Mysql 数据库安全

1.数据库 安全准则。

在讨论安全性时，有必要考虑完全保护整个服务器主机，而不仅仅是 MySQL 服务器，以防止所有类型的攻击：窃听，更改，重放和拒绝服务。我们并不涵盖可用性和容错的所有方面。

MySQL 使用基于访问控制列表（ACL）的安全策略管理所有的连接、查询以及用户尝试执行的其它操作。MySQL 还支持客户端和服务器之间的 SSL 加密连接。这里讨论的许多概念并不仅仅是针对 MySQL 的，而是几乎适用于所有应用程序的。

运行 MySQL 时，请遵循以下准则：

- 不要给除了 root 账号以外的任何人访问 mysql.user 表的权限！这是至关重要的。
- 了解 MySQL 访问权限系统的工作原理。使用 grant 和 revoke 语句来控制对 MySQL 的访问。不要授予比必要更多的权限。决不授予所有主机的访问权限。

检查清单：

- 运行 mysql -u root. 如果不需要密码即可连接成功，那么任何人都可以以 root 账号的身份连接到你的 MySQL 服务器；
- 使用 show grants 语句来检查所有账号的权限。然后使用 revoke 语句删除那些不必要的访问权限；
- 不要在你的数据库中存储明文密码。因为如果你的计算机遭到入侵，那么入侵者可以获取完整的密码列表并使用它们。相反，使用 SHA2()或其它单向散

列函数并存储散列值。

要防止使用彩虹表进行密码恢复，请勿直接在普通密码上使用这些函数；而是应该选择一些字符串作为 salt，并使用 hash (hash (password) + salt)的值。

- 不要从字典中选择密码。因为存在破解密码的特殊程序。即使像"xfish98"这样的密码也很糟糕。更好的是"duag98"，它包含的也是单词"fish"，但在标准键盘上取了"fish"中每个字母左侧的那个字母。另一种方法是使用句子中每个单词的首字母提取的密码(例如，"Four score and seven years ago"的密码是"Fsasya")。这样的密码很容易记住和键入，但对于不知道句子的人却很难猜出。而且，你还可以用数字取代数字单词，以获得短语"4 score and 7 years ago"，从而产生密码"4sa7ya"，这更难以猜测。

- 投资于防火墙。这可以保护你免受来自各种软件所有类型漏洞中至少 50%的伤害。将 MySQL 放在防火墙后面或放在非军事区域（DMZ）中。

检查清单：

- 尝试使用 nmap 等工具从 Internet 上扫描你的端口。MySQL 的默认端口为 3306。这个端口不应该能够被不受信任的主机所访问，即不受信任的主机应不能访问此端口。检查你的 MySQL 端口是否打开的一个简单方法是，尝试从一些远程计算机上执行以下命令，其中 server_host 是运行 MySQL 服务器的主机的主机名或 IP 地址

```
shell> telnet server_host 3306
```

如果 telnet 挂起或连接被拒绝，那么说明端口被限制了，这就是期望的结果。如果连接成功并收到一些垃圾字符，则说明端口是打开的，那么此时你应该在防火墙或路由器上关闭该端口，除非你真的有一个很好的理由保持打开。

- 访问 MySQL 的应用程序不应该信任用户输入的任何数据，应该使用适当的防御性编程技术编写这些应用程序。

- 不要在互联网上传输普通的（未加密的）数据。所有有时间和能力拦截它的人都可能获取到这些信息，并将其用于自己的目的。相反，我们应该使用加密的协议，如 SSL 或 SSH。MySQL 支持内部 SSL 连接。另一种技术是使用 SSH 端口转发来创建用于通信的加密（和压缩）隧道。

- 学习使用 tcpdump 和 strings 实用工具。在大多数情况下，你可以通过以下命令来检查 MySQL 数据流是否未加密：

```
shell> tcpdump -l -i eth0 -w - src or dst port 3306 | strings
```

如果你没有看到明文数据，这并不总是意味着信息是加密的。如果你需要高安全性，请咨询安全专家。

2. 保证密码安全

密码出现在 MySQL 内的几个上下文中。下面的部分提供了指导原则，使终端用户和管理员能够确保这些密码的安全，避免暴露它们。还有一个关于 MySQL 如何在内部使用密码散列的讨论。

2.1 终端用户密码安全指南

MySQL 用户应该使用以下准则来保证密码安全。

当你运行客户端程序连接到 MySQL 服务器时，以一种将其暴露给其他用户的方式来指定密码是不明智的。这里列出了在运行客户端程序时可以用来指定密码的方法，以及每种方法的风险评估。简而言之，最安全的方法是让客户端程序提示输入密码，或者在恰当保护的选项文件中指定密码。

