保障 MySQL 安全的 14 个最佳方法

MySQL 数据库一贯以高性能、高可性和易用性著称，它已经成为世界上最流行的开源数据库。大量的个人、WEB 开发者、大型公司等都在其网站、关键系统、软件包中广泛使用 MySQL 数据库。

通常，许多企业在部署一种产品时，安全性常常得不到应有的重视。企业最关心的是使其可以尽快地运行，企业由此也可以尽快赢利。

但有的企业在安装 MySQL 时用的是默认选项，由此造成其数据不安全，且服务器也面临被入侵的风险，并有可能在短时间内就出现性能问题。下面将提供保障 MySQL 安全的最佳方法。

1、避免从互联网访问 MySQL 数据库，确保特定主机才拥有访问特权

直接通过本地网络之外的计算机改变生产环境中的数据库是异常危险的。有时，管理员会打开主机对数据库的访问：

```
> GRANT ALL ON *.* TO 'root'@'%';
```

这其实是完全放开了对 root 的访问。所以，把重要的操作限制给特定主机非常重要：

```
> GRANT ALL ON *.* TO 'root'@'localhost';
> GRANT ALL ON *.* TO 'root'@'myip.athome';
> FLUSH PRIVILEGES;
```

此时，你仍有完全的访问，但只有指定的 IP(不管其是否静态)可以访问。

2、定期备份数据库

任何系统都有可能发生灾难。服务器、MySQL 也会崩溃，也有可能遭受入侵，数据有可能被删除。只有为最糟糕的情况做好了充分的准备，才能够在事后快速地从灾难中恢复。企业最好把备份过程作为服务器的一项日常工作。

3、禁用或限制远程访问

前面说过，如果使用了远程访问，要确保只有定义的主机才可以访问服务器。这一般是通过 TCP wrappers、iptables 或任何其它的防火墙软件或硬件实现的。

为限制打开网络 socket，管理员应当在 my.cnf 或 my.ini 的[mysqld]部分增加下面的参数：

```
skip-networking
```

这些文件位于 windows 的 C:\Program Files\MySQL\MySQL Server 5.1 文件夹中，或在 Linux 中，my.cnf 位于/etc/，或位于/etc/mysql/。这行命令在 MySQL 启动期间，禁用了网络连接的初始化。请注意，在这里仍可以建立与 MySQL 服务器的本地连接。

另一个可行的方案是，强迫 MySQL 仅监听本机，方法是在 my.cnf 的[mysqld]部分增加下面一行：

```
bind-address=127.0.0.1
```

如果企业的用户从自己的机器连接到服务器或安装到另一台机器上的 web 服务器，你可能不太愿意禁用网络访问。此时，不妨考虑下面的有限许可访问：

mysql> GRANT SELECT, INSERT ON mydb.* TO 'someuser'@'somehost';

这里，你要把 someuser 换成用户名，把 somehost 换成相应的主机。

4、设置 root 用户的口令并改变其登录名

在 linux 中，root 用户拥有对所有数据库的完全访问权。因而，在 Linux 的安装过程中，一定要设置 root 口令。当然，要改变默认的空口令，其方法如下：

```
    Access MySQL 控制台：$ mysql -u root -p
```

在 MySQL 控制台中执行：

```
    > SET PASSWORD FOR 'root'@'localhost' = PASSWORD('new_
password');
```

在实际操作中，只需将上面一行的 new_password 换成实际的口令即可。

在 Linux 控制台中更改 root 口令的另一种方法是使用 mysqladmin 工具：

```
    $ mysqladmin -u root password new_password
```

此时，也是将上面一行的 new_password 换成实际的口令即可。

当然，这是需要使用强口令来避免强力攻击。

为了更有效地改进 root 用户的安全性，另一种好方法是为其改名。为此，你必须更新表用户中的 mySQL 数据库。在 MySQL 控制台中进行操作：

```
    > USE mysql;

    > UPDATE user SET user="another_username" WHERE user="
root";

    > FLUSH PRIVILEGES;
```

然后，通过 Linux 访问 MySQL 控制台就要使用新用户名了：

```
    $ mysql -u another_username -p
```

5、移除测试(test)数据库

在默认安装的 MySQL 中，匿名用户可以访问 test 数据库。我们可以移除任何无用的数据库，以避免在不可预料的情况下访问了数据库。因而，在 MySQL 控制台中，执行：

```
> DROP DATABASE test;
```

6、禁用 LOCAL INFILE

另一项改变是禁用”LOAD DATA LOCAL INFILE”命令，这有助于防止非授权用户访问本地文件。在 PHP 应用程序中发现有新的 SQL 注入漏洞时，这样做尤其重要。

此外，在某些情况下，LOCAL INFILE 命令可被用于访问操作系统上的其它文件(如/etc/passwd)，应使用下现的命令：

```
mysql> LOAD DATA LOCAL INFILE '/etc/passwd' INTO TABLE
table1;
```

更简单的方法是：

```
mysql> SELECT load_file("/etc/passwd");
```

为禁用 LOCAL INFILE 命令，应当在 MySQL 配置文件的[mysqld]部分增加下面的参数：

```
set-variable=local-infile=0
```

7、移除匿名账户和废弃的账户

有些 MySQL 数据库的匿名用户的口令为空。因而，任何人都可以连接到这些数据库。可以用下面的命令进行检查：

```
mysql> select * from mysql.user where user="";
```

在安全的系统中，不会返回什么信息。另一种方法是：

```
mysql> SHOW GRANTS FOR ''@'localhost';
mysql> SHOW GRANTS FOR ''@'myhost';
```

