The MariaDB Audit Plugin

mariadb.com

Introduction

MariaDB and MySQL are used in a broad range of environments, but if you needed to record user access to be in compliance with auditing regulations for your organization, you would previously have had to use other database solutions. To meet this need, though, the developers of MariaDB have developed the MariaDB Audit Plugin. Although the MariaDB Audit Plugin has some unique features available only for MariaDB, it can be used also with MySQL.

Installation

The MariaDB Audit Plugin is provided as a dynamic library: `server_audit.so` (`server_audit.dll` for Windows). The plugin must and should be located in the plugin library. The file path of the plugin library is recorded in the `plugin_dir` system variable. To see the value of this variable and determine thereby the file path of the plugin library, execute the following SQL statement:

```
SHOW GLOBAL VARIABLES LIKE 'plugin_dir';

+---------------+-----------------------------+
| Variable_name | Value                       |
+---------------+-----------------------------+
| plugin_dir    | /usr/local/mysql/lib/plugin/ |
+---------------+-----------------------------+
```

The plugin can be loaded from the command-line as a start-up parameter, or it can be set in the configuration file (i.e., `my.cnf` or `my.ini`). Below is an excerpt from a configuration file, showing the relavent line to load this plugin. To use this option from the command-line at start-up, just add a double-dash (e.g., `--plugin-load`).

```
[mysqld]
...
plugin-load=server_audit=server_audit.so
...
```

Another way to install this plug-in is to execute the `INSTALL PLUGIN` statement from within MariaDB or MySQL. You would need to use an administrative account which has `INSERT` privilege for the `mysql.plugin` table. To do this, you would execute the following within the `mysql` client or an equivalent client:

```
INSTALL PLUGIN server_audit SONAME 'server_audit.so';
```

Note
The variables that will be used by the plugin (see the *Configuration* section) will be unknown to the server until the plugin has been loaded the first time. The database server will not start successfully if these variables are set in the configuration file before the audit plugin has been loaded at least once before.

The `UNINSTALL PLUGIN` statement may be used to uninstall a plugin. For the auditing plugin, you might want to disable this possibility. To do this, you could add the following line to the configuration file after the plugin is loaded once:

```
[mysqld]
...
plugin-load=server_audit=server_audit.so
server_audit=FORCE_PLUS_PERMANENT
...
```

Once you've added the option above to the server's configuration file and restarted the server, if someone tries then to uninstall the audit plugin, an error message will be returned. Below is an example of this with the error message:

```
UNINSTALL PLUGIN server_audit;

ERROR 1702 (HY000):
Plugin 'server_audit' is force_plus_permanent and can not be
unloaded
```

Configuration

After the audit plugin has been installed and loaded, new global variables will be registered within MariaDB or MySQL. These can be used to configure many factors, limits, and methods related to auditing the server. You may set variables for related logs: their location, size limits, rotation parameters, and method of logging information. You may also set what information should be logged: information related to connections (i.e., connects, disconnects, failed attempts to connect), queries, as well as read and write access to tables. You may also include or exclude user activity in the logs.

To see a list of related variables on your server and their values, execute the follow from a MySQL client while connected to the server:

```
SHOW GLOBAL VARIABLES like 'audit_server%';

+-------------------------------+----------------------------------+
| Variable_name                 | Value                            |
+-------------------------------+----------------------------------+
| server_audit_events           | CONNECT,QUERY,TABLE              |
| server_audit_excl_users       |                                  |
| server_audit_file_path        | /usr/local/mysql/data/audittest |
| server_audit_file_rotate_size | 1000000                          |
| server_audit_incl_users       | root,John                        |
| server_audit_logging          | ON                               |
| server_audit_mode_mysql       | ON                               |
| server_audit_output_type      | file                             |
| server_audit_file_rotate_now  | OFF                              |
| server_audit_file_rotations   | 1                                |
| server_audit_syslog_facility  | LOG_USER                         |
| server_audit_syslog_ident     | mysql-server_auditing            |
| server_audit_syslog_info      |                                  |
| server_audit_syslog_priority  | LOG_INFO                         |
+-------------------------------+----------------------------------+
```

The value of these variables can be changed by an administrator with the SUPER privilege, using the SET statement. Below is an example of how to switch off audit logging:

```
SET GLOBAL server_audit_logging=OFF;
```

Although it is possible to change all of the variables shown above, their values may be reset when the server restarts. You should therefore set them in the configuration file (e.g., my.cnf) to ensure the values are the same after a restart. You would not generally set variables related to the auditing plugin using the SET statement. However, you might to test settings before making them more permanent. Since one cannot always restart the server, you would use the SET statement to change immediately the variables and then include the same settings in the configuration file so that the variables are set again when the server is restarted.