- 在命令行上使用「-pyour_pass」或「--password=your_pass」选项。例如：

```
shell> mysql -u francis -pfrank db_name
```

这种方法很方便，但不安全。在某些系统中，系统状态程序可以看到你的密码，比如其他用户可以调用 ps，以显示命令行。在初始化序列期间，MySQL 客户端通常会用零覆盖命令行中的密码参数。但是，仍然有一段短暂的时间间隔，值是可见的。此外，在某些系统上，这种覆盖策略是无效的，密码对于 ps 仍旧可见。

如果你的操作环境设置为在终端窗口的标题栏中显示当前命令，那么只要命令正在运行，密码仍然可见，即使命令已经在窗口内容区域中滚动了。

- 在命令行上使用「-p」或「--password」选项，而不指定密码值。在这种情况下，客户端程序会以交互的方式请求密码：

```
shell> mysql -u francis -p db_name
Enter password: ********
```

*表示输入密码的位置。当你输入密码时，密码不会显示出来。

输入密码比在命令行上指定密码更安全，因为它对其他用户不可见。但是，输入密码的方法仅适用于以交互方式运行的程序。如果你想从非交互运行的脚本中调用客户端，则无法从键盘输入密码。在某些系统上，你甚至可能会发现脚本的第一行被读取并错误地解释为密码。

- 将你的密码存储在选项文件中。例如，在 Unix 上，你可以在主目录中的.my.cnf 文件的[client]部分列出密码：

```
[client]
password=your_pass
```

为了保证密码的安全，除了你以外，任何人都不能访问这个文件。为了确保这一点，请将文件访问模式设置为 400 或 600。例如：

```
shell> chmod 600 .my.cnf
```

要在命令行上指定包含密码的特定选项文件，请使用「--defaults-file=file_name」选项，其中 file_name 是文件的完整路径名。例如：

```
shell> mysql --defaults-file=/home/francis/mysql-opts
```

- 将你的密码存储在「MYSQL_PWD」环境变量中。这种指定 MySQL 密码的方法是非常不安全的，不应该被使用。某些版本的 ps 包含了显示运行进程环境的选项。在某些系统上，如果你设置了「MYSQL_PWD」，那么你的密码就会被暴露给运行 ps 的任何其他用户。即使在没有这种版本的 ps 的系统上，假设没有其他用户可以检查进程环境的方法也是不明智的。

在 Unix 上，MySQL 客户端会将执行过的语句写入一个历史文件。默认情况下，此文件名为「.mysql_history」，并在你的主目录中创建。例如 create user, grant 和 set password 之类的 SQL 语句中的密码是以纯文本的形式写入的，因此如果你使用了这些语句，则它们将被记录在历史文件中。为了保证这个文件的安全，请使用与「.my.cnf」文件相同的访问限制模式。

如果你的命令解释器被配置为维护历史记录，则用于保存命令的任何文件都将包含在命令行中输入的 MySQL 密码。例如，bash 使用「~/.bash_history」。任何此类文件都应该具有限制性访问模式。

2.2 管理员密码安全指南

数据库管理员应使用以下准则来保证密码安全。

MySQL 在 mysql.user 表中存储账号密码。不应向任何非管理账号授予访问该表的权限。

有权修改插件目录 (plugin_dir 系统变量的值) 或指定插件目录位置的 my.cnf 文件的用户可以替换插件，并修改插件提供的功能，包括验证插件。

应该保护可能写入密码的日志文件等文件。

2.3 密码和日志

在 create user, grant, set password 以及调用 password()函数的 SQL 语句中密码以纯文本的形式写入。如果这些语句被 MySQL 服务器记录，那么有权访问日志的任何人都可以看到密码。这适用于一般查询日志，慢查询日志和二进制日志。

审核日志插件生成的审核日志文件的内容未加密。为了安全起见，这个文件应该被写入一个只有 MySQL 服务器和有合法理由查看日志的用户才可以访问的目录中。

为了防范日志文件被曝光，请将它们放在只有 MySQL 服务器和数据库管理员才有权访问的目录中。如果服务器将日志记录到「mysql」库表中，则只将这些表的访问权限授予数据库管理员。

