melapress

UPDATED ON 25 NOVEMBER, 2024 BY JOEL BARBARA. | PUBLISHED ON 6 MARCH, 2024

Secure WordPress database: 13 MySQL security best practices (+1 bonus)

WordPress, the most popular CMS, runs on MySQL – one of the most popular databases out there. Spending time to secure your WordPress database and ensure it is adequately hardened against common attack vectors can help you reduce risks. This is especially true if you are managing your MySQL server yourself.

It is worth noting that many WordPress installations use MariaDB, which is a fork of MySQL. As both work very similarly, we will use MySQL to mean both MySQL and MariaDB. Regardless of which RDMS flavor you're running, hardening your MySQL can help you minimize the risks of attacks from hackers. However, this does not replace other security measures, such as installing a web application firewall, ensuring you have the latest version of plugins, themes, and WordPress, and hardening WordPress.

Heads up: This article is targeted at MySQL 8.0 running on Linux (Ubuntu). While the concepts will translate to other operating systems and MySQL/MariaDB versions, the commands and file paths used in these

examples may differ. You should always test any changes in a staging or pre-production environment before making any changes to a production system.

Table of contents

- Why secure the WordPress database?
- Best practice 1: Run MySQL on a dedicated machine
- Best practice 2: Use the mysql_secure_installation script
- Best practice 3: Ensure that mysqld is not started with the –skip-grant-tables argument
- Best practice 4: Keep MySQL up-to-date
- Best practice 5: Bind MySQL to an IP address
- Best practice 6: Limit the use of web-based GUI tools
- Best practice 7: Run the MySQL daemon using a dedicated user
- Best practice 8: Create a dedicated WordPress database user
- Best practice 9: Ensure that local_infile is disabled
- Best practice 10: Disable MySQL command history
- Best practice 11: Back up your database
  - Take frequent backups
  - Verify the integrity of your backups frequently
  - Store your backups securely
- Best practice 12: Enable and enforce TLS connections
- Best practice 13: Change the table prefix
- Bonus: Consider using a Database as a Service (DBaaS)
- How to implement WordPress database security changes
- Frequently Asked Questions
- Keeping your database and WordPress secure

Why secure the WordPress database?

A secure WordPress database is an important part of your overall WordPress site security. Just like proper file permissions, secure password policies, limiting failed login attempts, and everything in between, database security can help you protect sensitive data, avoid data corruption, and reduce certain security risks, such as SQL injection attacks.

In this article, which is primarily aimed at those managing their own MYSQL, we offer several tips and tutorials on how to secure MySQL. We cover everything from changing the default database prefix and default settings to which options you should enable or disable. Even so, the extensive list of best practices presented in this article is worth a read for anyone managing WordPress websites.

Best practice 1: Run MySQL on a dedicated machine

Many WordPress installations run MySQL, PHP, and a web server (such as Nginx or Apache HTTP Server) on the same machine. This is not optimal – both in terms of performance and security. MySQL should ideally run on a dedicated server to reduce the blast radius of an attack. If an attacker manages to compromise and escalate privileges on the web server, it would be much harder for that attacker to move laterally and compromise the MySQL server.

Best practice 2: Use the mysql_secure_installation script

The mysql-server package comes with a shell script utility called mysql_secure_installation. You can use this script to set up a secure starting point for the MySQL server. As such, you should run it after a fresh install of MySQL. This utility helps you:

- Set a password for root accounts
- Remove root accounts that are accessible from outside localhost
- Remove anonymous user accounts
- Remove the test database (which, by default, can be accessed by anonymous users)

To invoke mysql_secure_installation, run the following command:

```
sudo mysql_secure_installation
```

Once the setup process begins, you will be presented with several prompts asking you whether you want to enable the validate password plugin. This is used to test the strength of passwords you pick for MySQL users. Enabling this plugin is highly recommended.

```
Securing the MySQL server deployment.

Connecting to MySQL using a blank password.

VALIDATE PASSWORD PLUGIN can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD plugin?

Press y|Y for Yes, any other key for No: y
```

