# MySQL 最佳安全配置

# 指导手册

| | | | |
|---|---|---|---|
| ■ 文档编号 | RP-2016-01 | ■ 密级 | 完全公开 |
| ■ 版本编号 | V1.0 | ■ 日期 | 2017.12.14 |

# 目录

# 一. MySQL 数据库

MySQL 数据库由于其快速性能，高可靠性和易用性而成为世界上最流行的开源数据库。它被广泛应用在各个行业。从个人到世界级的公司及行业领导者都广泛使用 MySQL 数据库。例如 Yahoo, Alcatel-Lucent, Google, Nokia, YouTube 等。

像大多数产品一样，MySQL 安装极其简单。通常，在安装这种产品时安全性不是主要的考虑因素。由于对安全的不在意最终造成在 2016 年底到 2017 年初，出现多个黑客组织对 MySQL 数据库实施大规模入侵。最终累计造成经济损失达数千万。

本文旨在帮助用户在快速完成 MySQL 部署后，只需要很少的时间就可以完成 MySQL 的安全配置。这些安全配置可以帮助您抵御大多数常用形式的黑客攻击行为。我们选取现今最流行的 MySQL5.7 版本+Linux 系统作为安全配置演示对象。所有安全配置项都会向大家展示检测方式和修补方法。以帮助您快速判断是否存在配置问题并进行修补。

# 二. 安全配置项

## 2.1 操作系统级别安全配置

本节包含和 MySQL 数据库所运行的操作系统密切相关的安全配置项

### 2.1.1 确保数据文件在非系统分区

建议理由：

操作系统上有明确的系统分区和非系统分区。如果数据文件在系统分区，会提高整个系统因为磁盘空间用尽发生拒绝服务的几率。

检查手段：

1.进入数据库执行下列语句

show variables where variable_name = 'datadir';

2. df –h <上面的地址返回值>

如果发现结果中存在/ /var /usr 说明数据文件在系统分区建议换区

修复建议：迁移走数据文件

1. 停止数据库服务

service MySQL stop

2. 拷贝数据文件到非系统分区

cp –rp <老地址> <新地址>

3. 修改 MySQL 配置文件中 datadir 的值成新地址

4. 启动 MySQL 数据库

service MySQL start

## 2.1.2 确保 MySQL 操作系统账号权限最小化

建议理由：

MySQL 在操作系统上的账号权限最小化有助于减小 MySQL 数据库漏洞造成的影响。防止黑客利用 MySQL 漏洞入侵操作系统，造成更大损失。

检查手段：

假设 MySQL 账号为 MySQL

ps -ef | egrep "^MySQL.*$"

如果没有返回行，则权限存在问题

修复建议：创建一个仅用于运行 MySQL 和直接相关进程的用户。

## 2.1.3 禁止 MySQL 链接历史记录

建议理由：

MySQL 会把客户端登陆的交互执行记录保存在.MySQL_history 文件中。该记录有可能会暴露登陆过程中的敏感信息。建议删除该记录

检查手段：

检查.MySQL_history 文件是否存在（默认在 home 下）

find /home -name ".MySQL_history"

如果有返回行说明存在建议删除。

修复建议：删除并禁止继续记录

1. 如果存在.MySQL_history，则删除

2. 创建链接，防止.MySQL_history 再次生成

ln -s /dev/null $HOME/.MySQL_history

或采用修改 MySQL_history 的环境变量让他的值等于 /dev/null

## 2.1.4 禁止 MySQL_PWD 的使用

建议理由：

MySQL_PWD 是一种用于存储 MySQL 密码的环境变量。而且是以明文形式存储，带来了非常大的安全隐患。

检查手段：

检查 MySQL_PWD 环境变量是否存在于某个进程中

grep MySQL_PWD /proc/*/environ

如果有返回行说明那个进程使用了 MySQL_PWD 环境变量。

修复建议：对使用 MySQL_PWD 环境变量变量的脚本和进程，建议不在使用该环境变量。

## 2.1.5 禁止 MySQL 操作系统账号登陆

建议理由：

MySQL 的操作做系统账号在安装完数据库后，不应该有其他用途。建议禁止该账号登陆操作系统。此举在防止黑客利用 MySQL 数据库漏洞反射 shell 有极佳效果。

