为 pit-viper.snake.net，且在 user 表中有两个项，一个带有 Host 值或 localhost，而另一个带有 pit-viper.snake.net 值。带有 localhost 值的项仅当您连接到 localhost 时才进行匹配。另一个项仅当您连接到 pit-viper.snake.net 时才匹配。如果想让用户能够以任意一种方式进行连接，在 user 表中需要两个项。还可以用通配符指定 Host 的值。SQL 的模式字符‘ % ’和‘ _ ’也可以使用，并与查询中的 LIKE 操作符有相同的含义。SQL 模式（不允许 REGEX 模式）字符可用于名字和 IP 号。例如，%.wise.edu 将与 wise.edu 域中的任何主机相配。同样，192.168.% 与192.168 类 B 子网中的任何主机相配，因而 192.168.3.% 与192.168.3 类 C 子网中的任何主机相配。

  值‘ % ’总是与任何主机相配，用于允许某个用户从任何地方进行连接。空白的 Host 的值与‘ % ’含义相同（例外：在 db 表中，空白的 Host 值含义是“检查 host 表的进一步的信息”。

  自 MySQL 3.23 以来，还可以利用表明网络号的二进制位数的网络掩码指定 IP 号。例如，192.168.128.0/17 指定 17 位的网络号并用 IP 地址的前 17 位中的 192.168.128 与所有主机相配。
- User　用户名必须或者是直接量或者是空白的（空）。空白值与任何用户相配。作为 User 值的 % 不意味着空白，相反，它与带有 % 的直接名字的用户相配，这可能不是您想要的东西。

  当一个输入的连接对应 user 表进行检查并且相配的项包含一个空白的 User 值时，客户机被认为是匿名用户。
- Password　口令值或者是空白的（空）或者是非空的，但不允许通配符。空白的口令并不是说与任何口令相配。它的意思是用户不必指定口令。口令以加密值而不是以直接值被存储。如果在 Password 列存储了一个直接的口令，则该用户将不能连接！GRANT 语句和 mysqladmin password 命令自动对口令进行加密，但是，如果使用像 INSERT、REPLACE、UPDATE 或 SET PASSWORD 这样的语句，则必须用 PASSWORD(“NEW_PASSWORD”) 而不是简单地用“new_password”来指定口令。
- Db　在 columns_priv 和 tables_priv 表中，Db 值必须是直接的数据库名，不允许模式和空白名。在 dn 和 host 表中，Db 值可以用直接量指定，或通过使用指定通配符的 SQL 模式字符‘ % ’或‘ _ ’来指定。% 值或空白将与任何数据库相配。
- Table_name、Column_name　这些列中的值必须是直接量的表名或列名，模式值和空白是不允许的。

有些作用域列由服务器视为区分大小写的，其他的则不是。这些规则已在表 12-3 中总结。特别注意，Table_name 值始终是区分大小写的，即使查询中的表名处理是根据服务器运行的文件系统，其大小写敏感性也是如此（在 UNIX 中区分大小写，而在 Windows 中不区分大小写）。

表12-3 授权表作用域列大小写敏感性

列	是否区分大小写	列	是否区分大小写
Host	No	Db	Yes
User	Yes	Table_name	Yes
Password	Yes	Column_name	No

2. 查询访问检查

每当您发布查询时，服务器都要检查您是否有足够的权限来执行查询。它是通过依次检查 user、db、table_priv 和 columns_priv 表来进行的，直到它确定您有适当的访问权或徒然地搜索了所有的表为止。更准确地说：

1) 在开头进行连接以查看所拥有的全局权限时，服务器检查相匹配的 user 表项。如果有全局权限且满足该查询，则服务器执行查询。

2) 如果全局权限不满足，则服务器在 db 表中寻找一个项，并将该项的权限增加到您的全局权限中。若结果满足该查询，则服务器执行查询。

3) 如果全局和数据库级权限的组合不够，服务器将保持不断地查找，首先在 tables_priv 表中，然后在 columns_priv 表中查找。

4) 如果在所有表检查后发现您没有许可权，服务器将拒绝执行查询。

口令是怎样存储在 user 表中的

口令在 user 表中以加密串的形式出现，因此您不知道用户的口令是什么，即使已经拥有对该表的访问权也是如此。通常，似乎认为 PASSWORD（）函数执行与 UNIX 口令所使用的类型相同的加密方法，其实并不是这样的。

