

Securing MySQL databases

RESEARCH ON SECURING MYSQL DATABASES, FOR
DEVELOPERS WORKING ON THE DATASTREAMS CORONA
TESTRESULT PLATFORM

THOMAS VAN SCHIJNDEL, MICHEL BIJNEN, YOURI SAMAN, ROEL
LUCASSEN, KRISTIAN HANSEN, ALEXANDER LAMBOOIJ

Introduction

Datastreams is creating a corona test data platform to share corona test results easily and securely. Two Fontys graduates are working on the software, while a group of 6 security engineers advise and assess the security aspects of the project.

The graduates asked about advice in relation to securing a MySQL database. In this research document, research is done, and advice is given to improve the security of the storage of this data.

The following question is answered:

How can a MySQL database be secured?

The research is done using the DOT-framework. The following methods of the DOT-framework are used, problem analysis, literature study, community research, prototyping, and best good and bad practices. After using these methods, a conclusion and advice was formed.

Table of contents

Introduction-----	1
Preliminary investigation -----	3
Problem analysis-----	3
Defining sub questions -----	4
Research-----	5
How should access control be handled? -----	5
What is the best way to encrypt a database and its contents? -----	9
How can CRUD methods be applied in a secure way with MySQL? -----	14
How can the database be securely backed up regularly?-----	18
Conclusion-----	21
Advice -----	22

Preliminary investigation

This chapter describes the methods that are used in this research, as well as the research questions.

Problem analysis

The problem analysis is done by using the five W's: **who**, **what**, **where**, **when**, and **why**.

What is the problem? The MySQL database needs to be secured.

Why does the problem exist? The problem exists because the database contains valuable data which should not be stolen by hackers or other malicious actors.

Who suffers from the problem? At first, the users suffer from this problem, since they have very sensitive data which can lead to exclusion in the database. But in the end, Datastreams is ultimately responsible.

When does this problem arise? When the application releases, everything needs to be secure.

Where does this problem exist? This problem exists in the database that is used by the Datastreams test data platform.

The goal of this research is to inform the graduates about securing their MySQL databases against the most common threats. This will be done by listing and elaborating the most important security aspects. This advice will have to be delivered well before the end of the semester so that it can be considered and implemented.

Defining sub questions

According to IBM¹ and eSecurityPlanet², the following aspects are important in terms of database security:

- Physical security;
- Administrative and network access control;
- User account/device security;
- Encryption;
- Database software security;
- Application security;
- Backup security;
- Logging/monitoring.

The graduates have already begun development using MySQL databases in the cloud. A brainstorming session was done to determine which topics research topics were the most important to research. Physical security is out of scope because Amazon and Azure will handle this aspect. Logging and monitoring is important but will be excluded due to time constraints. From the points above, the following aspects were extracted:

- Access control;
- Encryption;
- Application level/CRUD;
- Backups.

Research questions

How can a MySQL database be secured?

- How should access control be handled?
- What is the best way to encrypt the database and its contents?
- How can the CRUD methods be applied in a secure way?
- How can the database be securely backed up regularly?

¹ "Database Security: An Essential Guide | IBM." 27 aug.. 2019, <https://www.ibm.com/cloud/learn/database-security>. Geopend op 3 dec.. 2020.

² "7 Database Security Best Practices | eSecurity Planet." 23 aug.. 2016, <https://www.esecurityplanet.com/network-security/6-database-security-best-practices.html>. Geopend op 3 dec.. 2020.

Research

How should access control be handled?

Library - Literature study

First, the official MySQL documentation³ will be consulted. Reliable and official information can be gathered this way. This information will later be confirmed using other methods.

MySQL has a privilege system in which users and hostnames are defined with the granted permissions. Before a connection is established, MySQL will check the username, hostname, and password. User `joe` from `host1.example.com` can have different permissions than user `joe` from `host2.example.com`.

To see what privileges a user has:

```
SHOW GRANTS FOR 'joe'@'host1.example.com';  
SHOW GRANTS FOR 'joe'@'host2.example.com';
```

The MySQL documentation states that denying access for a specific user is not possible. It also states that permissions are valid for a global user, and not for a specific database, table, or routine. However, creating database and table specific permissions is a common practice⁴ in MySQL for a while now. When looking further through the documentation, clarification on this can be found:

“Some MySQL releases introduce changes to the grant tables to add new privileges or features. To make sure that you can take advantage of any new capabilities, update your grant tables to the current structure whenever you upgrade MySQL. See Section 2.11, ‘Upgrading MySQL’⁵.”

When looking further into the GRANT statement documentation there is stated that MySQL offers global, database, table, column, stored routine, and proxy user privileges.

³ "Security in MySQL :: 4 Access Control and Account Management."
<https://dev.mysql.com/doc/mysql-security-excerpt/8.0/en/access-control.html>. Geopend op 13 nov.. 2020.

⁴ "MySQL 8.0 Reference Manual :: 6.2.2 Privileges ... - MySQL."
<https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html>. Geopend op 3 dec.. 2020.

