Firebird and Security

Pavel Císař
IBPhoenix

Firebird Conference, Hamburg 2007

Security problem domains

- Secure access to server and database
- Data security at database level
- Server exploits

Basic concepts

- All security is based on **user identity**
- Identity is represented by (login) name
- Identity is server-wide
- Identity is a gateway into system
- Identities may have various rights granted or revoked for operations in system
  - Only some operations are reserved for super user
  - Only some existing database objects are protected
- **Security stands and fall with identification and login procedure**

Secure access to server and database

- InterBase was designed for work on secure systems with trust relationships.
- Users and user groups are defined at OS level, database server uses these OS identities.
- This concept is still present in Firebird, but it was available only on Linux/UNIX. Starting from v2.1 it's also available on Windows (but with differences) and it was disabled for POSIX in v1.5.3.
- Security database (isc4.gdb) was added when version 4 was ported to Windows as weak replacement for weak (or absent) built-in OS security.

Physical access to database and server files

- **Server** must have read/write access to all database files.
- **Server** must have read access to all his files, along with executable rights for executables, and write access to some files (lock file etc.)
- User or any other user-space program do not need (and should not have) the physical access to these files at all.
- Using Embedded Firebird makes things more complicated.

Physical access to database and server files II.

- Server should always run under it's own OS identity separate from user's identity in OS.
- Never run the Firebird server under identities with administrative OS rights.
- Security problems:
  - Windows 95/98/ME and NT/2000/XP in application mode
  - Embedded server or local connection method used with Classic server (Linux/UNIX).
  - GSTAT needs physical access to database files.

User identity

- User identity could be taken from OS
  - Client node must be defined as trusted host in /etc/host.equiv or /etc/gds_host.equiv on POSIX
  - Firebird login name and password must not be provided
  - Users with root privileges will get SYSDBA rights
  - This method does not use security database
- User identity is verified against security2.fdb
- Never mix both methods to access databases

Security database

- In main server installation directory
  - isc4.gdb for Firebird 1.0
  - security.fdb for Firebird 1.5
  - security2.fdb for Firebird 2.0
- Tables owned by SYSDBA
  - USERS (RDB$USERS in 2.0): many fields, but only next are important:
    - USER_NAME(128): login name, only first 31 characters are significant
    - PASSWD(32/64): password, encoded, only first 8 characters are significant
    - FIRST_NAME, MIDDLE_NAME, LAST_NAME: user's real name

Super User

- SYSDBA identity
  or anyone with root/admin privileges
- Always have all rights to everything

Other users

- It's possible to define any number of additional users with unique identities
- **Any** known user can attach to **any** database at server
- Only super user can define new users
- Users can't change their security information

User groups and roles

- Two similar but distinct variants for assigning rights to users by membership in group:
  - OS user groups
  - SQL roles

User groups

- Available only for user identities taken from POSIX OS
- Rights assigned to user are inclusive with rights assigned to all groups where user is a member
- Group membership is managed by OS tools

SQL Roles

- Defined for database with CREATE ROLE *name* SQL statement
- Role is not an user group (user is not a member, but has rights to use a role)
- User can specify one role on database login
- User and role names must be different

Database level security I.

- Rights are based on „ownership"
- Object's owner and super user can grant rights on given object to other users, roles, views, procedures and triggers, including right to grant these rights to others
- Only existing databases, roles, user tables, user views and user stored procedures and triggers are protected, all other database objects are free targets (including metadata)
- **Anyone can create new database objects of any type**
- User identity is applied on views, procedures and triggers work. These objects can also have their own privileges to other objects.

Database level security II.

- Table/view privileges:
  - Select
  - Update (can specify columns)
  - Delete
  - Insert
  - References (can specify columns)
  - All (shortcut for all above)
- Stored procedure privileges:
  - Execute
- Role privileges:
  - Use role (role itself is a privilege to use it)

Database level security III.

