SQL Injection Attacks: Vulnerabilities, Detection and Prevention Techniques

Sonakshi
M.Tech. Student, Department of Computer Science & Applications, Kurukshetra University, Kurukshetra-136119
Email: sonakshik15@gmail.com
Rakesh Kumar
Professor, Department of Computer Science & Applications, Kurukshetra University, Kurukshetra-136119
Email: rakeshkumar@kuk.ac.in
Girdhar Gopal Bansal
Assistant Professor, Department of Computer Science & Applications, Kurukshetra University, Kurukshetra-136119
Email: girdhar.gopal@kuk.ac.in

--ABSTRACT--
The last decade has observed an exponential growth of World Wide Web. There are various vulnerabilities in the web based applications which have been generally exploited by the various attackers. One of the most common web application attacks is SQL injection attack. In this paper, a critical analysis has been performed with regard to, how the SQL injection attack is carried out, the vulnerabilities that are exploited by attackers and the detection and prevention techniques that have been used in past.

Keywords: SQL injection, SQL injection detection, SQL injection prevention, Vulnerabilities, Web application attacks, Web applications security

1. Introduction

Web applications are known to be as backbone of today‘s business environment. The rapid development of web technology has made wide use of web applications which are used in many fields, also the dependency on internet or web applications for most of the activities such as e-commerce, online banking, blogs, forums, social networking, online bill payments, web mail, online gaming etc. has increased in our daily life. So, it can be said that these activities are the key component of today's internet infrastructure. Internet has made one's lives much simpler but the privacy of sensitive data in backend databases is of big concern. As the web applications are accessible to everyone, so these are exposed to various forms of security threats like Denial of Service (DoS), Structured Query Language (SQL) injection, Cross-site Scripting (XSS), remote file inclusion attacks etc. (Kar & Panigrahi, 2012). Out of various attacks, SQL injection has been the most frequent among all. In fact, it has been on the top among top 10 security threats listed by OWASP (Anon., 2010). It has become more serious after 2008. Since attackers have started using sophisticated botnets (Maciejak & Lovet, 2009) to automatically find vulnerable websites from search engines and perform mass SQL injection attacks on them (Kar & Panigrahi, 2012). SQL injection attacks (SQLIAs) are considered as the most widespread attacks on web servers because SQL databases are attractive targets as they contain most important assets such as usernames, passwords, personal data and financial information etc. This way, a security breach exposing the database to outside world can be fatal to the business enterprise repudiation and liability. The attack can also be easily carried out with web browsers using port 80 which is frequently left open by firewalls (Yeole & Meshram, 2011). The increasing demand for web applications, also attract hackers that want to benefit from their weaknesses (Elia et al., 2010). According to the survey conducted in 2007, it was estimated that about 70% of websites are at a risk of being hacked (Anon., 2007), it is due to the fact that these applications can be accessed from anywhere in the world which makes them even more interesting for attackers (Elia et al., 2010). SQL injection is very profitable for attackers and there is black market which is most prevalent these days. The trades include all sorts of digitally stolen goods like credit card numbers and bank accounts numbers etc. (Anon., 2008). Many times attackers can even be escaped without being prosecuted due to application configurations or bad server, lack of logs and lack of security regulations from corporations and governments (Anon., 2008). It is therefore, a dire need to prevent such types of attacks and SQLIA prevention has become one of the most active topics in research in academia and industry both. There has been a significant progress in this field and a number of methods have also been proposed to counter SQLIAs but none of them is able to guarantee an absolute level of security in web applications due to diversity and scope of SQLIA's (Namdev et al., 2012).

This paper is organized in five sections as follows. Section 2 presents the overview of SQL injections. Various vulnerabilities of SQLIA are discussed in section 3. Related work of SQLIA detection and prevention techniques is provided in section 4. Finally, conclusion is provided in section 5.