⁵ "2.11 Upgrading MySQL - MySQL :: Developer Zone."
<https://dev.mysql.com/doc/refman/8.0/en/upgrading.html>. Geopend op 3 dec.. 2020.

So MySQL has a privilege system and possible database, table, column, stored routine and proxy user specific access control. How to apply this?

A basic security principle is to only allow strictly what is necessary. This is called the principle of least privilege⁶. This would mean that the database user that the application is using does not have any more permissions than needed. It is likely that the system creating the database records (researcher system) only must insert new records into the database. Researchers will probably not have to be able to view inserted data. Maybe, in addition to INSERT permissions, this user may be able to edit the last inserted record in case of a mistake. In that case the user also needs UPDATE permissions.

The system accessing and viewing the data (as a user/test subject) should not have any rights other than SELECT. This can even be narrowed down to table and column specific permissions.

When using multiple components that are accessing the database, the attack surface of the system can be reduced by creating component-specific database users with their own roles. A vulnerability in one of the components will then have a smaller impact because of the reduced permissions.

The MySQL authentication is already based on username, password, and hostname. Additional factors for authentication are not supported by MySQL but can be accomplished using Linux PAM's (Pluggable Authentication Module). This however does not seem necessary because the standard authentication is already based on the users' hostname.

Library - Community research

Users on Stackoverflow confirm⁷ that this is the way to set up access control in MySQL. Steps described are creating a user with a specific hostname:

```
CREATE USER 'user'@'hostname';
```

And granting certain access to specific resources:

```
GRANT ALL PRIVILEGES ON dbTest.* To 'user'@'hostname' IDENTIFIED BY  
'password';
```

ON dbTest.* grants the privileges to database dbTest on all tables. 'ALL' can be replaced by certain actions like for example 'SELECT, UPDATE'. After an action, columns can be added as follows: SELECT ('', COLUMN_NAME, ''). It is possible to grant privileges to a certain table by replacing the * character. Using IDENTIFIED BY the password can be set.

⁶ "Principle of least privilege - Wikipedia." https://en.wikipedia.org/wiki/Principle_of_least_privilege. Geopend op 3 dec.. 2020.

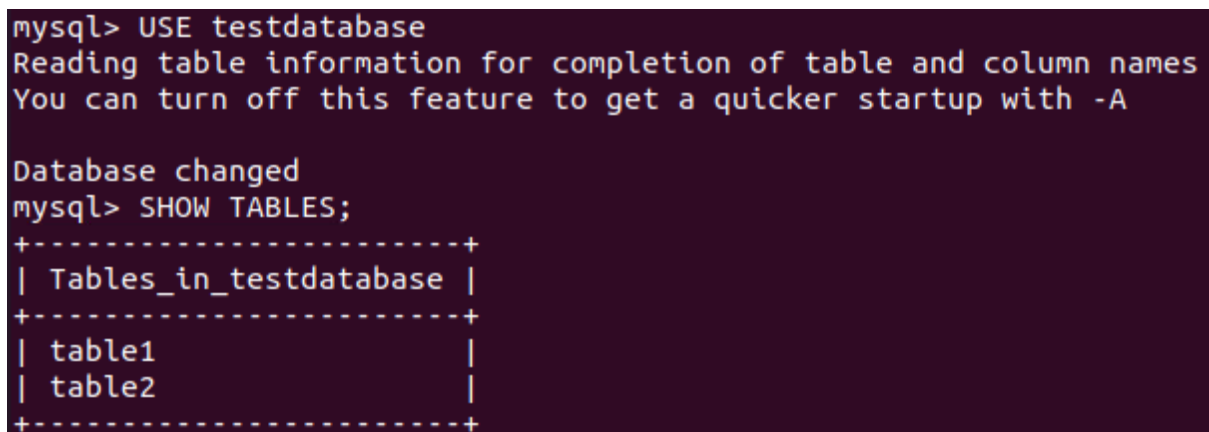
⁷ "Create new user in MySQL and give it full access to one" 24 mrt.. 2016, <https://stackoverflow.com/questions/1720244/create-new-user-in-mysql-and-give-it-full-access-to-one-database>. Geopend op 13 nov.. 2020.

Workshop - Prototyping

To confirm these findings a prototype has been set up. This is an Ubuntu Desktop 20.4 machine on host 192.168.44.10 with mysql-server installed, and an Ubuntu Desktop 20.4 machine on host 192.168.44.11 with mysql-client installed. On the mysql-server, two databases are created, and two tables are added to the first database with the following commands:

```
CREATE DATABASE testdatabase;  
CREATE DATABASE testdatabase2;  
USE testdatabase;  
CREATE TABLE table1 (int id);  
CREATE TABLE table2 (int id);
```

The creation of the two tables in testdatabase can be confirmed using the `SHOW TABLES;` query.



```
mysql> USE testdatabase  
Reading table information for completion of table and column names  
You can turn off this feature to get a quicker startup with -A  
  
Database changed  
mysql> SHOW TABLES;  
+-----+  
| Tables_in_testdatabase |  
+-----+  
| table1                  |  
| table2                  |  
+-----+
```