如果 grants 存在，那么任何人都可以访问数据库，至少可以使用默认的数据库“test”。其检查方法如下：

```
shell> mysql -u blablabla
```

如果要移除账户，则执行命令：

```
mysql> DROP USER "";
```

从 MySQL 的 5.0 版开始支持 DROP USER 命令。如果你使用的老版本的 MySQL，你可以像下面这样移除账户：

```
mysql> use mysql;
mysql> DELETE FROM user WHERE user="";
mysql> flush privileges;
```

8、降低系统特权

常见的数据库安全建议都有“降低给各方的特权”这一说法。对于 MySQL 也是如此。一般情况下，开发人员会使用最大的许可，不像安全管理一样考虑许可原则，而这样做会将数据库暴露在巨大的风险中。

为保护数据库，务必保证真正存储 MySQL 数据库的文件目录是由”mysql”用户和” mysql”组所拥有的。

```
shell>ls -l /var/lib/mysql
```

此外，还要确保仅有用户” mysql”和 root 用户可以访问/var/lib/mysql 目录。

Mysql 的二进制文件存在于/usr/bin/目录中，它应当由 root 用户或特定的”mysql”用户所拥有。对这些文件，其它用户不应当拥有“写”的访问权：

```
shell>ls -l /usr/bin/my*
```

9、降低用户的数据库特权

有些应用程序是通过一个特定数据库表的用户名和口令连接到 MySQL 的，安全人员不应当给予这个用户完全的访问权。

如果攻击者获得了这个拥有完全访问权的用户，他也就拥有了所有的数据库。查看一个用户许可的方法是在 MySQL 控制台中使用命令 SHOW GRANT

```
>SHOW GRANTS FOR 'user'@'localhost';
```

为定义用户的访问权，使用 GRANT 命令。在下面的例子中，user1 仅能从 dianshang 数据库的 billing 表中选择：

```
    > GRANT SELECT ON billing.dianshang TO 'user1'@'localh
ost';

    > FLUSH PRIVILEGES;
```

如此一来，user1 用户就无法改变数据库中这个表和其它表的任何数据。

另一方面，如果你要从一个用户移除访问权，就应使用一个与 GRANT 命令类似的 REVOKE 命令：

```
    > REVOKE SELECT ON billing.ecommerce FROM 'user1'@'loc
alhost';

    > FLUSH PRIVILEGES;
```

10、移除和禁用。mysql_history 文件

在用户访问 MySQL 控制台时，所有的命令历史都被记录在 ~/.mysql_history 中。如果攻击者访问这个文件，他就可以知道数据库的结构。

```
    $ cat ~/.mysql_history
```

为了移除和禁用这个文件，应将日志发送到/dev/null。

```
    $export MYSQL_HISTFILE=/dev/null
```

上述命令使所有的日志文件都定向到/dev/null，你应当从 home 文件夹移除。mysql_history：$ rm ~/.mysql_history，并创建一个到/dev/null 的符号链接。

11、安全补丁

务必保持数据库为最新版本。因为攻击者可以利用上一个版本的已知漏洞来访问企业的数据库。

12、启用日志

如果你的数据库服务器并不执行任何查询，建议你启用跟踪记录，你可以通过在/etc/my.cnf 文件的[Mysql]部分添加：log =/var/log/mylogfile。

对于生产环境中任务繁重的 MySQL 数据库，因为这会引起服务器的高昂成本。

此外，还要保证只有 root 和 mysql 可以访问这些日志文件。

错误日志

务必确保只有 root 和 mysql 可以访问 hostname.err 日志文件。该文件存放在 mysql 数据历史中。该文件包含着非常敏感的信息，如口令、地址、表名、存储过程名、代码等，它可被用于信息收集，并且在某些情况下，还可以向攻击者提供利用数据库漏洞的信息。攻击者还可以知道安装数据库的机器或内部的数据。

MySQL 日志

确保只有 root 和 mysql 可以访问 logfileXY 日志文件，此文件存放在 mysql 的历史目录中。

13、改变 root 目录

Unix 操作系统中的 chroot 可以改变当前正在运行的进程及其子进程的 root 目录。重新获得另一个目录 root 权限的程序无法访问或命名此目录之外的文件，此目录被称为“chroot 监狱”。

通过利用 chroot 环境，你可以限制 MySQL 进程及其子进程的写操作，增加服务器的安全性。

你要保证 chroot 环境的一个专用目录，如/chroot/mysql。此外，为了方便利用数据库的管理工具，你可以在 MySQL 配置文件的[client]部分改变下面的参数：

```
[client]
socket = /chroot/mysql/tmp/mysql.sock
```

14、禁用 LOCAL INFILE 命令

LOAD DATA LOCAL INFILE 可以从文件系统中读取文件，并显示在屏幕中或保存在数据库中。如果攻击者能够从应用程序找到 SQL 注入漏洞，这个命令就相当危险了。下面的命令可以从 MySQL 控制台进行操作：

> SELECT LOAD_FILE("/etc/passwd");

该命令列示了所有的用户。解决此问题的最佳方法是在 MySQL 配置中禁用它，在 CentOS 中找到/etc/my.cnf 或在 Ubuntu 中找到/etc/mysql/my.cnf，在[mysqld]部分增加下面一行：set-variable=local-infile=0。搞定。

当然，唇亡齿寒，保护服务器的安全对于保障 MySQL 数据库的安全也是至关重要的。服务器的安全对于数据库来说可谓生死攸关。