2. SQL Injection

SQL injection is the attack which is performed intentionally by an attacker either to gain unauthorized access to a database or to retrieve information directly from the database. In this attack, the attacker inserts a portion of SQL statement via not sanitized user input parameters into the original query and passes them to database server(Antunes & Vieira, 2012). The input is accepted from users by web applications and then it is incorporated into dynamically generated code (Son et al., 2013). For example, a web application asks the user to fill a form like submitting a username and password for authentication. Using an SQL query, the attacker can add, modify or delete data in a database with proper authorization. The SQL queries are dynamically constructed from user input, e.g., if the input from user includes SQL keywords like tautology statement, the dynamically generated SQL query will change the intended function of the SQL query in the application. An attacker can enter the following input through user interface:

Username: ‘OR ‘1=1-- Password: ‘OR ‘1=1--

It would generate the following query:

SELECT user info FROM users WHERE id=’1’ OR ‘1=1--’ AND password =’1’ OR ‘1=1--’;

The given input makes the WHERE clause in the SQL statement which always returns a true condition (i.e. the tautology statement). The database will return all the user information in the table. Therefore, the malicious user has been authenticated without a valid login id and password (MeiJunjin, 2009). If the application is vulnerable to SQLIA, then this malicious input will change the intended structure of SQL query which is then executed on back end database server. The major cause is the lack of input validation in web applications which causes hacker to be successful. Basically, the SQLIA process can be explained in three phases (Nithya et al., 2013):

a) An attacker sends the malicious HTTP request from a client to the web application as input.
b) Generates a SQL statement
c) Submits the SQL statements to the back end database server.

3. Vulnerabilities in Web Applications

Vulnerabilities are weakness or flaws in the system which an attacker can take advantage by exploiting it to gain unauthorized access (Buja et al., 2014). Web application vulnerabilities are the most serious threat for web application. The web programming language has vulnerabilities due to few syntax errors. The poor programming or coding practice leads to vulnerabilities such as improper sanitizations of inputs, type checking, over privilege accounts and detailed error messages (Sharma & S.C.Jain, 2014). The attacker can plan a specific attack according to the specific kind of vulnerability present in the application. The different kinds of vulnerabilities that may be present in an application are described as follows:

3.1 Insufficient input validation

Insufficient input validation will allow code to be executed with proper verification of its intention (Nithya et al., 2013). Attacker takes advantage of insufficient input validation and can utilize malicious code to conduct attacks (Kindy & Pathan, 2011). The intent of attackers is to bypass authentication.

3.2 Privileged account

A privileged account has a degree of freedom to do what normal accounts cannot do. Its action may also exempt from auditing and validation.

3.3 Error messages

Its basic idea is having knowledge of database through informative error messages. The intention of attacker is to upload files or executing remote commands to gain knowledge of database structure through the error messages generated (Sharma & S.C.Jain, 2014).

3.4 Injection with UNION query

In this, an attacker extracts data from a table which is different from the one that was intended in web application by the developer. A malicious user exploits a vulnerable parameter to change the data set returned for a given query (Sajjadi & Pour, 2013).

3.5 Uncontrolled variable size:

If variables allow storage of data to be larger than expected which consequently allows attackers to enter modified SQL statements due to which attack becomes successful, such as buffer overflow**.**

3.6 Dynamic SQL

The scripts are used to combine user input such as username and password and construct the WHERE clause of the query statement. Here, the main problem is that query building components can also accept SQL keywords. By exploiting this, the attacker can make a totally different query than what was intended.

4. Related Work

An extensive literature survey is carried out that covers the period of 2000-2015. Although researchers have proposed various methods to address the SQL injection problem and there are many solutions proposed in the literature. These are discussed below:

4.1 Detection Techniques

An SQL-Injection free (SQL-IF) secure algorithm was proposed by Kanchana et al.(Natarajan & Subramani, 2012) in their paper in 2012, to detect and prevent SQL injection attacks. The detection method consists of a Generic detection method known as check vulnerability which checks special characters like keywords & Boolean characters in the input fields. If there is any mismatch found in parametric values, it is directly sent to vulnerability data collector & HTTP request to warnings is reset. This algorithm detects SQL injection attacks and that can be applied to any web based applications wherever the user and database interacts. The proposed technique is generic and is language independent. Its advantage is that there is no need for further code modification and provides optimized runtime analysis.