Figure 1: Created tables in testdatabase

After these databases and tables were created, a user has been created that can access the database from the second host (192.168.44.11).

```
CREATE USER 'testuser'@'192.168.44.11' IDENTIFIED BY 'uwuowo';
```

Finally, this user gets privileges on testdatabase and on table1. This means that the user should not be able to access testdatabase1, and should also not be able to access table testdatabase.table2.

```
GRANT ALL PRIVILIGES ON testdatabase.table1 TO 'testuser'@'192.168.44.11';
```


Connecting from the second host, 192.168.44.11, to testdatabase1, gives the following error, as expected:

```
student@student-virtual-machine:~$ sudo mysql -u testuser -p'uwuowo' -h 192.168.44.10 -P 3306 -D testdatabase1
mysql: [Warning] Using a password on the command line interface can be insecure.
ERROR 1044 (42000): Access denied for user 'testuser'@'192.168.44.11' to database 'testdatabase1'
```

Figure 2: Access denied for user on different database

However, connecting to testdatabase authorizes the user:

```
student@student-virtual-machine:~$ sudo mysql -u testuser -p'uwuowo' -h 192.168.44.10 -P 3306 -D testdatabase
mysql: [Warning] Using a password on the command line interface can be insecure.
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 18
Server version: 8.0.22-0ubuntu0.20.04.3 (Ubuntu)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Figure 3: Access granted on testdatabase

After logging in to testdatabase, the user can retrieve the tables with `SHOW TABLES;`. Only the tables that this user has access to, will be shown:

```
mysql> SHOW TABLES;
+-----+
| Tables_in_testdatabase |
+-----+
| table1                  |
+-----+
1 row in set (0.01 sec)
```

Figure 4: Tables the user has access to

When the user tries to `SELECT` from table2, a permission error will be thrown:

```
mysql> SELECT * FROM table2;
ERROR 1142 (42000): SELECT command denied to user 'testuser'@'192.168.44.11' for table 'table2'
```

Figure 5: Access denied on tables the user does not have access to

However, `SELECT`ing from table1 does work, as configured:

```
mysql> SELECT * FROM table1;
Empty set (0.00 sec)
```

Figure 6: Access granted to table1

The last thing that can be testing is connecting the testuser from another host, to the correct database (testdatabase). The static IP has been changed to 192.168.44.12 and the following command was issued:

```
student@student-virtual-machine:~$ sudo mysql -u testuser -p'uwuowo' -h 192.168.44.10 -P 3306 -D testdatabase
mysql: [Warning] Using a password on the command line interface can be insecure.
ERROR 1130 (HY000): Host '192.168.44.12' is not allowed to connect to this MySQL server
```

Figure 7: Access denied from different host

As expected, this host is not allowed.

What is the best way to encrypt a database and its contents?

To answer this sub question, the first thing that was done is a literature study from the official documentation of MySQL on this subject. After that, a community research needs to be done. Finally, the results need to be tested, so a prototype is created.

Literature study

According to MySQL's official website, there are two different ways to encrypt the database: RSA encryption⁸ and TDE⁹.

First the RSA encryption, this method of encryption is done with a public and private key generation.



Figure 8: Asymmetric encryption (RSA)

MySQL Enterprise Encryption offers the following encryption tools regarding RSA:

- Asymmetric Public Key Encryption (RSA)
- Asymmetric Private Key Encryption (RSA)
- Generate Public/Private Key (RSA, DSA, DH)
- Derive Symmetric Keys from Public and Private Key pairs (DH)
- Digitally Sign Data (RSA, DSA)
- Verify Data Signature (RSA, DSA)
- Validation Data Authenticity (RSA, DSA)

In a nutshell, this is what MySQL Enterprise encryption offers:

- **Secure data using combination of public, private and symmetric keys** to encrypt and decrypt data.
- **Encrypt data stored in MySQL** using RSA, DSA, or DH encryption algorithms.
- **Digitally sign messages to confirm the authenticity** of the sender (non-repudiation) and the integrity of the message.
- **Eliminate unnecessary exposure to data** by enabling DBAs to manage encrypted data.
- **Interoperate with the other cryptographic systems** and appliances without changing existing applications.
- **Avoid exposure of asymmetric keys** within client applications or on disk.

⁸ "MySQL Enterprise Encryption - MySQL."

<https://www.mysql.com/products/enterprise/encryption.html>. Geopend op 5 nov.. 2020.

⁹ "MySQL Enterprise Transparent Data Encryption (TDE) - MySQL."

<https://www.mysql.com/products/enterprise/tde.html>. Geopend op 5 nov.. 2020.

MySQL Enterprise TDE offers a different way of encryption: via AES. They offer data-at-rest encryption by encrypting the physical files of the database. As a result, hackers and malicious users are unable to read sensitive data from tablespace files, database backups or disks.

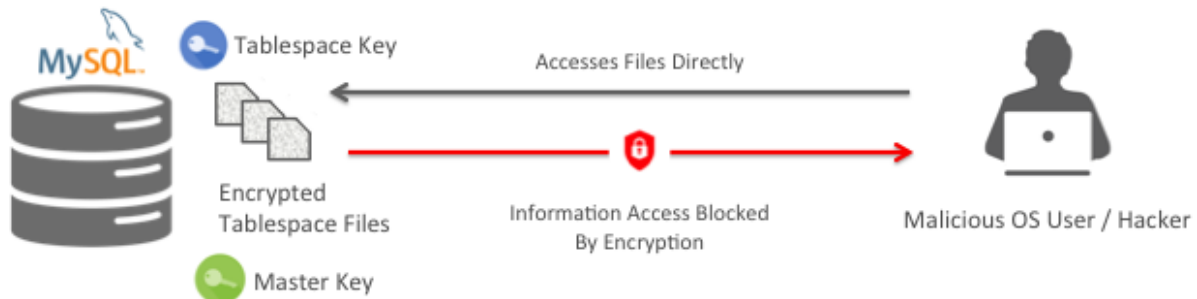


Figure 9: symmetric encryption (TDE)

TDE uses a two-tier encryption key architecture, consisting of a master encryption key and tablespace keys providing easy key management and rotation. Tablespace keys are managed automatically over secure protocols while the master encryption key is stored in a centralized key management solution such as:

- Oasis KMIP-protocol implementations
 - Oracle Key Vault
 - Gemalto KeySecure
 - Thales Vormetric Key Management Server
 - Fernetix Key Orchestration
 - Townsend Alliance Key Manager
- MySQL Enterprise TDE also supports HTTPS based APIs for Key Management such as:
 - Hashicorp Vault
 - AWS KMS

Community research

GioMac¹⁰ states that AES and DES encryption is available by default¹¹. GioMac says that normal SQL queries commands like AES_ENCRYPT and AES_DECRYPT for field encryption can be used. Some examples to do this:

```
INSERT INTO users (username, password) VALUES ('root',
AES_ENCRYPT('somepassword', 'key12346123'));
```

```
SELECT AES_DECRYPT(password, 'key12346123') FROM users WHERE username =
'root';
```

¹⁰ "Whats a good way to encrypt a mysql database, and is it worth"

<https://serverfault.com/questions/538715/whats-a-good-way-to-encrypt-a-mysql-database-and-is-it-worth-it>. Geopend op 13 nov.. 2020.

¹¹ "MySQL 8.0 Reference Manual :: 12.14 Encryption ... - MySQL."

<https://dev.mysql.com/doc/refman/8.0/en/encryption-functions.html>. Geopend op 13 nov.. 2020.

Some people give reasons to not use AES. Noach¹² says that it does not matter that AES is used, but it does matter that the AES encryption key is sent with the query in plaintext. To solve this, a secure connection is required. That is the reason why SSL is a must.

Shankar¹³ shows some different ways of encrypting full tables and schemas. The following command is for encrypting a table. The ENCRYPTION keyword can be changed to either 'y' (yes) or 'n' (no).

```
CREATE TABLE db1.t1 ENCRYPTION='y';
```

To create a schema with all tables inside encrypted, following command is used:

```
CREATE SCHEMA db1 DEFAULT ENCRYPTION='y';
```

Prototyping

A prototype is made to test which methods (still) work. The enterprise edition cannot be installed for this prototype, so the only methods to test are the AES function and the table/scheme encryption.

This prototype is created on an Ubuntu 20.04 server with the default MySQL package. Now the encryption schemes process can commence.

Encrypting schema's

The first command that is tested is the following command:

```
CREATE SCHEMA testDB_schema DEFAULT ENCRYPTION='y';
```

and the newly created schema needs to be used:

```
USE testDB_schema;
```

When creating a table, the following error occurred:

```
ERROR 3185 (HY000): Can't find master key from keyring, please check in the server log if a keyring plugin is loaded and initialized successfully.
```

To solve this error, a Keyring plugin is required. This plugin generates encryption keys and stores them properly. It can be installed using the following command.

```
INSTALL PLUGIN keyring_file SONAME 'keyring_file.so';
```

¹² "Is "Why Should You Avoid AES In MySQL?" true? - Information" 21 nov.. 2013, <https://security.stackexchange.com/questions/45838/is-why-should-you-avoid-aes-in-mysql-true>. Geopend op 13 nov.. 2020.

¹³ "Controlling table encryption in MySQL 8.0 | MySQL Server Blog." 17 mrt.. 2020, <https://mysqlserverteam.com/controlling-table-encryption-in-mysql-8-0/>. Geopend op 13 nov.. 2020.

Now to create the table:

```
CREATE TABLE table1(id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY);
```

This will return an OK query and the tables are encrypted.

Using AES in the query

First, an extra column needed to be added to the already existing table with the following command.

```
ALTER TABLE table1 ADD COLUMN enc VARBINARY(255) AFTER id;
```

Next, an item can be inserted. The following command should encrypt the data with AES. The string that is encrypted is 'apples' and it is encrypted with the key 'abcdefghijklmnopqrstuvwxyz'.