复制 slave 将 master 的密码存储在「master.info」文件中。限制此文件只能由数据库管理员访问。

使用受限访问模式来保护包含日志表或包含密码的日志文件的数据库备份。

3. 保护 MySQL 免受攻击

当连接到 MySQL 服务器时，你应该使用密码。密码不会在连接上以明文的

形式传输。在 MySQL4.1.1 中，客户端连接序列中的密码处理经过升级，变得非常安全。如果你仍然使用的是 4.1.1 之前的密码，则加密算法不如新算法那样强大。通过一些努力，聪明的攻击者可以嗅探客户端和服务器之间的流量，从而破解密码。

其它所有信息都以纯文本的形式传输，任何能够查看连接的人都可以阅读。如果客户端和服务器之间的连接通过了一个不受信任的网络，并且你为此感到担忧，那么你可以使用压缩协议使流量更加难以解密。你还可以使用 MySQL 的内部 SSL 支持，使连接更加安全。或者，使用 SSH 在 MySQL 服务器和 MySQL 客户端之间建立加密的 TCP/IP 连接。

要确保 MySQL 系统安全，你务必仔细考虑以下建议：

- 要求所有 MySQL 账号都要有密码。客户端程序不一定知道运行它的人的身份。对于客户端/服务器应用程序，用户可以为客户端程序指定任何用户名，这很常见。例如，如果 other_user 没有密码，那么任何人都可以通过调用 mysql -u other_user db_name 以其他人的身份连接到服务器。如果所有的账号都有密码，那么使用其他用户的账号进行连接就会变得更加困难。

- 确保唯一对数据库目录具有读或写权限的 Unix 账号是用于运行 mysqld 的账号。

- 不要以 Unix root 用户的身份运行 MySQL 服务器。这是非常危险的，因为这将使得任何具有「file」权限的账号都能够使 MySQL 服务器以 Unix root 用户的身份创建文件（例如，~root/.bashrc 文件）。为了防止这种情况，mysqld 拒绝以 root 身份运行，除非使用「--user=root」选项显式指明。

mysqld 应该作为一个普通的、非特权用户来运行。你可以创建一个单独的 Unix 账号，名为 mysql，该账号专门用于管理 MySQL，从而使一切更加安全。要以不同的 Unix 用户启动 mysqld，请在 my.cnf 选项文件的组[mysqld]中添加一个用户选项，例如：

```
[mysqld]
user=mysql
```

无论你是手动还是使用 mysqld_safe 或 mysql.server 启动服务器，该选项都将使得服务器以指定的用户身份启动。

以 Unix 用户而非 root 用户运行 mysqld，并不意味着你需要修改 mysql.user 表中的 root 账号。MySQL 账号的用户名与 Unix 账号的用户名没有任何关系。

- 不要将「file」权限授予非管理用户。具有此权限的任何用户都可以以 mysqld 进程的有效用户 ID 的身份在文件系统的任何位置写入文件。这包括服务器的数据目录，其中含有实现权限表的文件。为了使「file」权限操作更安全，select ... into outfile 生成的文件不会覆盖现有的文件。

「file」权限也可以用来读取全局可读或者是 MySQL 服务器（mysqld 进程的有效用户 ID）具有读权限的所有文件。使用此权限，你可以将任何文件读入数据库表。这可能会被滥用，例如，使用「load data」将/etc/passwd 加载到表中，然后使用 select 语句进行显示。

为了限制可以读取和写入文件的位置，请将系统变量「secure_file_priv」设置为特定的目录。

- 不要将「process」或「super」权限授予非管理用户。「mysqladmin processlist」和「show processlist」的输出显示了当前正在执行的所有语句的文本，所以能够

查看服务器进程列表的任何用户都可以看到其他用户发起的语句,如 update user set password=password('not_secure');

mysqld 为拥有「super」权限的用户保留了一个额外的连接，这样即使所有正常的连接都在使用中，MySQL 的 root 账号也可以登录并检查服务器的活动。

「super」权限可以终止客户端连接，通过更改系统变量的值来改变服务器的运行，以及控制服务器的复制。● 不允许使用到库表的符号链接。如果你以 root 身份运行 mysqld，这一点尤为重要，因为这会导致任何对服务器数据目录具有写权限的人都可以删除系统中的任何文件。可以使用--skip-symbolic-links 选项禁用此功能。

● 存储过程和视图应该参照第 20.6 节 "存储过程和视图的访问控制"给出的安全准则来编写。

● 如果你不信任你的 DNS，则应在权限表中使用 IP 地址而不是主机名。无论如何，在使用含有通配符的主机名创建权限表条目时，你都应该非常小心。

● 如果要限制单个账号的连接数量，可以通过设置系统变量「max_user_connections」来实现。grant 语句还支持资源控制选项，以限制账号允许使用的服务器资源。

