International Journal of Computer Science and Engineering (IJCSE)
ISSN(P): 2278-9960; ISSN(E): 2278-9979
Vol. 3, Issue 3, May 2014, 141-148
© IASET

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

IMPROVED AUTHENTICATION TECHNIQUE TO PROTECT WEB APPLICATIONS

SRINADH SWAMY[1], PAVAN KUMAR[2] & VASU DEV[3]

[1]Assistant Professor, Department of Computer Science & Engineering, Chirala Engineering College, Andhra Pradesh, India

[2,3]Under Graduate, Department of Computer Science & Engineering, Chirala Engineering College, Andhra Pradesh, India

ABSTRACT

In our daily life, web applications have become an integral part. The major challenge of security in web applications is SQL injection attack which is top ten attacks according to Open Web Application Security Project. SQL injection attacks mainly focuses databases that are accessible through a web front end and take advantage of weak points in the input validation logic of web components. In the last few months, vulnerabilities in the application level have been exploited with serious consequences by hackers.

But there are no correct approaches able to give proper solution to this problem. SQL injection attacks can be easily prevented by applying more secure authentication schemes in login page itself. While many approaches have been proposed to address the vulnerabilities in the web application but which approach is more convenient and can also provide fast access to application without compromising security is also major concern. In this paper, we proposed an authentication mechanism for web applications which encrypts the user's data like username and password by using SHA-3.

KEYWORDS: Authentication, SQL Injection Attack, Vulnerability, Web Application Security

INTRODUCTION

SQL injection vulnerabilities have been marked as most serious threats for Web applications. SQL injection vulnerabilities for Web applications [1] may allow an attacker to get complete access to their underlying databases. Since these databases often contain sensitive information of the consumer or user, results the security violations like identity theft, loss of confidential information, and fraud. In some situations, attackers even use an SQL injection vulnerability to take control and corrupt the resources that hosts the Web application. Web applications which are vulnerable to SQL Injection Attacks (SQLIAs) [2] are spread a wide. In general, SQLIAs have successfully focused the high-profile victims such as Travelocity, www.FTD.com, and Guess Inc. SQL injection defines to a class of code-injection attacks in which data of the user is included in an SQL query in such that part of the user's input will treated as SQL code.

So, to eliminate vulnerabilities, developers addressed defensive coding practices but they are not sufficient. To prevent the SQLIAs, defensive coding must provides a solution but it is too difficult. In this situation, not only developers try to put some controls in their source code but also attackers are continue to bring some new ways to violate these controls. Hence it is difficult to keep developers up to date, according the last and the best defensive coding practices. Implementing the defensive coding is very difficult and require special skills.These problems motivate the need for a solution to the SQL injection problem.

BACKGROUND ON SQLIA

Definition of SQL Injection Attack

Generally, web applications now-a-days uses a multi-tier architecture, usually with three tiers: a presentation tier, a processing tier and a data tier. The presentation tier is the web interface uses HTTP, the application tier provides the software functionality based on the requirement, and the data tier maintains data structured and results to requests from the application tier [3]. Many companies developing databases which are SQL-based rely heavily on hardware to ensure the required performance [4]. SQL injection is a type of attack which the attacker adds Structured Query Language code to input box of a web form to gain access or make changes to data. SQL injection vulnerability allows an attacker to flow commands directly to web application's underlying database and destroy confidentiality or functionality.

Why SQL Injection is a Major Threat?

Injecting a code into the web application is the synonym of having access to the data stored in the database [8]. The data resided in the database may be confidential and of high value like bank secret details or list of transactions etc. An unauthorized access to this data resided in the database by a unauthorized user can threat their confidentiality, authority and integrity. As a result, the system could bear heavy loss in giving services to its users or it may face complete destruction. Sometimes such type of collapse of a system can threaten the existence of a company or a bank or an industry. If this type of situation happens against the information system of a hospital, the private information of the patients may be leaked out which could threaten their reputation or may be a case of defamation. Attackers may even use such type of attack to get confidential information that is related to the national security of a country. Hence, SQL Injection attacks are very dangerous in many cases depending on the platform where the attack is launched and it gets success in injecting rogue users to the target system.