```
INSERT INTO table1 (enc) VALUES (AES_ENCRYPT('apples', 'abcdefghijklmnopqrstuvwxyz'));
```

That way, it can be seen that the string is encrypted in the database as seen in figure 10.

```
mysql> SELECT * FROM table1;
+-----+-----+-----+
| id | enc |
+-----+-----+-----+
| 123 | 0x |
| 125 | 0x21C5CE68CC388714B19FA27732C43763 |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

Figure 10: the encrypted displayed string

Now to decrypt the string. This can be done via the following command with the right key.

```
SELECT id, AES_DECRYPT(enc, 'abcdefghijklmnopqrstuvwxyz') FROM table1 WHERE id=125;
```

This returns a not readable hexadecimal string as seen in figure 11.

```
mysql> SELECT id, AES_DECRYPT(enc, 'abcdefghijklmnopqrstuvwxyz') FROM table1 WHERE id=125;
+-----+-----+-----+
| id | AES_DECRYPT(enc, 'abcdefghijklmnopqrstuvwxyz') |
+-----+-----+-----+
| 125 | 0x6170706C6573 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

Figure 11: decrypted hexadecimal string

To solve this problem and make the string readable, the AES binaries shouldn't be stored as hexadecimals. This can be changed in the file in /etc/mysql/conf.d/mysql.cnf. The following snippet should be added to the configuration file. After that, restart the mysql instance.

```
[mysql]
binary-as-hex = false
```

This will store the data as seen in figure 12 and eventually, when decrypting, give back the result as seen in figure 13.

```
mysql> SELECT * FROM table1;
+-----+-----+
| id  | enc                |
+-----+-----+
| 123 | NULL              |
| 125 | !\h8w27c          |
+-----+-----+
2 rows in set (0.01 sec)
```

Figure 12: the encrypted string

