Journal of Mobile, Embedded and Distributed Systems, vol. VII, no. 1, 2015
ISSN 2067 – 4074

Banking Software Applications Security

Ioan Alexandru BUBU
Department of Economic Informatics and Cybernetics
The Bucharest University of Economic Studies
ROMANIA
alexandru.bubu@gmail.com

Abstract: Computer software products are among the most complex artifacts, if not the most complex artifacts mankind has created. Securing those artifacts against intelligent attackers who try to exploit flaws in software design and construct is a great challenge too.
The purpose of this paper is to introduce a secure alternative to banking software applications that are currently in use. This new application aims to cover most of the well-known vulnerabilities that plague the majority of current software.
First we will take a quick look at current security methods that are in use, and a few known vulnerabilities. After this, we will discuss the security measures implemented in my application, and finally, we will the results of implementing them.

Key-Words: C#, SQL Server, Security, DBMS, Token, Credentials, MD5

1. Introduction

Software Security is the process of engineering software so that the software can continue to withstand and function correctly under malicious attacks.
Today's software is troubled with both design flaws and implementation bugs, resulting in unacceptable security risks.
The notion of software security risk has become common, yet we have only recently begun to systematically investigate how to build secure software system. The practice of software security remains still in its infancy.
Software defects with security ramifications-including implementation bugs (such as buffer overflows) and design flaws (such as inconsistent error handling)-promise to be with us for years. Internet based software applications are easy to exploit and have become common attack targets. Good software security practice leverages good software engineering principles and practices. It involves thinking security early in the software lifecycle, knowing and understanding common problems (such as language-based flaws and pitfalls), designing for security, and subjecting all software artifacts to thorough risk analysis, review and testing.
Software can be intentionally malicious such as viruses, Trojans and software containing logic bombs (malicious functions set off when specified conditions are met). However, attacks against computer systems are not limited to intentionally malicious software. Benign software can contain vulnerabilities and such vulnerabilities can be exploited to make the benign software do malicious things. A successful exploit has traditionally been the same as an intrusion.
SQL injection for example, is a code injection technique, used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker). SQL injection must exploit a security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed.

2. Security of Banking Software Applications

Banking software applications, like Internet banking, which are exposed to users on public networks, are vulnerable to security threats.

The evolution of information technologies has facilitated the development of complex applications, primarily for large companies. In a banking system, customer records, bank loans and other processes cannot be recorded otherwise than by a complex computer applications. The application made in this study is a simplified form of a real life one and focused on implementing similar functions. Database design and implementation of the application will be based on the theoretical aspects of the security problem under review. Thus, every table in the database must retain information on the entities involved in the process of lending (e.g. borrowers), types of credit products launched by a certain bank branches and bank officers. The application also implements the loan approval process and the workflow associated with it. The application also must have a user friendly interface to simplify interaction with the user, which should not require thorough preparation prior to its use. Many program options must be intuitive and very easy to access.
Evidence of bank loans are of special importance for a bank, being its main activity and making most of the banks profit. One such application is very useful because it has a clear record of persons with late payment of loans, and allows the bank to initiate actions to recover the amounts owed by the customer.
The objective of this application is related to the processing of input data and storing them in the database of the program to the management of information relating to individuals who took out loans.
Another purpose of the software is to realize an application that is built on the same model as a real application and intended to perform each operation used by a normal bank.

3. Security Solutions

Security is a very important feature for any software and essential for a banking application. A secure application behaves correctly and predictable despite the influence factors.
In the context of information, security is the process of preventing and protecting against access to information by unauthorized recipients and preventing information destruction or alteration of important data. Security can be seen as the ability of a system to protect information and system resources in terms of confidentiality and integrity. Basically, security must be met at each component of the application from the database until the GUI.
Among the advantages of a software product that meets this feature, we can mention:

- protecting bank and customer data
- minimize costs regarding remediation of vulnerabilities discovered during the lifetime of the product;
- avoiding penalties caused by attacks
- avoiding loss of image and customer confidence caused by attacks;
- differentiation from competitors.

3.1 Database Security

Along the objectives of a database management system, protecting data ranks the highest. In database theory, data protection has two aspects: security and integrity. Security is the fact that access to the data is controlled, being the task of the database administrator to ensure security. In this respect, SQL Server enables authorization and access control data views using data encryption. Authorization and data access control is achieved through passwords. There are classes of users with certain access rights implemented. A subject (user) can perform certain actions on certain objects, within certain restrictions (additional conditions). User profile is given name, password, group name and access level. Using views aims to fulfill security features by allowing certain users to have access to a logical representation of a portion of the database according to their requirements for access. Users' rights are ensured by the database administrator, who can grant or cancel them at any time if necessary. Encryption in SQL Server is ensured by providing data encryption routines that can be called automatically or on demand, and by allowing the developer to carry out its own encryption

