At Rest Encryption with MySQL & Vault

Charles Thompson  Empowered
Tate McDaniel  Percona
Welcome and Introductions
Charles Thompson
Empowered

Senior MySQL DBA with over seven years of experience in the industry. Proficient in server tuning, query/schema optimization, and troubleshooting. I've worked with small & large (400+ instance) scale organizations. I really enjoy scripting solutions to my problems!
Senior MySQL DBA with 15 years in the industry, 2 years at Percona. Expert at query tuning, schema design, and optimization.

In my spare time I do a lot of traveling. A lot. I actually live on a boat and sail around, so I guess you could say I’m always travelling. How is that for freedom? Want to work from “anywhere”, for real? Come talk to me about working for Percona!
Working From Home…
Agenda
What Is This Talk About?

1. Reasons for encryption using MySQL and other databases
2. What types of encryption are available to use?
3. When to use the different types of encryption (pros and cons)
4. An overview of a health care use case
5. Percona Server working with Hashicorp’s Vault to achieve data at rest encryption
6. Questions?
Reasons for Encryption

Modern demands for data security
Common Compliance Issues

- GDPR
- PCI DSS
- HIPAA
- Specialized industry (eg. DOD)
GDPR Requirements

“What level of encryption is required for GDPR?

32(1) of the General Data Protection Regulation to implement appropriate technical and organisational measures to secure personal data. The GDPR deliberately does not define which specific technical and organisational measures are considered suitable in each case, in order to accommodate individual factors."

Source: https://gdpr-info.eu/issues/encryption/
“PCI encryption requirements

Requirement 3 of the Payment Card Industry's Data Security Standard (PCI DSS) is to “protect stored cardholder data.” The public assumes merchants and financial institutions will protect data on payment cards to thwart theft and prevent unauthorized use.

... Encryption of cardholder data with strong cryptography is an acceptable method of rendering the data unreadable in order to meet PCI DSS Requirement 3.4”

Source: https://www.pcisecuritystandards.org/pdfs/pci_fs_data_storage.pdf
HIPAA Requirements (US healthcare)

“Furthermore, the HIPAA encryption requirements for transmission security state that covered entities should “implement a mechanism to encrypt PHI whenever deemed appropriate”. This instruction is considerably vague and open to interpretation”

Source: https://www.hipaajournal.com/hipaa-encryption-requirements/
Take Away…

Policy makers have “kicked the can” down the road and put the onus on the industry to protect data.
Other Reasons

- Data breach can ruin a business
- Ethical or moral responsibility to protect people
- Protecting business secrets from competition
Types of Encryption

Available in the MySQL Ecosystem
Types of Encryption

- Encryption at rest
- Encryption in transit
- Encryption in use
Encryption in Use

Applications can encrypt data before storing it and decrypt it once retrieved. The application takes responsibility for the data security.
Encryption in Use
Encryption in Transit
Encryption in Transit

Followed by use of a UNIX SOCKET connection instead of the TCP/IP mysql connection.
Encryption at Rest

Full Disk Encryption
Encryption at Rest

Encrypted Database Files
(Database Encryption)
Encryption at Rest

Application encryption still counts!
Choosing the Right Encryption

Pros and cons of the different types of encryption
One Thing We Can All Agree On

No matter who you are or what you do in the IT world, it is probably best to encrypt data in transit. For MySQL this means TLS + enforcing SSL on user accounts as well as replication users.

