Security in Firebird 2

Summary of changes

Improving security has had a lot of focus in Firebird 2.0 development. The following is a summary of the major changes.

New security database

The new security database is renamed as security2.fdb. Inside, the user authentication table, where user names and passwords are stored, is now called RDB$USERS. There is no longer a table named "users" but a new view over RDB$USERS that is named "USERS". Through this view, users can change their passwords.

For details of the new database, see New security database in the section about authentication later in this chapter.

For instructions on updating previous security databases, refer to the section Dealing with the new security database at the end of this chapter.

Better password encryption

A. Peshkov

Password encryption/decryption now uses a more secure password hash calculation algorithm.

Users can modify their own passwords

A. Peshkov

The SYSDBA remains the keeper of the security database. However, users can now modify their own passwords.

Non-server access to security database is rejected

A. Peshkov

gsec now uses the Services API. The server will refuse any access to security2.fdb except through the Services Manager.

Active protection from brute-force attack

A. Peshkov

Attempts to get access to the server using brute-force techniques on accounts and passwords are now detected and locked out.

- Login with password is required from any remote client.
- Clients making too many wrong login attempts are blocked from further attempts for a period.

Support for brute-force attack protection has been included in both the attachment functions of the Firebird API and the Services API. For more details, see Protection from brute-force hacking.

back to top of page

Vulnerabilities have been closed

A. Peshkov, C. Valderrama

Several known vulnerabilities in the API have been closed.

Caution: It must be noted that the restoration of the server redirection (“multi-hop”) capability to Firebird 2 potentially throws up a new vulnerability. For that reason, it is controlled by a parameter (Redirection) in firebird.conf, which you should not enable unless you really understand its implications.

These days, the ability to redirect requests to other servers is dangerous. Suppose you have one carefully protected firebird server, access to which is possible from the Internet. In a situation where this server has unrestricted access to your internal LAN, it will work as a gateway for incoming requests like firebird.your.domain.com:internal_server:/private/database.fdb.

Knowing the name or IP address of some internal server on your LAN is enough for an intruder: he does not even need login access to the external server. Such a gateway easily overrides a firewall that is protecting your LAN from outside attack.

back to top of page

Details of the security changes in Firebird 2.0

Security focus was directed at some recognised weaknesses in Firebird's security from malicious attacks:

- the lack of brute-force resistant passwords encryption in the security database,
- the ability for any remote user with a valid account to open the security database and read hashes from it (especially interesting in combination with the first point),

- the inability for users to change their own passwords,
- the lack of protection against remote brute-forcing of passwords on the server directly.

Authentication

Firebird authentication checks a server-wide security database in order to decide whether a database or server connection request is authorised. The security database stores the user names and passwords of all authorised login identities.

Firebird 1.5 authentication

In Firebird 1.5 the DES algorithm is used twice to hash the password: first by the client, then by the server, before comparing it with the hash stored in security database. However, this sequence becomes completely broken when the SYSDBA changes a password. The client performs the hash calculation twice and stores the resulting hash directly in the security database. Therefore, hash management is completely client-dependent (or, actually, client-defined).

Firebird 2: Server-side hashing

To be able to use stronger hashes, another approach was called for. The hash to be stored on the server should always be calculated on the server side. Such a schema already exists in Firebird – in the Services API. This led to the decision to use the Services API for any client activity related to user management. Now, gsec and the isc_user_add(modify, delete) API functions all use services to access the security database. (Embedded access to Classic server on POSIX is the exception – see below).

It became quite easy to make any changes to the way passwords are hashed - it is always performed by the server. It is no longer gsec's problem to calculate the hash for the security database: it simply asks services to do the work!

It is worth noting that the new gsec works successfully with older Firebird versions, as long as the server's architecture supports services.

The SHA-1 hashing algorithm

This method leads to the situation where

1. a hash valid for user A is invalid for user B,
2. when a user changes his password – even to exactly the same string as before – the data stored in RDB$USERS.RDB$PASSWD is new.

Although this situation does not increase resistance to a brute-force attempt to crack the password, it does make “visual” analysis of a stolen password database much harder.

back to top of page

The new security database

The structure of the security database was changed. In general, now it contains a patch by Ivan Prenosil, with some minor differences, enabling any user to change his/her own password.

- In Firebird 1.5 the table USERS has to be readable by PUBLIC, an engine requirement without which the password validation process would fail. Ivan's patch solution used a view, with the condition "WHERE USER = ''". That worked due to another bug in the engine that left the SQL variable USER empty, not 'authenticator', as it might seem from engine's code.

Once that bug was fixed, it was certainly possible to add the condition "USER = 'authenticator'". For the short term, that was OK, because the username is always converted to upper case.

- A better solution was found, that avoids making user authentication depend on an SQL trick. The result is that the non-SYSDBA user can see only his own login in any user-management tool (gsec, or any graphical interface that use the Services API). SYSDBA continues to have full access to manage users' accounts.

New security database structure

The Firebird 2 security database is named security2.fdb. For user authentication it has a new table named RDB$USERS that stores the new hashed passwords. A view over this table replaces the old USERS table and enables users to change their own passwords.