ISSN 2067 – 4074

routines.
Data integrity refers to the correctness (consistency) of the data and is secured by protecting them against intentional or unintentional incidents. Components of DBMS ensures data integrity by treating separately the cases that can damage database: semantic integrity, access control competitor, saving / restoring.
Semantic integrity is ensured by the operations performed by the database management system. These operations make up a set of rules called integrity constraints.
Concurrent access ensures data consistency. In this respect, Sql Server uses a separate data processing unit called transactions, which consists of a sequence of operations (that run entirely or not at all) marked with start and ending points. The transaction can be controlled by default, when the start and end points are automatically defined, or explicitly when the start and end points are defined by specific commands. Concurrent execution of transactions ensures blocking the used data for a period of time. This means you are prohibited access to the same data to other concurrent transactions until the current transaction ends. This blocking technique can be applied to the entire database, a file, a record or a field. It can be read (shared) or write (exclusive).
Backup / Restore (backup / recovery) allows restoring data consistency that have been physically damaged for various reasons. Saving data storage is the process of making backups copies and storing them for later use.
At the database level, the following measures are implemented to ensure security:

- Server Login;
- Database Login;
- Fixed database roles;
- User-defined roles;
- Roles at the application level;
- Object-level permissions;

3.2 GUI Security

Application Authentication is done by using a code generator token.

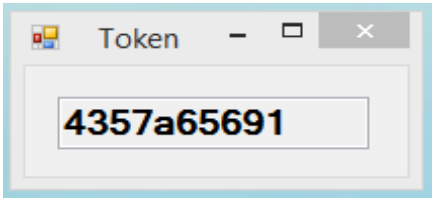

Figure 1. Token generator

It generates a sequence of code valid only for a minute, by using an MD5 encryption algorithm applied to the workstation date, which has a format of: *dd- mm - yyyy hh: mm*. From this sequence of characters resulted, the first ten characters are selected and displayed in the token application GUI. The user has a span of 60 seconds in order to enter its username, password concatenated with the sequence of code generated by the token and choose the type of account he has in order to be able to log into the application.

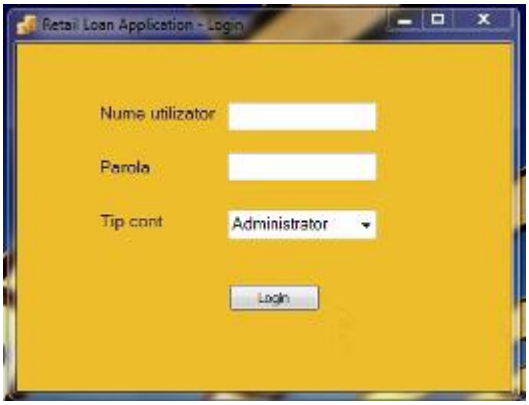

Figure 2. Log in form.

Users are uniquely registered at the application level and the password is randomly generated when the administrator creates the account. It does not have access to the user's passwords, which are generated and sent automatically by e-mail.

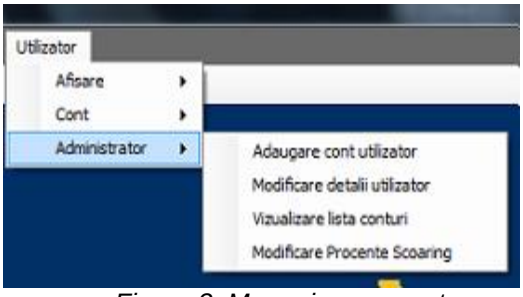

Figure 3. Managing accounts.

In conclusion, the GUI security is achieved by:

- Application Login using token generated codes;
- Compliance data formats introduced in application;
- Input validation;
- User accounts with different permissions.

The validation methods used in my application, are needed both to improve application response time and to increase security. Data validation is an activity with important implications for civil computer products.

Validate the user input data can be achieved by two methods:

- Integrity restrictions applied to the fields in the database;
- validation checks on user data input before saving the data in the database.

The following examples of data validation can be applied to the database on each individual field and at the application level:

- the social security number of the client must be numeric and must not exceed 13 characters;
- telephone numbers for customers or bank branches, must not exceed 10 characters;
- No credit may be granted to a person if the outcome of the scoring process is one below 50% of the maximum score;
- Bank officer must have only one set of login credentials;
- Dates will be format dd / mm / yyyy ;

3.3 Data flow analysis

For an overview of how the system will be used, the following UML diagrams have been created.
Figure 4 shows the general diagram of use cases.

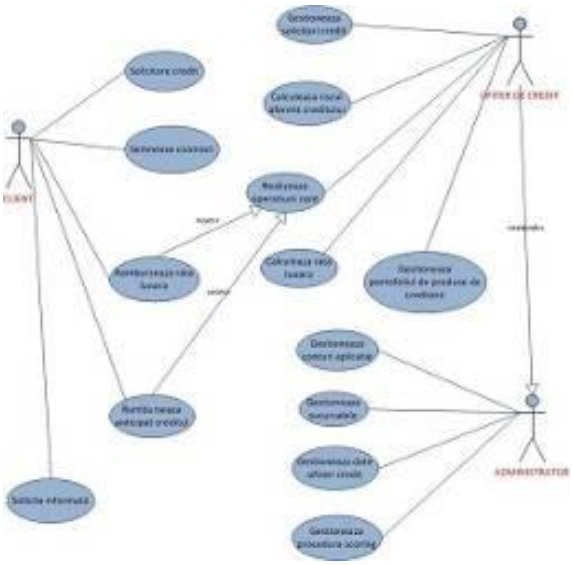

Figure 4. General diagram of use cases.