● 如果插件目录对于服务器来说是可写的，那么用户可以使用 select ... into dumpfile 将可执行代码写入目录中的文件。为了防止这种情况，你可以设置「plugin_dir」目录，使其仅对服务器开放读权限，或者将「--secure-file-priv」设置为可以安全地进行 select 写入的目录。

4. 以普通用户运行 MySQL

在 Linux 上，对于使用 MySQL 仓库、RPM 软件包或 Debian 软件包执行的安

装，MySQL 服务器 mysqld 应以本地用户「mysql」的身份启动。以其他操作系统用户的身份启动服务器不受 init 脚本的支持，其作为安装的一部分被包含在系统中。

在 Unix 上（或使用 tar 或 tar.gz 包执行安装的 Linux 上），MySQL 服务器可以由任何用户启动和运行。但是出于安全考虑，你应该避免以 Unix root 身份运行服务器。要将 mysqld 更改为以普通的非特权用户 user_name 的身份运行，你必须执行以下操作：

① 如果服务器正在运行，请使用「mysqladmin shutdown」停止服务器。

② 更改数据库目录和文件，以便 user_name 对其具有读权限和写权限（你可能需要使用 Unix root 用户身份执行此操作）：

```
shell> chown -R user_name /path/to/mysql/datadir
```

如果不执行此操作，服务器将无法访问数据库或表，当它以 user_name 运行时。

如果 MySQL 数据目录中的目录或文件是符号链接，则 chown -R 可能不会跟随符号链接。这时，你需要查看这些链接，手动更改它们指向的目录和文件。

③ 以 user_name 用户身份启动服务器。另一种方法是以 Unix root 用户运行 mysqld，同时使用「--user=user_name」选项。这样的话，mysqld 首先启动，然后在接受任何连接之前切换为 user_name 用户。

④ 要在系统启动时自动以特定的用户运行 MySQL 服务器，请在/etc/my.cnf 选项文件或数据目录中的 my.cnf 选项文件的[mysqld]组中添加一个用户选项，例如：

```
[mysqld]
user=user_name
```

如果你的 Unix 机器本身不安全，你应该为权限表中的「root」账号分配密码。否则，在该机器上具有登录账号的任何用户都可以使用--user=root 选项运行 mysql 客户端并执行任何操作。无论什么时候，你都应该主动为 MySQL 账号分配密码，特别是当主机上存在其他 Unix 账号时，更应如此。

5. load data local 的安全问题

「load data」语句可以加载位于服务器主机上的文件；如果指定了「local」关键字，则可以加载位于客户主机上的文件。「local」版的「load data」有两个潜在的安全问题：

- 从客户主机到服务器主机的文件传输由 MySQL 服务器发起。理论上，服务器可以命令客户端程序传输服务器指定的任何文件，而不是由客户端程序在「load data」语句中指定的文件。所以，攻击者可以使用一个虚假的服务器冒充 MySQL，或者直接在 MySQL 服务器外部添加补丁，以达到访问客户主机文件的目的。这样的服务器可以访问客户主机上客户端用户有权访问的任何文件。更糟糕的是，除了「load data local」，实际上服务器可以对任何的语句响应一个文件传输请求，所以一个更根本的问题是客户端不应该连接到不受信任的服务器。
- 在 Web 环境中，客户端程序是从 Web 服务器进行连接的。用户可以使用「load data local」读取 Web 服务器有权访问的任何文件（假设用户可以对 SQL Server 运行任何语句）。在这种环境中，相对于 MySQL 服务器，客户端是 Web 服务器，而不是连接到 Web 服务器的用户运行的远程程序。

为了避免出现「load data」问题，客户端应避免使用「local」。为避免连接到不受信任的服务器，客户端可以使用「--ssl-verify-server-cert」选项和相应的

CA 证书来建立安全的连接并验证服务器身份。

为了使管理员和应用程序能够管理本地数据加载功能，「local」配置如下：

5.1 在服务器端：

■「local_infile」系统变量控制服务器端的「local」功能。根据「local_infile」设置的不同，服务器拒绝或允许由已开启「local」的客户端发起的本地数据加载。「local_infile」默认开启。

■ 要显式地使服务器拒绝或允许「load data local」语句（不管客户端程序和库在构建时或运行时如何配置），请在启动 mysqld 的同时禁用或启用「local_infile」。「local_infile」也可以在运行时设置。