A method that detects and prevents SQL injection attacks by checking whether user inputs cause any changes in query's intended result, is proposed by NTAGWABIRA Lambert et al.(Lambert & Lin, 2010) in the year 2010. The Query tokenization is implemented by using a method called QueryParser. This method firstly creates tokens of original query and injected query. Then, the tokens are made for every string which detects the spaces, single quotes or double dashes which are used for commenting the text. After completion of tokenization, every token is considered as element of the array. The lengths of obtained arrays are compared and if lengths are different, injection is detected otherwise it is considered that there is no injection in the original query i.e. equality in lengths leads to absence of injection.

Yeole et al. (Yeole & Meshram, 2011) has analyzed various techniques such as mutation testing, tokenization, multi-layer defense mechanism, service based approach and syntactic and semantic analysis for detection of SQL injection. The author has observed that most of SQL injection detection tools are anomaly base or signature base. The major problem with these techniques is that they compare input with saved patterns but the hackers are constantly looking for new vulnerabilities to attack, even the previous validation functions are also exploited.

An injection detection method by removing SQL query attribute values is presented by Jeom-Goo Kim (Kim, 2011). The method utilizes both static and dynamic analysis by comparing and analyzing the removed SQL query attribute value. The detection method can also delete the attribute values of static SQL query in web application and those generated at runtime will be deleted. The attribute value removing function '*f*' will remove attribute values in fixed SQL query and generated dynamic query from user input. If an invalid SQL query is applied by attacker to function *f*, a syntax error will be occurred. To determine if the SQL query is normal query or an attack, after the attribute value is removed, XOR calculation is applied onto removed attribute value of the fixed SQL query & removed attribute value of dynamic SQL query.

Sruthy Manmadhan et al. (Manmadhan & T, 2012) have proposed a different method for preventing SQL injection attacks in JSP web applications. The proposed technique is based on *dynamic query structure validation*, which is used for mining programmer-intended query structures at each SQL query location. The paper makes use of semantic comparison to detect SQL injection. The statements are parsed and syntax tree structure is compared. It then makes the SQL injection by generating a benign query from the final SQL query and then the inputs from users, the semantics of safe query and SQL query are then compared. Finally, it has focused on stored procedure attacks and how to get query structure before actual execution which is not easier task.

4.2 Prevention techniques

An authentication mechanism to prevent SQL injection attack using Advance Encryption Standard (AES) is proposed by Indrani Balasundaram et al.(Indrani & E., 2011) in 2011. It is a more secure authentication schemes in login phase. In this method, encrypted username and password are used to improve the authentication process with minimum overhead. The method has proposed three phases, in the first phase i.e. *registration phase*, server sends a registration conformation. In the second phase i.e. *login phase*, user can access the database from server. The username and password is encrypted by using Advance Encryption Standard (AES) algorithm by applying user secrete key and the SQL query is generated using encrypted username and password. Then the query will be sent to server. In the third phase i.e. *verification phase*, server gets the login query and verifies the corresponding users secrete key. If they matches then the decrypted username and password is checked from user account table. If it matches, then user is accepted otherwise rejected.

Mayank Namdev et al. (Namdev et al., 2012) have given a model to block SQL injections which is based on verification information. They have combined two approaches and created a new hybrid algorithm which works by applying the hash code with encryption for more security. In this approach, two extra columns are needed, one for storing the hash values of username and another one for storing the hash values of password. When the account of users is created for the first time, the hash values are calculated and stored in user table. The hash values are calculated at runtime using stored procedure when user logs into the database. The values which are calculated at runtime are matched with stored hash values in database table. Thus, if user tries to inject to the query, the proposed method will automatically detect the injections as malicious content & rejects the values. Therefore, it cannot bypass the authentication process. Its advantage is that hackers do not know about the hash value concept.

A similar approach to protect web applications against SQL Injection attacks is proposed by Neha Mishra et al.(Mishra & Gond, 2013) in their paper, which has discussed some predefined methods and also proposed an integrated approach of encryption method with secure hashing. The technique works by creating two columns by DBA for storing username and password, in the same way as done in previous approach by (Namdev et al., 2012). The secure hash values are generated at runtime using stored procedure if a user wants to login to the database. These values are stored in login table when the user's account is created first time. If a user wants to login to the database, his/her identity is verified via username, password and secure hash values. The combined approach of secure hashing on encryption is provided. The purpose of encrypting data is that it helps to change the data into a form that is not readable (Priyanka & Bohat, 2013).

