BetterDocumentation SqlReadmeBayes

sql/README.bayes

The following is the text of sql/README.bayes, based off of revision 160246.

Please feel free to edit it as much as you like to make it more useful. Periodically the version in Subversion will be updated to incorporate some of the changes.

To see the latest version in Subversion, click here

Feel free to write comments about your changes in the Comments section (at the bottom).

```
Using A SQL Database for Bayesian Storage Module
-------------------------------------------------------

SpamAssassin can now store users' bayesian filter data in a SQL
database. The  most common use for a system like this would be for
users to be able to have per user bayesian filter data on systems
where users may not have a home directory to store the data.

In order to activate the SQL based bayesian storage you have to
configure spamassassin and spamd to use a different bayes storage
module.  This can be done via a setting in the global configuration
file.

The directives required to turn on the SQL based bayesian storage are:

bayes_store_module                   Mail::SpamAssassin::BayesStore::SQL

This directive is used by the Bayes module to determine which storage
module should be used.  If not set it will default to:
Mail::SpamAssassin::BayesStore::DBM

PostgreSQL users will want to use the PostgreSQL specific storage
module:
bayes_store_module               Mail::SpamAssassin::BayesStore::PgSQL
This module provides a slightly different interface to makes better
use of the resources that PostgreSQL offers.  In addition, please make
sure that you follow the instructions below for loading the proper
procedural language and installing the tables and stored procedure.

There is also a MySQL specific storage driver available to provides a
small boost in performance.  It requires version 4.1 or above of the
MySQL database software to work properly.  In addition, it provides
rollback on error functionality if you create your bayes database
table using the InnoDB storage engine (ie s/MyISAM/InnoDB/ on the
bayes_mysql.sql file).  WARNING: Using this module with a version of
MySQL < 4.1 could have unexpected results.  To use the MySQL 4.1+
specific module set your bayes_store_module directive accordingly:
bayes_store_module               Mail::SpamAssassin::BayesStore::MySQL

Additional configuration directives provided by BayesSQL:

bayes_sql_dsn                           DBI:driver:database:hostname[:port]
bayes_sql_username                   dbusername
bayes_sql_password                   dbpassword

The bayes_sql_dsn directive describes the data source name that will
be used to create the connection to your SQL server.  It MUST be in
the format as listed above.  <driver> should be the DBD driver that
you have installed to access your database (initially tested with
MySQL, PostgreSQL, and SQLite).  <database> must be the name of the
database that you created to store the bayes data tables. <hostname>
is the name of the host that contains the SQL database  server.
<port> is the optional port number where your database server is
listening.

For an example of connection to PostgreSQL, see the main README file.
```

In addition to the global configuration directives there is a user preference:

```
bayes_sql_override_username           someusername
```

This directive, if used, will override the username used for storing data in the database. This could be used to group users together to share bayesian filter data. You can also use this config option to trick sa-learn to learn data as a specific user.

Requirements

In order for SpamAssassin to work with your SQL database, you must have the perl DBI module installed, AS WELL AS the DBD driver/module for your specific database. For example, if using MySQL as your RDBMS, you must have the DBD::mysql module installed. Check CPAN for the latest versions of DBI and your database driver/module.

The BayesStore::SQL module was tested with:

```
DBI-1.38
DBD-mysql-2.9002
perl v5.8.0
```

But older versions should work fine as the SQL code in SpamAssassin is as simple as could be.

NOTE: Some users have reported problems using DBD::Pg v1.22. There appears to be a bug in how parameters are quoted in that version, we recommend you upgrade to at least v1.31.

In addition, there appears to be a quote bug in some versions of PostgreSQL. It's unclear when the bug was fixed. 7.4.2 seems to work just fine, however some have reported problems with 7.3.6. Your milage may vary with versions less than 7.4.2. If you happen to know when the specific bug was fixed please feel free to notify the dev team so they can update this documentation.

Database Schema

The database schema for storage of the bayesian filter data contains several different tables. Several sample SQL schemas have been included in to help in setting up your database. The schemas contain the minimum tables and columns necessary to work with the code as written. You are free to add other columns as needed for your local implementation. Presently there is no way to override the table and column names used by the BayesStore::SQL code, this feature may be added in the future.