当两种加密方法是单向的且不可逆时它们是相同的，但是， MySQL 不使用与 UNIX 相同的加密算法。这意味着，即使您使用 UNIX 口令作为 MySQL 的口令，也别希望加密的口令串相配。如果想对应用程序使用 UNIX 的加密方法，应使用 CRYPT() 函数而非 PASSWORD()。

在逻辑条件下，授权表由服务器按以下形式使用：

```
user OR db OR tables_priv OR columns_priv
```

您可能正在奇怪，为什么上面的描述只涉及到五个授权表中的四个。正击中要害，服务器实际检查访问许可权的方法如下：

```
user OR (db AND host) OR tables_priv OR columns_priv
```

笔者先给出了较简单的表达式，因为 host 表不受 GRANT 或 REVOKE 语句的影响。如果一直保持使用 GRANT 和 REVOKE 来管理用户权限，将不需要考虑 host 表。但下面还是给出了对该表的讨论，以便您有所了解：

- 当服务器检查数据库级的权限时，它查看客户机的 db 表的项。如果 Host 列值是空白，其含义是“检查 host 表查找哪些服务器可访问该数据库”。
- 服务器利用与来自 db 表项相同的 Db 列为项查看 host 表。如果 host 表项与客户机主机不相配，就不授予数据库级权限。如果这些项都与客户机正在连接的主机的 Host 列值相配，则 db 表的项和 host 表的项进行组合以产生客户机的数据库级权限。

但是，权限用逻辑 AND 进行组合，其意思是，除非客户机同时出现在两个项中，

否则它没有指定的权限。这样，可以在此 db 表项中授予一组基本的权限，然后用 host 表项有选择地对特定的主机禁止使用这些权限。例如，允许从域的所有主机中访问某个数据库，但是对于定位在非安全范围内的主机禁止使用数据库权限。

以上描述无疑要进行访问检查，这听起来好像是相当复杂的过程，尤其是考虑到服务器要为客户机发布的每一个单独的查询检查权限时。但是，该过程相当地快，因为服务器不对每个查询实际检查授权表的信息。相反，在服务器启动时，它将该表的内容读到内存中，然后用内存中（in-memory）的拷贝来检验查询。这对于访问检查操作性能产生了推进，但有相当重要的负面效应：如果您直接修改了授权表的内容，则服务器将不会注意到权限的改变。

例如，如果通过用 INSERT 语句增加一个新记录到 user 表中来增加了一个用户，在该项中指定的该用户将不能连接到该服务器。这是迷惑新管理员的问题（有时比较有经验的管理员也是如此），但是解决方案很简单：告诉服务器在您修改授权表后重新加载其内容即可。可以通过发布 FLUSH PRIVILEGES 语句或通过执行 mysqladmin flush-privileges（或 mysqladmin reload，如果是不支持 flush-privileges 的旧版本的话）来进行。

3. 作用域列匹配顺序

MySQL 服务器以一种特定的方式对授权表的项进行排序，然后试着通过按顺序查看所有的项来匹配输入的连接。所发现的第一个匹配确定被使用的项。重要的是要理解 MySQL 使用的排序顺序，尤其是对于 user 表的排序。这好象是在为难试图理解 MySQL 安全性的许多人们。

当服务器读取 user 表的内容时，它根据 Host 和 User 列的值对项进行排序。Host 值是域值（带有相同 Host 值的项被一起存放，然后按照 User 值进行排序）。但是，排序不是词典式的，或不完全是这样的。要记住的原则是直接值比模式值优先。这意味着如果正从 boa.snake.net 中进行连接并且带有 boa.snake.net 和 %.snake.net 的 Host 值，则直接值将是首选的。同样，%.snake.net 优先于 %.net，如此类推，%.net 优先于 %。IP 号也以这种方法进行匹配。对于从带有 192.168.3.14 的 IP 号的主机中连接的客户机，具有 192.168.3.4、192.168.3.%、192.168.%、192.% 和 % 的 Host 值将按此顺序匹配。