Location and Duration

Logs can be written to a separate file or to the system logs. If you prefer to have the logging separated from other system information, the value of the variable, server_audit_output_type should be set to file. This is the only option on Windows systems.

You can force a rotation by using set global server_audit_file_rotate_now=yes

Separate Log Files

In addition to setting the variable, server_audit_output_type to a value of file, you will have to provide the file path and name of the audit file. This is set in the variable, server_audit_file_path. You can set the file size limit of the log file with the variable, server_audit_file_rotate_size. So, if rotation is on and the log file has reached the size limit you set, a new file is created with a new file extention, leaving the old file with its current name and extension. To limit the number of log files created, set the variable, server_audit_file_rotations. You can force log file rotation by setting the variable, server_audit_file_rotate_now to a value of YES. When the number of files permitted is reached, the oldest file will be overwritten. Below is an example of how the variables described above might be set in a server's configuration files:

```
[mysqld]
...
server_audit_file_rotate_now=ON
server_audit_file_rotate_size=1000000
server_audit_file_rotations=5
...
```

Status Monitoring

You may want to monitor routinely the status of the auditing plugin. The `SHOW GLOBAL STATUS` statement can assist you in this. On the server, execute the following SQL statement:

```
SHOW GLOBAL STATUS LIKE "server_audit%";

+----------------------------+---------------------------------+
| Variable_name              | Value                           |
+----------------------------+---------------------------------+
| server_audit_active        | ON                              |
| server_audit_current_log   | /usr/local/mysql/data/audittest |
| server_audit_last_error    |                                 |
| server_audit_writes_failed | 0                               |
+----------------------------+---------------------------------+
```

The results above show that the auditing plugin is running, as indicated by the variable, `server_audit_active`. The variable, `server_audit_current_log` provides the path and name of the log file on the server that is currently in use.

System Logs

For security reasons, it can be better sometimes to use the system logs instead of a local file owned by the `mysql` user. To do this, the value of the variable, `server_audit_output_type` needs to be set to `syslog`. Advanced configurations such as using a remote `syslogd` service is part of the `syslogd` configuration.

The variables, `server_audit_syslog_ident` and `server_audit_syslog_info` can be used to identify a system log entry from the audit plugin. If a remote `syslogd` service is used for several MariaDB Servers, these same variables are used also to identify the MariaDB Server.

Below is an example of a system log entry taken from a server which had the variable, `server_audit_syslog_ident` set to the default value of `mysql-server_auditing`, and the variable, `server_audit_syslog_info` set to `<prod1>`.

```
Aug  7 17:19:58 localhost mysql-server_auditing:
<prod1> localhost.localdomain,root,localhost,1,7,
QUERY,mysql,'SELECT * FROM user',0
```

Although the default values for `server_audit_syslog_facility` and `server_audit_syslog_priority` should be sufficient in most cases, they can be changed based on the definition in `syslog.h` for the functions `openlog()` and `syslog()`. See Appendix B for more information on this.

Information to Log

The events that are logged can be grouped into the different types: connect, query, and table events. These are also the values which can be set for the variable, `server_audit_events`, in a comma-separated list.

Logging Connect Events

If the MariaDB Audit Plugin has been configured to log connect events, it logs connects, disconnects, and failed connects. For a failed connect, the log includes also the error code.

It's possible to define a list of users, for which events can be excluded or included for tracing their database activities. This list will be ignored for the loggings of connect events. This is because auditing standards distinguish between technical and physical users. Connects need to be logged for all type of users; access to objects need to be logged only for physical users.