After you enable the validate password plugin, the script will ask you to specify a password validation policy. Here, you should choose a strong password policy. You will subsequently be asked to reset the root user's password.

```
There are three levels of password validation policy:

LOW    Length >= 8
MEDIUM Length >= 8, numeric, mixed case, and special characters
STRONG Length >= 8, numeric, mixed case, special characters and dictionary
file

Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 2
Please set the password for root here.

New password:
```

```
Re-enter new password:

Estimated strength of the password: 100
Do you wish to continue with the password provided?(Press y|Y for Yes, any
other key for No) : y
```

Next, the script will prompt you to remove anonymous MySQL users. This is important to reduce any chance of attackers gaining access to the database server by leveraging an anonymous MySQL user.

```
By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to have
a user account created for them. This is intended only for
testing, and to make the installation go a bit smoother.
You should remove them before moving into a production
environment.

Remove anonymous users? (Press y|Y for Yes, any other key for No) : y
```

The next prompt will ask you if you would like to disable logins using the root user when authenticating remotely to the MySQL server. Remote authentication using the root user is dangerous and rarely required. Instead, you should either SSH onto the MySQL and use the MySQL client on the server to authenticate as the root user or, preferably, use an SSH tunnel to forward the remote MySQL port to your local machine and connect using a local client.

```
Normally, root should only be allowed to connect from
'localhost'. This ensures that someone cannot guess at
the root password from the network.

Disallow root login remotely? (Press y|Y for Yes, any other key for No) : y
```

Next, you'll be asked to delete the default databases (if they exist) that MySQL ships with. This is the recommended practice for production MySQL servers.

```
By default, MySQL comes with a database named 'test' that
anyone can access. This is also intended only for testing,
and should be removed before moving into a production
environment.


Remove test database and access to it? (Press y|Y for Yes, any other key
for No) : y
```

Finally, you'll be asked if you want to reload the privileges tables for all changes that have been applied to take effect.

```
Reloading the privilege tables will ensure that all changes
made so far will take effect immediately.

Reload privilege tables now? (Press y|Y for Yes, any other key for No) : y
Success.

All done!
```

Best practice 3: Ensure that mysqld is not started with the –skip-grant-tables argument

Should the MySQL's root password get misplaced, while not the preferred method, some MySQL administrators may resort to setting MySQL to start with the –skip-grant-tables argument. When starting MySQL with this parameter, it will avoid checking its grant tables when a client connects or runs a query, effectively allowing anyone, anywhere (provided they can reach the database over the network), to do anything on the database server.

You should ensure that the *–skip-grant-tables* option is not enabled. First, open your server's */etc/mysql/mysql.conf.d/mysqld.cnf* configuration file and look for skip-grant-tables. The value should either not be set or set to *skip-grant-tables = FALSE*.

While MySQL does not support two-factor authentication, WordPress does, thanks to WP 2FA – a feature-rich plugin that will enable you to add 2FA to your WordPress websites in minutes.

Add two-factor authentication and increase security with multiple methods to choose from.

Get WP 2FA

Best practice 4: Keep MySQL up-to-date

Just as it's important to ensure you're running the latest WordPress updates, it's important to keep MySQL up-to-date. Like most other software, updates to the MySQL server are released periodically. These updates address bugs, mitigate vulnerabilities, and provide new features. You should keep MySQL up-to-date with the latest security patches to reduce the risks of running software with known vulnerabilities. Bear in mind that once updated, you will be required to restart the 'mysql daemon.' This is a process that may incur some downtime. As always, plan ahead.

Best practice 5: Bind MySQL to an IP address

You can configure MySQL to only accept TCP/IP connections from a specific IPv4 or IPv6 interface. All you need to do is set the *bind-address* configuration option to a specific IP address. This provides additional controls and restrictions on how client applications (in our case, WordPress) can connect to MySQL. By default, this is set to *, meaning that out-of-the-box MySQL will listen on all interfaces.

If not configured to listen to a specific IP, all IPs can be used to connect to MySQL. This setting is especially important to set if you are running MySQL on

the same machine as a web server that you are exposing to the Internet (in this case, you should set the bind-address to **127.0.0.1** so MySQL only listens on localhost).

For example, if you want the MySQL server to only accept connections on a specific IPv4 address, you can add an entry similar to the example below. You should enter this under the [mysqld] option group in your server's */etc/mysql/mysql.conf.d/mysqld.cnf* configuration file.

```
bind-address=192.168.0.24
```

Once you set this, you will need to reconfigure WordPress to connect to the database using this IP address (unless it's doing so already). The server will not permit connections from any other addresses.

