Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 53 (2012) 11819-11823

ISSN: 2229-712X

Preventing SQL injection attacks using Blowfish and RSA

Aakash Ahuja, Pulkit Arora, Shashank Singh, Shobhit Srivastava, Saravanakumar Kandasamy
School of Information Technology and Engineering, VIT University, Vellore, India.

ARTICLE INFO

Article history:
Received: 25 October 2012;
Received in revised form:
20 November 2012;
Accepted: 3 December 2012;

Keywords
SQL injection attacks,
Web security,
Authentication,
Blowfish, RSA,
Unique key.

ABSTRACT
SQL injection attacks on the web databases are mainly due to the application development process where the coding process is vulnerable as it was not secured. This however can be prevented by various methods. One of the techniques is to limit the access of database to authorized users only. Database contents are encrypted so as to allow a secure way of efficient query processing directly on the encrypted database. SQL attacks can be prevented through highly secure authentication schemes in the login phase itself. In this paper, we have presented one such technique. Our scheme proposes that access be provided to verified users only. That is, at the time of creation of the user account, a user key is generated for every user where the user name and password at the time of login is encrypted by Blowfish encryption and RSA technique at different levels of the total encryption process. The access is provided by the server after confirming the user's authenticity. On server side the encrypted data will be decrypted using the user key. The decrypted data will be checked and if the user is genuine, further access will be granted to the database. The RSA encryption will work as a protective cover for the SQL query generated by the user at the client's end.

Introduction

Internet is growing day by day but most of the people are not aware of security and privacy. It is not a secure channel for exchange of information. Nowadays most of the applications are vulnerable making them a threat possible. An attack may be possible due to poor design, configuration mistakes, or poor written code of the web application.

An attacking technique that is being widely used is Structured Query Language (SQL) Injection. It is a method for exploiting web applications by inserting SQL meta-characters and commands into Web-based input fields in order to manipulate the execution of the back-end SQL queries [1]. It is too vulnerable that it can bypass many traditional security layers like Firewall and encryption and also bypassing the authentication of user, which is a big flaw in the web applications [2].

Login page is the most complicated web application which allows users to enter into the database after authentication. In this page, the user provides his identity like username and password. There might be some invalid input validations which can bypass the authentication process using some mechanism like SQL injection. Whenever a user wants to enter into the database, he/she inputs his/her authentication information (Username and Password). If result of the query is true then the user is authenticated otherwise, denied. But, there are some attacks which can mislead the database server. The user can enter malicious code through SQL injection which will always return true results of the authentication query. For example the user enters the expression in the Username field like " ' OR 1=1 – – ' ". Here, in this query the mark (‘) tells the SQL parser that the user name string is finished and " OR 1=1 " statement is appended to the statement which always results in true. The (– –) is comment mark in the SQL which tells the parser that the statement is finished and the password will not be checked. So, the result of the whole query will return true always.

Many techniques have been proposed for controlling SQL injection. Major problems with these techniques are either high code modifications or it takes large extra time overhead. Cryptographic support is another important aspect of database security. Database encryption mechanism could provide database security. Illegal users cannot access the database without the proper key to decrypt it [3] [4] [5] [6].

Blowfish: Blowfish [7] is a 64-bit symmetric block cipher with a variable length key. The algorithm operates with two parts: a key expansion part and a data encryption part. The role of key expansion part is to convert a key of maximum 448 bits into several sub key arrays totalling 4168 bytes. The data encryption occurs via a 16-round Feistel network. Each round consists of a key dependent permutation, a key and data-dependent substitution. All operations are EX-ORs and additions on 32-bit words.

RSA: One of the best-known asymmetric-key cryptography processes is the RSA, named after its originators Rivest, Shamir, and Adleman [8] [9] [10]. RSA is easy to understand and relatively easy to implement. It is widely used in many cryptographic systems. RSA gets its security from factoring large prime numbers. The RSA public and private keys are derived from two randomly selected large prime numbers.

This paper proposes a new and better technique for query encryption using a two level dual encryption. I. Balasundaram et al. (2011) [11] proposed an encryption method for query encryption, which provides an authentication and access control for web applications and prevent from unauthorized access by using malicious query. We extend this approach and propose an alternate method. In our proposed method, two stage of encryption is applied:

1) Encrypt the username and password by blowfish encryption algorithm.
2) Encrypt the login query result by using asymmetric key encryption algorithm.