Logging Query Events

If query event logging is enabled, queries that are executed will be logged for the defined users. The queries will be logged exactly as they are executed. Someone who was access to the log file can therefore read the queries and data in plain text. This is a security vulnerability. So make sure only trusted users have access to the log files and that the files are in a protected location. An alternative is to not use query event logging, but to use only table event logging (covered in the next section).

Queries are also logged if they cannot be executed. For example, a query will be logged because of a syntax error or because the user doesn't have the privileges necessary to access an object. These queries can be parsed by the error code that is provided in the log. The log format is explained in detail in section, *Audit Trail Format*.

> **Note**
> Unsuccessful queries might be more interesting than successfully executed ones. They can reveal applications that are executing queries which don't match the current schema. This can be useful in discovering problems with applications. Unsuccessful queries can also reveal sometimes someone who is guessing at the names of objects to get access to data. That can be useful in discovering in progress the IP address and other source information of a malicious user.

Logging Table Events

MariaDB has the ability to record table events in the logs—this is not a feature of MySQL. This feature is the only way to log which tables have been accessed when a query has used a view, a stored procedure, stored function, or a trigger. Without this new feature, a log entry for a query shows only the view used, or the stored procedure or function which was called, not the underlying tables. Of course, you could create a custom application to parse each query executed to find the SQL statements used and the tables accessed, but that would be a drain on system resources. Table event logging is much simpler: it adds a line to the log for each table accessed, without any parsing. It includes notes as to whether it was a read or a write. MariaDB version 5.5.31 or newer is required to be able to use this feature.

If you want to monitor user access to specific databases or tables (e.g., `mysql.user`), you can search the log for them. Then if you want to see a query which accessed a certain table, the audit log entry will include the query identificaiton number. You can use it to search the same log for the query entry. This can be useful when searching a log containing tens of thousands of entries.

Because of the option to log table events, you may disable query logging and still know who accessed which tables. You might want to disable query event logging to prevent sensitive data from being logged, to resolve the security vulnerability with query logging mentioned earlier. Since table event logging will log who accessed which table, you can still watch for malicious activities with the log. This is often enough to fulfill auditing requirements.

Logging User Activities

The audit plugin will log the database activities of all users, or only the users that you specify. With the MariaDB Audit Plugin, a database activity is defined as a query event or a table event. Connect events are logged for all users.

You may specify users to include in the log with the `server_audit_incl_users` variable or exclude users with the `server_audit_excl_users` variable. This can be useful if you would like to log entries from persons, but are not as interested in entries from trusted applications and would like to exclude them from the logs.

> **Note**
> One would use typically either the `server_audit_incl_users` variable or the `server_audit_excl_users` variable. You may though use both variables. If a username is listed in both variables, database activities for that user will be logged —`server_audit_incl_users` takes priority.

Although MariaDB and MySQL consider a user as the combination of the username and hostname, the audit plugin logs only based on the username. MariaDB and MySQL use both the username and hostname so as to grant privileges relevant to the location of the user (e.g., `root` access from a remote location is inadvisable). Privileges are not relevant though for tracing the access to database objects. The host name is still recorded in the log, but logging is not determined based on that information.

The following example shows how to add a new username to the `server_audit_incl_users` variable without removing previous usernames:

```
SET GLOBAL server_audit_incl_users =
CONCAT(@@global.server_audit_incl_users, ',Maria');
```

Remember to add also new users to be included in the logs to the same variable in MariaDB or MySQL's configuration file. Otherwise, when the server restarts it will discard the setting.

Audit Trail Format

The audit plugin logs user access to MariaDB and it's objects. The audit trail (i.e., audit log) is a set of *records*, written as a list of *fields* to a file in a plain-text format. The fields in the log are separated by commas. The format used for the plugin's own log file is slightly different from the format used if it logs to the system log because it has its own standard format. The general format for the logging to the plugin's own file is defined like the following:

```
[timestamp],[serverhost],[username],[host],[connectionid],
[queryid],[operation],[database],[object],[retcode]
```

If the variable, `server_audit_output_type` is set to `syslog`, the general format looks like this:

```
[timestamp] [syslog_host] [syslog_ident]: [syslog_info]
[serverhost],[username],[host],[connectionid],[queryid],
[operation],[database],[object],[retcode]
```