Best practice 6: Limit the use of web-based GUI tools

Many WordPress installations include web-based front-end graphical management tools. Common examples include Cpanel, phpMyAdmin, or Adminer. These tools make it easier to manage MySQL and other aspects of the underlying infrastructure. While a web-based graphical interface can help you manage your MySQL databases, these interfaces can increase the attack surface by adding another vector. Furthermore, there is a risk that they'll be discovered and abused by attackers to run destructive or malicious SQL queries against your database. Attacks may even result in a full takeover of your WordPress website.

The only safe server is the one that's switched off and unplugged – however, risk can be managed. Uninstalling non-critical systems is one option. You can also lock these down and restrict them to minimize the risk.

It is possible to restrict access to these tools in a variety of ways. You can install phpMyAdmin for WordPress remotely, thus minimizing risk to the web server. Alternatively, you might also want to consider using tools such as MySQL Workbench or Beekeeper Studio on your local machine and connect to your database server over an SSH tunnel.

Best practice 7: Run the MySQL daemon using a dedicated user

As with other services running on a server, you can run the MySQL daemon under a dedicated user instead of the root account. When you run MySQL using a dedicated user, you can precisely define what permissions that user is given within the system. Running MySQL under a dedicated user also follows the principle of least privilege since this reduces the blast radius of a MySQL vulnerability. It also decreases the possibility of a misconfiguration being taken advantage of since a restricted user will be unable to access resources unrelated to MySQL (such as operating system configurations and secrets).

The good news is that installations via package managers (such as apt or yum) take care of this step automatically when installing MySQL. A quick way to verify if MySQL is running under a dedicated user is to run the following on the machine running the MySQL daemon.

```
ps -ef | egrep “^mysql.*$”
```

If MySQL is running using a dedicated user, you should expect to see at least one line from ps’s output returned.

Best practice 8: Create a dedicated WordPress database user

Security best practices dictate segregating users and privileges by duties or roles. This means that every application that makes use of the database should have its own dedicated user with the minimum MySQL database permissions required to carry out its job. As such, you’ll need to ensure user privileges do not go over and above what is required.

This practice should extend to deployments running multiple WordPress websites — each WordPress website should have its own dedicated database and MySQL user. This ensures that at any time, only one user has access to one database at a time, and users cannot access other databases, avoiding unauthorized access and data breaches.

You can usetThe following SQL statement (substitute *host, password,* and *database* to fit your needs) to create a dedicated user for your WordPress website and grant privileges for regular use. Keep in mind that some WordPress plugins, themes, and WordPress updates may occasionally need additional privileges to operate correctly (see the official WordPress guidance on this for more information).

```
CREATE USER 'wordpress'@'<host>' IDENTIFIED BY '<password>';
GRANT SELECT, INSERT, UPDATE, DELETE ON '<database>'.* TO
'wordpress'@'<host>'
```

Best practice 9: Ensure that local_infile is disabled

The *LOAD DATA* statement allows you to load data files into database tables. Under specific conditions, this can be abused to read files from the MySQL server. As such, unless you have a specific use case for this in your WordPress site, you should disable this feature.

If MySQL and the web server are running on the same machine, it may allow an attacker to use the *LOAD DATA LOCAL* statement to read arbitrary files that the web server process has read access to. This assumes that an attacker has the ability to run arbitrary SQL statements against MySQL. Such may be the case with an SQL injection vulnerability or through the installation of a malicious WordPress plugin. This is yet another reason to keep your web server and database servers separate.
By default, local_infile is disabled in MySQL 8.0 (it used to be enabled by default in earlier versions of MySQL). To prevent the MySQL server from accepting LOAD DATA LOCAL statements, ensure that the mysqld daemon is started with *local_infile* disabled.

Best practice 10: Disable MySQL command history

On Linux, the MySQL client logs statements executed interactively are saved to a history file (typically located in $HOME/.mysql_history). As such, you should disable the MySQL command history since this reduces the likelihood of exposing sensitive information, such as passwords, encryption keys, or other secrets.

