Starting to Secure MySQL (Part 2 - Principle of Least Privilege)

Author	josh
Date	2017-09-22 19:00:29

Overview

In the last article, we discussed a solid starting point for securing MySQL installations using `mysql_secure_installation`. Running this script is great way to eliminate some default security flaws with MySQL, but it is just a starting point. In MySQL, multiple database users can be created for different hosts and privileges can be set for different databases. In order to further secure each database created for a particular application, it is important to look at how users are created and what privileges are granted on that database.

Creating users for the database

Why isolated users improve security

Say you're creating a web application and need to supply a database user and password in your backend code so you can connect to the database and select some records to display. It might be tempting to just plug in the root user. After all, root has all the privileges you need and it's already built in, so it should just work. But what happens if say, PHP isn't properly configured and a user requests your webpage from the server? Now they have the root username and password since the source code was returned to them instead of executed! A malicious attacker could gain access to your server through another vulnerability and wreak havoc on your databases. This is the first part of using the "Principle of Least Privilege" to secure your database. It's a really good idea to have a user account (or multiple user accounts) with access to only *one* database. Don't configure this user to have privileges on any other database. In the event that user's credentials are compromised, the damage can at least be *isolated* to only one database.

What about hosts?

Another important consideration for creating users is *where* they have privileges to access the database from. With a database engine like MySQL, it's possible to connect directly to the database from a remote location. You could configure a user for your web application's database to connect from any host, so you could always work on the database from a coffee shop or a friend's house. However, this is another important place to limit privileges to the *least amount* needed to be operational. In the above scenario, if an attacker gets the credentials for your database user, they could log in from their bedroom in their PJ's and start messing with your database! Instead of allowing a user to connect from any host, consider the only place a user should be able to connect from to make the database functional. In a web application scenario `localhost` is often a good choice because it limits the user to connecting from the web server the application is running on. Now if an attacker gets access to your user's credentials, they would have to also gain shell access to the server. This presents an additional obstacle between them and your database that can be difficult to overcome with a properly configured server.

How to create a user for a specific host

In MySQL, creating a user for a specific host only requires one command that is fairly straightforward. For example: `CREATE USER "MyUser"@"localhost" IDENTIFIED BY "MyLongAndStrongPassphrase!";` The above example creates a user called `MyUser` that can only log in from `localhost`. The `IDENTIFIED BY` part tells the database engine what password the user will log in with. Be sure to create a *unique*, strong password for this user.

Configuring user privileges

What privileges should a user have?

Much like the question of "what hosts should I allow my user to connect to?", the question of what privileges a user should be granted comes down to the *least amount of privileges needed for the user to do its job*. Just like it's easy to use the root user for everything, it's easy to grant all privileges to a user for your database so everything just works. But again consider the scenario where an attacker gains access to that user's credentials. Even if that user can only log into one database, they could drop tables or modify records maliciously. Imagine an attacker being able to modify the price of an expensive item in your web store to any amount they want! Now in that scenario, the only access your web store needs to the database is to `SELECT`the prices of available items. Why give the user access to `update` or `delete` records at all? This is the core of the "Principle of Least Privilege". Only give users the minimum amount of privileges they need to do the job they do. In our web store example, that means they only need the ability to do a `SELECT` on the database and nothing more.

How to grant user privileges in MySQL

In MySQL, specifying user privileges is done using the `GRANT` statement. For example: `GRANT SELECT ON MyWebStore.PRICES to "MyUser"@"localhost";` The first part of this command (the `GRANT SELECT`) tells the engine that this user will get the SELECT privilege only. There are a bunch more, but some of the common ones include `INSERT, UPDATE, DELETE, SELECT, DROP, CREATE`. The second part of this command (`MyWebStore.PRICES`)specifies which database and which table(s) the privileges will apply to. If we're following the "Principle of Least Privilege", this user should only be granted these privileges on one database. As for tables, you may want to specify one table or all tables in the database depending on your application. You can apply the privileges to all tables using the `*` wildcard operator, like `MyWebStore.*`. The final part of this command (`"MyUser"@"localhost"`) specifies which user (on which host) the privileges will apply to. There is one final step, and that's to immediately apply all these privileges. In MySQL the command `FLUSH PRIVILEGES` immediately commits the changes to the database.

Being mindful

Applying the "Principle of Least Privilege" is another important methodology for securing MySQL database installations against attackers. It's important to be mindful of the needs of your application when creating users and granting privileges, so that each user has the minimal amount of access needed to do their job. By isolating users to one database, one or few hosts, with a minimal amount of privileges, we can isolate and minimize the potential done by an attacker that gains access to database user credentials.