Authentication in Firebird 3

by Helen Borrie & Alex Peshkov

Developments introduced in Firebird 3 provide many options for authenticating users attempting to attach to a database. No longer is authentication restricted to a single method: the methods available are determined by the plug-in authentication modules configured in the global and/or database config files.

Plug-in Authentication Provider Modules

Authentication-specific calculations, hashes and related activity in the security databases are performed by a provider module invoked according to the configuration of the AuthServer and AuthClient parameters. Configuration and its effects in different .conf files is discussed here, later. Suffice it to say, at this point, that AuthServer is configured at the server side, while AuthClient is configured in a client-side configuration file when the client connection is to be configured differently from the global default. The AuthClient settings configured in firebird.conf at the server side initially represent the protocols available to Firebird 3 clients, although any settings there have no effect on clients..

All of the code related to authentication is conducted by the provider. Generic work, like extracting authentication data from a network message or inserting data into one, is performed in the interface code of the server application.

The Firebird 3 Authentication Providers

Firebird 3 comes with two providers for authentication: srp.dll and legacy_auth.dll on Windows, libsrp.so liblegacy_auth.so on POSIX, libsrp.dylib and liblegacy_auth.dylib on MacOSX. Respectively, these modules encode the work for the newly-introduced SRP protocol and the legacy authentication protocol. On Windows systems, a security support provider interface (SSPI) is enabled for server-wide users already logged into the system to be accepted as trusted users in Firebird.

Firebird 3 is not limited to these three authentication protocols. A third-party provider (sometimes referred to as a 'pluggable authentication module' or 'PAM') could be plugged into the server interface with little effort or one could be written from scratch. In this paper, we talk only about the three providers supported out-of-the-box.

SRP Protocol

Firebird 3's new user authentication protocol, SRP, is installed as the default provider, meaning that it is configured so that clients logging in through the Firebird 3 client library will need to supply SRP credentials. These credentials (user name and password) cannot be inherited from a legacy security database (security2.fdb or older). One way or another, new credentials must be installed into security3.fdb. A script exists in the Firebird 3 file structure for converting the users in a restored backup of a security2.fdb database to the record format for Firebird 3 security databases. The old passwords do not survive, since long passwords are the key to the security of this protocol.

Quoting from Wikipedia:

"The SRP protocol creates a large private key shared between the two parties in a manner similar to DiffieHellman key exchange, then verifies to both parties that the two keys are identical and that both sides have the user's password.

In cases where encrypted communications as well as authentication are required, the SRP protocol is more secure than the alternative SSH protocol and faster than using Diffie-Hellman key exchange with signed messages.

It is also independent of third parties, unlike Kerberos."

SRP is resistant to "man-in-the-middle" attacks and does not need the key pre-exchange between server and client that SSH would require. A client needs only user name and password. Exchange occurs when the connection is established.

Legacy_Auth Protocol

The legacy authentication protocol is a survivor from older Firebird versions and, of course, their predecessor, InterBase. Passwords, limited to 8 characters, are hashed using a DES algorithm and are relatively easy to sniff and crack. Retaining the legacy protocol is not recommended but it remains available for use where SRP is unavailable to clients, typically where applications are connecting pre-V.3 clients to a V.3 server.

As with SRP, legacy user accounts must be created anew or converted using the upgrade script. The old passwords are lost.

User Managers

The configuration parameter UserManager determines the default protocol for which user accounts are created or altered. In a new installation, it is set to Srp. To change the default user manager, uncomment

```
#UserManager = Srp
```

and change it to

```
UserManager = Legacy_UserManager
```

Alternatively, if you have clients needing to access both, then include both, separated by a comma:

```
UserManager = Srp, Legacy_UserManager
```

The symbols Srp and Legacy_UserManager, along with Win_sspi, are used in the USING <protocol> clause of DDL commands for managing user accounts and for setting up access mappings for externally authenticated users,
Note, when enabling trusted user authentication on Windows, the current setting of UserManager is irrelevant.

Creating or Updating User Accounts

User accounts for Firebird 3 are database objects created, altered and dropped using a DDL syntax while logged in to any database as a user with suitable privileges. That change first appeared in V.2.5 and was enhanced in V.3.0. User names, previously strings, are now object identifiers of up to 31 characters. The syntax has expanded to accommodate accounts for either the default SRP user manager or the legacy one or, indeed, a third-party user manager. They are not mutually exclusive: srp and legacy accounts can coexist in the same security database. However, they are not interchangeable.