5.2 在客户端：

■「enabled_local_infile」CMake 选项控制 MySQL 客户端库「local」功能的静态编译。没有显式指定的客户端，根据 MySQL 构建时指定的「enabled_local_infile」设置，禁用或启用「local」功能。

MySQL 二进制发布版中的客户端库在编译时默认启用了「enabled_local_infile」。如果从源码编译 MySQL，则根据没有显式指定的客户端是否应该禁用或启用「local」功能，将「enabled_local_infile」配置为禁用或启用。

■ 使用 C API 的客户端程序可以通过调用 mysql_options()禁用或启用「mysql_opt_local_infile」选项，显式地控制数据加载。

■ 对于 mysql 客户端，默认禁用本地数据加载。要显式地禁用或启用它，请使用「--local-infile=0」或「--local-infile[=1]」选项。

■ 对于 mysqlimport 客户端，默认禁用本地数据加载。要显式地禁用或启用

它，请使用「--local=0」或「--local=[1]」选项。

■ 如果你是在 Perl 脚本或读取选项文件[client]组的其它程序中使用「load data local」，则可以向该组添加「local-infile」选项。为了防止不理解该选项的程序出现问题，请使用「loose-」前缀：

```
[client]
loose-local-infile=0
```

```
[client]
loose-local-infile=1
```

■ 在所有情况中，客户端成功使用「ocal」加载操作也要求服务器允许它。

如果服务器或客户端禁用了「local」功能，那么尝试发起「load data local」语句的客户端将收到以下错误消息：

```
ERROR 1148: The used command is not allowed with this MySQL version
```

6. 客户端编程安全指南

访问 MySQL 的应用程序不应该信任用户输入的任何数据，他们可以通过在 Web 表单，URL 或你构建的任何应用程序中输入特殊的或转义的字符序列来尝试欺骗你的代码。确保你的应用程序在用户输入类似「; drop database mysql;」的语句时仍然是安全的。这是一个极端的例子，但如果你不为此做准备，黑客使用类似技术将引发极大的安全漏洞和数据丢失。

一个常见的错误是只保护字符串数据值。记得要检查数值型数据。如果应用程序在用户输入「234」时生成诸如「select * from table where id = 234」的查询，则用户可以输入「234 or 1 = 1」，使应用程序生成查询「select * from table where id = 234 or 1 = 1」。结果，服务器检索表中的每一行。这会暴露所有数据行，并

导致服务器负载过大。防止此类攻击的最简单的方法是在数值常量周围使用单引号：select * from table where id = '234'。如果用户输入额外的信息，它们都将称为字符串的一部分。在数值上下文中，MySQL 会自动将此字符串转换为数字，并自动剥离末尾的非数字字符，类似于「atoi」或「atof」函数。

有时人们认为，如果一个数据库只包含公开可用的数据，那么它就不需要被保护。这是不正确的。即使允许显示数据库中的任何数据行，你仍然应该防止拒绝服务攻击（例如，那些基于上一段中的技术导致服务器资源浪费的攻击）。否则，你的服务器就无法对合法用户作出响应。

检查清单：

- 启用严格的 SQL 模式，以告知服务器对其接受的数据值进行更严格的限制。参见第 5.1.8 节 "服务器 SQL 模式"。
- 尝试在所有 Web 表单中输入单引号和双引号。如果你遇到任何类型的 MySQL 错误，请立即调查问题。
- 尝试通过向其中添加%22（"），%23（#）和%27（'）来修改动态 URL。
- 尝试使用前面示例中所示的字符，将动态 URL 中的数据类型从数值类型修改为字符类型。你的应用程序应该对这些和类似的攻击是安全的。
- 尝试在数值字段中输入字符、空格和特殊符号，而不是数字。你的应用程序应该在将字段值传递给 MySQL 之前删除这些非法字符，或者直接生成一条错误提示。将未经检查的值传递给 MySQL 是非常危险的！
- 在将数据传递给 MySQL 之前检查数据的大小。
- 应用程序连接数据库的账号不应是管理账号。不要给你的应用程序任何不需要的访问权限。

许多应用程序编程接口提供了转义数据值中特殊字符的方法。如果使用得当，这将防止应用程序用户输入导致应用程序生成与你打算的效果不同的语句的值：

- MySQL C API：使用 mysql_real_escape_string() API 调用。
- MySQL++：对查询流使用 escape 和 quote 修饰符。

其它编程接口可能具有类似的功能。