Proceedings of the 8th International Conference on Applied Informatics
Eger, Hungary, January 27–30, 2010. Vol. 1. pp. 345–352.

Data Protection on Progress Databases and Progress 4GL Environment

Attila Hadházi

1. Introduction

This paper describes a protection mechanism implemented on a Progress RDBMS and in Progress 4GL development environment. The method described below is intended to protect the stored data at a level that to not allow direct access to the database, but only using the application functionality and its user interface. The method ensures that even the database administrator has no direct access to the database, despite the fact that for application upgrade and recompilation reasons higher privileges are needed. The paper only deals with protection against unauthorized access, and does not deal with protection against physical damage of the database.

Due to the internal structure of Progress databases, the simple possession of the files is not sufficient to retrieve the data, this is why the paper does not deal with the protection of the database files themselves, since they are protected on operating system level. This paper describes the protection against intrusion to the database, based on the information related to the schema of the database or to the application accessing the database.

The first part of the document describes the tools provided by the Progress RDBMS - from the point of view of the database itself and from the point of view of the application which access the database.

The second part of the document details the areas which were needed to be covered for protecting the database and application in a real environment - a medication database used in an integrated Health-Care Information System.

The third part describes the steps of technical and organizational implementation of the security system. In particular, it was important to establish a mechanism, which is used in a real environment, having regard to the development workflow, as well as the upgrade process performed by clients of the live system.

Finally, there are discussed the used encrypting algorithms as well as the ways of possible further improvements of the presented security mechanism.

2. The features of the Progress environment

The Progress RDBMS is a product of the Progress company, being a stable and relatively inexpensive product on database management systems' market. At the same time contains a 4GL development environment, which allows a much more effective application development than a simple SQL interface.

The database access can be protected on multiple levels. Given the close relationship between a database of a defined schema and the 4GL program which runs on the database, the Progress database protection has to cover the protection of the actual data files against unauthorized access and protection of the associated application as well.

2.1. Protection of the Database

The Progress database has a protection mechanism covering several areas:

- The restriction and authentication of database connection

  The connection to the database occurs through the servers serving the Progress 4GL clients or through JDBC, ODBC drivers. The system administrator can set which kind of connections are allowed.

  The Progress stores the registered users in a user table in the database. In cases where users are created in the database, the anonymous access to the database can be disabled. There can be assigned passwords to the users which are stored in the database using a 16 characters length one-way encryption. Connecting to the database does not mean that the user can automatically access the database schema and the data stored in the database. Some users can be endowed with a database administrator role. The additional users and schema access rights can be set only by these administrators.

- Protection of the database schema

  After the 4GL-side connection to the database there is started a Progress delivered application, through which 4GL programs can be edited and run, and – owning the appropriate permissions – is possible to review and edit the schema.

  Protection of the database schema means that only certain users can adjust tables, fields and indexes stored in the schema. The individual tables (including itself the table containing the schema definitions can be "frozen". Then, the schema can not be changed.

After connecting to the database through JDBC drivers we have a user interface where SQL commands can be invoked. In this case, using the GRANT and REVOKE commands can be set the schema access permissions in the same way as for other databases.

2.2. Protection of the Application

This section presents the opportunities of 4GL applications protection.

The 4GL source files has to be compiled before runtime. Before compilation the compiler has to connect to the database, on which the program will run. During compilation an intermediate code is produced which is called r-code (runtime code). To run this code just database connection licenses are needed, and no developer licenses.

Thus, running an application is performed in two steps, and based on this the protection of the application has two aspects:

- compilation and creation of r-code - the compile-time protection.
- the actual run - the run-time protection

- Compile-time protection:

  To each table and each field of them there are defined different access modes, and to each access mode there can be attached individual users or user lists (using eve pattern matching). The following access modes are available: read, write, create, delete, data load and data dump. The first four modes does not apply directly to the data content but to the definition of the data.

  For example, if someone does not have 'READ' access to a table, then he/she simply do not even knows what kind of fields are in that table and is not able to compile a program against that table. Since every data access in fact is a piece of 4GL program, without the appropriate privileges, corresponding data cannot be accessed.

- Run-time protection:

  Due to performance reasons, the privileges associated with the schema are checked only at compile time, and are not rechecked when executing the r-code. When the r-code is created by the compiler, in the r-code there is stored a CRC value as well, which is characteristic to the given database schema. That's why a program compiled against a database with a given schema can be run on another database having the same schema - even though the security settings of the two databases differ.