Below is a list of these items which are logged and what information is provided for each:

timestamp	Data and time in which the event occurred. If syslog is used, the format is defined by `syslogd`.
syslog_host	The host from which the syslog entry is received.
syslog_ident	Used to identify a system log entry, including the MariaDB server.
syslog_info	Used to provide information for identify a system log entry.
serverhost	Host name on which MariaDB is running.
username	Username of the connected user.
host	Host from which the user has connected.
connectionid	Connection identification number to which the operation is related.
queryid	The query identification number—multiple lines will be added to the log for table events. It can be used to find the relational table events and the related queries.
operation	The type of action recorded: CONNECT, QUERY, READ, WRITE, CREATE, ALTER, RENAME, DROP.
database	The active database (e.g., as set usually by `USE` command).
object	The executed query or table name for table events.
retcode	The return code of the operation logged.

Various events will result in different audit records. Some events will not return a value for some fields (e.g., when the active database is not set when connecting to the server).

Connect Events

Below is a generic example of the output for connect events, with placeholders representing data. These are events in which a user connected, disconnected, or tried unsuccessfully to connect to the server.

```
[timestamp],[serverhost],[username],[host],[connectionid],0,CONNECT,[database],,0
[timestamp],[serverhost],[username],[host],[connectionid],0,DISCONNECT,,,0
[timestamp],[serverhost],[username],[host],[connectionid],0,FAILED_CONNECT,,,[retcode]
```

Query Events

Here is the one audit record generated for each query event:

```
[timestamp],[serverhost],[username],[host],[connectionid],[queryid],QUERY,[database],[object],
[retcode]
```

Table Events

Below are generic examples of records that are entered in the audit log for each type of table event:

```
[timestamp],[serverhost],[username],[host],[connectionid],[queryid],CREATE,[database],[object],
[timestamp],[serverhost],[username],[host],[connectionid],[queryid],READ,[database],[object],
[timestamp],[serverhost],[username],[host],[connectionid],[queryid],WRITE,[database],[object],
[timestamp],[serverhost],[username],[host],[connectionid],[queryid],ALTER,[database],[object],
[timestamp],[serverhost],[username],[host],[connectionid],[queryid],RENAME,[database],
[object_old]|[database_new].[object_new],
[timestamp],[serverhost],[username],[host],[connectionid],[queryid],DROP,[database],[object],
```

Appendix A - Log File Examples

The following are examples of an audit log, if the variable, `server_audit_output_type` is set to `file`.

Connect Events

Below is an excerpt from a log file showing connect events. A larger excerpt including table events and query events is provided at the end of this appendix. Notice in the excerpt here that a failed connect also includes the error code.

```
20130810 00:05:30,localhost.localdomain,root, localhost,2,0,CONNECT,db1,,0
20130810 00:05:53,localhost.localdomain,root, localhost,2,0,DISCONNECT,,,0
20130810 00:06:28,localhost.localdomain,unknownuser,localhost,3,0,FAILED_CONNECT,,,1045
20130810 00:06:28,localhost.localdomain,unknownuser, localhost,3,0,DISCONNECT,,,0
```

Table Events

Below is an excerpt from a log file showing table events. Table events do not return any error code, as they only exist when a query could be executed successfully.

```
20130810 02:21:06,localhost.localdomain,John,localhost,3,25,CREATE,db1,services,
20130810 02:21:06,localhost.localdomain,John,localhost,3,27,READ,db1,services,
20130810 02:21:07,localhost.localdomain,John,localhost,3,29,WRITE,db1,services,
20130810 02:21:27,localhost.localdomain,John,localhost,3,35,ALTER,db1,services,
20130810 02:21:27,localhost.localdomain,John,localhost,3,36,RENAME,db1,services|db1.services_new,
20130810 02:21:45,localhost.localdomain,John,localhost,3,38,DROP,db1,services_new,
```

Query and Table Events using Views

A query which accesses a view does not include information about the underlying table. Table events, though, provide the name of the underlying database and table in the log.

```
20130810 02:21:07,localhost.localdomain,John,localhost,3,31,READ,db1,services,
20130810 02:21:07,localhost.localdomain,John,localhost,3,31,READ,db1,services_types,
20130810 02:21:07,localhost.localdomain,John,localhost,3,31,QUERY,db1,'SELECT * from myview',0
```

Notice in this log excerpt that for the query identified as 31, the query event entry shows that a view was accessed (i.e., `myview`), but it doesn't show the name of the underlying table. The table events which were logged, though, provide the name of the database and table (i.e., `db1`, `services`, and `services_types`).

