How to cite this paper:

Rua Mohamed Thiyab, Musab A. M. Ali, Farooq Basil, & Abdulqader. (2017). The impact of SQL injection attacks on the security of databases in Zulikha, J. & N. H. Zakaria (Eds.), Proceedings of the 6th International Conference of Computing & Informatics (pp 323-331). Sintok: School of Computing.

THE IMPACT OF SQL INJECTION ATTACKS ON THE SECURITY OF DATABASES

RUA MOHAMED THIYAB[1], MUSAB A. M. ALI[1,2], FAROOQ BASIL[3] ABDULQADER[3]

ruaalkarkhi1991@yahoo.com, drmusab@msu.edu.my, farouqshukri@siswa.ukm.edu.my
[1]Management & Science University, Malaysia,
[2]Associate Research Fellow,Center of Scientific Research and development Nawroz University-Kurdistan Region, Iraq,
[3]Universiti Kebangsaan Malaysia

ABSTRACT. SQL injection Attack (SQLIA) can be detected in many web applications that lack of input variable filtering. The problem of this study is the weak input filtration and validation of forms in dynamic web applications and using a single detection and prevention technique against SQL injection attacks. The aim of this study is to investigate the effect of poor input validation of SQL query to discriminate the parameters used for injection malicious SQL on the security of server database and to improve the filtration level of a user input from real one and a malicious one on dynamic web applications in e-commerce, and to proposes a technique called CombinedDetect based on two methods based on JavaScript and PHP coding to detect malicious SQL query and isolate it before sending to the server. The result of this study shows that many web developers neglect the high risks of SQL injection attacks on the security and confidentially of data stored in databases. The injection of malicious SQL parameters pass to the database in the server could damage the whole database or steal data. The method used in this study is based on JavaScript and PHP codes enable the dynamic web application to separate between normal data and malicious data, nevertheless of what user input is entered through input fields. The study recommended avoiding any weakness in SQL server by providing effective input validation to discriminate the malicious parameters used for injection SQL attack queries and using multiple detection methods for SQL injection.

Keywords: SQL Injection Attack (SQLIA), SQL Queries, Vulnerability, Dynamic Applications, Input Validation.

INTRODUCTION

The technology of Internet is a prevalent information infrastructure in today business and education world. Ignorant of the essential security and privacy issues is a critical matter for web developers. The spread of internet services is becoming a main source of information worldwide. It is found that one of the core threats to most web application nowadays is SQL Injection Attack (SQLIA). For example, Open Web Application Security Project (OWASP) has identified the primary key threats (top 10) for the security of web application found in 2013, where SQL injection attacks scored first[1]. Actually SQLIA is a familiar type of web application that cause real vulnerability to source of data such as database storing information of users in the server, SQLIA exists in applications that suffer from suitable input filtering for user's main variables. For evaluating this kind of vulnerability, the SQL attacker always attempts to insert a part of malicious SQL commands by using special variables and inserting them into the application. Thus, it causes the web application to sends these malicious commands to the target database in the server and start to execute them in a different purpose using legitimate query[2].

The issue of SQLIA is still exist and has become a giant online threat to many companies who become a victim of SQLIA vulnerability and cost them a lot of money. This mainly because of the high flexibility for programmer in SQL language[3]. Researcher agree that the main method for sending a malicious query is variables that might come inform of a GET request, a POST request, HTTP Cookies or HTTP Headers, mainly in Ajax XMLHttpRequest object that is a technique for building dynamic webpages using GET request, a POST request[3,4,5]. SQL injections is considered as one of the main web attack which include various mechanisms implemented by hackers so that they are able to steal data from target database and in particular e-commerce websites. If SQLIA happens against the target database for example banks, hospitals, online shops, and government websites, the private information can be highly vulnerable to the attackers[6]. Therefore, it is essential step during the developing of to restrict input fields for certain malicious codes or commands that can be used to exploit SQL injection vulnerability to the server and execute it which may delete all stored data in the database. It is concluded that in the absence of input purification the malicious query will be vulnerable to SQL injection attack.

TYPES OF SQL INJECTION ATTACKS

There various types of SQLIA have ascended during the past 10 years. To better identify SQLIAs, and well understand the current issues and developed techniques requires for knowing the main types of SQLIA. In this section the researcher presents the main types of SQLIA.