Except for ALTER CURRENT USER, these operations require SYSDBA privileges.

The syntax is:

```
CREATE USER username [ options_list ] [ USING PLUGIN plugin_name ]
ALTER USER username [ SET ] [ options_list ] [ USING PLUGIN plugin_name ]
DROP USER username [ USING PLUGIN plugin_name ]
CREATE OR ALTER USER username [ SET ] [ options_list ] [ USING PLUGIN
plugin_name ]
```

Also:

```
ALTER CURRENT USER [ SET ] [ options_list ]
```

which allows the logged-in user to change some of his/her own account options, e.g., password. However, note that it is not recommended for use in Firebird 3.0 if the server is configured with multiple user managers. This caveat may be removed in a later release.

`options_list` is a (possibly empty) list with the following options:

```
PASSWORD 'password'
FIRSTNAME 'string value'
MIDDLENAME 'string value'
LASTNAME 'string value'
ACTIVE
INACTIVE
USING PLUGIN plugin_name
```

The plugin_name argument for the `USING PLUGIN` clause can be `Srp` or `Legacy_UserManager`. If you omit the `USING PLUGIN` clause, the request will use the

configured user manager by default, or the first (leftmost) one if two or more user managers are configured. Because it is possible to duplicate user names under different user managers, it is recommended to specify the user manager whenever you make these requests.

SYSDBA

Be aware that both user managers provide a ready-created SYSDBA user if an installer kit was used. The SYSDBA password under Legacy_UserManager would be preset to 'masterke', as before. Under Srp, a password is set for SYSDBA during installer execution, where it may be either entered by the user or generated automatically and saved to disk.

If you did not use an installer or the installation was broken then, to initialise authentication and thus make the server usable, one must connect to a database in embedded mode with SYSDBA as user and set a password, e.g.

```
> isql employee -user SYSDBA
Database: employee, User: SYSDBA
SQL> create or alter user SYSDBA set password 'sardineBurger'
CON> using plugin srp;
SQL> commit;
```

Tip: it is recommended to use the `CREATE OR ALTER` verb for this operation.

The extensions include the ability to include attribute tags, although we do not discuss that here. You can look up those details in the Security chapter of the Firebird 3 release notes.

gsec

The old *gsec* utility can still be used, although only with the "global" security3.gdb database. It is deprecated and there are no good reasons to use it. It does not support all of the security features in Firebird 3; nor can it be used to manage users in dedicated or embedded security databases.

Upgrading a Security Database – Notes

- The script in the /misc/upgrade/security sub-directory of the Firebird 3 installation is accompanied by a text file with instructions. More detailed instructions can be found In Chapter 12 of the Firebird 3 release notes, in the topic Upgrading a v.2.x Security Database.
- The upgrade script generates new long passwords for each user, one by one, based on a GUID, which the administrator will need to record manually as they are output. There is no way to retrieve either legacy or Srp passwords from any security database.

- If you want the upgrade script to create legacy user accounts, you must first change the UserManager parameter to Legacy_UserManager, on its own. Only the first 8 characters of the password will be recognised. Remember to restart the Firebird server after any change to a config file.

Trusted User Authentication

A third option for user authentication is Win_Sspi which, if configured, will enable trusted user authentication for Windows clients on a Windows server. With this style of authentication, the security database is not consulted: the engine relies on the login security of the OS to be trustworthy. Since Windows users can be set up to log in without passwords, be careful about this.

Database Access for Trusted Users

Understand that trusted user authentication—like other server-wide methods— gives the users access only to the server. An attempt to connect to a database returns the error "Statement failed, SQLSTATE = 28000 Missing security context for ..{database filename or alias}". A mapping must exist to allow a trusted user access to a specific database or, optionally, to any database on the server. Mapping is a new feature in Firebird 3, explained later. As with native users, trusted users will then require privileges within each database to access objects that are not accessible to PUBLIC.

Configuring Authentication

Because of the granularity introduced in Firebird 3—per-database configuration and multiple authentication protocols—the configuration for authentication in Firebird 3 is split into two parts: AuthServer governs the network protocols that the server can accept, while AuthClient governs the protocol[s] that the client may use. The Authentication parameter in the Firebird 2 series is gone.