12.2.3 授权表应避免的风险

本节讨论在授权时应遵守的某些预防措施以及维护人员作出轻率选择所冒的风险。通常，给用户授超级用户权限应保守。也就是说，不要在 user 表中放置权限。取而代之的是使用其他的授权表来限制用户对特定数据库、表或列的权限。 user 表中的权限可令用户影响服务器的操作或访问任何数据库中的任一表。

不要对 mysql 数据库授权。该数据库连接授权表，在该数据库上具有权限的用户可以修改这些表来获得任何其他数据库的权限。允许用户修改 mysql 数据库表的授权还能给该用户授予一个全局 GRANT 权限。如果该用户可直接修改表，就相当于可以发布任何的 GRANT 语句。

FILE 特权相当危险,不要轻易授权。以下是一个 FILE 特权用户的例子：

```
CREATE TABLE etc_passwd (pwd_entry TEXT);
LOAD DATA INFILE "/etc/passwd" INTO TABLE etc_passwd;
SELECT * FROM etc_passwd;
```

发布这些语句后，该用户拥有您的口令文件的内容。事实上，在服务器中，任何公用的

可读文件的内容都可以通过网络由任何有 FILE 权限的用户访问。

FILE 权限也可以用来对系统中的、不能用足够的限制文件许可权来建立的数据库进行折衷。这就是为什么应该将数据目录的内容设置成为只能由服务器可读的原因。如果与数据库表相对应的文件是全球范围内可读的，则不仅有该服务器主机账号的用户可读取它们，而且有 FILE 权限的任何客户机也都可通过网络连接并读取它们！下面的过程说明如何进行此项操作：

■ 创建有 LONGBLOB 列的表：

```
USE test
CREATE TABLE tmp (b LONGBLOB)
```

■ 用此表读取您要窃取的表的每个文件的内容，然后将该表的内容写到您自己的数据库文件中：

```
LOAD DATA INFILE "./other_db/x.frm" INTO TABLE tmp
    FIELDS ESCAPED BY "" LINES TERMINATED BY ""
SELECT * FROM tmp INTO OUTFILE "y.frm"
    FIELDS ESCAPED BY "" LINES TERMINATED BY ""
DELETE FROM tmp
LOAD DATA INFILE "./other_db/x.ISD" INTO TABLE tmp
    FIELDS ESCAPED BY "" LINES TERMINATED BY ""
SELECT * FROM tmp INTO OUTFILE "y.ISD"
    FIELDS ESCAPED BY "" LINES TERMINATED BY ""
DELETE FROM tmp
LOAD DATA INFILE "./other_db/x.ISM" INTO TABLE tmp
    FIELDS ESCAPED BY "" LINES TERMINATED BY ""
SELECT * FROM tmp INTO OUTFILE "y.ISM"
    FIELDS ESCAPED BY "" LINES TERMINATED BY ""
```

■ 现在，您有了一个新表 y，它包含 other_db.x 的内容，您可以对它进行完全访问。

为了避免某人用同样的方式攻击您，应根据“内部安全性：安全数据目录访问”的指导设置数据目录的许可权。当启动服务器以限制用户利用 SHOW DATABASE 和 SHOW TABLES对他们不能访问的数据库进行访问时，还可使用 -skip-show-database 选项。这将有助于防止用户查找不应访问的数据库和表。

ALTER 权限可能以您没有设想的任何方法被使用。假定您要 user1 能访问 table1 但不能访问 table2。但带有 ALTER 权限的用户可能会通过使用 ALTER TABLE 将 table2 重新命名为 table1 来打乱您的这一设想。

要小心使用 GRANT 权限。如果两个用户有不同的权限但都具有 GRANT权限，则他们能使互相的访问权更强大。

12.2.4 不用 GRANT 建立用户

如果是 MySQL 3.22.11 以前的版本，则不能用 GRANT（或 REVOKE）语句建立用户和访问权限，但是可以直接修改授权表的内容。如果理解 GRANT 语句是怎样修改授权语句的，这个问题就很容易理解。可以通过手工发布 INSERT 语句自己来进行（INSERT 语句很难正确输入，但这是另外的问题。事实上这个问题就是为什么 GRANT 语句设计得如此易于使用的原因）。