Query and Table Events using `DROP`

When a database is removed, that affects obviously the tables in the database. Table events logging will record that a database was deleted and the tables that were involved.

```
20130810 02:21:45,localhost.localdomain,John,localhost,3,38,READ,mysql,proc,
20130810 02:21:45,localhost.localdomain,John,localhost,3,38,DROP,db1,services_types,
20130810 02:21:45,localhost.localdomain,John,localhost,3,38,DROP,db1,services_new,
20130810 02:21:45,localhost.localdomain,John,localhost,3,38,DROP,db1,myview,
20130810 02:21:45,localhost.localdomain,John,localhost,3,38,WRITE,mysql,proc,
20130810 02:21:45,localhost.localdomain,John,localhost,3,38,WRITE,mysql,event,
20130810 02:21:45,localhost.localdomain,John,localhost,3,38,QUERY,db1,'drop database db1',0
```

The second and third line from the log excerpt above shows that two tables were deleted from the database, `db1`. The fourth entry above shows that a view for the same database was also dropped. The last line above notes that a query was run which dropped the database.

Query and Table Events calling a Stored Procedure

A query which calls a stored procedure does not give the details you need for auditing. Having table events enabled in the plugin will provide the names of the tables and the operations executed.

```
20130810 03:00:36,localhost.localdomain,John,localhost,3,58,READ,db1,services,
20130810 03:00:36,localhost.localdomain,John,localhost,3,59,QUERY,db1,'call read_services()',0
```