  To solve this security issue, Progress provides the possibility to attach a database authentication key to each database (DBAUTHKEY), and at compile time this DBAUTHKEY is added to the r-code. This r-code can be replaced or added to the r-code afterwards as well. If a database has such a DBAUTHKEY attached, then only those r-codes can be run against it, which contains the same authentication key.

  Further CRC code is the r-code specific CRC, which should not be confused with the previously described CRC-s. This is unique and can be interrogated from the r-code. Using this code, an application can store all the valid r-code filenames and corresponding CRC-s, and in this way can control the programs which are run against the database.

3. The specific frame of the deployment and usage of the MedSolution drug database

The International System House Ltd. provides the drug database to the MedSolution Hospital Information System from an external source. There was a contracted condition to ISH to protect this database in such a way, that the content of it to be accessible by the customers only using the application, and in no other way.

During designing the protection system, the following issues had to be be considered:

- It is possible that the hospital administrators owns Progress developer licenses.
- It is possible to embed new functions into the application even by the administrators with medium MedSolution application level knowledge and Progress 4GL knowledge.
- The compilation process is performed on clients' servers and source code is distributed to the client sites using a special encryption provided by Progress.
- The application is used from Windows fat clients and Unix terminal based thin clients.
- The protection system should fit in the supplier's development and deployment processes.
- Uniform system has to be used which is applied at all customer sites.
- Ensure fast drug database upgrade.

4. The applied data protection system

4.1. The protection of database and application

- The database access protection and authentication

  The medication database has to be connected for two reasons:

  a. To recompile the application after upgrades
  b. To access the drug database through the application functionality

  This usage is more restricted than the access to other tables, where it is needed sometime to perform ad-hoc queries. To handle this on given sites the system administrators have development licenses and access to the schema of the tables. To restrict direct access to drug database tables (around a dozen tables) these were put in a separate physical database. There were only two users defined in the database, a 'dba' and a 'user'. The dba user has access to the schema (so can compile programs against it), while the plain user is able only to connect to the database. Of course, none of these two users password is not known by customers, but were implemented in the compiler and installer tool, and the application itself. In this way direct connection to the database was disabled.

When running the application each user logs in with his own account and authenticates. After successful authentication, the application connects and authenticates to the separate drug database with the 'user' account. This allows running the already compiled r-codes.

- The protection of the schema

  In the absence of appropriate passwords the database schema can not be accessed. But during upgrades the schema information is provided in plain text files. It would be possible that having a full schema upgrade these files to be accessed at customer sites. Having the schema a database can be built and r-codes compliant with the drug database can be produced.

  To avoid this, the schema text files are deployed with a bit level XOR ciphering (CBC - cipherblock chaining mode) having a relatively long cipher key. The decoding algorithm together with the key were embedded into the installer application. During installation the installer decodes the schema text file, connects with dba access to the drug database and applies the schema changes to the database. During installation, the decoded data content is not accessible.

- The protection of the r-codes

  The drug database can be accessed with 'dba' access by the compiler application as well. Since the application is enough freely configurable, it is possible that somebody writes a program, which for example lists the content of the tables, compiles it to r-code and configures this r-code in the menu system of the application, and then runs it. To write such a program is enough to know the name of the different tables. To avoid this, the compiler application was enhanced in the following way:

  - In the compiler there are included all r-codes' name which accesses the drug database.
  - To avoid the case that somebody overwrites a registered program with his own code, the r-code CRC-s of the registered programs were added to the compiler as well. The r-code CRC can be extracted from the r-code with a Progress statement. The compiler performs the validation of a program in the following steps:
    i. The compiler is normally not connected to the drug database, and the not registered programs are compiled only against the other tables of the database of the information system.
    ii. Before the deployment of a registered program there is added to it a header macro. Activating it the registered program can be compiled without any connection to the drug database, of course these r-codes has no real functionality at all.
    iii. The r-code CRC of these programs is extracted from the r-code file, and compared to the value stored in the compiler.

iv. If the values are matching the compiler connects as 'dba' to the drug database and compiles the program deactivating the header macro first. In this way a 'real' r-code is produced.

- So the compiler does not check the CRC of the 'real' programs, but only the CRC of such protected ones, and in this way real compilation against the drug database occurs only in the case of registered programs having valid CRC-s. Using the presented mechanism it is allowed to compile only those programs against the drug database which were deployed by the ISH development staff.