The DDL for the new structures can be found in the Security upgrade script in the Appendix to Firebird 2 Release Notes.

back to top of page

gsec in Firebird 2

Special measures were thus taken to make remote connection to the security database completely impossible. Don't be surprised if some old program fails on attempting direct access: this is by design.

Users information may now be accessed only through the Services API and the equivalent internal access to services now implemented in the isc_user_* API functions.

Protection from brute-force hacking

Current high-speed CPUs and fast WAN connections make it possible to try to brute-force Firebird server users' passwords. This is especially dangerous for the Superserver which, since Firebird 1.5, performs user authentication very fast. Classic is slower, since it has to create a new process for each connection, attach to the security database within that connection and compile a request to the table RDB$USERS before validating login and password. Superserver caches the connection and request, thus enabling a much faster user validation.

Given the 8-byte maximum length of the traditional Firebird password, the brute-force hacker had a reasonable chance to break into the Firebird installation.

The v.2.0 Superserver has active protection to make a brute-force attack more difficult. After a few failed attempts to log in, the user and IP address are locked for a few seconds, denying any attempt

to log in with that particular user name or from that particular IP address for a brief period.

No setup or configuration is required for this feature. It is active automatically as soon as the Firebird 2.0 Superserver starts up.

back to top of page

Classic Server on POSIX

For reasons both technical and historical, a Classic server on POSIX with embedded clients is especially vulnerable to security exposure. Users having embedded access to databases MUST be given at least read access to the security database.

This is the main reason that made implementing enhanced password hashes an absolute requirement. A malicious user with user-level access to Firebird could easily steal a copy of the security database, take it home and quietly brute-force the old DES hashes! Afterwards, he could change data in critical databases stored on that server. Firebird 2 is much less vulnerable to this kind of compromise.

But the embedded POSIX server had one more problem with security: its implementation of the Services API calls the command-line gsec, as normal users do. Therefore, an embedded user-maintenance utility must have full access to the security database.

The main reason to restrict direct access to the security database was to protect it from access by old versions of client software. Fortuitously, it also minimizes the exposure of the embedded Classic on POSIX at the same time, since it is quite unlikely that the combination of an old client and the new server would be present on the production box.

Caution: However, the level of Firebird security is still not satisfactory in one serious respect, so please read this section carefully before opening port 3050 to the Internet.

An important security problem with Firebird still remains unresolved: the transmission of poorly encrypted passwords “in clear” across the network. It is not possible to resolve this problem without breaking old clients.

To put it another way, a user who has set his/her password using a new secure method would be unable to use an older client to attach to the server. Taking this into account with plans to upgrade some aspects of the API in the next version, the decision was made not to change the password transmission method in Firebird 2.0.

The immediate problem can be solved easily by using any IP-tunneling software (such as ZeBeDee) to move data to and from a Firebird server, for both 1.5 and 2.0. It remains the recommended way to access your remote Firebird server across the Internet.

back to top of page

Dealing with the new security database

A. Peshkov

If you try to put a pre-Firebird 2 security database – security.fdb or a renamed isc4.gdb – into Firebird's new home directory and then try to connect to the server, you will get the message *Cannot attach to password database*. It is not a bug: it is by design. A security database from an earlier Firebird version cannot be used directly in Firebird 2.0 or higher.

The newly structured security database is named security2.fdb.

In order to be able to use an old security database, it is necessary to run the upgrade script security_database.sql, that is in the ../upgrade sub-directory of your Firebird server installation.

Note: A copy of the script appears in the Appendix to these notes: Security upgrade script.

Doing the security database upgrade

To do the upgrade, follow these steps:

1. Put your old security database in some place known to you, but not in Firebird's new home directory. Keep a copy available at all times!
2. Start Firebird 2, using its new, native security2.fdb.
3. Convert your old security database to ODS11 (i.e. backup and restore it using Firebird 2.0). Without this step, running the security_database.sql script will fail!
4. Connect the restored security database as SYSDBA and run the script.
5. Stop the Firebird service.
6. Copy the upgraded database to the Firebird 2 home directory as security2.fdb.
7. Restart Firebird.

Now you should be able to connect to the Firebird 2 server using your old logins and passwords.

Nullability of RDB$PASSWD

In pre-2.0 versions of Firebird it was possible to have a user with NULL password. From v.2.0 onward, the RDB$PASSWD field in the security database is constrained as NOT NULL.

However, to avoid exceptions during the upgrade process, the field is created as nullable by the upgrade script. If you are really sure you have no empty passwords in the security database, you may modify the script yourself. For example, you may edit the line:

```
RDB$PASSWD RDB$PASSWD,
```

to be

```
RDB$PASSWD RDB$PASSWD NOT NULL,
```

Caution with LegacyHash

As long as you configure LegacyHash = 1 in firebird.conf, Firebird's security does not work completely. To set this right, it is necessary to do as follows:

1. Change the SYSDBA password.
2. Have the users change their passwords (in 2.0 each user can change his or her own password).
3. Set LegacyHash back to default value of 0, or comment it out.
4. Stop and restart Firebird for the configuration change to take effect.