SQLIA PROCESS

SQLIA is a hacking technique which the attacker or unauthorized person adds SQL statements through input fields or hidden parameters of the web application to access the resources. Lack of input validation and lack of security measures in web applications causes hacker to successful [6]. For the following example we will assume that a web application which receives a HTTP request from a client as input data and generates a SQL statement as output for the backend database server.

- Application presents a form to the attacker
- Attacker sends an attack in the form data
- Application forwards attack to the database in a SQL query
- Database runs query containing attack and sends encrypted results back to the application
- Application decrypts data as normal and send results to the user.

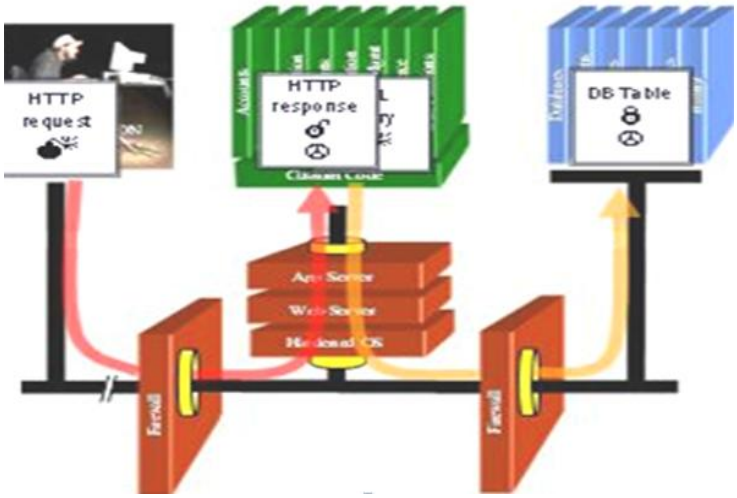


Figure 1: SQL Injection Attacking Process

CONSEQUENCES OF SQLIA

The results of SQLIA causes severe losses because from the database, a successful SQL injection can read sensitive data, it can modify database, or it can execute administrative operations on the database. The main tradeoffs of this vulnerability attacks on the database are on authorization, authentication, confidentiality and integrity [5].

Existing Encrypted Systems to Protect Web Applications

Over the last few years, both the industries and academic institutes doing research to prevent injection attacks. Following are existing encryption mechanisms proposed by researchers in the past.

- An algorithm which uses both Advance Encryption Standard (AES) and Rivest-Shamir-Adleman (RSA) cryptographic encryption algorithms to prevent SQL injection attacks. In this method, a secret key (unique) is fixed or assigned for every client or user. On the other hand, server uses private key and public key combination for RSA encryption. In this method, two level of encryption is applied:
  - To encrypt the user name and password provided by user, symmetric key encryption is used with the help of user's secret key.
  - To encrypt the query, this technique uses asymmetric key encryption by using server's public key.

The disadvantages of this approach:

  - Difficult to maintain every user secret key
  - No security mechanism at registration phase at client side.
  - Not efficient when large number of users are accessing the application simultaneously.

- To encrypt the SQL words, SQL rand uses randomization[7]. The main drawbacks of this approach are
  - Needs an additional proxy.
  - Computational overhead.

- Random4 approach is based on SQL rand [10] and randomization algorithm uses the salt concept for converting the input into a cipher text. The main flaws in this are:
  - Use of lookup table that is not an efficient way.
  - Can't handle second order SQL injection attack.

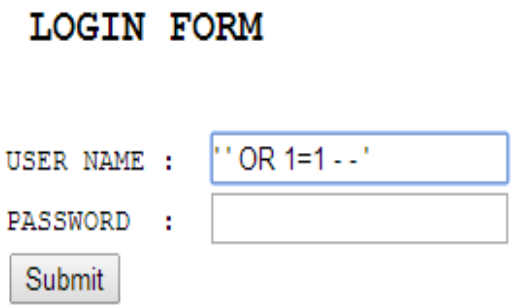
PROPOSED SYSTEM

This paper proposes an improved technique for preventing database from SQL injection attacks. In the proposed approach, we are encorporating two extra columns in User_details table. The first one is for the hashed value of user name and other column for the hashed value of password entered by user. At the time of registration, the hashed values for user name and password are calculated and stores it in the User_details table. Whenever user wants to login to the web application, the database checks the identity using user name and password and their hash values. These hashed values are calculated at runtime using SHA-3 and it is in stored procedure when user wants to login into the database. Generally, whenever user enters the user details like user name and password in the fields at login form, the query [9] at the back end server will be created. The hashed values are generated by using SHA-3 encryption algorithm.