SQL injection attacks can be classified into 3 main categories (out-of-band, in-band, and inferential)[7]:

- In-band: The information is extracted from the identical channel that implemented for the SQL attack where it is the simplest method known in the world nowadays.
- Out-of-band: This category presents where an extracted information is delivered back to the attacker depending on a different channel such as an email.
- Inferential: This category is known as Blind injection, where an attacker does not depend on returning data from the server. However, attacker main goals are to reconstruct the data stored in the database by attempting to implement different attack and observe all the possible behavior from the server as well as the web application.

Today, SQL injection (SQLI) is recorded as one of the main and top 10 vulnerabilities to web application between 2007 -2010 which is certified by the Open Web Application Security Project. Where, in 2013, SQLI was rated to take number one malicious attack on the OWASP[1].

There are four main sub-classes of SQL injection[7]:

1. Classic SQLI

2. Blind or Inference SQL injection
3. Database management system-specific SQLI
4. Compounded SQLI:

- SQL injection + insufficient authentication
- SQL injection + DDoS attacks
- SQL injection + DNS hijacking
- SQL injection + XSS.

THE TECHNIQUES FOR SQL INJECTION

The various forms of SQL injection attacks are happening when a user input is not filtered very well to establish an escape characters and then did pass them into an SQL statement to the server. This issue using a potential technique for manipulation of the SQL statements are mostly performed on the target database directly by the end-user of the application[5].

The following line of code illustrates this vulnerability:

statement = "SELECT * FROM users WHERE name = '" + userName + "';"

This SQL code is specially designed to tow up the records of the definite username from a list of table of users. However, if the variable of "username" is designed in a good specific way through a malicious user, then an SQL statement may achieve a different code as the primary

author intended. For instance, setting a "userName" variable as follow: ' OR '1'='1

For all kinds of SQL malicious injection, there should be no method where an attacker can achieve malicious injection by avoiding to insert (a space, single quotes or double dashes) in a query.

It is found that attacker firs check of the web application is vulnerable by observing error messages [8]. Attackers then can send their original malicious query such as the following:

Name: ExampleName'; drop table users—

As a result of execution of above SQL statement, the single quote after the 'ExampleName' will close the opened quote in the original query and “drop table” will remove the users table from the database which may lead to large damage to vender as the database may store confidential information about users and sometimes financial information. It should be noted that the two dashes (—) at the end of input will escape any remaining part of query by commenting them.

Based on the above, it is found that SQLIA can be achieved using various techniques. Some of the most attacks detailed as follows[4]:

Tautologies: The main objective of this type of attack is to use an injection of code in a conditional statement in order they are always able to evaluate as true.

Logically incorrect query attacks: This method is mainly to achieve a Logically/Illegal/ Incorrect Queries which is mainly based on SQL Attacks and designed to collect the data associated with the target database associated with the Web Application

Union Query: The main target of the Union Query is mainly to trick the target database where the attacker wants to achieve malicious damage to the data by returning the results from a table which is different to the one originally intended. However, adopting this technique, the attackers can join injected query to obtain a safe query then by using the word UNION and then attacker will be able to get data about other tables from the application. This method is performed to achieve bypass verification as well as extract data.

Timing Attacks: In this type of attack the attacker collects special information from a target database by spotting timing delays when the response from the database is generated.

Stored Procedures: The main objective of this technique is to achieve privilege escalation and attempt to perform undetected SQL procedure.

Alternate Encodings: The essential objective of this attack is to evade being identified through the use of a code based on secure defensive way and then start to implement a prevention mechanism. Hence this technique helps the attackers to be undetected.

Blind Injection: It is found that developers usually hide the error details which help attackers to use a method of compromise to cheat the target database. So, in such scenario the attacker may face to a common page which are provided by developer at the start of writing the code, instead of any error message that might happen.

DETECTION TECHNIQUES

The various Detection techniques are found these techniques that used by web developers to detect attacks typically at runtime of JavaScript code at the client side. Today, there are many techniques and most of them are different from each other in detecting and preventing SQLIA using special coding tools and mechanisms for detecting or preventing malicious SQL attacks on target database on the server. Before introducing detection technique, it is important to show the highly common types of security vulnerabilities which were found in the literature [9] (see Table 1 below).