- The protection of drug data upgrades
  - The drug database tables were received initially in plain CSV files and it was needed to be loaded using the application. So, these files had to be coded in such a way that only the application could decode them. Unfortunately the size of drug database is more than 70 MB and the coding used at schema text files in this case would take too much time. The solution of the transfer was the usage of binary files which is another feature of Progress RDBMS:
    i. Binary export and import is allowed only to one single table at a time, so before migration the structured tables were moved in a migration table.
    ii. This table is exported in binary format which is deployed to the customers.
    iii. The application at customer sites loads the binary file in the migration table and rebuilds the new version of the structured drug database.
  - This process can be performed only if the schema at customer site is the same as the structure existing at the distribution site. To ensure this a version mechanism of the loader program is implemented. If the binary file version does not match with the version of application at client sites, first the application has to be upgraded.

- Usage of encrypted columns

Unfortunately this feature known in Oracle databases cannot be applied here.

4.2. The changes related to the workflow of development

During the implementation of the security mechanism it had to be ensured that the preparation of registered programs etc. will not create a significant overhead on the development process. Therefore, the following changes of development process were applied and new development tools were used:

- On the development versions where the access to the drug database is more often needed, the security mechanism was deactivated.
- The schema upgrade files are encrypted with a specific tool in one step.

- The registration of the specific program files was performed with another tool, which:
  - Adds to the source program the header macro which ensures the protected and 'sharp' compilation.
  - Compiles the source program in protected mode.
  - From the created r-code extracts the CRC value.
  - Registers the program and the CRC value into the compiler application which will be deployed together with the source file.
- There were minimized the number of programs which access the drug database, creating general usage programs which can be invoked in different modes. In this way the number of programs which have to be registered was reduced.

There can be deduced, that when a program which accesses the drug database is changed, there has to be deployed a new version of the compiler application as well.

5. Ciphering methodology

To cipher the schema text files, different ciphering methods were checked:

- Electronic CodeBook mode (ECB) - in this case the data to be coded is split in blocks of equal size with the length of a predefined key. The key is applied to each block consecutively using bitwise XOR operator. There was taken a key of 30 bytes length.
- CipherBlock Chaining mode (CBC) - the process is similar to the previous one, but the applied key for each block is different, being the result of the ciphering of the previous block. In this case the key is context based and practically impossible to be guessed. Same length was chosen for the initial code.
- Affine Cipher mode (AFC) - having some predefined key values to code a text on byte level the following conversion is applied: $y = (a * x + b) \bmod m$; In this case decoding can be done using the following formula: $x = c * (y - b) \bmod m$; where $a * c \bmod m = 1$, b being an arbitrary chosen value. m should be at least as high as the number of different characters.

The performance related results are shown in the following table:

		Progress 4GL		C	
algorithm	size (bytes)	msec	byte/msec	msec	byte/msec
ECB	10 000	954	10	2	5 000
ECB	1 000 000	100 706	10	244	4 098
ECB	70 000 000	7 200 314	10	17 144	4 083
CBC	10 000	1 667	6	2	5 000
CBC	1 000 000	111 197	9	260	3 846
CBC	70 000 000	-	-	19 205	3 645
AFC	10 000	155	65	1	10 000
AFC	1 000 000	12 593	79	383	12 611
AFC	70 000 000	-	-	4 913	14 248

Comments:

- AFC ciphering is significantly faster than the other two methods, however it was not used, being not as safe as the other ciphering modes.
- The reason of the big difference in performance among implementation of ciphering algorithms in Progress 4GL and C is caused by the fact that in Progress 4GL there are no bitwise operations implemented.
- For files of size of just a few kilobytes ciphering in 4GL is acceptable, but for files with bigger size another implementation is needed.

6. Conclusions, possible enhancements

At the time of creating this protection mechanism the remote access technologies were not mature and not sufficiently fast. This is why the coding and decoding of the database had to be performed directly on the customer servers.

Currently it is appropriate to analyse a SOA based architecture to refresh the drug database. In this way the data is completely hidden from the customers and there is provided just a SOA service which is able to refresh the database.

References

[1] Johannes Buchmann, Introduction to cryptography. Second edition. Undergraduate Texts in Mathematics. Springer-Verlag, New York, 2004.

[2] Progress Version 9.1 Documentation - Progress Language Reference

[3] Progress Version 9.1 Documentation - Progress Programming Handbook

[4] Progress Version 9.1 Documentation - Progress Client Deployment Guide

[5] Progress Version 9.1 Documentation - Progress Database Administration Guide and Reference