检查手段：

假设 MySQL 数据库操作系统账号就是 MySQL 执行下列命令

getent passwd MySQL | egrep "^.*[ \ /bin\ / false| \ / sbin \ / nologin]$"

如果没有返回行则说明存在安全隐患。

修复建议：执行下列语句禁止 MySQL 登陆

usermod -s /sbin/nologin MySQL

## 2.1.6 禁止 MySQL 使用默认端口

建议理由：

使用默认端口，会更容易被黑客在网络中发现数据库。改成其他端口有助于隐藏数据库，防止被黑客入侵。

检查手段：

执行 sql 确认端口

show global variables like 'port';

如果返回是 3306，则说明需要修改端口

修复建议：修改配置文件更改端口

修改为非 3306 端口，

## 2.2 文件系统权限安全配置

本节包含 MySQL 关键文件的安全配置项。

### 2.2.1 确保数据文件最小权限

建议理由：

限制数据文件的访问权限，有助于阻碍不法分子直接从数据文件中读取数据甚至读取或替换 MySQL.user 表中的用户和密码信息。

检查手段：

1.执行 sql 定位数据文件地址

show variables where variable_name = 'datadir';

2.检查路径权限是否符合最小权限原则

ls -l /.. | egrep "^d[r|w|x]{3}------\s*.\s*MySQL\s*MySQL\s*\d*.*MySQL"

如果没有返回行，则说明存在安全问题

修复建议：请执行以下语句

chmod 700 <'datadir'>

chown MySQL:MySQL <'datadir'>

### 2.2.2 确保 log_bin_basename 文件最小权限

建议理由：

限制日志文件的权限将有益于保护数据信息不泄露，或被恶意修改。

检查手段：

1.执行 sql 定位日志文件地址

show variables like 'log_bin_basename';

2.检查日志文件的权限是 660 属于 MySQL

修复建议：请执行以下语句

chmod 660 <'log file'>

chown MySQL:MySQL <"log file '>

### 2.2.3 确保 log_error 文件最小权限

建议理由：

限制日志文件的权限将有益于保护数据信息不泄露，或被恶意修改。

检查手段：

1.执行 sql 定位日志文件地址

show global variables like 'log_error';

2.检查日志文件的权限是 660 属于 MySQL

修复建议：请执行以下语句

chmod 660 <'log file'>

chown MySQL:MySQL <"log file '>

## 2.2.4 确保 slow_query_log 文件最小权限

建议理由：

限制日志文件的权限将有益于保护数据信息不泄露，或被恶意修改。

检查手段：

1.执行 sql 定位日志文件地址

show variables like 'slow_query_log_file';

2.检查日志文件的权限是 660 属于 MySQL

修复建议：请执行以下语句

chmod 660 <'log file'>

chown MySQL:MySQL <"log file '>

## 2.2.5 确保 relay_log_basename 文件最小权限

建议理由：

限制日志文件的权限将有益于保护数据信息不泄露，或被恶意修改。

检查手段：

1.执行 sql 定位日志文件地址

show variables like 'relay_log_basename';

2.检查日志文件的权限是 660 属于 MySQL

修复建议：请执行以下语句

chmod 660 <'log file'>

chown MySQL:MySQL <"log file '>

## 2.2.6 确保 general_log_file 文件最小权限

建议理由：

限制日志文件的权限将有益于保护数据信息不泄露，或被恶意修改。

检查手段：

1.执行 sql 定位日志文件地址

show variables like 'general_log_file';