It describes how external users with role players interact with computer application, serving to present graphically functionality that the system must meet its final phase. In this chart there are all the actors that are using the application, namely: customer loan officer, application administrator and application supervisor. The administrator is the one who can operate the maximum number of options in the application, having access to the user role management, information about branches and loan officers as well as the percentages of the scoring procedure.
The loan officer is the operator of the application itself, as their portfolio includes client's management, potential clients and banking actions, credit risk calculation, implementation of projects on the repayment of loans etc. Loan officer have access to a number of reports on credits already under way, which relate to the number and value of loans per bank officer or amount of loans by currency.

4. Database architecture

The application consists of the actual programs (main program, coordinating programs for menus, screens, queries, updates, reports etc.) and the database . To design a database one must start from the main objectives. In the case of our application, the main goal is that the database provides data independence from the programs that uses it, both

ISSN 2067 – 4074

logically and physically, and to use a language as close as possible to the natural language in order to enable easy operation on the database to end users. This will enable access to data more efficiently, by using data manipulation language to centrally manage data and ensure minimum redundancy in stored values. Another important aspect of the database design, is data security and confidentiality. DBMS must ensure the physical and logical security of databases and ensure that only authorized users perform operations on the data.

The first step towards creating the database relates to system analysis and includes three aspects: structural, dynamic and functional. The purpose of system analysis is to highlight the application requirements and the resources used. Structural analysis shows how the data is structured and the links between them through the entity - association technique. This technique involves identifying entities (borrowers, loans, branches, criteria and sub-criteria), the associations between them, the attributes characterizing each entity and the establishment of unique identification attributes.

Dynamic analysis shows the behavior of system components at certain events and is represented by a transition states diagram. To achieve this diagram one must identify the states in which they can find events that cause system components and component switching from one state to another. Functional analysis highlights the processing flow within the system or the way in which inputs are transformed into outputs.

The second stage in creating a database is the design structure. Design work is based on two aspects: the choice of DBMS and the creation of database functions. In choosing the DBMS one must take into account the user requirements and technical performance. In the case of an application that tracks bank loans, it is estimated that it would need to handle a large volume of data to be stored and processed in the database. The SQL Server DBMS is a suitable choice when designing such applications, offering opportunities for large volumes of data storage.

After the design phase, next comes the creation of logical components and program applications. Programs take into account the flow of inputs / outputs, the processing of data and the data collection.

The fourth step in the creation of a DB consists of its commissioning and its daily use. At this stage, the testing of the database functions occurs, first with test data and after with real data. Data is loaded and procedures simulating the data daily are employed and the application documentation is completed. Commissioning the application does not mean that this will be its final form, changes often occur over time, during the operating cycle.

In the process of creating the conceptual scheme, we need to identify the necessary data system.

An application used for bank loans management is required to store personal data about its clients, bank loan characteristics, bank branches and its employees, the loan officer who submits credit application, data about current contracts etc.

These data are structured in a scheme normalization technique shown below.

5. Conclusions

The goal of providing a totally secure banking environment continues to be a moving target exacerbated by the free and readily available flow of information at the disposal of would be attackers. As new security measures are devised, new attack vectors are developed to thwart these measures. For too long, code writers have been measured on the features built into their applications and not on the potential security vulnerabilities. It is time to change that perspective and treat security as business enabler.

The evolution of information technologies has facilitated the development of complex applications, primarily for large companies. In a banking system, customer records and processes can be recorded only by using complex applications. The software program analyzed in this study is focused on

lowering the risks assumed by the bank. The application also has a friendly interface which does not require prior training.
The advantages of this computer product are:

- substantial reduction in processing time information;
- automation of manual processes;
- reducing the risks assumed by the banks when granting loans;
- reducing the number of staff dealing with the customers;
- friendly and intuitive interface.

The theoretical aspects are combined with the practical implementation of the computer program and the technologies used are among the newest and most used in this field.

Acknowledgement

Parts of this paper were presented at The 7th International Conference on Security for Information Technology and Communications (SECITC 2014), Bucharest, Romania, 12-13 June 2014.

References

[1] Michael G. Solomon, Security Strategies in Windows Platforms and Applications, *Jones & Bartlett,* 2013
[2] M.Velicanu, I.Lungu, M.Muntean, S. Ionescu, – *Sisteme de baze de date – teorie şi practică*, *Petrion*, 2003
[3] Ion Smeureanu, Marian Dârdală – *Visual C#.Net, ASE, 2004*