Srivastava et al. (Srivastava & Tripathi, 2012) have proposed a new technique for prevention of SQL Injection attacks which is more secure mechanism that limits the access to those who are authorized to access the application at login phase. For this, the users are authenticated for limiting access. Then, the final hash code value is calculated at runtime. Hash function is a subroutine that maps a large data set to a small data set. This technique uses a hash function and salt, which consists of random bits creating one of the inputs, to a one way function.

Mihir Gandhi et al. have discussed in their paper about Advance SQL injection (ASQLIA) (Gandhi & Baria, 2013). They had identified the types of attacks and prevention measures are suggested accordingly. Some new features like web crawling, web services are also added to it, which emphasize more on security of web applications. The architecture of proposed technique i.e. Preventing SQL Injection Attacks in Web Application (PSIAW) consists of three components: user login interface, SQL query component & User Account Table. The user login interface is just the user entry form. The main component of PSIAW is SQL Query Component, where hash value of username and password is calculated. These values are then combined with username and password using AND operator. Every time, the username and password is entered, their hash values are calculated. The query formed is then sent to the database. If there is SQL injection attack, string is entered for logging into the database, its hash values do not match with the hash values stored in the table and hence, the attacker cannot access the database.

An application may also be vulnerable to SQL injection including stored procedure since the stored procedures are stored at server side, these are available easily to all clients. A technique proposed by Deevi Radha Rani et al. to prevent SQL injection is stored procedure by encrypting user input fields (Rani et al., 2012). Whenever a user submits the registration form, a unique secret key will be generated corresponding to entered username and password. A stored procedure will be called by passing the three parameters namely, username, password and secret key. Then username and password are encrypted with the secret key generated. The encrypted values along with username are stored into user table. The basis of such a technique is that effect of malicious code can be avoided by using encryption algorithm.

5. Conclusion and future work

SQL injection is a technique that an attacker employs to attack web based applications. SQL injection statements are used to retrieve information from back end database and can manipulate the data also. The deficiency of security policy in developing web application is an extra advantage to attackers and nowadays it has become a profitable business as an underground economy such as trades of bank accounts, credit card numbers and personal information. In this paper, a survey of the techniques involved to detect and prevent SQL injections in past is presented. To perform this, different kinds of vulnerabilities of SQL injection has been identified. Then, the SQL injection detection and prevention techniques proposed by various researchers, has been investigated. Different authors have presented their work at different levels of detail, extracting uniform data from such a diverse literature was very tedious task. The future work will be to add validations & encryption, so that unauthorized user won't be allowed to login and also detection of malicious activity becomes possible by incorporating validation at login phase. Encryption of confidential data lying in database won't allow unauthorized user to read confidential data even if it gets access by employing any kind of malicious technique or SQL injection. In this way, the web-based applications can be protected from such an attack.

References

Agarwal, U., Rana, K.S. & Saxena, M., 2015. A survey of SQL injection attacks. *International Journal of Advanced Research in Computer Science and Software engineering*, 5(3), pp.286-89.

Anjugam, S. & Murugan, A., 2014. Efficient Method for Preventing SQL Injection Attacks on Web Applications Using Encryption and Tokenization. *International Journal of Advanced Research in Computer Science and Software Engineering*, 4(4), pp.173-77.

Anon., 2007. *Acunetix Ltd*. [Online] Available at: http://www.acunetix.com/news/security-audit-results.htm.

Anon., 2008. *Symantec Report on the Undergroung Economy*. Symantec.

Anon., 2010. *OWASP, "Top 10 Security threats 2010"*. [Online] Available at: http://www.owasp.org/index.php/Top_10_2010-Main.

Antunes, N. & Vieira, M., 2012. Defending against Web Appliaction Vulnerabilities. 45(2), pp.66-72.