2.检查日志文件的权限是 660 属于 MySQL

修复建议：请执行以下语句

chmod 660 <'log file'>

chown MySQL:MySQL <"log file '>

### 2.2.7 确保密钥文件最小权限

建议理由：

限制密钥文件的访问权限，防止密钥文件被盗取，被替换，被破解等情况的发生。

检查手段：

1.执行 sql 定位密钥文件地址

show variables where variable_name = 'ssl_key';

2.检查路径权限是否符合最小权限原则

ls -l | egrep "^-r--------[ \t]*.[ \t]*MySQL[ \t]*MySQL.*$"

如果没有返回行，则说明存在安全问题

修复建议：请执行以下语句

chmod 400 <ssl_key_value>

chown MySQL:MySQL < ssl_key_value >

### 2.2.8 确保插件目录最小权限

建议理由：

限制插件目录的权限，防止有人恶意添加插件，这些插件会在 MySQL 启动时和 MySQL 同时启动。如果被插入恶意插件，可能会导致 MySQL 被控制。

检查手段：

1.执行 sql 定位插件目录地址

show variables where variable_name = 'plugin_dir';

2.检查路径权限是否符合最小权限原则

ls -l /.. | egrep "^drwxr[-w]xr[-w]x[ \t]*[0-9][ \t]*MySQL[ \t]*MySQL.*plugin.*$"如果没有返回行，则说明存在安全问题

修复建议：请执行以下语句

chmod 755 < plugin_dir Value>

chown MySQL:MySQL < plugin_dir Value >

## 2.3 MySQL 基本安全配置

本节包含 MySQL 数据库自身的基本安全配置项

### 2.3.1 确保使用最新版 MySQL 数据库

建议理由：

新版 MySQL 数据库会修复一些 bug 和所有已知数据库漏洞。能极大的提高数据库的安全性

检查手段：

执行 sql 检查数据库版本

SHOW VARIABLES WHERE Variable_name LIKE "version";

修复建议：存在新版本请替换成新版本数据库。

### 2.3.2 确保样例数据库删除

建议理由：

样例数据库可以被所有数据库用户访问，并且可以用来消耗系统资源删除样例库可以减少黑客攻击面。

检查手段：

执行 sql 检查数据库是否存在样例库

SHOW DATABASES LIKE 'test';

修复建议：如果存在删除样例库

DROP DATABASE "test";

### 2.3.3 修改 root 用户名

建议理由：

MySQL 默认用户 root 应该修改名称，以减小攻击面。防止黑客针对用户名进行密码猜测攻击。

检查手段：

执行 sql 检查数据库是否有默认用户 root

SELECT user from MySQL.user where user='root;

如果有返回行则需要修改

修复建议：修改用户名

update user set name='newname' where name='oldname';

flush privileges;

### 2.3.4 确保 allow-suspicious-udfs 设置成 false

建议理由：

关闭 allow-suspicious-udfs，可以防止通过共享对象文件加载存在威胁的 UDFs 函数。

检查手段：

1. 检查配置文件确定这个参数被设置成 false（没有通过命令检查的方式吗？）
2. 检查 MySQLd 的启动参数中 没有 allow-suspicious-udfs 参数

修复建议：

1. 从配置文件中把 allow-suspicious-udfs 设置成 false
2. 启动参数中剔除 allow-suspicious-udfs

### 2.3.5 禁用 local_infile

建议理由：

禁用 local_infile 可以阻止黑客利用 sql 注入来读取数据库文件，减小黑客给数据库带来的安全损失。

检查手段：用 sql 语句检查参数状态

```
SHOW VARIABLES WHERE Variable_name = 'local_infile';
```