For the three authentication protocols that ship with Firebird 3, the options in the global firebird.conf are Srp, Legacy_Auth and Win_Sspi. These are comma-separated string symbols (not quoted strings), representing the secure remote password, legacy and trusted user protocols, respectively.

AuthServer

By default, at installation AuthServer is set to Srp, allowing the server to accept only secure connections.

When more than one plugin is configured for authentication, in most cases, the server will try to process a connection by attempting to match the connection credentials with the symbols listed in the AuthServer parameter in left-to-right order. However, the processing order is implementation-defined. The first working plugin from the client's AuthClient list is tried by the client in the very first (CONNECT) packet sent to the server

and it will take precedence if that plugin is present in the server configuration. For example:

Server side:
```
AuthServer = Srp, Win_Sspi
```
Client side:
```
AuthClient = Srp, Legacy_Auth, Win_Sspi
```

When the user connects without providing login credentials, the client tries Srp and Legacy_Auth, both of which fail because of the missing credentials. The connection succeeds with Win_Sspi, the server accepting that the connection request is not saying "try first with Srp".

AuthClient

In a configuration file on the server side, the values for AuthClient in a new installation simply reflect the protocols available by default to connect from a Firebird 3 client, i.e., all of them. Any reconfiguration of this parameter in a server-side file has no effect on clients requesting connection from remote network locations: it governs only non-embedded client connections that are requested from the server, typically from an EXECUTE STATEMENT call in a PSQL module or block.

Any client application that needs to have its connection protocol restricted to just one protocol can do so in one of two ways:

- directly in the API, DPB or SPB of the connection request, by passing a configuration fragment as a string argument on the connection tag isc_dbp_config or isc_spb_config, as the case may be. The form is:
  ```
  isc_dbp_config <string-length> "AuthClient=Legacy_Auth"
  ```
  E.g.,
  ```
  isc_sbp_config 22 "AuthClient=Legacy_Auth"
  ```

  When multiple parameters are configured, use the \n (newline) symbol to separate the parameters within the string, counting each \n as one character.

- by including a configuration file (firebird.conf or databases.conf) that contains the desired AuthClient configuration in the client application root directory.

A parameter that is configured in the isc_xxx_ config tag takes precedence over the equivalent setting in a conf file. As with the server-side conf files, a setting in databases.conf take precedence over any setting for the same parameter in firebird.conf.

Other Parameters

Certain other parameters are pertinent to configuration files that are split between the server and the client. Of particular interest are :

- Wirecrypt, which sets encryption across the wire. In the default installation it is set as Required on the server side and Enabled on the V.3 client side. If you expect applications to connect using the V.2 client, or the V.3 client with win_sspi, then it should be downgraded to Enabled in the conf file on the server. The default on the client side is Enabled, which is OK, so there is no need to reconfigure it there.

- WireCompression, a True/False value, determines whether data should be compressed across the wire. It is really only useful where the client has extremely slow communication with the server. It is a client-side-only setting and is false by default in a new server installation. On the client side, if needed, it can be set true if the application uses the V.3 client; otherwise, don't include it at all.

- IpcName, RemotePipeName, RemoteServiceName and RemoteServicePort should be configured at both client and server. Take into account that RemoteServicePort can also be set in the connection string host/{RemoteServicePort}:employee. If the client and server sides do not match, a connection cannot be established.
- RemoteAuxPort is configured only at the server side.

- SecurityDatabase. This is the parameter that provides the location and name of the security database file. In a new installation,in firebird.conf, it is configured to be the global database:
  ```
  #SecurityDatabase = $(dir_secDb)/security3.fdb
  ```

  It is a server-side setting only. But, of course, Firebird 3 allows a variety of alternatives to the older "one-fits-all" model, including embedding the authentication account structures inside a user database or having separate security databases for individual databases or groups of databases on the same server. Its scope in firebird.conf, is server-wide; in databases.conf, it can be used in the configs for individual databases, to direct the provider to a non-global stand-alone or embedded security database.