GRANT ... REQUIRE SSL;
GRANT ... REQUIRE X509;
The Exception Proves the Rule
Another Thing We Can All Agree On

MySQL hasn’t got to a point of maturity where “in use” data is encrypted. Use of debuggers like strace can give access to the unencrypted data in memory, etc.
The Exception Proves the Rule

Application encryption can ensure data in memory is encrypted.
Encryption at Rest - Costs
Encryption at the Volume/OS/Block Level

**PROs:**

- We just encrypt the volume or disk using one of the many tools available
- MySQL isn’t aware of any change
- Application isn’t aware of a change
- Cheap to do

**CONs:**

- Doesn’t protect us from insider threats
- Centralized key storage and compliance is problematic sometimes
Modern Considerations

<table>
<thead>
<tr>
<th>CONTAINER</th>
<th>CONTAINER</th>
<th>CONTAINER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tomcat</td>
<td>PHP</td>
<td>Static Binary</td>
</tr>
<tr>
<td>Java</td>
<td>MySQL</td>
<td>Alpine</td>
</tr>
<tr>
<td>Debian</td>
<td>Ubuntu</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kernel</td>
<td></td>
</tr>
</tbody>
</table>

kubernetes
Encryption at the Database Level

**PROs:**

- Protects from “inside” threats
- Can encrypt across volumes you don’t control
- Backup/restores are automatically encrypted
- Lower overhead 3-5% performance hit
- DBA controlled (no sys admin needed)
- Centralized key storage and compliance ready.
Encryption at the Database Level

CONs:

Still vulnerable to in memory attacks

More setup/complication

Loss of keys can be catastrophic
Encryption at the Application Level

PROs:

Database servers are protected at all levels automatically since data cannot be used without decryption

Most flexible

Very little overhead on databases (which are usually the choke point)
Encryption at the Application Level

CONs:

Many applications are not built with this in mind and are hard to change.

Full text and partial text search can be a nightmare.

Application shoulders the responsibility for key security.
What is Hashicorp’s Vault?

- Software for securely managing secrets
- A secret is anything that you want to tightly control access to, such as:
  - API Keys
  - Passwords
  - Certificates
- Vault can be controlled via Web UI or command-line
- Strong API using curl with lots of ways to authenticate
  ```
  curl -k -X GET -H X-Vault-Token:$TOKEN "$VAULT_ADDR/v1/secret/mysql" | python -m json.tool
  ```
- Updated regularly
## Pros of Vault Versus Keyring File

<table>
<thead>
<tr>
<th>Pros of Using Vault</th>
<th>Pros of Using Keyring File</th>
</tr>
</thead>
<tbody>
<tr>
<td>One centralized location for all keys that is easily managed</td>
<td>Do not have a single point of failure if Vault goes down</td>
</tr>
<tr>
<td>No backups of keyring file</td>
<td>No additional setup/overhead of Vault</td>
</tr>
<tr>
<td>Better security! Key is nowhere on the MySQL server itself</td>
<td></td>
</tr>
<tr>
<td>Powerful auditing capabilities</td>
<td></td>
</tr>
</tbody>
</table>
Why did we choose Vault?

- SOC 2 Compliance
- Fastest/easiest solution to put into production
- We already wanted a secrets management tool in-house
Setting Up Percona Server with Vault

- Must be running Percona Server 5.7.20+
- Configuration setup documentation can be found here:
- Backup/restore tutorial:
- Things that can be encrypted:
  - Binary Log
  - General Tablespaces
  - Tables
  - Temporary Files
  - Undo Tablespace
  - Redo Log
Encrypting Tables

ALTER INSTANCE ROTATE INNODB MASTER KEY;

Check Vault Web UI to ensure the key has been created. If you don’t see an entry for it, do not start encrypting tables!

If you see an entry, you’re good to start encrypting tables!

ALTER TABLE `db`.`table` ENCRYPTION = ‘Y’;

Be careful - Converting a table locks the table completely for duration of execution
Lessons Learned and Items to Note

- PS doesn’t support KV Version 2
- MySQL only uses Vault when it’s started
- Error messages are extremely vague
- No additional memory overhead
- Performance decrease is about 3-5% due read/write to disk and buffer pool operations
- Vault encryption key names are Base64 encoded
- Logical backups (mysql_dump/mydumper) are not affected by encryption
Questions and Discussion

Ask away!
Thank You to Our Sponsors
Rate Our Session

Introducing gh-ost: triggerless, painless, trusted online schema migrations

- Time: 11:20 - 12:10
- Location: Matterhorn 2

Tap to rate and review.