如果返回值不是 off，则存在安全问题

修复建议：在配置文件中加入下列内容，然后重启数据库

```
Local_infile=0
```

### 2.3.6 确保 skip-grant-tables 设置成 false

建议理由：

如果不关闭此参数，所有账号可以不受限制的访问任意数据库。会导敏感数据外泄。

检查手段：

打开配置文件检查 skip-grant-tables 是否被设置成 false（没有通过命令检查的方式吗？）

修复建议：

在配置文件中把 skip-grant-tables 设置成 false

### 2.3.7 确保 skip-symbolic-links 开启

建议理由：

开启 skip-symbolic-links 可以禁止数据库用户删除或重名数据文件目录之外的文件。

检查手段：用 sql 语句检查参数状态

```
SHOW variables LIKE 'have_symlink';
```

修复建议：

如果没有启动 skip-symbolic-links 建议把 MySQL 配置文件中的值设置成 yes。

### 2.3.8 确保插件 daemon_memcached 被禁用

建议理由：

任何人可以利用 daemon_memcached 来访问或修改一部分数据，给数据库造成信息泄漏的隐患。

检查手段：用 sql 语句检查参数状态

SELECT * FROM information_schema.plugins WHERE PLUGIN_NAME='daemon_memcached';

如果有返回行数说明有插件，需要删除

修复建议：删除插件语句如下

uninstall plugin daemon_memcached;

### 2.3.9 确保 secure_file_priv 不是空

建议理由：

secure_file_priv 限制客户端可以读取数据文件的路径。secure_file_priv 设置合理的值可以有效降低 sql 注入后黑客读取数据库数据的可能性。

检查手段：用 sql 语句检查参数状态

SHOW GLOBAL VARIABLES WHERE Variable_name = 'secure_file_priv' AND Value<>'';

如果有返回内容说明安全，否则需要修复

修复建议：在配置文件中添加如下语句，然后重启数据库

secure_file_priv=<path_to_load_directory>>

### 2.3.10 确保 sql_mode 是 STRICT_ALL_TABLES 模式

建议理由：

sql_mode 模式有三种，STRICT_TRANS_TABLES 是其中一种模式。STRICT_TRANS_TABLES 模式会检查所有更新的数据，在一定程度可以给入侵者规避检测带来阻碍。

检查手段：用 sql 语句检查参数状态

SHOW VARIABLES LIKE 'sql_mode';

如果有返回的是 STRICT_TRANS_TABLES 说明安全，否则需要修复

修复建议：在配置文件中添加如下语句，然后重启数据库

sql_mode=STRICT_ALL_TABLES

### 2.3.11 确保 disconnect_on_expired_password 参数是 ON

建议理由：

disconnect_on_expired_password 是用来控制客户端用失效密码来访问数据库的。关闭这个参数会给数据库带来安全风险。

检查手段：用 sql 语句检查参数状态

SHOW GLOBAL VARIABLES like 'disconnect_on_expired_password';

如果有返回的是 ON 说明安全，否则需要修复

修复建议：在配置文件中添加如下语句，然后重启数据库

disconnect_on_expired_password =ON

## 2.4 MySQL 权限安全配置

这一节主要包含各种权限的安全配置

### 2.4.1 确保只有管理员账号有所有数据库的访问权限

建议理由：

除了管理员账号，其他用户没必要有所有数据库的访问权限。过高的权限会导致安全问题。

检查手段：用 sql 语句检查

SELECT user, host FROM MySQL.user WHERE (Select_priv = 'Y') OR (Insert_priv = 'Y') OR (Update_priv = 'Y') OR (Delete_priv = 'Y') OR (Create_priv = 'Y') OR (Drop_priv = 'Y');

SELECT user, host FROM MySQL.db WHERE db = 'MySQL' AND ((Select_priv = 'Y') OR (Insert_priv = 'Y') OR (Update_priv = 'Y') OR (Delete_priv = 'Y') 45 | Page OR (Create_priv = 'Y') OR (Drop_priv = 'Y'));