Rights are managed by GRANT and REVOKE SQL statements. Because they are well known and documented, I'll concentrate only on special, not so well known and documented facts about these statements

Facts about GRANT statement

- Granted privileges are stored in RDB$USER_PRIVILEGES system table
- Privileges are internally identified by distinct combination of:
  - User name
  - Privilege name (SELECT, UPDATE, INSERT, DELETE, REFERENCES, EXECUTE)
  - Table/View/Procedure name
  - Column name (could be empty)
- GRANTing UPDATE and REFERENCES privileges with and without columns are two distinct privileges stored separately, and may result in unwanted side effects

Privileges on Views

- To create a View, user needs the SELECT privilege on selected tables. This privilege could be revoked after view is created
- To use a View, user needs appropriate privileges on View, and also User, View's owner or View itself must have appropriate privileges on base tables
- When View's owner is not SYSDBA, he/she must possess privileges to base tables with GRANT OPTION to grant privileges on View to other users
- Views could be used to restrict user's access to base tables beyond limits of standard privileges

Facts about REVOKE statement

- REVOKE will never tell you what he really performed
- Only user who granted the privilege or super user can revoke it
- REVOKE executed by user only revoke privileges granted by this user. The same privileges granted by other user will remain in effect
- REVOKE executed by SYSDBA revoke all specified privileges regardless who granted them
- Privileges granted to PUBLIC could be revoked only from PUBLIC, not individual users

Building Secure System

If it's ever possible...

Basic server and db security I.

- Use Firebird 2.x
- Forbid physical access to database
  - Someone can steal or damage it
  - Do not allow even the read access
  - Shadow files are database as well
- Hide the database location
  - Unknown databases cannot be used in attack
  - Server-side database aliases
  - Encrypted in client software
- Restrict database location(s)

Basic server and db security II.

- Forbid any access to your backup files
  - Most easy route to have your data stolen
- Forbid any access to server software files
  - Someone may replace your system with Trojan horse, damage it or change your configuration
  - No one should have right to change executable or configuration files (including Firebird itself) except entrusted administrators

Basic server and db security III.

- Use separate OS identity to run the server
  - Never use any identity with administrative privileges
- Set only necessary system rights to server's identity
  - Damaged or compromised server may cause more damage to other systems as well

Server and db access security I.

- Let in only authorized users
  - **You can't stop malicious users once they are in**
  - Do not mix trusted and security db authentication methods
- Protect your SYSDBA account
  - As well known and powerful account, it's a natural target for attacks
  - (At least) change your SYSDBA password
  - You can block SYSDBA account at database level with SYSDBA role

Blocking SYSDBA account for a database with SYSDBA role

- You can't create SYSDBA role with CREATE ROLE statement, so you need to work with system tables directly
- First, create new user account that would grant you access to remove the SYSDBA role (for example LOCKSMITH)
- Execute next commands:
  ```
  INSERT INTO RDB$ROLES
  (RDB$ROLE_NAME, RDB$OWNER_NAME)
  VALUES ('SYSDBA', 'LOCKSMITH');
  COMMIT;
  ```
- Now you can't attach to this database as SYSDBA
- To restore SYSDBA access, log in as LOCKSMITH and execute:
  ```
  DROP ROLE SYSDBA;
  ```

Server and db access security II.

- Secure user accounts
  - If possible, use only internal user accounts known only to your applications (Firebird's password protection is still weak, so anyone can use brute-force to get in with knowledge of user name).
  - Use Trusted Authentication

Server and db access security III.

- Secure your network
  - Allow network access only from trusted nodes
  - Change the TCP/IP port number for communication between clients and server if your network is open to Internet
  - Use encryption at network level for communication over insecure network (paper about how to use the open source ZeBeDe product is on Firebird Project or IBPhoenix sites)

Last defence at database level I.

- Whenever possible, tighten up user privileges to individual database objects
- Never give your users any privilege to grant their privileges or roles to other users
- Protect system tables from direct access

Protecting system tables from direct access

- It's not possible to revoke rights on system tables directly, you must grant these right first:
  - GRANT ALL ON <RDB_table_name> TO PUBLIC
  - From this point onward, access to system table is controlled by normal SQL rights instead internal defaults
- Now you can revoke any right from public with REVOKE statement or grant it with GRANT
- Revocation of SELECT right from PUBLIC may cause troubles with client applications
- This doesn't last the backup/restore, so you must renew your settings after restore
- This only protects system tables from direct changes with DML statements, but doesn't protect from regular changes with valid DDL statements.

Last defence at database level II.

- Restrict use of UDF's and external tables
  - UDF's together with external tables are a mass-destruction weapon that may be used by any attached user to nuke your system
  - Use OS rights or Firebird configuration options to restrict locations for UDF's and external tables only to controlled areas
  - Restrict user's access to these storage areas
  - Firebird should never posses privileges to access user-controlled or system disk storage

That's all (for now)

http://www.ibphoenix.com
We have answers