Securing MySQL on Linux Systems

May 11, 2012 / By Matt Reid

Tags: MySQL

MySQL is one of the most, if not the most, popular relational databases chosen by internet based startups in the last decade. Although it is a very robust platform and offers many of the necessary features to support the database needs of today's internet giants, it also suffers from some security issues that must be addressed prior to production use.

This article will discuss the necessary steps to secure a basic MySQL installation and additionally cover more advanced topics for general database security as would be applied to general production environments.

Functionality

The following general statements apply to this security discussion:

•This is a discussion of MySQL as installed on Linux; Windows, OSX, Solaris, and BSD installations may differ and the differences are not covered here.

•MySQL will be handling generic web traffic as is commonly found on PHP,

Recommended

Welcome to Blackbird.io Employees and Clients

Life at Pythian as a MySQL DBA

The Hadoop and the Hare

SMB File Share and SQL Server: An option to consider

Do You Know Your Customers? The Truth Behind Millennial Shopping Habits

Recent Posts

Python, Perl, and Java web applications.

•The standard, and included, MySQL administration tools will be used as necessary for daily contact with the database server. Non-standard tools are not discussed.

•Any remote connections not done over SSH, either explicitly or via tunnel, are assumed to be a security risk and are not advised, as they lack encryption and allow passwords to be read in transit by potential attackers.

Pre-Installation Security Recommendations

Before installing any software it is a good idea to harden the operating system. This includes filesystem security features:

•Encrypted filesystems to prevent inspection of data by unauthorized parties

•Intrusion detection systems like Tripwire, which watch for changes to critical system files

•Port lockdown via firewall software to prevent access to system services

•Strong root and OS-user passwords

•Disallowing OS login by application users: set to “nologin” or “/bin/false”

•Setting up sudo for privileged accounts and requiring a password for sudo access

•Running scheduled rootkit detection scripts

•Running an operating system level Access Control List process: SELinux extensions or AppArmor. These programs will restrict system or server processes from accessing data and resources that is not explicitly defined. Due to general misunderstanding or lack of desire to maintain the access controls, these programs are very often disabled by system administrators.

Post Install Security

After MySQL is installed, either via RPM, Deb package, or other means, there are different approaches to securing the initial database. The first option is to execute the script provided with MySQL, named “mysql_secure_installation”. This will go through the following steps, which can also be taken manually if you prefer:

•Checking for existence of the Root password, and if not found it will set one

```
?mysql> SELECT * FROM mysql.users WHERE User=’root’;
```

```
?mysql> UPDATE mysql.users SET Password=password(‘your password here’) WHERE User=’root’;
```

•Delete anonymous users

```
?mysql> DELETE FROM mysql.users WHERE User=’’;
```

•Removing remote access for the root account

Licensing Oracle in a public cloud: the CPU calculation impact

Log Buffer #418: A Carnival of the Vanities for DBAs

Log Buffer #417: A Carnival of the Vanities for DBAs

Disable Lock Escalation in SQL Server

OakTable World at IOUG COLLABORATE15

Most Read

How to Install a Clustered SQL Server 2012 Instance – Part 2

How to Install a Clustered SQL Server 2012 Instance – Part 1

TEXT vs. VARCHAR

Step-by-Step Installation of an EBS 12.2 Vision Instance

T-SQL: Retrieve all users and associated roles for ALL databases

Recent Comments

?ukasz on How to Install a Clustered SQL Server 2012 Instance – Part 3

Mark Gruenberg on An Open Letter to Larry Ellison on AWR and ASH Licensing

bplmanju on How to Install a Clustered SQL Server 2012 Instance – Part 1

jobsearch on Quick Install Guide for Oracle 10g Release 2 on Mac OS X Leopard & Snow Leopard

Varun on Oracle Database or

```
?mysql> DELETE FROM mysql.user WHERE User='root' AND Host NOT IN ('localhost', '127.0.0.1', '::1');
```