如果返回的都是管理员账号说明安全，否则需要对用户清除权限

修复建议：

清除非管理员账号的过高部分权限

### 2.4.2 非管理员账号 file_priv 不应该设置成 Y

建议理由：

File_priv 权限允许 MySQL 用户对磁盘进行读写操作。黑客很可能利用这一点盗取数据库中敏感数据。

检查手段：用 sql 语句检查

select user, host from MySQL.user where File_priv = 'Y';

如果返回的都是管理员账号，否则需要对用户清除权限

修复建议：

REVOKE FILE ON *.* FROM '<user>';

### 2.4.3 非管理员账号 process_priv 不应该设置成 Y

建议理由：

process_priv 权限允许委托账号查看当前正在执行的 sql 语句。使用超越当前用户权限的权利。可以被攻击者所利用。

检查手段：用 sql 语句检查

select user, host from MySQL.user where Process_priv = 'Y';

如果返回的都是管理员账号则是安全的，否则需要对用户清除权限

修复建议：

REVOKE PROCESS ON *.* FROM '<user>';

### 2.4.4 非管理员账号 super_priv 不应该设置成 Y

建议理由：

super_priv 权限允许委托账号执行任意语句，非管理员不应该具备该权限。

检查手段：用 sql 语句检查

select user, host from MySQL.user where Super_priv = 'Y';

如果返回的都是管理员账号则是安全的，否则需要对用户清除权限

修复建议：

REVOKE SUPER ON *.* FROM '<user>';

### 2.4.5 非管理员账号 shutdown_priv 不应该设置成 Y

建议理由：

shutdown_priv 权限允许委托账号关闭数据库，会造成一定安全隐患。

检查手段：用 sql 语句检查

SELECT user, host FROM MySQL.user WHERE Shutdown_priv = 'Y';

如果返回的都是管理员账号则是安全的，否则需要对用户清除权限

修复建议：

REVOKE SHUTDOWN ON *.* FROM '<user>';

### 2.4.6 非管理员账号 create_user_priv 不应该设置成 Y

建议理由：

create_user_priv 权限允许委托账号创建任意用户，会造成一定安全隐患。

检查手段：用 sql 语句检查

SELECT user, host FROM MySQL.user WHERE Create_user_priv = 'Y';

如果返回的都是管理员账号则是安全的，否则需要对用户清除权限

修复建议：

REVOKE CREATE USER ON *.* FROM '<user>';

### 2.4.7 非管理员账号 grant_priv 不应该设置成 Y

建议理由：

Grant_priv 权限允许委托账号对其他用户赋权，可能会被黑客利用造成一定安全隐患。

检查手段：用 sql 语句检查

SELECT user, host FROM MySQL.user WHERE Grant_priv = 'Y';

如果返回的都是管理员账号则是安全的，否则需要对用户清除权限

修复建议：

REVOKE Grant ON *.* FROM '<user>';

### 2.4.8 非管理员账号 reload_priv 不应该设置成 Y

建议理由：

reload_priv 权限可以对本地文件进行操作，可能会被黑客利用造成一定安全隐患。

检查手段：用 sql 语句检查

SELECT user, host FROM MySQL.user WHERE reload_priv = 'Y';

如果返回的都是管理员账号则是安全的，否则需要对用户清除权限

修复建议：

REVOKE reload ON *.* FROM '<user>';

### 2.4.9 非管理员账号 repl_slave_priv 不应该设置成 Y

建议理由：

repl_slave_priv 用于从主服务器上获得更新的数据。黑客很可能利用这一点盗取数据库中敏感数据。

检查手段：用 sql 语句检查

select user, host from MySQL.user where repl_slave_priv = 'Y';

如果返回的都是管理员账号，否则需要对用户清除权限

修复建议：

REVOKE repl_slave ON *.* FROM '<user>';

### 2.4.10 确保 DML/DDL 权限只在特定用户手上

建议理由：

限制用户有 INSERT，SELECT，UPDATE，DELETE，DROP，CREATE 和 ALTER 权限。这些权限都会导致数据泄密等。

检查手段：用 sql 语句检查

SELECT User,Host,Db FROM MySQL.db WHERE Select_priv='Y' OR Insert_priv='Y' OR Update_priv='Y' OR Delete_priv='Y' OR Create_priv='Y' OR Drop_priv='Y' OR Alter_priv='Y';