Tele:
E-mail addresses: aakashahu@gmail.com

The proposed method is secure and it needs a low computational cost.

Related work

Many of the existing techniques can detect and prevent a subset of the vulnerabilities that lead to SQL Injection Attacks. In this section, we will highlight the most relevant techniques-

M. Anand Kumar et al [12] - gave a comparison study which presented the performance evaluation of the two commonly used encryption algorithms, namely Blowfish and AES (Rijndael). Series of results based on the experimental procedures such as encoding techniques, packet size, data types and keys were found. Throughput was found to be high for Blowfish when compared to that of AES.As throughput is increased power consumption of decryption technique decreases. Blowfish encryption also had better performance than AES for text and document data files. Overall AES can be used in circumstances where high security is needed.

Gang Chen Gang Chen et al. (2006) - [13] proposed a Database Encryption Scheme. This was adopted to improve or enhance the way data is shared inside a database as well as preserving data privacy. It combines public key encryption with conventional encryption. User encrypts private data with the help of a conventional encrypted algorithm. If user has to see the encrypt data ,private key is decrypted first with the pass phrase, the working can then be decrypted to access the key with the help of this private key.

Islam - [14] proposed a framework to make E-Government Procurement Secure. This framework helped data protection in which encryption based Private Information Retrieval is used along with compression. It allows data storing, processing and retrieving in a secure way.

SAFELI [15] - proposes Static Analysis Framework which can detect the various vulnerabilities of SQL injection. The main aim is to identify the type of SQL injection attacks during compile time. This framework uses a Hybrid-Constraint Solver, implementing an efficient string analysis tool which deals with Boolean, integer and string variables.

AMNESIA (Analysis and Monitoring for Neutralizing SQL-Injection attacks) - In [16], Junjin proposes AMNESIA approach. The authors check the query during runtime and then it is declared as either valid or malicious. Various steps are involved during query checking. First a "hotspot" is identified. These are application codes which issues SQL query to database. Second NDFA (Non-Deterministic Finite Automata) is created which checks for a valid query.

Ali et al [2] - Proposed a scheme which adopts the hash value approach. This approach improves the user authentication mechanism. When the user account is created for the first time, the hash values for username and password are created and are stored in the User account table.

MeiJunjin [17] has used static, dynamic and automatic testing method for the detection of SQL injection vulnerabilities. This approach traces user queries to a vulnerable location. Although these techniques are effective, they cannot capture more general forms of SQLIAs.

Ezumalai et al (R. Ezumalai 2009) - [18] used a signature based approach for the protection of SQL injection. To detect security issues three modules were used: A monitoring module which takes the input from web application, an Analysis module which finds out the 'hotspots' from web application. This module uses Hirschberg algorithm (Hirschberg 1975) which works on 'divide and conquer' rule. It stores all the keywords in the Specifications module.

Parse Tree Validation Approach - Buehrer et al. [19] adopted the parse tree framework. Parse tree of a particular statement at runtime is compared with its original statement. They stopped the execution of statement unless there is a match. This method was first tested on a student Web application using SQL Guard. Although there were a few drawbacks to this approach: additional overhead computation and listing of input (black or white).

Pan (2008) [20] proposed an approach which was based on criteria access control. This approach was to deal with multilevel database security. Various authorization rules are transformed to security criteria, security criterion expressions, and security criterion subsets which serve as locks and keys. This approach is easier as only one mechanism is used and also reduces the cost of storage as only one row and one column is added to the table.

Proposed model

We propose a model for preventing SQL injection attacks by combining two well-known encryption techniques Blowfish (Symmetric key encryption) and RSA (Asymmetric key encryption) at different levels of the proposed model. In the proposed scheme, access to the database will be provided only by the server to all the authenticated users. If a new user wants to access the database he will have to register himself with the server.

During the registration process, we require every new user to provide username and password. Along with its regular process of checking the availability of user name, on successful completion, the server generates user key which is hexadecimal value of the password and stores it in the user table along with user name and password as shown in the Table 1.