The query given was static. So, it can easily be deceived using SQL injection and created every time during the authentication process. For example, if some one enters the details like user name as " 'OR 1 = 1 – – ", password as "Password". The query gives the result as true, as tautology is used in the code. Hence, the authentication process can easily be violated through SQL injection.

In the proposed technique, using SHA-3 first the hashed values of user name and password are calculated at runtime and checked with stored hash values in the User_details table. In case or the query if a hacker enters the SQL injection query still he/she cannot bypass the authentication process [14]. The advantage of this proposed technique is that the hackers do not know what algorithm used for hashing. So, it is not possible for the hacker to violate the authentication process through the ordinary SQL injection techniques [12]. The SQL injection attacks can only be done on codes which are entered through user login form but the hash values for the data are calculated at run time at backend before creating SELECT query to the underlying database [13, 15] So, the hacker or unauthorized user cannot calculate the hash values as it dynamic at runtime.

- **User Login Form**



- **SQL Query Containing Input Violation**

Query_result = "SELECT * FROM User_details WHERE name = "OR 1=1 – – " AND password= 'Password'

- **Proposed Technique Which Uses SHA-3**

Query_result = SELECT * FROM User_account WHERE Username_Hash_value="Username_Hash_value" AND Password_Hash_value = "Password_Hash_value" AND Username = '' OR 1=1 – – AND password = 'Password'

ABOUT SHA -3

SHA-3, belongs to the cryptographic primitive family(keccak) is a hash function in cryptography designed by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche, building upon Radio Gatún in 2012.

In this the message blocks are XORed which uses the sponge construction into the initial bits of the state. SHA-3contains state consists of 64-bit words of 5×5 array, 1600 bits total. "sponge construction",are used in SHA-3, In these the input can be "absorbed" into the hash state at a given rate, then an output hash is "squeezed" from it at the same rate. It take r bits of data for absorb, the XoRed data has the leading bits of the state, then the permutation block is applied. The first r bits to be squeeze state are produced as output, if any adiditional output is desired block permutation is applied.

Central to this is the "capacity" of the hash function, which is the $c=25w-r$ state bits that are not handled by input or output. This can be adjusted depend on security requirements, but the SHA-3 proposal sets a conservative $c=2n$, where n is the size of the output hash. Thus r, the number of message bits processed per block permutation, based on the output hash size. To ensure the message can be evenly divided into r-bit blocks, requires, padding.

To compute a hash, initialize the state to 0, pad the input, and break it into r-bit pieces. Absorb the input into the state; that is, for each piece, XOR it into the state and then apply the block permutation. After the completion of final block permutation, the leading n bits of the state are the desired hash. Since r is always greater than n, there is actually never a need for additional block permutations in the squeezing phase. However, arbitrary output length is useful in applications such as optimal asymmetric encryption padding. In this case, n is a security parameter rather than the output size.

CONCLUSIONS

In this paper, we proposed new authentication technique which protects the web applications at the user validation page by using SHA-3 algorithm. This technique is tested on sample data of different user details in user details table and it takes less time to produce hash values when compared with existing systems and protects the web applications from the vulnerabilities like SQL injection attack.

ACKNOWLEDGEMENTS

Our sincere thanks to our college principal **Dr. V. Ranga Rao** for the immense support you have provided us for publishing this paper. We hereby take the privilege to show our gratitude towards our college management – **Mr. Naga Malleswara Rao** (President), **Mr. K. Ravi kumar**(secretary) and **Mr. P.Sunil kumar** (Executive Member) for encouraging us We couldn't publish this paper without the support & encouragement of our Joint secretary and Head of the Department **Mr. Tella Ashok Kumar**.