```
mysql> SELECT id, AES_DECRYPT(enc, 'abcdefghijklmnop') FROM table1 WHERE id=125;
+-----+-----+
| id  | AES_DECRYPT(enc, 'abcdefghijklmnop') |
+-----+-----+
| 125 | apples                             |
+-----+-----+
1 row in set (0.00 sec)
```

Figure 13: the decrypted string

How can CRUD methods be applied in a secure way with MySQL?

For this research question literature study and community research is used to read about what others have used/told about this problem. Also, best, good, and bad practices are used to answer this question. All in all, other ones their experiences are used to answer this question. Three methods of communication to a database came out of a brainstorm session.

Workshop - Brainstorm & Library - Community Research

In this sub question, the following methods are researched and compared to each other to see which is the best to use in this project for the most secure way of processing CRUD methods:

- SQL queries embedded in code
- Stored procedures
- Using an ORM

Looking at applying CRUD methods in a secure way, SQLi (SQL injection) is a vulnerability on the top ten list of most common database vulnerabilities on zdnet¹⁴ that can be a problem for doing a secure CRUD operation with the database. So, the methods have to be prevented against this.

¹⁴ "The top ten most common database security vulnerabilities" 26 jun.. 2013, <https://www.zdnet.com/article/the-top-ten-most-common-database-security-vulnerabilities/>. Geopend op 2 dec.. 2020.

Library - Community Research & Best good and bad practices

Stored procedures vs queries embedded in code

Bert Wagner¹⁵ (developer, filmmaker, and teacher) did some research on the speed difference of stored procedures and queries that are embedded in code.

According to Wagner¹⁶, stored procedures are faster than SQL queries because of the database caching. It uses the same query out of the cached plans in the database. When using queries embedded in code with one little difference, the database will not reuse the query, so it must load it again. It is much less likely that a query inside of a stored procedure will change compared to a query that is embedded in code. Because of this, it is more likely that the stored procedure plans are being run from cached plans.

Stored procedures vs ORM^{17 18 19}

- Stored procedures are precompiled so are faster than the SQL queries that are generated by the ORM.
- Stored procedures are harder to maintain because they need to be typed by hand. An ORM will be generating queries and tables by looking at the class code.
- By using an ORM no queries must be created, and it is easier to change to another database.
- ORM comes with more code base, so a (little) slower application.
- Both stored procedures and ORM eliminate many SQLi vulnerabilities.
- Sometimes it takes a lot of tweaking to make more difficult queries to work by using an ORM.
- Sometimes it is not even possible to make a complex query work with an ORM and it must be typed out (as a Stored procedure) by hand, because the ORM can't make it work.
- "ORMs(Object-relational mapping) are not mutually exclusive with Stored Procedures. Most ORMs can utilize stored procedures. Most ORMs generate Stored Procedures if you so choose."
- "ORMs may generate unacceptable SQL (in terms of performance) and you may sometimes want to override that SQL with hand-crafted SQL. One of the ways to accomplish this is by using SPs(stored procedures)."

¹⁵ "Are Stored Procedures Faster Than Stand-Alone Queries?." 15 okt.. 2019, <https://bertwagner.com/posts/are-stored-procedures-faster-than-stand-alone-queries/>. Geopend op 2 dec.. 2020.

¹⁶ "Whitespace, Letter Case, and Other Things That Prevent Plan" 2 sep.. 2019, <https://bertwagner.com/2019/09/03/whitespace-letter-case-and-other-things-that-prevent-plan-reuse/>. Geopend op 2 dec.. 2020.

¹⁷ "When not to use ORM and prefer stored procedures" <https://softwareengineering.stackexchange.com/questions/139319/when-not-to-use-orm-and-prefer-stored-procedures>. Geopend op 2 dec.. 2020.

¹⁸ "Stored Procedures, ORMs, and GraphQL | Blog - Ardalís." 26 apr.. 2020, <https://ardalis.com/stored-procedures-orms-and-graphql/>. Geopend op 2 dec.. 2020.

¹⁹ "Stored Procedures and ORM's - Stack Overflow." 18 mrt.. 2011, <https://stackoverflow.com/questions/5346601/stored-procedures-and-orms>. Geopend op 2 dec.. 2020.

ORM vs queries embedded in code^{20 21}

- Working with an ORM and knowing what the strengths and weaknesses are, can save a lot of time creating tables with correct data and relations.
- For applications that make simple requests for getting some data and show it on a page, in many cases an ORM can be the best way to go. Sometimes it even faster because it will cache objects it has seen before, that otherwise would have queried the database multiple times.
- For applications that are reporting-heavy or deal with many database rows per request, an ORM has more load, and the caching that they do are mostly big and useless. So, in that case it's better to use queries embedded in code.
- An ORM eliminates many SQLi vulnerabilities that queries embedded in code may have.

Library - Literature study

Stored procedure against SQLi

As explained in Stored Procedure SQL Injection - Understand Attacks²², it is possible to create procedures vulnerable to SQLi. As recommended in their previous defense articles, using parameterized statements (see figure 14) is always the best option. It also applies to stored procedure code; however input sanitizing is still a valid alternative.

But as said by marcelm²³ about sanitizing input for parameterized queries:

"Input sanitization is a horrible term that pretends you can wave a magic wand at data and make it "safe data". The problem is that the definition of "safe" changes when the data is interpreted by different pieces of software.

So, what should we do? Make sure the data is never in a position to do harm.

The best way to achieve this is to avoid interpretation of the data in the first place. Using parameterized SQL queries is an excellent example of this; the parameters are never interpreted as SQL, they're simply put in the database as, well, data."

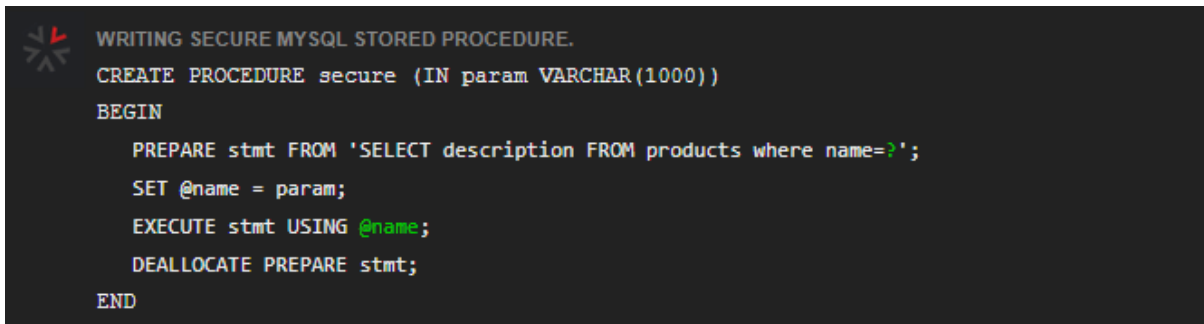
²⁰ "Using an ORM or plain SQL? - Stack Overflow." 30 jan.. 2009, <https://stackoverflow.com/questions/494816/using-an-orm-or-plain-sql>. Geopend op 2 dec.. 2020.

²¹ "What is your personal preference, orm vs plain sql query?: PHP." https://www.reddit.com/r/PHP/comments/9p7lys/what_is_your_personal_preference_orm_vs_plain_sql/. Geopend op 2 dec.. 2020.

²² "Stored Procedure SQL Injection - Understand Attacks." <https://www.sqlinjection.net/advanced/stored-procedure/>. Geopend op 2 dec.. 2020.

²³ "Sanitizing input for parameterized queries - Information" 27 okt.. 2017, <https://security.stackexchange.com/questions/172297/sanitizing-input-for-parameterized-queries>. Geopend op 10 dec.. 2020.

In the following stored procedure on the figure that is shown on [sqlinjection.net](https://www.sqlinjection.net)²⁴ is immunized against SQLi because of using parameterized statements.