To verify that .mysql_history files do not exist on the system, run the following commands:

```
find /home -name “.mysql_history”
find /root -name “.mysql_history”
```

If the above commands return any output, remove any *.mysql_history* files. Additionally, you can set *$HOME/.mysql_history* as a symlink to */dev/null* as follows:

```
ln -s /dev/null $HOME/.mysql_history
```

Best practice 11: Back up your database

Backing up your WordPress database does not help you secure the WordPress database. However, it is absolutely crucial to be able to recover promptly from a disaster or an attack. While there is a myriad of ways to back up your WordPress database – from WordPress backup plugins and services to homegrown scripts that take a database dump periodically — the following are a few salient tips to bear in mind.

Take frequent backups

Taking regular backups is pretty obvious and self-explanatory — the more frequently you take database backups, the easier it will be to recover from a data loss incident. While the frequency of backups will depend on the type of WordPress site you are running, as a rule of thumb, taking a backup daily serves most use cases well.

Verify the integrity of your backups frequently

Your backups are only useful if they work. You would likely prefer not to find out whilst you're in the middle of an incident trying to recover data. The simple remediation to this is to frequently verify that your backups actually work by doing test restores every so often. A good way to do this is to set a calendar event every few months to go through a restore procedure to ensure your backups are still working as expected. Additionally, documenting database restoration steps is also a good idea — the less guesswork when responding to an incident, the better.

Store your backups securely

Never keep backups of your WordPress site on your web or database server (especially on your web server). Backups are a great place for attackers to go dumpster diving. Storing your backups in a secure offsite location is highly advisable. If you are taking periodic database dumps, consider storing your

database dumps on an object storage service. These can include Amazon S3, Cloudflare R2, DigitalOcean Spaces, Linode Object Storage, etc. Taking this route can be a great, cost-effective way to store your database backups. However, do be extra careful that you do not make the storage bucket you are using publicly accessible.

Best practice 12: Enable and enforce TLS connections

Unless you are running MySQL on the same machine as your web server (which, as we already covered above, is not an ideal security practice), you should encrypt data between WordPress and MySQL. You can do this by using Transport Layer Security (TLS certificate). It was formerly referred to as Secure Socket Layer (SSL certificate).

By default, when you install MySQL, it will generate a self-signed certificate for you automatically. You can verify this by running the following (alternatively, you can use the mysql_ssl_rsa_setup script to generate new certificates).

```
ls -l /var/lib/mysql/*.pem
-rw------- 1 mysql mysql 1705 Nov 21 10:45 /var/lib/mysql/ca-key.pem
-rw-r--r-- 1 mysql mysql 1112 Nov 21 10:45 /var/lib/mysql/ca.pem
-rw-r--r-- 1 mysql mysql 1112 Nov 21 10:45 /var/lib/mysql/client-cert.pem
-rw------- 1 mysql mysql 1705 Nov 21 10:45 /var/lib/mysql/client-key.pem
-rw------- 1 mysql mysql 1705 Nov 21 10:45 /var/lib/mysql/private_key.pem
-rw-r--r-- 1 mysql mysql  452 Nov 21 10:45 /var/lib/mysql/public_key.pem
-rw-r--r-- 1 mysql mysql 1112 Nov 21 10:45 /var/lib/mysql/server-cert.pem
-rw------- 1 mysql mysql 1705 Nov 21 10:45 /var/lib/mysql/server-key.pem
```

You will need to copy over *ca.pem* from the above list (for example, via SCP) to the server running your WordPress website. Once you upload the *ca.pem* file to your WordPress server, you will need to move the certificate over to the operating system's certificate trust store. Then, you'll need to update the certificate trust store as follows.

Heads up, the file name of the CA certificate must end with a *.crt* file extension. As an example, *mysql-ca.crt* is valid, but m*ysql-ca.pem.crt* or *mysql-ca.pem* are invalid.

```
sudo mv ca.pem /usr/local/share/ca-certificates/mysql-ca.crt
sudo update-ca-certificates
```

Next, you need to configure WordPress to use TLS when connecting to MySQL. You'll need to add the following to the wp-config.php file of your

WordPress installation.

```
define('MYSQL_CLIENT_FLAGS', MYSQLI_CLIENT_SSL);
```

Once you update wp-config.php, WordPress will initiate connections to your MySQL server using TLS.