Table 1. Types of Vulnerabilities

Vulnerability Types	Description
Type I	The validation of input is a trial to check or screen any suspicious input for possible malicious conduct. Inadequate input field validation may permit a malicious code to be executed many times without proper and exact verification on the original intention. So, SQL attacker may wish taking benefit of inadequate validation to an input field so that they he is able to utilize a malicious code and then conduct a malicious attack on the target database.
Type II	The lack of perfect division between data types which are accepted as an input after writing a programming language utilized for the development of web application.
Type III	Any delay of processes in the analysis until the runtime stage as the present variables are measured despite than source code using an expression to achieve the attack.

Some papers in literature even refer to stored procedures as a remedy against SQLIA. As stored procedures reside on the database front, the methods proposed by them cannot be applied to secure stored procedures themselves[10]. For example, some studies proposed a novel technique to defend against the attacks targeted at stored procedures[5]. This technique combines static application code analysis with runtime validation to eliminate the occurrence of such attacks. Other researchers developed a method that detects and prevents SQL injection attacks by checking whether user inputs cause changes in the query's intended result[11]. They proposed a method to detect SQL injection attacks by using Query tokenization that is implemented by the Query Parser method[12] used Black Box Testing technique, for testing Web applications for SQL injection vulnerabilities. The technique uses a Web crawler to identify all points in a Web application that can be used to inject SQLIAs. Other researchers[13] used a technique called Dynamic Analysis, where in this is very useful for conducting analysis of runtime or dynamic SQL query, it is generated with user input data first by implementing a web application. The Detection techniques is also under post-generated category a could be executes the query before sending the query to the target database in the server side. There is another proposed technique called SQL-IDS, it is a specification based approach to detect malicious intrusions[14]. Some researchers in SQLIA suggest using new methodology based on specification as well as characteristic of exploitations and detection of malicious SQL injection so that to avoid vulnerabilities. It is found that a detection based on query-specific allow the system to achieve analysis that is focused on negligible computational without any need for producing false negatives or false positive. The study focus of detection technique types I, a proposed a detection technique called CombinedDetect based on JavaScript and two solutions, where the depending on input validation as an effort to prove or filter any suspicious input found bear a specification of malicious conduct[9]. Most of methods showed that insufficient input validation detection technique adopted will allow malicious code to be executed without proper verification of its intention.

Therefore, any detection technique should prevent attacker to take great advantages of insufficient input validation which threat the target database and help the attacker to utilize malicious SQL code to conduct attacks easily[15].

METHOD OF THE STUDY

The method proposed in chapter can improve the protection level against SQLIA and ensure large enter of user data at the same time. The proposed method in this study is termed CombinedDetect composed Parameterized filtration queries: Using of parameterized queries is a secure technique against SQL injection.

In this method the developer used JavaScript to validate input field after the login stage. In this method there are some placeholders in SQL query preserved particularly for the variables of SQL statement.

The SQL engine first parse and compile the query without the variables and keep the result. Following this stage, the technique adds variables and compile them again. In this case, the attacker may intend to insert a malicious query directly using the variable as a transport mean to send the malicious code to the SQL query that uses this variable, later the SQL engine will the treat content of this variable as an ordinary string.

The method is composed of the four stages starting with inputting the data in the input field of the web page. The user then starts to register his data in the username and password input fields, then start login. After this stage the method developed in this study will begins input validations by removing any suspected characters like ' OR '1'='1. A smart hacker might get access to all the user names and passwords in a database by simply inserting 105 or 1=1 into the input box, or by simply inserting " or ""=" into the user name or password text box. The result as follow:

1. *Registration Stage*

In registration phase a user fills the form by relevant details like username, password, date of birth, address, and other data related to application. Now in order to avoid SQLIA we need to encrypt only those fields that we will use in conditional clause of SQL query and include variables to be sent to a PHP file because with these fields only there will be a chance of SQLIA.