```
WRITING SECURE MYSQL STORED PROCEDURE.  
CREATE PROCEDURE secure (IN param VARCHAR(1000))  
BEGIN  
    PREPARE stmt FROM 'SELECT description FROM products where name=?';  
    SET @name = param;  
    EXECUTE stmt USING @name;  
    DEALLOCATE PREPARE stmt;  
END
```

Figure 14: Example of creating a stored procedure in MySQL database

As said on [sqlshack](https://www.sqlshack.com)²⁵: "Stored procedures are more rigid than the application code. They accept a known set of parameters and return predictable results. For very common or sensitive processes, this can be a positive way to ensure that important code executes and performs the same way every time. In addition to writing secure TSQL within the proc, we can assign permissions to a stored procedure separately from underlying tables, allowing us to greatly restrict and customize who can or cannot access it."

²⁴ "Secure Stored Procedure Against SQL Injection - Defense."
https://www.sqlinjection.net/defense/stored-procedure/?utm_source=rss&utm_medium=rss&utm_campaign=secure-stored-procedure-programming. Geopend op 2 dec.. 2020.

²⁵ "SQL Injection: Detection and prevention - SQLShack." 30 aug.. 2019,
<https://www.sqlshack.com/sql-injection-detection-and-prevention/>. Geopend op 2 dec.. 2020.

How can the database be securely backed up regularly?

To prove that this research is valid there will firstly be done with community research. After the community research the findings will be discussed with the stakeholders and the most suitable methods will be selected. Finally, the most suitable methods will be prototyped in order to prove that the methods are valid.

Library - Literature study

According to Jotform²⁶, there are different methods to securely backup databases on a regular basis. The best technique will depend on the specifications of the database as well as the host machine. If different servers are used as the database servers a service like BackupBird²⁷ should be used. Backupbird is a paid service so this option should only be considered if the necessary funds are acquired. AutoMySQLBackup²⁸ is an alternative which is free but there have been multiple comments on the application as it doesn't smoothly support multiple server locations. If the databases are installed on a single location with multiple servers, it would be sufficient to use AutoMySQLBackup.

MySQLDump²⁹ could also be used. This is a utility from SQL itself used for backing up. A backup can be written into a file, another server or compressed gzip file. This process can be automated on a time basis, cron could be used on Unix/Linux OS to do time-based scheduling. Another tool is sqlbackupandftp³⁰. This is a tool with an UI to use for backing up. automated backups can also be made using this tool.

For a backup which is not posted to a server but into a file is also possible. As an example, create a backup using phpMyAdmin. Go to export in phpMyAdmin and export the database into a file, this action will also automatically make the default browser download the aforementioned file. Besides this method a backup could also be created with the use of ssh but a tool like MySQLDump will still be required to create a backup.

MySQLDump, sqlbackupandftp and file backup are secure because it is performed on the host machine and does not have to be transferred through the internet. This does mean that the host machine itself needs to be secure.

²⁶ "10 Ways to Automatically & Manually Backup MySQL Database." 15 Mar. 2009, <https://www.jotform.com/blog/how-to-backup-mysql-database/>. Accessed 2 Dec. 2020.

²⁷ "Backup Bird." <https://www.backupbird.com/>. Geopend op 2 dec.. 2020.

²⁸ "AutoMySQLBackup download | SourceForge.net." <https://sourceforge.net/projects/automysqlbackup/>. Geopend op 2 dec.. 2020.

²⁹ "Diary of A Webmaster Part 4 - Backing Up With MySQLDump" 27 feb.. 2002, <https://www.sitepoint.com/backing-up-mysqldump/>. Geopend op 2 dec.. 2020.

³⁰ "Free MySQL Backup Tool - SQL Backup and FTP." <https://sqlbackupandftp.com/mysql-backup>. Geopend op 2 dec.. 2020.

Field - Stakeholder analysis

Question: Which server(s) are chosen as database host(s), or will you host the database yourself?

Answer: Azure is being used by 1 of the stakeholders while AWS is used by the other stakeholder.

Workshop - Prototyping

If used through AWS or Azure, there are options from the provider itself to backup. Those options are selected at the creation of the DB service. If the right setup was not chosen at the start, it can always be changed later in the settings.

If Azure is used for the database, the settings will be set during the setup of the database.

Home > Azure Database for MySQL servers > Select Azure Database for MySQL deployment option > Create MySQL server >

Pricing tier

Basic
Up to 2 vCores with
Variable I/O performance (1-2 vCores)

General Purpose
Up to 64 vCores with
predictable I/O performance (2-64 vCores)

Memory Optimized
Up to 32 memory optimized vCores with
predictable I/O performance (2-32 vCores)

Please note that changing to and from the Basic compute tier or changing the backup redundancy options after server creation is not supported.

Compute Generation - [Learn more about compute generation](#)

Gen 5

vCore - [What is a vCore?](#)

4 vCores

Storage cannot be scaled down

Storage (Type: General Purpose Storage)

100 GB

Backup Retention Period

CAN USE UP TO 300 available IOPS 22 Days

Backup Redundancy Options - [Learn more about backup redundancy options](#)

Locally Redundant
Recover from data loss within region

Geo-Redundant
Recover from regional outage or disaster

PRICE SUMMARY

Gen 5 Compute generation

Cost per vCore

54.91

vCores selected

4

General Purpose Storage

Cost per GB / month

0.10

Storage selected (in GB)

100

EST. MONTHLY COST

229.32 EUR

Additional charge per usage

See pricing details for more detail.

Figure 15: azure database settings with backup settings on the bottom