Table I User table

Username	Password	Key
Abhishek	Aloha	$User\_key_{abhishek}$
Shubham	Mindbogller	$User\_key_{shubham}$
Usere354	Appleshare	$User\_key_{user354}$

The information of the registered users is stored in the server database, in the user table with other user details like user name, user password and the key. Once the user is registered he can access the database by requesting a log-in to the server and the server responds with the access control once the request is validated.

There are two stages in the process:

1) Access Request Process
2) Access Grant Process

Access Request process:

Access is provided to the database by the server after verifying user's authentication. Every new user is required to register with the database first. Existing users can directly access the database by providing their username and password. Following are the steps:

Registration

1. A new user has to register himself/herself with the database server.
2. For registration a user has to provide valid username and password of minimum 4 characters.
3. User also has to fill in a randomly generated CAPTCHA to ensure that it's a human attempt or a computer generated attempt, to prevent hackers creating malicious accounts.

4. Hexadecimal value of the password provided by the user is stored in the user authentication table, which will act as a key for blowfish decryption.

Login

1. User enters username and password.
2. Password is converted into hexadecimal.
3. Username and password is encrypted using the hexadecimal value of the password as a key, by blowfish encryption technique.
4. An SQL query is the generated for the user request with username, password and encrypted username and password.
5. SQL query is then encrypted with RSA encryption using a public key for further security.
6. Encrypted query is then sent over to the server.

Since a two level encryption technique is used in this scheme the level of security is very high. The server end has to decrypt and verify user's authentication and provide further access to the database.

Access Grant process:

Server provides access to the requested user only after the verification is complete. Following are the steps included:

1. Server receives encrypted data.
2. Data is decrypted to get SQL query using a private server key.
3. From the decrypted SQL query the username and password are used to fetch the key (hexadecimal value of password) stored in the user authentication table.
4. Key is used to decrypt the encrypted username and password in the query.
5. If the decrypted user name and password matches any of the authorized users in the user table stored in the server then access is granted to the user or else the query is rejected.

Blowfish algorithm will only be able to decrypt successfully the data if the hexadecimal value of the password and the hexadecimal value stored in the user table match.

An example of SQL query generated and encrypted query figure 2 and figure 3 respectively.

*SQL_query=SELECT * FROM user_table WHERE username = 'user123' AND password='alohA' AND encrypted_username = 'key_user123' AND encrypted_password = 'key_alohA';*

Fig 1: Sample query generation in the proposed model using unique user key.

Encrypted_SQL_query= $Encrypted_{SQL\_query}$

Fig 2: Encrypted SQL Query sent to the server side.

Figure 1 shows a sample SQL query with user key and password, figure 2 shows an encrypted SQL query, which server receives.

Architecture/design

Figure 1 and 2 show the access request architecture and access grant architecture respectively.

Results and discussion

The reason for using Blowfish over Rijndael (AES) was because Blowfish is considered faster. As per the findings made by M. Anand Kumar et al [12], we observe that time taken by Blowfish to encrypt and decrypt the information is less than that what Rijndael takes for the same sized data. Two levels of encryption maintains ample security with low computation time, if we go for more levels of encryption then computation time will increase also increasing the complexity of the model, making it slower on devices with low memory space and low end processors. The following Table II is showing encryption and decryption time taken by Blowfish and Rijndael. Table II shows a comparative study between the times taken by both the encryption methods.