•Removing the default "test" database

```
?mysql> DROP database test;
```

•Reloading the user privilege tables

```
?mysql> FLUSH PRIVILEGES;
```

By default, MySQL will run via the "mysqld_safe" which contrary to the name does not make MySQL safer for security reasons aside from ensuring that the mysqld process is not running under the root user. The mysqld_safe script provides the functionality as follows: "Script to start the MySQL daemon and restart it if it dies unexpectedly". As such, if one attempts to run mysqld as the root user it will complain and refuse to start. If you are troubleshooting the mysqld process and want to run without mysqld_safe you can run it as follows, via sudo. Your binary location may differ from /usr/local/bin/mysqld so replace as necessary.

```
$>  sudo -u mysql /usr/local/bin/mysqld
```

MySQL Configuration Considerations

There are several configuration settings that can further secure the database during operation. These settings will be found in the /etc/my.cnf or /etc/mysql/my.cnf file depending on the version of Linux being used.

old-passwords: this allows MySQL to create and authenticate users via the outdated and insecure password hashing from version 4 and older. It is strongly recommended against using this in production.

http://dev.mysql.com/doc/refman/5.5/en/server-options.html#option_mysqld_old-passwords

sql-mode: this allows the administrator to run MySQL in various operating modes. There are many options but for security the recommended minimum setting is "NO_AUTO_CREATE_USER" or "TRADITIONAL"

http://dev.mysql.com/doc/refman/5.5/en/server-options.html#option_mysqld_sql-mode

skip-grant-tables: this starts the mysqld process without any authentication tables, which is useful if one needs to recover or reset a lost root password. However, if you see this option enabled on a production server that is not undergoing recovery it should be seen as a critical security issue and dealt with immediately.

http://dev.mysql.com/doc/refman/5.5/en/server-options.html#option_mysqld_skip-grant-tables

bind-address: this can be used to restrict network and socket access to

Hadoop?

Pythian Services

Data Management Services

Database Performance Management

Big Data Consulting

System Administration Services

System Administration Services

MongoDB Consulting

specific interfaces, thus ensuring that traffic can only originate through the desired interface. An example is a server with multiple network interfaces

and thus the mysqld process needs to be restricted to listen for connections on only the required subnet.

http://dev.mysql.com/doc/refman/5.5/en/server-options.html#option_mysqld_bind-address

Data Encryption

MySQL offers in-row data encryption functions. These can be used to ensure that even in the event of a security breach, that the attacker cannot access the data in the database tables without an encryption key. Although not a MySQL specific functionality, the SQL functions for AES_ENCRYPT and AES_DECRYPT, along with a multitude of HASH mechanisms exist for employing in-row data encryption. Read more about this topic here: http://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html

Database User and Connection Security Considerations

MySQL users, by definition, can connect to the database server and access data. How much or how little access your users are granted will determine the level of security compromise that they pose to potential attackers. The most insecure user that can be created is a user with wildcard remote host connection ability (the ability to connect from any location, any host, any IP address) who has SUPER and GRANT options; this user can create other users and has the ability to shut down the database server process. As such, it is wise to limit the amount of access that a user is granted when creating accounts. Here are some things to keep in mind:

•Limit user access to specific schemas: granting global access to a user allows the user to access all current and future schemas as opposed to a single defined schema.

•Limit originating users connections from specific IP addresses or subnets instead of allowing connections from any host or any network.

•Where possible limit users to connect only from localhost (127.0.0.1) and advise ssh tunnels to be utilized to gain access to localhost.

•Do not grant SUPER privileges to non-administrative users

•Do not grant replication privileges to non replication process users.

•When granting replication processes limit them to only replication privileges.

•When possible, do not use hostnames for host connection privileges, instead specify IP addresses. This removes the risk of DNS spoofing and removes the requirement for a DNS lookup during connection initiation which saves time and resources.

•Require strong passwords for all users and rotate passwords on a defined schedule.

•If running connections over the internet without a VPN, SSH tunnel, or other encrypted means, consider using SSL for all connections. Otherwise passwords can be seen in transit by attackers, similar to FTP.

References for further reading:

- http://dev.mysql.com/doc/refman/5.5/en/general-security-issues.html
- http://dev.mysql.com/doc/refman/5.5/en/security-against-attack.html
- http://www.sans.org/reading_room/whitepapers/application/

Share this article

0 Tweet 0 Like 1 g+1 Share

Leave a Reply

Name (required) Your Name*

Mail (required) Your E-Mail* (will not be published)

Website Got a website?

Your Comment here...

Submit

XHTML: You can use these tags: `<a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code> <del datetime=""> <em> <i> <q cite=""> <strike> <strong>`