Backup Retention Period

CAN USE UP TO 300 available IOPS 22 Days

Backup Redundancy Options - [Learn more about backup redundancy options](#)

Locally Redundant
Recover from data loss within region

Geo-Redundant
Recover from regional outage or disaster

Figure 16: close-up of azure database backup settings

In figure 15 the settings are determined as well as how to backup automatically. Even the basic tier has automated backups, but these will only be a snapshot of the database. Database snapshots have less functionality than database backups. The snapshot is done on a specific time slot which means that you do not have the privilege of selecting separate time slots. A snapshot also must be on the host machine while a backup can but does not


have to be on the host machine. For more information visit SQLShack³¹. The settings can always be tweaked later if the requirements change.

At AWS there are the following backup settings for a MySQL Databases³², see figure 17. If a MySQL server is bought on AWS,³³ there are no options for backups as it is already included in the product. All the databases which are bought at AWS regardless of which type it is will be covered. So, if a service of PostgreSQL is run with 250 GiB-month and a MySQL service of 500 GiB-month, a backup availability of 750 GiB-month will be available for the database.

Backup

Creates a point-in-time snapshot of your database

- ☒ **Enable automatic backups**
Enabling backups will automatically create backups of your database during a certain time window.

 Please note that automated backups are currently supported for InnoDB storage engine only. If you are using MyISAM, refer to details [here](#).

Backup retention period [Info](#)

Choose the number of days that RDS should retain automatic backups for this instance.

1 day ▼

Backup window [Info](#)

Select the period for which you want automated backups of the database to be created by Amazon RDS.

- ☐ Select window
☒ No preference

- ☒ Copy tags to snapshots

Figure 17: AWS database backup settings

³¹ "Understanding Database snapshots vs Database backups in" 4 May. 2018, <https://www.sqlshack.com/understanding-database-snapshots-vs-database-backups-in-sql-server/>. Accessed 10 Dec. 2020.

³² "Backup and restore - Azure portal - Microsoft Docs." 30 Jun. 2020, <https://docs.microsoft.com/en-us/azure/mysql/howto-restore-server-portal>. Accessed 2 Dec. 2020.

³³ "Amazon RDS for MySQL Pricing – Amazon Web Services." <https://aws.amazon.com/rds/mysql/pricing/>. Accessed 2 Dec. 2020.

Conclusion

Now to answer the main question: How can a MySQL database be secured? This main question is divided into multiple sub questions. The results of these sub questions are found here, as well as the conclusion of the main research question.

First, access control should be handled correctly. Database users should only be allowed to do what they are supposed to do. This is the principle of least privilege and can be realized with the MySQL privilege system. Users can be given specific permissions, allowing access only to specific tables, columns or even actions. Separate software components of a system can be assigned specific users with only the required permissions to reduce the attack surface of the system.

When someone has access to the database device, they should not have or gain access to the database files. This can be solved by encrypting the database. There are three types of database encryption:

- Database/schema encryption
- Table encryption
- Field encryption

MySQL Enterprise edition offers multiple ways to encrypt the database. Table encryption can be done easily by adding "ENCRYPTION='y'" when creating a table. Field encryption can be done by adding "AES_ENCRYPT" around the function but do pay attention since the requests are sent with the AES-key with the request thus using SSL/TLS is a must. AES_ENCRYPT is not hashing, so it should not be used for hashing passwords.

There are three common ways of applying CRUD methods to databases: embedded in code, using an ORM, or using stored procedures. In terms of security, embedded in code fails. This is because of the big risk of human mistakes, which can cause vulnerabilities like SQLi. ORM takes care of this, but because of big amounts of caching causes ORMs to be quite slow. Stored procedures are a lot quicker because they are pre-compiled. To prevent vulnerabilities in stored procedures, prepared statements and/or input sanitizing must be used. This means that stored procedures would be best if they are written securely. Otherwise, in terms of security, an ORM should be used. Also, a combination of using stored procedures and an ORM can be used as long as the known vulnerability (SQLi) is kept in mind.

There are different tools and methods for backing up MySQL databases, like BackupBird, AutoMySQLBackup, MySQLDump and sqlbackupandftp. In terms of security, MySQLDump and sqlbackupandftp are best, because these are used on the host machine, so data does not have to be transferred over the internet. When using a cloud provider like Azure, there is often a setting to enable back-ups of the database.

Advice

In relation to the corona test data platform, a few design concepts should be kept in mind. First, access control should be handled properly, according to the principle of least privilege. Depending on where the data is stored, we think that data should be encrypted, but most services like AWS and Azure will probably do this already. We strongly recommend you use SSL when connecting to your database. For implementing the CRUD methods using stored procedures is the best option when written securely. But using an ORM can also be done when a good performance is not a big must. Finally, backing up is very important. Most cloud providers will have built in functionality for this. When hosting the database locally, a secure and regular back-up tool should be used, like MySQLDump.