Firebird 3 Split Configurations

With advent of database-level and connection-level configuration and dedicated security databases, Firebird 3 provides plenty of granularity to suit the authentication scenarios of most deployments. Splits can be distributed four ways, depending on how you plan to manage authentication (as well as some other cross-border elements) for your deployment.

Server Side

On the server side, firebird.conf stores all of the parameters that apply to all databases to which it can attach, i.e., databases with the on-disk structure (ODS) 12. This is the first Firebird version that cannot attach to databases having an older ODS. Many of the configuration parameters, including those related to authentication, can be overridden for specific databases by setting them in databases.conf. Setting up configurations in databases.conf is explained in some detail in the Chapter 6 of the Firebird 3 release notes, under the topic Per-database Configuration.

Client Side

You can also have a firebird.conf or a databases.conf, or both, at the client side. On Windows, these files should be in the same location as the executable that connects to databases and it is advisable to place fbclient.dll in that location, too. On POSIX clients the files can be placed in /opt/firebird or in some other directory, according to how the binaries are configured. This might be something like /etc/firebird/firebird.conf or something else.

In these client-side .conf files you configure the behaviour you want for that application for parameters that cross borders. If you want the application to attach to more than one database and you want different behaviour for connection to one database than for another, you would configure the global behaviours in the client-side firebird.conf and the database-specific ones in the client-side databases.conf.

Mapping SSPI-Authenticated Users

The only way to enable trusted users to attach to a database is for the SYSDBA or equivalent to create a mapping in that database. The most generalised mapping is a global one providing access for all Windows user accounts to all databases:

```
create global mapping trusted_users
using plugin win_sspi
from any user to user
```

- The identifier used in the example, `trusted_users`, could be any valid identifier of 31 or fewer characters.
- `from any user` makes the mapping apply to any user that has been authenticated by the SSPI
- `to user` specifies that the user's own Windows account name will be recognised and applied to `CURRENT_USER` in any database to which he/she connects.

  Once the mapping is committed, the trusted user should be able to connect to a database without any login credentials:

```
SQL> connect 'DBSERVER:employee';
Database:DBSERVER:employee, User: OURDOMAIN\JACK
SQL> select current_user from rdb$database;

USER
==============================
OURDOMAIN\JACK
```

Reminder: make sure that the config parameter WireCrypt is either 'Enabled' or 'Disabled' at both client and server sides.

At this point, of course, the user will be able to access only those objects that have privileges granted to PUBLIC. To go further will require SYSDBA or another user with owner privileges to create one or more roles packaging the special privileges and granting those roles to the appropriate user names.

If you want the actual account name for some users to be substituted with something in common for each specified user, invent a name and add it as an argument to this clause, e.g.

```
...
from user "ourdomain\sherlock" to user dog_walker
```

As an example of how a name switch might be useful, one could define a role named Accounts, consisting of the privileges needed by users in a group named "Accounts". The role ACCOUNTS would then be granted to accounts_user.

A mapping in the Accounts database for each user in that group would switch the name to the common name accounts_user. E.g.

```
connect accounts.db;
create mapping accounts_users
using plugin win_sspi
from user "ourdomain\Fred" to user accounts_user;
```

Note: The double-quotes around the SSPI user name are needed in order for the backslash to be handled correctly.

Mapping SSPI User Groups to SQL Roles

The example above provides a workaround for mapping of SSPI user groups to SQL roles indirectly. As at Firebird 3.0.3, the ability to map them directly is still awaiting implementation in a later sub-release.

However, one group, Windows domain administrators, can be mapped so that its members assume the built-in RDB$ADMIN role automatically if they do not connect with any other role. A notional "well-known RID" group, DOMAIN_ANY_RID_ADMINS, is applied for this specific purpose. This is how it is done:

```
create global mapping win_admin
using plugin win_sspi
from predefined_group DOMAIN_ANY_RID_ADMINS
to role RDB$ADMIN;
```

Note, even with this mapping in place, the Windows administrator will not be assigned the RDB$ADMIN role, globally or in any database, if a role is passed in the DPB. In that case, the client application will be able to call SET TRUSTED ROLE during the session if the admin user needs it.

Giving SYSDBA Rights to a Specific User

If you have a SSPI user who is competent and trusted to have SYSDBA rights in all databases, including the security database, it can be done with a mapping:

```
create global mapping cto_sysdba
using plugin win_sspi
from user "ourdomain\cto"
to user sysdba
```

Examples

These examples are far from being exhaustive—merely a source of guidance for a few of the many possible deployment conditions. It is assumed that any necessary privileges and mappings are already present in the databases.