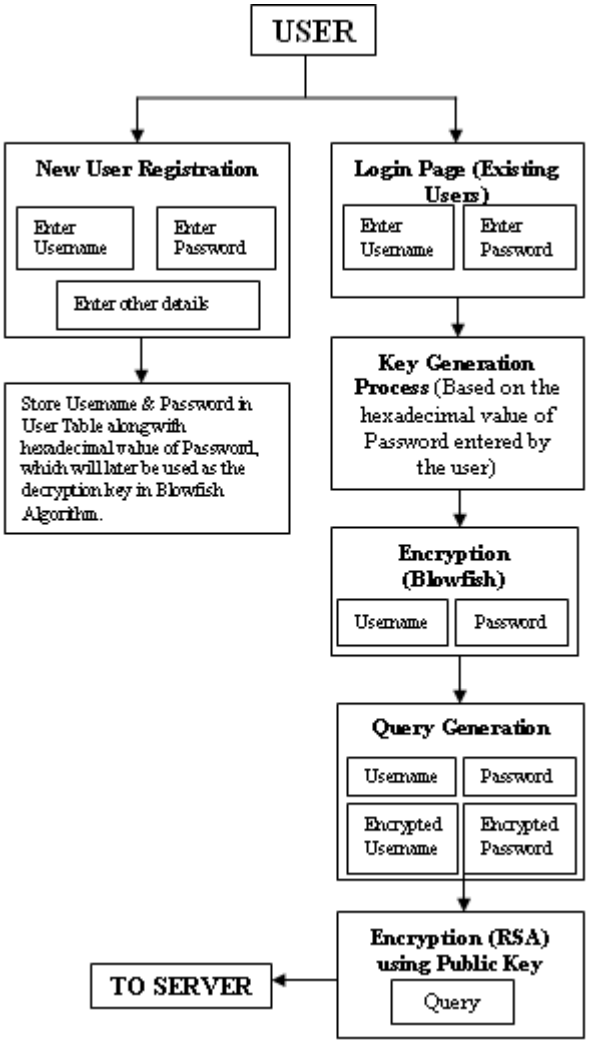

Fig 3: Access Request Process

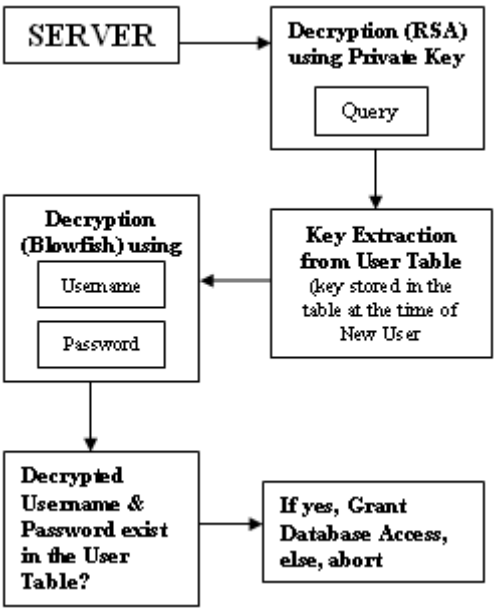

Fig 4: Access Grant Process

Table II Time taken for encryption

Input Size (kb)	Time (ms) Blowfish	Time (ms) Rijndael
74	72	87
500	121	134
1025	310	364

Encryption occurs at the client side and faster encryption makes the process faster and faster access is provided to the client. It does not create a problem even for the low end processors and devices with low memory space as the memory footprint of blowfish is just over 4KB. On server side information is decrypted. Time taken by the methods to decrypt is shown in Table III.

Table III Time taken for decryption

Input size (kb)	Time (ms) Blowfish	Time (ms) Rijndael
76	86	94
500	130	167
1025	300	301

It is clear from these results that in the cases, viz. Encryption & Decryption, the time taken by Blowfish Algorithm is observed to be lesser than the time taken by AES Algorithm (a well-established encryption/decryption technique). Moreover, it was also found that Blowfish Algorithm has a higher throughput when compared to AES Algorithm, i.e. the power consumption for decryption in Blowfish is lower than AES Algorithm. Hence, based on the performance comparison, Blowfish algorithm is considered as a better option for implementation. The table below, based on the findings made by Indrani Balasundaram et al [11], gives an analysis of the RSA Execution Time for different key sizes

TABLE IV Time taken by RSA

Key size (bit)	Execution time (ms)
128	19.015
512	36.085

Hence, Blowfish algorithm combined with RSA Algorithm, results in a highly secure and efficient method to prevent SQL injection attacks. On the other hand Blowfish encryption is an unpatented technique available at public domain making it easier to use and cheap so that any organization can use it.

Conclusion And Future Work

SQL injection attacks make the database vulnerable to unwanted access by non-reliable users that may not be good in terms of security. A secure database needs to restrict its user's activity according to the authentication of the user in order to work efficiently. In our scheme the proposed authentication process ensures user authenticity and efficient SQL query generation and in turn efficient database access and usage. The encryption on the client side and the decryption on the server side makes it hard for the non-reliable and malicious users to make an attempt to access the server database. Since the user name and SQL query both are encrypted and the query is only executed after the authenticity of the user is verified, hence making the process highly secured. In future, this project may be enhanced by improving its security and making it more secure. Other enhancement that can be done in the proposed project is to find a better encryption technique so that cost can be decreased and efficiency can be increased.