2. *Login Stage*

In a login stage where a user intends to enter username/password. Now the requirement of the protection against the attack require to match user's credentials from the target database but originally the password and username in database are encoded because of the registration phase. So before starting to match the username and password with the main database before encryption of both the input fields. The vital point here can be formed in the same way as in registration phase. Here there is no essential to save in a secret key in the target database.

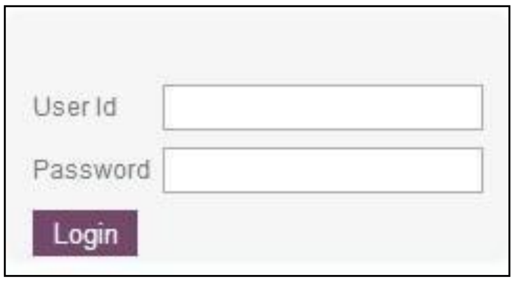

Figure 1. Login Fields

3. *Search Stage*

During the search stage the user usually interacts frequently with the target database and can then bring any saved data based on particular input fields provided in search form. As understood these particular fields are saved in an encrypted form related to the database so that the developer need to encrypt these fields before executing SQL query.

RESULT AND DISCUSSIONS

The first step in study methodology is validating the input fields of a form used by the website to let site registered users to login for their personal accounts.The technique of filters in the variable txtUserId to disnguish from any suspicious parameter or maybe a text that can be used incorrectly for malicious purpose to access the target data saved in all rows or they are called records of the database. This method is important as an SQL statement to be used as an intermediate variable (in this case txtUserId) through web page input as well initiate a shield in client side to provide protection to a web application from any negotiation to the level of security of the target database in the server.

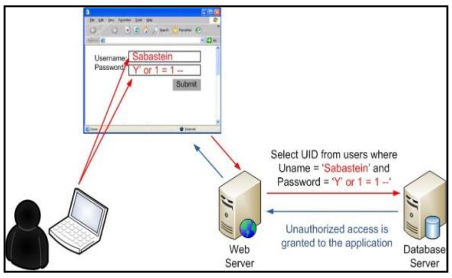
Username: Sabastein
Password: 'Y' or 1 = 1 --
Submit
Select UID from users where Uname = 'Sabastein' and Password = 'Y' or 1 = 1 --'
Unauthorized access is granted to the application
Web Server
Database Server

Figure 2. A diagram showing steeling of information through malicious injection in the username and password field

The filtration of study method includes the following three categories:

1. SQL Injection Based on 1=1 is Always True
2. SQL Injection Based on ""="" is Always True
3. SQL Injection Based On Batched SQL Statements

Based on the above three compromises the security threats using special parameters in the input field, the following is the first stage in study method:

In the client-side, the study method is based on form validation and filtering malicious parameters before sending to the server to prevent SQL attack and injection. In the client-side, the study method is based on form validation and filtering malicious parameters before sending to the server to prevent SQL attack and injection. In this stage the method establishes a blacklist of characters/parameters to provide fist level of protection against SQL injection attack.

In this code the entered variables (username, password) of the user is sent by JavaScript Ajax object to the server in order to process by PHP and SQL statements. The filtration is based on one of the following three options:

Option 1: User is a real person and enter incorrect username or password. The JavaScript code send the username and password to the server through AJAX object and check these input data with the database in the server the response from the page

Option 2: The user is not a real user and either a hacker or auto spambot. A spambot is a computer program designed to assist in the sending of spam.

Option 3: The user is a hacker aimed to use either username and password inputs to inject malicious SQL code, and start the attack. In this case the JavaScript code will filter the input field if any malicious parameters are found.

Following the above filtrations, a PHP code and SQL statement is executed in the server to fetch the data from the database after verification of username and password from any malicious code.

CONCLUSION

It is concluded that many existing techniques, such as filtering, information-flow analysis, penetration testing, and defensive coding, can detect and prevent a subset of the vulnerabilities that lead to SQLIAs. However, filtering (input validation) seems very efficient and simplest technique. Despite it is found that SQLIA detection techniques that employ input validation are disposed to a large number of false positives and yet there is no 100% guarantee that there are no false negatives.