Example setup of bayes tables for MySQL:

This assumes that you have already created a database for use with spamassassin and setup a username/password that can access that database. (See "Creating A Database", in "sql/README", if you don't have a suitable database ready.)

To install the tables using the included example, use the following command:

```
mysql -h <hostname> -u <adminusername> -p databasename < bayes_mysql.sql
Enter password: <adminpassword>
```

To install the tables for PostgreSQL, use:

You need to install the plpgsql procedural language (assuming it hasn't already been installed in template1). You can do this by running the following command:

```
createlang plpgsql <databasename>
```

and then:

```
psql -U <username> -f bayes_pg.sql <databasename>
```

From a security viewpoint, it is wise to grant the minimum privileges
needed to the user on the bayes tables. These grants largely depend
on your system policies but here are some example grants to get you
started, please review carefully before putting into production:

```
GRANT SELECT, UPDATE, DELETE, INSERT ON TABLE bayes_token TO <username>;
GRANT SELECT, UPDATE, DELETE, INSERT ON TABLE bayes_vars TO <username>;
GRANT SELECT, DELETE, INSERT ON TABLE bayes_seen TO <username>;
GRANT SELECT, DELETE, INSERT ON TABLE bayes_expire TO <username>;
GRANT SELECT ON TABLE bayes_global_vars TO <username>;
GRANT UPDATE, SELECT, INSERT ON bayes_vars_id_seq TO <username>;
```

Once you have created the database and added the tables, just add the
required lines to your global configuration file (local.cf).

Testing SpamAssassin/SQL

To test your SQL setup, and debug any possible problems, you should
start spamd with the -D option, which will keep spamd in the
foreground, and will output debug message to the terminal. You should
then test spamd with a message by calling spamc. You can use the
sample-spam.txt file with the following command:

```
cat sample-spam.txt | spamc
```

Watch the debug output from spamd and look for the following debug
line:

```
debug: bayes: Database connection established
debug: bayes: Using username: <username>
```

If you do not see the above text, then the SQL query was not
successful, and you should see any error messages reported. Note that
the "unable to initialize database for <username> user" message can be
ignored at this point -- see below for more details.

This code has been tested using MySQL as the RDMS, with basic tests
against PostgreSQL and SQLite. It does require a database that allows
you to refer to a column on the right hand side of an expression (ie
update foo set bar = bar + 1). Any database driver that allows for
that usage should work with the BayesStore::SQL code. NOTE: You may
find that some implementations do not provide a significant advantage
over using the default DBM implementation. If you find a driver that
should work and has issues, please report them to the SADev list.

******
NB: This should be considered BETA, and the interface, schema, or
overall operation of SQL support may change at any time with future
releases of SA.
******

Converting Bayes Data From a DBM Database

Converting your bayes database data from Berkeley (DBM) based storage
to SQL based storage is as simple as a backup and then restore.

If you are upgrading from a previous version of SpamAssassin you
should first follow any recommended upgrade instructions for that
release, in most cases this will be as simple as running an
sa-learn --sync

Once you have performed this upgrade, for each bayes database follow
this procedure:

o Run 'sa-learn --backup > backup.txt' which will backup your bayes

```
  data into a text file.
o Optionally you can run 'sa-learn --clear' to remove the DBM based
  bayes files.
o Modify your local.cf file according to the directions above.
o Run 'sa-learn --spam <sample spam message>' to initialize the
  database.
o Run 'sa-learn --restore backup.txt' to restore your bayes data to
  the SQL database.

NOTE: sa-learn must be run as the user who's data you are loading, or
      you must make use of the bayes_sql_override_username config
      option.

NOTE: failure to use 'sa-learn --spam <msg>' on an initial spam message
      will result in the error message
      "bayes: unable to initialize database for <username> user, aborting!"
```

Comments

Please enter comments here. You can type @"SIG@ to insert your signature. – DuncanFindlay <<DateTime(2005-08-22T02:48:38Z)>>

I found "Migrating our Debian Anti-Spam Anti-Virus Gateway Email Server's Bayes database to MySQL" very helpful in stepping me through MySQL setup & migration of my existing DBM Bayes database. – TimHunter <<DateTime(2006-12-13T17:07:21Z)>>