Bytebase

Documentation menu

MySQL security best practices

APR 10, 2025

1. Secure Installation and Initial Configuration

Minimal Installation

- Install only what you need to reduce your attack surface.
- Avoid unnecessary components, demo databases, and sample data.
- Secure file permissions on MySQL directories and configuration files.

Initial Hardening Checklist

1. **Secure the Root Account**: Set a strong password for the root user and implement password validation.
2. **Remove Anonymous Accounts**: Check for and remove any anonymous users that might exist by default.
3. **Remove Test Database**: Drop the test database and remove associated privileges.
4. **Restrict Network Access**: Configure MySQL to bind only to necessary network interfaces.
5. **Disable Remote Root Login**: Ensure root can only connect from localhost.
6. **Change Default Port**: Consider changing the default port (3306) to reduce automated scanning.
7. **Configure Error Logging**: Enable detailed error logging for security monitoring.
8. **Set Secure File Privileges**: Restrict where MySQL can read and write files.
9. **Disable LOCAL INFILE**: Prevent the potential exploitation of the `LOCA`

Book Demo

Bytebase

2. Authentication and User Management

Role and User Management

- Create dedicated user accounts with appropriate privileges.
- Implement role-based access control (MySQL 8.0+).
- Regularly audit user accounts to identify and remove unnecessary or outdated accounts.
- Set resource limits to prevent abuse.

Password Policies and Secure Password Storage

- Use strong authentication plugins (`caching_sha2_password` in MySQL 8.0+).
- Implement password validation with strong policies.
- Configure password expiration to force regular password changes.

Account Locking on Suspicious Activity: MySQL 8.0.19+ supports automatic account locking after failed login attempts. Configure this feature to protect against brute force attacks.

Restricting Superuser Access

- Limit root login locations to localhost only.
- Create administrative users with limited privileges instead of using root.
- Enable logging for all root activities.

3. Authorization and Access Control

Role-Based Access Control (RBAC): Create role hierarchies for more complex permission structures and assign roles to users based on their job functions.

Using GRANT and REVOKE Properly

- Grant only the specific privileges needed.

- Regularly review and revoke unnecessary privileges.
- Use `WITH GRANT OPTION` carefully.

Implementing Row-Level Security: MySQL 8.0 doesn't have built-in row-level security like PostgreSQL, but you can implement it using views with WHERE clauses or through stored procedures.

Dynamic Privileges: MySQL 8.0 introduced dynamic privileges for fine-grained control over administrative operations, allowing you to grant specific administrative capabilities without providing full admin access.

Securing Stored Procedures and Functions: Be cautious with SECURITY DEFINER in stored procedures, as it allows procedures to run with the privileges of the creator, which can be a security risk if not properly managed.

4. Data Encryption

Encryption in Transit

- Configure SSL/TLS in MySQL.
- Create and properly manage SSL/TLS certificates.
- Require encrypted connections for sensitive users or globally.
- Configure appropriate TLS versions and ciphers.

Encryption at Rest

- Enable tablespace encryption for database files.
- Configure the keyring for key management.
- Encrypt binary logs, redo logs, and undo logs.
- Consider encrypting the system tablespace and temporary tablespace.

Column-Level Encryption: For more granular control, encrypt specific columns containing sensitive data using MySQL's encryption functions or application-level encryption.

Encrypting Backup Files: Ensure that backup files are also encrypted to protect sensitive data in backups.

5. Network Security

- Bind MySQL to specific IP addresses instead of all interfaces.
- Change the default port to reduce automated scanning.
- Limit which hosts can connect to specific MySQL users.

Firewall Configuration: Configure host-level firewall rules to restrict MySQL access to only necessary sources.

Using VPNs and SSH Tunnels: Use VPNs or SSH tunneling to provide encrypted access to MySQL without exposing the port publicly.

Network Monitoring and Intrusion Detection

- Regularly monitor connections to your MySQL server.
- Set appropriate connection limits.
- Implement connection control to prevent brute force attacks.

6. Auditing and Monitoring

Check **MySQL Auditing**

7. Patching and Maintenance

Staying Informed About Security Announcements: Subscribe to the MySQL security announcements mailing list and regularly check the MySQL Security Advisories page.

Regular Configuration Reviews: Schedule regular reviews of your MySQL configuration to ensure security settings remain appropriate.

Maintaining User Accounts and Privileges: Regularly audit user accounts and privileges to identify and remove unnecessary access.

Upgrade Strategies

- For minor version updates, follow standard upgrade procedures.
- For major version upgrades, use a more cautious approach.
- Consider blue-green deployment for upgrades in production environments.
- Always test upgrades in staging environments first.

Bytebase

8. Backup and Disaster Recovery

Implementing Encrypted Backups: Always encrypt your MySQL backups to protect sensitive data.

Always encrypt your MySQL backups to protect sensitive data.

Secure Backup Storage: Apply the 3-2-1 backup rule:

- Keep 3 copies of your data
- Store backups on 2 different storage types
- Keep 1 copy offsite

Backup Authentication and Authorization: Create a dedicated backup user with minimal necessary privileges.

Automated Backup Verification: Regularly verify your backups to ensure they can be successfully restored.

Implementing High Availability: Consider high availability configurations to reduce downtime during incidents.

Testing Disaster Recovery Procedures: Regularly test your disaster recovery procedures to ensure they work when needed.

9. Advanced Security Techniques

MySQL Enterprise Firewall: MySQL Enterprise Edition includes a firewall that provides protection against SQL injection attacks by learning legitimate query patterns.

MySQL Enterprise Data Masking: For environments handling sensitive data, MySQL Enterprise Edition offers data masking capabilities to protect sensitive information.

OS-Level Protections: Enhance MySQL security with operating system level protections like AppArmor or SELinux.

Bytebase

10. Common Mistakes and Vulnerabilities

Even with the best intentions, MySQL databases often remain vulnerable due to common security mistakes.

Using Default or Weak Credentials: One of the most common and dangerous security mistakes is using default or weak credentials.

Excessive Privileges: Granting excessive privileges violates the principle of least privilege and increases the potential damage from compromised accounts.

SQL Injection Vulnerabilities: SQL injection remains one of the most dangerous threats to database security. Always use prepared statements in application code.

Unencrypted Connections: Transmitting data in plaintext exposes it to interception. Enable SSL/TLS and require encrypted connections.

Failing to Apply Security Patches: Failing to apply security patches promptly leaves databases vulnerable to known exploits.

Insecure Configuration Settings: Default MySQL configurations often prioritize convenience over security. Review and secure your configuration.

Lack of Monitoring and Auditing: Without proper monitoring, security breaches can go undetected. Enable appropriate logging and regularly review logs.

Bytebase enhances database security by streamlining access controls, data masking, audit trails, and ensuring data integrity across all changes.

Edit this page on GitHub

Next article
How to Configure MySQL SSL Connection →

← Previous article
How to install MySQL Shell on Mac

Bytebase

All-in-One Database Workflows

Schema migration, data fix, just-in-time access, data masking, and audit logging in one place.

LIVE DEMO

AICPA SOC

COMPARISONS

vs. Liquibase

vs. Flyway

vs. DataGrip

vs. DBeaver

vs. CloudBeaver

vs. Navicat

vs. Metabase

vs. schemachange

vs. Jira

PRODUCT

Pricing

Changelog

Documentation

API

Supported Databases

RESOURCES

Resources

Terms

Policy

Partners

Bytebase

COMPANY

About

Brand

Contact

Bytebase