The method developed in this study provided strong filtering to input fields and forms validation that prevent any injection of malicious parameter to the database or text that can be used incorrectly by hackers to access the data stored in all rows (records). The proposed method

in this study is termed CombinedDetect. It is found that using of parameterized queries is a secure technique against SQL injection. The method is composed of three stages (registration, login, and search) and during these stages a filtration of input fields is done by using a "blacklist" of words or characters to search for in SQL input, to prevent SQL injection attacks. After a user login in web application there is a need for more interaction with database and more number of field are present in other web pages.

Finally, it evident that encryption can be used effectively in future methods to prevent SQL injection attack in wide range of programming environment and applications. The prevention method considered at all the phases of the application starting by Registration, Login and Search phase.

RECOMMENDATIONS

Based on the findings and result, the study suggests the following recommendations:

1. Avoiding any weakness in SQL server by providing effective input validation to discriminate the malicious parameters used for injection SQL attack queries that may damage the whole data in a target database. The most important precautions are data sanitization and validation, which should already be in place.

2. Using multiple detection methods for SQL injection. Today, no single solution is sufficient to defeat SQL injection attacks. This is due to the power of SQL and the flexibility that it gives to the user.

3. Reducing SQL attacking surface by getting rid of any poor database functionality that may enable the hacker to take advantage of it and start injection his malicious code.

4. It is very important to update and patch the database security by enhancement the vulnerabilities in web applications and databases that hackers could exploit through SQL injection from time to time on regular bases when new injection techniques are discovered, so it's vital to apply patches and updates as soon as practical.

REFERENCES

Aleksiander, Xhuvani and Sheehu Bojkenn, 2014, the Computer Engineering Department, University of Tirana Tirana, Albania. IJCSI International Journal of Computer Science Issues, Vol. 11, Issue 4, No 1.

B.I.E. Barrey and H.F. Chaan, 2009, "Syntax, and Semantics-Base Signature Database of Hybrid Intrusion Detection Systems," Security and Comm. Networks, Vol. 2, no. 6, pp. 456-474.

Bojjken Shi. , A. Shqiponjja, A. Marein, and Xh. Alekksandder , 2013," Pr otection of Personal Data in Information Systems", International Journal of Computer Science, Vol. 10, No. 2, pp1694-0784. Foccardi, R.; Luccio, FL.; Sequarcina, M., 2012, Fast SQL blind injections in high latency networks,

Braad Warrneck, 2007, "Defeating SQL Injection IDS Evasion." The Institute Information Security Reading Room.

Computer: pp66-72.

Guaan, Shhengdoing and Waenng, Lix, 2005, "Analysis of restrictive factors of e-commerce implemented in traditional retail trade in China". Proceedings of the 7th International Conference on Electronic Commerce, China. pp. 819-821

Hattoon Matbouly, Qiegang Giao, 2012, "An Overview on Web Security Threats and Impact to E-Commerce Success", International Conference on Information Technology and e-Services, pp.978-1166-IEEE Journal.

IEEE First AESS European Conference on, vol., no., pp.1-6.

K. Kemalies, and T. Tezouramanis, 2008, "SQL-IDS: A Specification-based Approach for SQL Injection Detection", SAC, ACM Journal, pp.2154-2157.

Nunno Anttunes, Marrco Vieirra, 2009, “Detecting SQL Injection vulnerabilities in web services”, IEEE Journal, pp.201-222.

Open Web Application Security Project (OWASP). “Top 10 web application vulnerabilities, Nuneo, Anteunes, and Marcco Vieira, 2012, “Defending against web application vulnerabilities.

Seixaas, J. Fonsicca, Vieiraa, and Madeiraa, 2009, “Looking at Web Security Vulnerabilities from the Programming Language Perspective: A Field of Study”, pp.128-136.

Weii, Kee, M. Surraj Koothari, Muuthuprassanna, 2013. Preventing SQL Injection Attacks in Stored Procedures. Iowa State University

Yanng, H., Y. Heuangg, H. Yu, F. Tesai, S. Ku, 2009, “Securing Web Application Code by Static Analysis and Runtime Protection”, International Word Wide Web Conference.

Yuenqia Wienm Chuunhui Zhoue, Juen Maa, Keizhong Liu, 2008, "Research on E-Commerce Security Issues", the Journal of Business and Information Management, Vol.1.