MySQL Security Tips

(2009-11-05) - Contributed by Codex-M

If you are a web developer or administrator, aside from administering your web server, you should also be administering your MySQL database in terms of security. This database is open source and is commonly used with the PHP web server scripting language; tons of useful applications are being developed with this kind of setup. This is good, but it opens up issues, which we'll discuss here along with their solutions.

Because of its immense popularity, MySQL is also a regular target for malicious users or hackers wanting to exploit your system and steal data. This type of exploit can be serious; it can include putting malicious software on your web server and using the website to host malware.

To avoid letting your database be compromised, this article outlines useful tips which every webmaster should be implementing to maximize database security. These tips come from experience, and from the MySQL developers themselves.

We will outline possible security issues with using MySQL and then propose feasible corrective actions.

Network eavesdropping to compromise connection

This is particularly true if you belong to an untrusted network. Someone may have the time to sniff your connection to the MySQL server and compromise the connections.

The default connection to the MySQL server is not encrypted (for most hosting companies), which could be a problem if you are sending sensitive data to the MySQL server.

To correct this problem, encrypt the connection or session to the MySQL server. You have two options when encrypting data, SSH or SSL.

SSH is short for "secure shell." If you prefer to work with a command line rather than a GUI, then you'll find that using SSH is great for encrypting the packets going in and out of the MySQL server.

SSL stands for "secure socket layer." It is very important for commercial websites, especially if you are running a business that accepts sensitive information, such as credit card numbers.

If you want to look at it as a system, MySQL server serves two entities: the connection between MySQL server and the MySQL client, and the connection between MySQL server and the web server. See the screen shot below for a visual representation.

Not all web hosts offer SSL connections in phpMyadmin. So if you need to encrypt some of your connections, you need to check with your web hosting company to see if they enable SSL by default when connecting to phpmyadmin.

Some free hosting companies disable this feature, so a free hosting account is more susceptible to an eavesdropping problem.

If you use Putty, you can use it to connect to the remote MySQL server using SSH protocol. A discussion of this topic is beyond the scope of this article, though you can easily download Putty. You can also find some Putty security tutorials if you're interested in pursuing this topic further.

Then, if you accept sensitive data from customers, you can purchase an SSL certificate and install it in your website, so that communication between web forms and the web server is encrypted.

{mospagebreak title=Open port in your MySQL server}

If you have configured your MySQL server to connect to the Internet, make sure it is behind a firewall to prevent remote attacks. If you are not protected, an attacker can initiate many attacks, including denial of service (DoS) attacks. If your server does not have proper authentication routines configured, they can even penetrate and compromise your server.

MySQL commonly uses port 3306. If you are running an Apache home-based server with MySQL in it, this port number might be exposed to the Internet. So make sure you have checked to see if this port is open, and that you only allow connections from trusted hosts. If not, you should close this port in a firewall, and open it only when it is highly necessary.

You can use this tool to scan your MySQL server for open ports.

Brute force attacks to your MySQL server

MySQL commonly asks for authentication if someone needs to log in to the server and access sensitive data. However, if you are using phpMyadmin to connect to the database, this is susceptible to brute force attacks.

To prevent brute force attacks, especially in a web -based interface using phpmyadmin, you need to add the patch mentioned in this update.

If you do not have full control of the phpmyadmin, which may be true in limited hosting accounts, then you cannot add that patch. Your only option is to contact your web host and ask if they have provided a layer of security to prevent brute force attacks, especially with the MySQL server. If they haven't, you can either ask them to upgrade, or switch to another hosting provider if you are really in need of brute force prevention.

The best way to provide brute force protection without changing phpMyadmin scripts or switching web hosts is by adding extremely strong passwords. In this case, any brute force attempts can take years before they succeed, if ever -- and most attackers will give up in that time.

Try to monitor your logs. If there is a series of failed login attempts, it is a sign that someone is planning to brute force the account. And then you can easily ban the IP using .htaccess or other methods.

{mospagebreak title=Weak MySQL password administration}

Do not ever try to store passwords in plain text. It is highly recommended to encrypt the password when it is stored in a MySQL database. If your database has been compromised, the passwords in plain text can be used to log in to the CMS panel, further compromising your website.

This is why, if you are using WordPress, by default they are stored as MD5, which is a one-way encryption and cannot be reversed. In this method, there is no way for a hacker to reverse engineer a password encrypted using MD5.

It is also extremely important that all users in the database have strong passwords that are stored in encrypted form the MySQL database.

MySQL injection issue

This is so far the biggest MySQL security problem. A lot of websites get compromised because of hackers using SQL injection methods. The root cause is improper validation of user inputs. It is always important to NEVER trust user data

coming from forms or URLs.

You can easily prevent MySQL injection. Just follow four simple steps.

First, designing the database in such a way that it will only accept valid data entries for storing. If hackers inject code which fails to validate in the database fields design, it will be rejected and fail to be stored to the database. You can read more about MySQL database design.

Second, make sure you validate form user inputs. If you have strong validation of user inputs, it will be very hard to compromise the forms by entering malicious scripts. You can read more about PHP form validation.

Third, monitor your logs for MySQL injection entries. If you see an IP address engaging in possible MySQL injection, block the IP. That will make it harder for hackers to break into your website.

Fourth, prefer POST over using GET for common form submission. Although this is not a strict requirement, using POST will hide variables from being shown in the URL. GET will expose the variables and other values in the URL, making it very easy for hackers to look for possible exploits.

{mospagebreak title=Properly assigned privileges}

When working with a developer or other people who need special access to your web site, do not always give them the root access (full access) for the MySQL server. Instead, you can assign privileges and provide them limited access to prevent possible leaking of accounts which can compromise the system.

For example, a developer's privileges can be limited to inserting tables, editing tables and creating databases. You could withhold the privilege of looking into other sensitive databases in the server.

You can learn more about user privileges.

Restrict access to MySQL database by allowed IP

One effective way to reduce the possibility of compromising your database is to restrict access to it using the IP address. You need to define your IP address and use that to configure MySQL so that, when other IP addresses try to access the database, they will be denied.

You can read more about restricting access by IP address.

Also, if you use a web form to communicate with your web server, below are some tips you can use to restrict access to certain users, based on the specific country. For example, if you have a website that accepts form inputs from people living only in the USA, you can:

- Grab the IP address of the form user.

- Analyze the geo location of the IP address. You can use a database shown here.

- If the IP is not allowed, form inputs will not be allowed to enter the MySQL database.

You can also use a firewall to allow only certain IP address to pass through to the MySQL server. In this way, access is controlled and security will be improved.