Appelt, D., Nguyen, C.D. & Briand, L., 2015. Behind an Application Firewall, Are we safe from SQL injection attacks? *IEEE*.

Bhanderi, A. & Rawal, N., 2015. A Review n detection Mechanism for SQL Injection Attacks. *International Journal of Innovative Research in Science, Engineering and Technology*, 4(12), pp.12446-52.

Buja, G., Jatil, D.K.B.A., Ali, D.F.B.H.M. & Rahman, T.F.A., 2014. Detection Model for SQL injection attack. *IEEE Symposium on Computer Applications & Industrial Electronics(ISCAIE)*, pp.60-64.

Elia, I.A., Fonseca, J. & Vieira, M., 2010. Comparing SQL Injection Detection Tools using Attack Injection:An Experimental Study. *IEEE 21st International Symposium on Software Reliablity Engineering*, pp.289-98.

Firdos, M. & Sheikh, A., 2011. Secure Query Processing by Blocking SQL Injection Attack. *International Journal of Research in management*, 3(1).

Gandhi, M. & Baria, J., 2013. SQL INJECTION Attacks in Web Application. *International Journal of Soft Computing and Engineering*, 2(6), pp.189-91.

Gaur, R. & Bhushan, R., 2014. Protecting SQL Injection Attack on Web Applications using AES and Stored Procedure. *International Journal Of Advanced Research in Computer Science and Software Engineering*, 4(5), pp.550-54.

Halfond, W.G.J. & Orso, A., 2005. Combining static Analysis and Runtime monitoring to counter SQL Injection Attacks. *ACM*, pp.1-7.

Halfond, W.G., Viegas, J. & Orso, A., 2016. A classification of SQL injection Attacks and countermeasures. In *International Symposium on Secure Software Engineering*., 2016.

Indrani, B. & E., R., 2011. An Authentication Mechanism to prevent SQL injection Attacks. *International Journal of Computer Applications*, 19(1), pp.30-33.

Johari, R. & Sharma, P., 2012. A Survey on Web application Vulnerabilities (SQLIA,XSS) Exploitation and Security Engine for SQL Injection. In *IEEE International Conference on Communication Systems and Network Technologies*., 2012.

Kar, D. & Panigrahi, S., 2012. Prevention of SQL Injection Attack Using Query Transformation and Hashing. *3rd IEEE International Advance Computing Conference*, pp.1317-23.

Kar, D. & Panigrahi, S., 2013. Prevention of SQL Injection Attack Using Query Transformation and Hashing. *IEEE International Advance Computing Conference(IACC)*, pp.1317-23.

Kar, D. & Panigrahi, S., 2013. Prevention of SQL Injection attack using query transformation and hashing. In *3rd IEEE International Advance Computing Conference(IACC)*., 2013.

kiani, M., Clark, A. & Mohay, G., 2008. Evaluation of Anomaly Based character distribution models in the detection of SQL Injection attacks. In *The third International Conference on Availabilty, Reliability and security IEEE Computer Society*., 2008.

Kim, J.-G., 2011. Injection Attack Detection using the Removal of SQL Query Attribute Values. *IEEE*.

Kim, J.-G., 2011. Injection Attack detection using the removal of SQL query Attribute values. *IEEE*.

Kindy, D.A. & Pathan, A.-S.K., 2011. A surevey on SQL injection : Vulnerabilities, Attacks and Prevention Techniques. In *IEEE 15th International Symposium*., 2011.

Kumar, P. & Pateriya, R.K., 2012. A Survey on SQL injection attacks, detection and prevention techniques. *IEEE ICCCNT*.

Lambert, N. & Lin, K.S., 2010. Use of Query Tokenization to detect and prevent SQL Injection Attacks. *IEEE*, pp.438-40.

Maciejak, D. & Lovet, G., 2009. Botent-Powered SQL injection Attacks:A deeper look within. In *Virus Bulletin Conference*., 2009.

Manmadhan, S. & T, M., 2012. A method of detecting SQL Injection attack to secure web applications. *International Journal of distributed and Parallel Systems(IJDPS)*, 3(6).