As a general rule of thumb, it is unnecessary to configure on both sides. The capabilities are there to meet the needs for unusual deployments. Each of the examples here will work fine with the default configuration in the client-side firebird.conf. It does no harm to configure the defaults explicitly at both sides, reflecting the defaults, as long as they are correct and consistent, of course.

Firebird 3 clients, native Srp Authentication

In our simplest scenario, the application uses the Firebird 3 client and server on any supported OS platform. Workstations could be on Windows, Linux or MacOSX. House rules require legacy and Windows SSPI authentication routes to be blocked and for data across the wire to be encrypted.

This is native SRP authentication, using the installation defaults. There is no need change anything in firebird.conf on the server side, although here we chose to remove the symbols for the AuthServer plugins we want to block.

Configuration Files

Server Side		Client Side	
firebird.conf	UserManager = Srp AuthServer = Srp AuthClient See note 1 WireCrypt = Required	firebird.conf	AuthClient = Srp WireCrypt = Required

At the client side we are blocking the unused protocols in AuthClient and upgrading WireCrypt from Enabled to Required. The latter is not essential: Enabled will do, since the server requires it by default. Thus, we reinforce security and avoid attachments to pre-V.3 servers.

Note 1: Set AuthClient to enable such authentication protocol[s] as you require for server-based operations, such as an EXECUTE STATEMENT task, to attach to databases (self or other).

Firebird 3 clients, SSPI Authentication

For this one, our application is using the Firebird 3 client from users on Windows workstations in the Windows network to connect to one database on a Windows server.

We want all users to connect using their SSPI accounts.

Configuration Files

Server Side		Client Side	
firebird.conf	UserManager doesn't matter AuthServer = Win_Sspi AuthClient See note 1 above WireCrypt = Enabled	firebird.conf	AuthClient = Win_Sspi WireCrypt = Disabled

With Windows SSPI as the authentication protocol, Firebird's authentication process will actually skip checking the connection packet against the unusable protocols and proceed with Win_Sspi from the outset.

An Unholy Mix of Everything

Now, we look at a slightly more complicated scenario. We have two databases on a Windows server, aliased as Sales.db and Accounts.db. Versions of several client applications are running from Windows, Linux and MacOSX workstations: point-of-sale, sales ordering, stock control, accounts receivable, purchasing, accounts payable, all accessing Sales.db; payroll, HR, assets management, investments and other stuff accessing Accounts.db.

Windows clients using the Sales applications are to be allowed access to the Sales database without having to log in: mappings and privileges will be needed for them. Users connecting from their Sales applications on Linux, Android, MacOSX or POS devices will be required to use one or other of the native authentication protocols. The preference is for Srp but the applications on our old POS terminals and PDE devices still use the Firebird 2 client.

Only selected members of the Windows Administrator group may access the Accounts database, using their elevated OS credentials. We have to set up mappings for that and privileges to control each user's specific access to the non-public database objects.

Configuration Files

Server Side		Client Side	
firebird.conf	UserManager = Srp,Legacy_UserManager AuthServer = Srp,Legacy_Auth,Win_Sspi AuthClient See note 1 above WireCrypt = Enabled	firebird.conf	WireCompression = False WireCrypt = Disabled

databases.conf	Accounts.db = ...
	{
	AuthServer = Win_Sspi
	..
	}
		databases.conf	Accounts = ... {
			AuthClient = Win_Sspi
			WireCrypt = Disabled
			..
			}

	Sales.db = ...
	{
	AuthServer = Srp, Legacy_Auth
	...
	}
		Deploy version of databases.conf to suit the client version used with the application	V.3 client apps:
			Sales = ...
			{
			AuthClient = Srp
			}
			
			V.2 client apps:
			Sales = ...
			{
			AuthClient = Legacy_Auth
			}

Conclusion

New capabilities in Firebird 3 to choose a provider for authenticating users and to restrict or enable access to databases can seem overwhelming at first. We hope this article helps to answer some of the questions that have appeared as developers set about migrating databases and applications to Firebird 3 and addressing the security aspects of the task.