当发布 GRANT 语句时，指定了用户名和主机号，可能还指定了口令。 user 表项对该用

户建立，这些值记录在该项的 User、Host 和 Password 列中。需要仔细考虑的一件事是，GRANT 语句对口令进行加密，而 INSERT 则不能，它需要用 PASSWORD() 函数对 INSERT 语句中的口令进行加密。

如果已经指定了数据库级的权限，用户名和主机名将记录在 db 表项的 User 和 Host 列中。已授权的数据库记录在 Db 列中，所授予的权限记录在权限列中。

对于表级和列级权限，效果是类似的。在 tables_priv 和 columns_priv 表中创建项来记录用户名、主机名和数据库以及需要的表或者表和列。授予的权限记录在权限列中。

如果还记得上述讨论，即使不用 GRANT 语句也能够进行 GRANT 的工作。请记住，当直接修改授权表时，需要告诉服务器重新加载授权表，否则它将不会留意这些变化。可以通过执行 mysqladmin flush-privileges 或 mysqladmin reload 命令施加重新加载操作。如果忘记这步操作，服务器将不按您的设想进行工作。

以下 grant 语句创建有全部权限的超级用户，其中包括向其他用户授权的能力：

```
GRANT ALL ON *.* TO ethel@localhost IDENTIFIED BY "coffee"
    WITH GRANT OPTION
```

该语句将创建 user 表的ethel@localhost 的项，并开启所有权限，因为超级（全局）用户将存储在那里。用 INSERT 进行同样工作的语句是：

```
INSERT INTO user VALUES("localhost","ethel",PASSWORD("coffee"),
    "Y","Y","Y","Y","Y","Y","Y","Y","Y","Y","Y","Y","Y","Y")
```

这是一个很难看的 INSERT 语句！您甚至会发现它在您的 MySQL 版本上不工作。授权表的结构可能已经改变，在 user 表中可能不会有 14 个权限列。此时，应使用 SHOW COLUMNS 来查找每个授权表包含些什么权限列，然后调整 INSERT 语句。

下列 GRANT 语句还创建具有超级用户状态的用户，但是只对单个的权限：

```
GRANT RELOAD ON *.* TO flush@localhost IDENTIFIED BY "flushpass"
```

您可能还记得我们曾在第11章建立 flush 用户时使用过该语句。本例中的 INSERT 语句比前一个更简单，可以很容易地列出列名并指定唯一的一个权限列。所有其他的列都将设置为缺省值“ N ”：

```
INSERT INTO user (Host,User,Password,Reload_priv)
    VALUES("localhost","flush",PASSWORD("flushpass"),"Y")
```

数据库级的权限是用 ON db_name.* 子句而不是用 ON *.* 授予的：

```
GRANT ALL ON samp_db.* TO boris@localhost IDENTIFIED BY "ruby"
```

这些权限都不是全局的，因此不能存储在 user 表中。我们仍需要在 user 表中创建一个项（使得用户可以连接），但我们还需要创建一个 db 项来记录数据库级的权限：

```
INSERT INTO user (Host,User,Password)
    VALUES("localhost","boris",PASSWORD("ruby"))
INSERT INTO db VALUES("localhost","samp_db","boris",
    "Y","Y","Y","Y","Y","Y","N","Y","Y","Y")
```

“ N ” 列是针对 GRANT 权限的，对于在末尾带 WITH GRANT OPTION 的数据库级的 GRANT 语句，应该将该列设置为“ Y ”。

为了设置表级或列级权限，对 tables_priv 或 columns_priv 表使用 INSERT 语句。当然，如果没有 GRANT 语句，也将没有这些表，因为它们都是在同一时间出现在 MySQL 版本中的。如果有这些表但由于某种原因想手工操纵它们，那么您不能使用单个的列使权限有效。

您或者设置 tables_priv.Table_priv 或者设置 columns_priv.Column_priv 为一个 SET 值，该 SET 值由您允许的权限组成。例如，为了允许对某个表的 SELECT 和 INSERT 权限，应在相对的 tables_priv 项中设置 Table_priv 列为“ Select, Insert ”值。