如果返回的都是管理员账号，否则需要对用户清除权限

修复建议：

REVOKE SELECT ON . FROM ; REVOKE INSERT ON . FROM ; REVOKE UPDATE ON . FROM ; REVOKE DELETE ON . FROM ; REVOKE CREATE ON . FROM ; REVOKE DROP ON . FROM ; REVOKE ALTER ON . FROM ;

## 2.5 审计和日志安全配置

### 2.5.1 确保 log_error 日志启动

建议理由：

启用错误日志有可能会增加检测到针对 MySQL 的恶意攻击行为机会。为日后安全检查提供更多线索和证据。

检查手段：用 sql 语句检查

SHOW variables LIKE 'log_error';

如果返回是空否，则存在安全问题需要修复

修复建议：

打开 MySQL 配置文件把 log_error 配置到一个有效路径

### 2.5.2 确保日志文件在非系统分区

建议理由：

操作系统上有明确的系统分区和非系统分区。如果日志文件在系统分区，会提高整个系统因为磁盘空间用尽发生拒绝服务的几率。

检查手段：

1.进入数据库执行下列语句

SELECT @@global.log_bin_basename;

2. df –h <上面的地址返回值>

如果发现结果中存在/ /var /usr 说明数据文件在系统分区建议换区

修复建议：打开 MySQL 配置文件

将 log_bin 设置成非系统分区路径

### 2.5.3 确保 log_raw 被设置成 off

建议理由：

语句中的密码在写入一般查询日志时会被服务器重写，不会以明文方式记录。但如果 log-raw 被设置成 ture，则会记成明文。

检查手段：

打开数据库配置文件

确定 log_raw 被设置成 off

如果是 off 是安全的，但如果是 on 则需要修复

修复建议：打开 MySQL 配置文件

Log_raw = OFF

### 2.5.4 确保 log_warnings 被设置成 2

建议理由：

log_warnings 适用于决定日志中记录的内容的。默认是 1 随着级别的调整会记录更多信息。调整到 2 有助于通过日志追查安全问题。

检查手段：

通过 sql 读取信息

SHOW GLOBAL VARIABLES LIKE 'log_warnings';

如果返回值是 1 需要调整到 2

修复建议：打开 MySQL 配置文件

log_warnings = 2

### 2.5.5 确保 audit_log_connection_policy 被设置成 ERRORS 或 ALL（企业版独有）

建议理由：

audit_log_connection_policy 分为三种模式：NONE，ERRORS 和 ALL。建议选用 ERRORS（只记录登录失败事件）或 ALL。更有利于追踪安全问题。

检查手段：

通过 sql 读取信息

show variables like '%audit_log_connection_policy%';

如果返回值是 none 或者空，则需要调整成 all 或 ERRORS

修复建议：使用 sql 语句

set global audit_log_connection_policy = ERRORS

set global audit_log_connection_policy = ALL

### 2.5.6 确保 audit_log_exclude_accounts 中设置的内容不会让某些用户的行为逃避审计（企业版独有）

建议理由：

audit_log_exclude_accounts 中的内容会不审计。检查参数的内容防止某些用户的操作逃避审计。

检查手段：

通过 sql 读取信息

SHOW VARIABLES LIKE '%audit_log_exclude_accounts%';

检查返回值是否有需要审计的内容。

修复建议：

SET GLOBAL audit_log_exclude_accounts = NULL

### 2.5.7 确保 audit_log_include_accounts 中设置为空（企业版独有）

建议理由：

audit_log_include_accounts 中的内容才会审计。audit_log_include_accounts 支持两种参数，一种是一个用户列表，另一种是一个参数 null。列表的方式很容易导致有部分用户逃避审计。所以建议使用 null