Our sincere thanks to CECC's Department of library and information sciences for providing all necessary resources, accessing of e-journals like IEEE journals, Springer, Elsevier etc that leads to successful completion of this research.Finally, our thanks to our college faculty members, students and our parents for successful completion of my work.

REFERENCES

1. Shubham Srivastava, Rajeev Ranjan Kumar Tripathi "Attacks Due to SQL Injection & Their Prevention Method for Web-Application" (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (2), 2012, 3615-3618.
2. Shaukat Ali, Azhar Rauf, Huma Javed "SQLIPA: An authentication mechanism Against SQL Injection"
3. Bogdan Carstoiu, Dorin Carstoiu, "Zatara, the Plug-in-able Eventually Consistent Distributed Database", AISS, Vol. 2, No. 3, pp. 56 ~ 67, 2010.
4. Dorin Carstoiu, Elena Lepadatu, Mihai Gaspar, "Hbase - non SQL Database, Performances Evaluation", IJACT, Vol. 2, No. 5, pp. 42 ~ 52, 2010.
5. P. Grazie., PhD SQL Prevent thesis. University of British Columbia (UBC) Vancouver, Canada. 2008.
6. C. Gould, Z. Su, and P. Devanbu. JDBC Checker: A Static Analysis Tool for SQL/JDBC Applications. Proceedings of the 26th International Conference on Software Engineering (ICSE 04) Form l Demos, pp 697–698, 2004.
7. Stephen W. Boyd, Angelos D. Keromytis "SQL and: Preventing SQL injection Attacks
8. Rahul Johari, Pankaj Sharma, A Survey On Web Application Vulnerabilities(SQLIA, XSS) Exploitation and Security Engine for SQL Injection, 2012 International Conference on Communication Systems and Network Technologies
9. A Kiezun, P. J. Guo., K. Jayaraman and M. D. Ernst (2009). Automatic Creation of SQL Injection and Cross-Site Scripting Attacks. International Conference on Software Engineering. Vancouver, Canada, IEEE: pp. 199-209.
10. Boyd, S. W and A. D. Keromytis (2004). SQLrand: Preventing SQL Injection Attacks. 2nd Applied Cryptography and Network Security (ACNS) Conference Yellow Mountain, Chine: pp. 292-302
11. K. Wei, M. Muthuprasanna. and S. Kothari (2006). Preventing SQL Injection Attacks in Stored Procedures. Australian Software Engineering Conference (ASWEC'06) Australia, IEEE: pp. 191 – 198
12. 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.
13. Indrani Balasundaram, Dr.E.Ramaraj "An Approach to Detection of SQL Injection Attacks in Database Using Web Services" (IJCSNS, VOL. 11 No. 1, January 2011).
14. Rahul Shrivastava, Joy Bhattacharyji, Roopali Soni "SQL INJECTION ATTACKS IN DATABASE USING WEB SERVICE: DETECTION AND PREVENTION – REVIEW" Asian Journal Of Computer Science And Information Technology 2: 6 (2012) 162 – 165
15. Shubham Srivastava, Rajeev Ranjan Kumar Tripathi "Attacks Due to SQL Injection & Their Prevention Method for Web-Application" (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (2), 2012, 3615-3618.

AUTHOR'S DETAILS

Mr. Srinadh Swamy Majeti working as Assistant Professor in Chirala Engineering College, Chirala (CECC) in the department of Computer Science & Engineering. His experience in teaching is 3 years. He published two international journals in IJCA in the research area of Image processing & Data mining. As Assistant professor, he taught Computer Organization, Design patterns, web technologies, Computer Architecture, Software engineering, Software project management.

Mr. Pavan Kumar Annam studying 3rd year B.Tech course in Computer Science & Engineering stream in Chirala engineering College, Chirala. His areas of interest are Security in database applications.

Mr. Vasu Dev studying 3rd year B.Tech course in Computer Science & Engineering stream in Chirala engineering College, Chirala. His areas of interest are Security in database applications and networking.