Query and Table Events - Longer Excerpt

Here is a longer excerpt from the same audit plugin log file as shown above in smaller excerpts. It may be useful to see all of them together like this.

```
20130810 02:21:06,localhost.localdomain,John,localhost,3,25,CREATE,db1,services,
20130810 02:21:06,localhost.localdomain,John,localhost,3,25,QUERY,db1,'CREATE TABLE services (id
int(10) primary key, typeid int(10), name varchar(50))',0
20130810 02:21:06,localhost.localdomain,John,localhost,3,26,CREATE,db1,services_types,
20130810 02:21:06,localhost.localdomain,John,localhost,3,26,QUERY,db1,'CREATE TABLE
services_types (id int(10) primary key, name varchar(50))',0
20130810 02:21:06,localhost.localdomain,John,localhost,3,27,READ,db1,services,
20130810 02:21:06,localhost.localdomain,John,localhost,3,27,READ,db1,services_types,
20130810 02:21:06,localhost.localdomain,John,localhost,3,27,QUERY,db1,'CREATE VIEW db1.myview AS
SELECT * FROM services WHERE typeid IN (SELECT id FROM services_types WHERE name="consulting")',0
20130810 02:21:07,localhost.localdomain,John,localhost,3,28,WRITE,db1,services_types,
20130810 02:21:07,localhost.localdomain,John,localhost,3,28,QUERY,db1,'INSERT INTO services_types
VALUES (1,"support"),(2,"training"),(3,"consulting")',0
20130810 02:21:07,localhost.localdomain,John,localhost,3,29,WRITE,db1,services,
20130810 02:21:07,localhost.localdomain,John,localhost,3,29,QUERY,db1,'INSERT INTO services
values (1,1,"Remote DBA"),(2,3,"Health Check")',0
20130810 02:21:07,localhost.localdomain,John,localhost,3,30,READ,db1,services,
20130810 02:21:07,localhost.localdomain,John,localhost,3,30,READ,db1,services_types,
20130810 02:21:07,localhost.localdomain,John,localhost,3,30,QUERY,db1,'SELECT * FROM services
WHERE typeid IN (SELECT id FROM services_types WHERE name="consulting")',0
20130810 02:21:07,localhost.localdomain,John,localhost,3,31,READ,db1,services,
20130810 02:21:07,localhost.localdomain,John,localhost,3,31,READ,db1,services_types,
20130810 02:21:07,localhost.localdomain,John,localhost,3,31,QUERY,db1,'SELECT * from myview',0
20130810 02:21:07,localhost.localdomain,John,localhost,3,32,WRITE,db1,services,
20130810 02:21:07,localhost.localdomain,John,localhost,3,32,QUERY,db1,'UPDATE services SET
name="Time Hire" WHERE id=2',0
20130810 02:21:07,localhost.localdomain,John,localhost,3,33,WRITE,db1,services,
20130810 02:21:07,localhost.localdomain,John,localhost,3,33,QUERY,db1,'REPLACE INTO services
VALUES (2,3,"Health Check")',0
20130810 02:21:09,localhost.localdomain,John,localhost,3,34,WRITE,db1,services,
20130810 02:21:09,localhost.localdomain,John,localhost,3,34,QUERY,db1,'DELETE FROM services WHERE
id=1',0
20130810 02:21:27,localhost.localdomain,John,localhost,3,35,READ,db1,services,
20130810 02:21:27,localhost.localdomain,John,localhost,3,35,ALTER,db1,services,
20130810 02:21:27,localhost.localdomain,John,localhost,3,35,QUERY,db1,'ALTER TABLE services
MODIFY name VARCHAR(64)',0
20130810 02:21:27,localhost.localdomain,John,localhost,3,36,READ,db1,services,
20130810 02:21:27,localhost.localdomain,John,localhost,3,36,ALTER,db1,services,
20130810 02:21:27,localhost.localdomain,John,localhost,3,36,RENAME,db1,services|db1.services_new,
20130810 02:21:27,localhost.localdomain,John,localhost,3,36,QUERY,db1,'alter table services
rename to services_new',0
20130810 02:21:37,localhost.localdomain,John,localhost,3,37,QUERY,db1,'drop db1',1064
20130810 02:21:45,localhost.localdomain,John,localhost,3,38,READ,mysql,proc,
20130810 02:21:45,localhost.localdomain,John,localhost,3,38,DROP,db1,services_types,
20130810 02:21:45,localhost.localdomain,John,localhost,3,38,DROP,db1,services_new,
20130810 02:21:45,localhost.localdomain,John,localhost,3,38,DROP,db1,myview,
20130810 02:21:45,localhost.localdomain,John,localhost,3,38,WRITE,mysql,proc,
20130810 02:21:45,localhost.localdomain,John,localhost,3,38,WRITE,mysql,event,
20130810 02:21:45,localhost.localdomain,John,localhost,3,38,QUERY,db1,'drop database db1',0
```

Appendix B

The header file, `syslog.h` uses the values set in the variable, `server_audit_syslog_facility`. This in turn is used by the function, `openlog()`. There are several possible values for the variable:

LOG_USER	LOG_MAIL	LOG_DAEMON	LOG_AUTH
LOG_SYSLOG	LOG_LPR	LOG_NEWS	LOG_UUCP
LOG_CRON	LOG_AUTHPRIV	LOG_FTP	LOG_LOCAL0
LOG_LOCAL1	LOG_LOCAL2	LOG_LOCAL3	LOG_LOCAL4
LOG_LOCAL5	LOG_LOCAL6	LOG_LOCAL7	

The header file, `syslog.h` uses also the values set in the variable, `server_audit_syslog_priority`, in conjunction also with the function, `syslog()`. There are also several values possible for this variable:

LOG_EMERG	LOG_ALERT	LOG_CRIT
LOG_ERR	LOG_WARNING	LOG_NOTICE
LOG_INFO	LOG_DEBUG	

References

Below is a list of links to web sites where you may find more information on the audit plugin:

- MariaDB Website: http://www.skysql.com
- MariaDB Knowledgbase: https://mariadb.com/kb/en/mariadb/mariadb-documentation/mariadb-plugins/server_audit-mariadb-audit-plugin/
- Syslog.h: http://pubs.opengroup.org/onlinepubs/7908799/xsh/syslog.h.html