检查手段：

通过 sql 读取信息

SHOW VARIABLES LIKE '%audit_log_include_accounts%';'

检查返回值是空或 null。

修复建议：

SET GLOBAL audit_log_include_accounts = NULL

### 2.5.8 确保 audit_log_policy 设置为 LOGINS 或更高级别（企业版独有）

建议理由：

audit_log_policy 的值指定记录哪些事件，ALL(所有事件)，LOGINS(仅仅用户登录事件)，QUERIES(仅仅查询语句)，NONE(不记录任何事件)，建议使用 LOGINS。

检查手段：

通过 sql 读取信息

SHOW GLOBAL VARIABLES LIKE 'audit_log_policy';

检查返回值是 all,如果不是建议调整。

修复建议：

SET GLOBAL audit_log_policy='ALL';

SET GLOBAL audit_log_policy='LOGINS';

### 2.5.9 确保 audit_log_statement_policy 中设置为 ALL（企业版独有）

建议理由：

audit_log_statement_policy 分为三种模式：NONE，ERRORS 和 ALL。建议选用 ERRORS（只记录登录失败事件）或 ALL。使用 errors 或 all 模式更有利于追踪安全问题。audit_log_policy 如果设置会覆盖 audit_log_statement_policy 的效果。

检查手段：

通过 sql 读取信息

SHOW GLOBAL VARIABLES LIKE 'audit_log_statement_policy';

检查返回值是 all 如果不是建议调整。

修复建议：

audit_log_statement_policy='ALL'

### 2.5.10 确保 audit_log_strategy 设置为 SYNCHRONOUS 或 SEMISYNCHRONOUS（企业版独有）

建议理由：

audit_log_strategy 分为 4 种模式：ASYNCHRONOUS，PERFORMANCE，SEMISYNCHRONOUS 和 SYNCHRONOUS 建议使用 SEMISYNCHRONOUS 和 SYNCHRONOUS。

检查手段：

通过 sql 读取信息

SHOW GLOBAL VARIABLES LIKE 'audit_log_strategy';

检查返回值是 SEMISYNCHRONOUS 或 SYNCHRONOUS 如果不是建议调整。

修复建议：

打开 MySQL 配置文件

设置 audit_log_strategy =SEMISYNCHRONOUS 或 SYNCHRONOUS

## 2.6 身份认证安全配置

本节包含属于 MySQL 认证配置的安全配置

### 2.6.1 确保密码不在全局变量中

建议理由：

MySQL 配置文件（客户端部分）允许设置用户名和密码。使用密码参数可能会对用户的机密性造成负面影响。

检查手段：

打开数据库配置文件

检查用户名和密码参数

如果是空的是安全的，但如果不是空的则需要修复

修复建议：打开 MySQL 配置文件

清理参数内容

### 2.6.2 确保 sql_mode 中含有 NO_AUTO_CREATE_USER

建议理由：

NO_AUTO_CREATE_USER 是 sql_mode 的一个选项，可以阻止 grant 语句在特定情况下自动创建用户。给数据库带来安全隐患。

检查手段：

通过 sql 观察参数

SELECT @@session.sql_mode;

如果返回值包含 NO_AUTO_CREATE_USER 是安全的，但如果不包含则需要修复

修复建议：打开 MySQL 配置文件

在 sql_mode 中添加参数 NO_AUTO_CREATE_USER

### 2.6.3 确保没有用户使用空密码

建议理由：

如果密码被设置成空密码，入侵者只要知道密码和主机允许列表，就可以绕过身份验证随意登录数据库。进行违规操作。

检查手段：

通过 sql 观察参数

SELECT User,host FROM MySQL.user WHERE authentication_string='';

没有行数返回说明安全，否则需要配置

修复建议：给空白密码的账号配上密码

SET PASSWORD FOR @'' = '<user>@;<host>' = '<clear password>'