References

[1] W.G.J. Halfond, A. Orso, "AMNESIA: analysis and monitoring for Neutralizing SQL-injection attacks," 20th IEEE/ACM International Conference on Automated Software Engineering, Long Beach, CA, USA, 2005, pp. 174–183.

[2] Shaukat Ali, Azhar Rauf, Huma Javed "SQLIPA : An authentication mechanism Against SQL Injection".

[3] Henry Brown, (2003), "Considerations in implementing a Database Management System Encryption Security solution," A Research Report presented to The Department of Computer Science at the University of Cape Town.

[4] George l Davida, David L Wells, and John B Kam, (1981), "A Database Encryption System with Subkeys," ACM Transactions on Database Systems, 6:2(1981), pp. 312–328.

[5] Hakan Hacigumus, Bala Lyer, and Sharad Mehrotra, (2002), "Providing Database as a Service", in: Proc. of ICDE 2002, pp. 29–38.

[6] Jingmin He, Min Wang, (2001), "Cryptography and Relational Database Management System", IDEAS, pp. 273–284.

[7] Tingyuan Nie Teng Zhang, A study of DES and Blowfish encryption algorithm, Tencon IEEE Conference, 2009.

[8] Schneier, B., Applied Cryptography, 2nd edition, John Wiley & Sons Inc, 1996.

[9] ANSI Standard X9.31-1998, Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA).

[10] Rivest, R.L., Shamir, A., and Adleman, L.M., "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", Communications of the ACM, v.21, n.2, February1978, pp. 120-126.

[11] Indrani Balasundaram and E. Ramaraj (2011), "An Authentication Scheme for Preventing SQL Injection Attack Using Hybrid Encryption (PSQLIA-HBE)" European Journal of Scientific Research Vol.53 No.3 (2011), pp.359-368.

[12] M. Anand Kumar and Dr. S. Karthikeyan (2012), "Investigating the Efficiency of Blowfish and Regindael (AES) Algorithms". I. J. Computer Network and Information Security, 2012, 2, 22-28 Published Online March 2012 in MECS (http://www.mecs-press.org/) DOI: 10.5815/ijcnis.2012.02.04

[13] Gang Chen; Ke Chen; Jinxiang Dong, (2006), "A Database Encryption Scheme for Enhanced Security and Easy Sharing; Computer Supported Cooperative Work in Design". CSCWD '06, ppt 1 – 6.

[14] Islam, M.S.; Dey, S.; Kundu, G.; Hoque, A.S.M, (2008), "A Solution to the Security Issues of an E-Government Procurement System," Electrical and Computer Engineering, ICECE 2008. International Conference on; Publication Year: 2008, pp 659 – 664.

[15] X. Fu, X. Lu, B. Peltsverger, S. Chen, K. Qian, and L. Tao. A Static Analysis Framework for Detecting SQL Injection Vulnerabilities, COMPSAC 2007, pp.87-96, 24-27 July 2007.

[16] Halfond, W. G. J. and A. Orso (2005). AMNESIA: analysis and monitoring for Neutralizing SQL-injection attacks. . ASE'05. Long Beach, California, USA.

[17] Mei Junji, An approach for SQL injection vulnerability detection. Sixth International Conference on Information Technology, (2009): New Generations: pp. 1411-1414.

[18] R. Ezumalai, G. A. (2009). Combinatorial Approach for Preventing SQL Injection Attacks.2009 IEEE International Advance Computing Conference (IACC 2009). Patiala, India: pp.1212-1217.

[19] G. Buehrer, B.W. Weide, P.A.G. Sivilotti, Using Parse Tree Validation to Prevent SQL Injection Attacks, in: 5th International Workshop on Software Engineering and Middleware, Lisbon, Portugal, 2005, pp. 106–113.

[20] Pan L. (2008), "Using Criterion-based access control for multilevel database security," Electronic Commerce and Security, 2008 International Symposium, ppt 518 – 522.