Next, you should enforce TLS connections to your MySQL server using the *require_secure_transport* system variable. You'll need to add the following to your */etc/mysql/mysql.conf.d/mysqld.cnf* file:

```
require_secure_transport = ON
```

Finally, restart MySQL for changes to take effect:

```
systemctl restart mysql
```

Best practice 13: Change the table prefix

By default, WordPress creates all tables with the 'wp_' prefix. This can make it easier for attackers to succeed in certain attacks, such as SQL injection, since they would know the default prefix and, by extension, the names of the database tables. This alone is not going to protect you. However, it's a straightforward exercise, recommended by many as a WordPress security best practice.

You can change the default WordPress database prefix during the installation process. You can also do so at any point thereafter, although this is slightly more complex. Either way, you can find helpful tutorials on changing WordPress database prefix.

Bonus: Consider using a Database as a Service (DBaaS)

Database as a Service is well worth considering if you're not hosting WordPress on a managed plan. It replaces the traditional model of installing MySQL locally with a service you connect to. This might suit your use case if you are running your WordPress site with a hosting provider that offers managed database services. Available options generally include Amazon RDS, DigitalOcean Managed MySQL, and Linode Managed MySQL). At face value, these services can be costlier than running MySQL yourself. However, they do all the heavy lifting of running production-grade databases. Most services include security best practices presets, ongoing security patches and maintenance, and backups.

Using a Database as a Service (DBaaS) is one of the best options in terms of security and reliability. While this is not mandatory, it's still good to have. However, if you are looking to manage MySQL yourself, the following is a collection of hardening tips to keep in mind.

How to implement WordPress database security changes

Hopefully, this article has provided you with an overview of MySQL security hardening in the context of running a WordPress website. There are no silver bullets in website security. However, taking a layered, defense-in-depth approach to security will make attacking your website significantly more difficult for attackers.

While this guide presents a number of hardening techniques for MySQL, MySQL is just one component of the WordPress ecosystem. As such, you should also consider other aspects of WordPress security covered in our WordPress security hardening guide. This, coupled with proven security measures such as WordPress two-factor authentication, will help you ensure you're as safe as you can be.

This might feel like a lot to take in. Remember that you can (and probably should) apply the various hardening techniques covered in this guide gradually.

Frequently Asked Questions

How do I make MySQL more secure?

What is the security standard of MySQL?

How do I change the prefix in WordPress database?

Keeping your database and WordPress secure

Remember that attackers are often after soft targets. This allows them to exploit weakly secured websites with little effort. Being one step ahead of the next WordPress website's security posture makes you a less attractive target.

One topic that often comes up in MySQL security is that of using strong passwords. Strong passwords mitigate many of the risks associated with being online. This also applies to WordPress passwords.

Make sure all of your users, customers, and employees are using strong passwords with Melapress Login Security. This plugin enables you to effortlessly deploy strong password policies, among other policies designed to keep your WordPress login secure.

Get started with Melapress Login Security today.

WORDPRESS SECURITY

Joel Barbara

Joel was our technical writer who wrote a number of articles on our blog. With a background in tech and content, he has a passion for making technology accessible and understandable for everyone.

Related

WordPress username disclosure, is it a vulnerability or not?

WORDPRESS SECURITY

How to change the WordPress database prefix & improve security

WORDPRESS SECURITY

Stay in the loop

Subscribe to the Melapress newsletter and receive curated WordPress management and security tips and content.

Enter your email here **Subscribe**

Read our **Privacy Policy**

It's free and you can unsubscribe whenever you want. Check our blog for a taste.

We're dedicated to building great plugins but we're not antisocial!

- Follow us on LinkedIn
- Connect with us on Facebook
- Follow us on Twitter
- Follow us on Instagram
- Subscribe to us on YouTube

CONTACT US >

Our plugins

WP Activity Log

WP 2FA

Melapress Login Security

Melapress File Monitor

Admin Notices Manager

User Roles Editor

Support

Open a ticket

Knowledge base

Plugins support

WordPress

WordPress security & hardening

The guide to WordPress security plugins

Subscribe to newsletter

Melapress

About us

Become an affiliate

Careers

My account page

Partners

Blog

Copyright © 2025 Melapress. All rights reserved.
Enjoy the rest of your Tuesday.

Terms and conditions | Privacy policy | Report a security issue | Contact us | Cookie Policy | Privacy Statement