MeiJunjin, 2009. Anon vulnerability detection approach for SQL inject. *IEEE*, pp.1411-14.

Mishra, N. & Gond, S., 2013. Defenses to protect against SQL Injection Attacks. *International Journal of Advanced research in Computer and Communication Engineering*, 2(10), pp.3829-33.

Monticelli, F., 2008. *SQL Prevent*. PhD. SQLPrevent thesis. Vancouver, Canada: University of British Columbia (UBC).

Namdev, M., Hasan, F. & Shrivastav, G., 2012. Review of SQL Injection Attack and Proposed method for detection and Prevention of SQLIA. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(7), pp.24-28.

Namdev, M., Hasan, F. & Shrivastav, G., 2012. Review of SQL Injection Attack and Proposed Method for Detection and Prevention Techniques. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(7), pp.24-27.

Natarajan, K. & Subramani, S., 2012. Generation of SQL-injection free secure algorithm to detect and prevent SQL-injection attacks. *Elsevier*, pp.790-96.

Nithya, V., regan, R. & Vijayaraghavan, J., 2013. A survey on SQL injection attacks, their Detection and Prevention techniques. *International Journla of Engineering and Computer Science*, 2(4), pp.886-905.

Pietrazek, T. & Berghe, C.V., 2006. Defendinding against Injection attacks through context sensitive string evaluation. *Recent Advances in Intrusion detection*, 3858, pp.124-45.

Priyanka & Bohat, V.K., 2013. Detection of SQL Injection Attack and Various prevention Stategies. *International Journal of Engineering and Advanced Technology*, 2(4).

Rani, D.R. et al., 2012. Web Security by Preventing SQL Injection Using Encryption in Stored Procedures. *International Journal of Computer Science and Information Technologies*, 3(2), pp.3689-92.

Sajjadi, S.M.S. & Pour, B.T., 2013. Study of SQL Injection Attacks and Countermeasures. *International Journal of Computer and Communication Engineering*, 2(5), pp.539-42.

Sharma, C. & S.C.Jain, D., 2014. Analysis and Classification of SQL injection vulnerabilities and Attacks on Web Applications. In *IEEE International Conference on Advances in Engineering and Technology research(ICAETR)*. Kota, 2014.

Shehu, B. & Xhuvani, A., 2014. A Literature review and Comparative Analyses on SQL Injection: Vunerabilities, Attacks and their Prevention and Detection Techniques. *International Journal of Computer Science Issues(IJCSI)*, 11(4), pp.28-37.

Son, S., Mckinley, K.S. & Shmatikov, V., 2013. Diglossia: Detecting Code injection attacks with precision and efficiency. *ACM*, pp.1181-91.

Srivastava, M., 2014. Algorithm to Prevent Back end database against SQL Injection Attacks. *IEEE*, pp.754-57.

Srivastava, S. & Tripathi, R.R.K., 2012. Attacks due to SQL Injection & their Prevention Method for Web-Application. *International Journal of Computer Science and Information Technologies*, 3(2), pp.3615-18.

Tajpour, A., Ibrahim, S. & Sharifi, M., 2012. Web application Security by SQL Injection Detection Tools. *International Journal of Computer Science Issues(IJCSI)*, 9(2), pp.332-39.

Tajpour, A., Ibrahim, S. & Sharifi, M., 2012. Web Applications Security by SQL Injection Detection Tools. *International Journal of Computer Science Issues*, 9(2).

Tajpour, A., Suthaimi & Masrom, M., 2011. SQL injection detection and prevention techniques. *Inetrnational journal of Advancements in Computing Technology*, 3(7).

Thomas, S. & Xie, L.W.a.T., 2009. On Automated prepared statement generation to remove SQL injection vulnerabiliies. pp.589-98.

Wei, T. et al., 2010. Research on mock attack testing for SQL injection vulnerability in multi-defense level web applications. *IEEE*.

Yeole, A.S. & Meshram, B.B., 2011. Analysis of Different Technique for Detection of SQL Injection. In *zinternational Conference and workshop on Emerging Trends in Technology(ICWET 2011)*. Mumbai, 25-26 February 2011.