### 2.6.4 确保 default_password_lifetimes 少于或等于 90 天

建议理由：

密码需要定期更换，才有意义，也才能更有效的防止黑客破解。

检查手段：

通过 sql 观察参数

SHOW VARIABLES LIKE 'default_password_lifetime';

看返回值和 90 的关系如果大于 90 就需要修复

修复建议：设置全局变量

SET GLOBAL default_password_lifetime=90

### 2.6.5 确保用户不允许所有 ip 访问

建议理由：

某一数据库用户支持所有 ip 访问，一旦账号密码泄露，数据库就变得很不安全。

检查手段：

通过 sql 观察参数

SELECT user, host FROM MySQL.user WHERE host = '%';

结果集为空说明不存在问题，否则需要修复

修复建议：删除该用户会或通过 alter 删除 %,指定特定 ip

### 2.6.6 确保无匿名帐户存在

建议理由：

匿名用户是空的，也没有密码。安全性很差，任意人员都可以利用匿名用户访问数据库。

检查手段：

通过 sql 观察参数

SELECT user,host FROM MySQL.user WHERE user = '';

结果集为空说明不存在问题，否则需要修复

修复建议：删除匿名用户。

## 2.7 网络安全配置

### 2.7.1 确保 have_ssl 设置成 yes

建议理由：

所有网络请求必须走 SSL/TLS 访问数据库。杜绝网络劫持和网络拦截

检查手段：

通过 sql 观察参数

SHOW variables WHERE variable_name = 'have_ssl';

返回 yes 不存在问题，否则需要修复

修复建议：开启 ssh。

### 2.7.2 确保 ssl_type 是 'ANY', 'X509', or 'SPECIFIED'

建议理由：

所有网络请求必须走 SSL/TLS 访问数据库。SSL 提供多种算法，其中一些算法安全性并不高不能帮助用户杜绝网络劫持和网络拦截。建议设置 ssl_type 为高安全类型的加密算法。但这其中有一个隐患，如果客户端使用较低加密算法，会由于算法无法匹配导致链接失败。

检查手段：

通过 sql 观察参数

SELECT user, host, ssl_type FROM MySQL.user WHERE NOT HOST IN ('::1', '127.0.0.1', 'localhost');

确保每个用户返回的 ssl_type 等于 ANY, X509, or SPECIFIED

修复建议：使用 GRANT 语句来使用要求的 SSL

GRANT USAGE ON *.* TO 'my_user'@'app1.example.com' REQUIRE SSL;

## 2.8 复制数据传输中的安全配置

### 2.8.1 确保 MASTER_SSL_VERIFY_SERVER_CERT 设置成 yes 或 1

建议理由：

在使用 SSL 时，证书验证对于验证正在进行连接的一方很重要。 在这种情况下，从属（客户端）应该在继续连接之前验证主服务器的证书以验证主服务器。MASTER_SSL_VERIFY_SERVER_CERT 主要是用于检查证书，确保证书的合法性。

检查手段：

通过 sql 观察参数

select ssl_verify_server_cert from MySQL.slave_master_info;

ssl_verify_server_cert 返回是 1，否则需要修复

修复建议：修改 MASTER_SSL_VERIFY_SERVER_CERT 配置。

STOP SLAVE; -- required if replication was already running

CHANGE MASTER TO MASTER_SSL_VERIFY_SERVER_CERT=1;

START SLAVE; -- required if you want to restart replication

### 2.8.2 确保 master_info_repository 设置成 table

建议理由：

master_info_repository 设置成 table.客户端使用的密码存储在表中。相较于文件系统表中更为安全。

检查手段：

通过 sql 观察参数

SHOW GLOBAL VARIABLES LIKE 'master_info_repository';

master_info_repository 返回是 table，否则需要修复

修复建议：打开配置文件

set the master_info_repository value to TABLE

注：参考链接：https://dev.MySQL.com/doc/refman/5.7/en/