如果要想修改某个用户的权限（该用户的 MySQL 账号已经存在），应使用 UPDATE 而非 INSERT。无论是增加还是取消权限都是这样。为了删除一个完整的用户，使用 DELETE 从该用户所在的每个授权表中删除其项。

如果要想避免发布查询来直接修改授权表，应看一看与 MySQL 分发包一起的 mysqlaccess 和 mysql_setpermissions 脚本。

一个权限难题，第二部分

在第11章的“一个权限难题，第一部分”中，我们看到了一种授权情况，在这种情况中权限没有预想的效果。我们来简单地叙述一下问题，经常与新的 MySQL 安装一起出现的一个问题是管理员想要为用户增加一项，使用户们可从几个主机中进行连接。这样做最直接的方法是使用包含‘ % ’通配符的主机名说明符，因此管理员用类似以下的语句创建一个用户：

```
GRANT ALL ON samp_db.* TO fred@%.snake.net IDENTIFIED BY "cocoa"
```

fred 碰巧有一个服务器主机的账号，因此他试图从这里进行连接：

```
% mysql -u fred -pcocoa samp_db
ERROR 1045: Access denied for user: 'fred@localhost'
(Using password: YES)
```

为什么会这样？为了继续理解，您必须同时考虑 mysql_install_db 是怎样建立初始授权表和服务器怎样使用 user 表的项匹配客户机连接的。当通过运行 mysql_install_db 对数据库初始化时，它创建带有如下 Host 和 User 值的 user 表的项：

```
+--------------------+------+
| Host               | User |
+--------------------+------+
| localhost          | root |
| pit-viper.snake.net | root |
| localhost          |      |
| pit-viper.snake.net |      |
+--------------------+------+
```

前两项通过指定 localhost 或主机名允许 root连接到本地主机。后两项允许来自本地主机的匿名连接。当为 fred 增加此项时，则 user 表将包含这些项：

```
+--------------------+------+
| Host               | User |
+--------------------+------+
| localhost          | root |
| pit-viper.snake.net | root |
| localhost          |      |
| pit-viper.snake.net |      |
| %.snake.net        | fred |
+--------------------+------+
```

当服务器启动时，它读取这些项并存储它们（先主机，然后是主机的用户），其顺序是比较详细的值在前，不详细的在后：

```
+--------------------+------+
| Host               | User |
```

(续)

```
+--------------------+------+
| localhost          | root |
| localhost          |      |
| pit-viper.snake.net | root |
| pit-viper.snake.net |      |
| %.snake.net        | fred |
+--------------------+------+
```

带 localhost 的两项一起存储，有 root 的那个项放在前，因为这项比空白的更具体、更详细。带有 pit-viper.snake.net 的项以类似的方法一起存储。这些项都是不带任何通配符的直接 Host 值，因此它们都优先于 fred 的项进行存储，而 fred 项使用了一个通配符。特别地，匿名用户项在存储顺序上优先于 fred 的项。

此结果是，当 fred 试图从本地主机上进行连接时，带有空白用户名的项匹配先于在 Host 列中包含 %.snake.net 的项。该项的口令是空白的，因为缺省的匿名用户没有口令。由于 fred 在连接时指定了口令，则产生一个错误匹配和错误连接。

这里要记住的事情是，尽管使用通配符指定用户可连接的主机这种方式非常方便，但只要您在user 表中保留匿名用户，就可能会出现从本地主机中连接的问题。

通常，笔者建议删除匿名用户项。这将使您的工作更轻松一些：

```
mysql> DELETE FROM user WHERE User = "";
```

如果想更彻底的话，可将其他授权表的匿名用户项也删除。有 User 列的表是 db、tables_priv 和 columns_priv。

本节给出的难题是一种特殊的情况，但包含了更为普遍的内容。如果某个用户的权限不按所希望的方式工作，则检查授权表，看看是否有某些项包含了比所讨论的该用户的项更详细的 Host 值，以及试图通过那个用户匹配连接的 Host 值。如果是这样的，问题的原因可能就是它。您可能需要使该用户的项更详细（或增加另一个项来包括更详细的情况）。