ISSN (e): 2250 – 3005 || Volume, 05 || Issue, 09 ||September – 2015 ||
International Journal of Computational Engineering Research (IJCER)

Prevention of SQL injection in E- Commerce

Sanchit Narang[1], Shivam Sharma[2], Rajendra Prasad Mahapatra[3]
[1,3] Computer Science & Engineering Department, 2 IT Department. SRM University ,NCR Campus, Modi Nagar , (UP) India.

ABSTRACT

Structured Query Language (SQL) injection, in present scenario, emerges as one of the most challenging fact to effect on the online business, as it can expose all of the business transaction related sensitive information which is stored in online database, inclusive of most highly secured sensitive information such as credit card passwords , usernames, login ids, credentials, phone, email id etc. Structured Query Language injection remain a responsibility that when intruder gets the ability with SQL related queries which is passed to a back-end database. The query which is passed by the intruder to the data, can allow the query to data which is an assisting element with database and required operating system. Every SQL Query that allows the inputs from the attacker sides can defect our real web application. Intruder which attempts to insert defective SQL query into an entry field to extract the query so that they can dump the database or alter the database which is known as "code injection technique" and this type of attacker is also called attack vector for websites and usually used by any type of SQL database. Through this research paper, our endeavour is to understand the methodology of SQL injection and also to propose solution to prevent SQL Injection in one of the most vulnerable field of E commerce.

Keywords: *Business, database, effect, Intruder, injection, query, solution, etc*

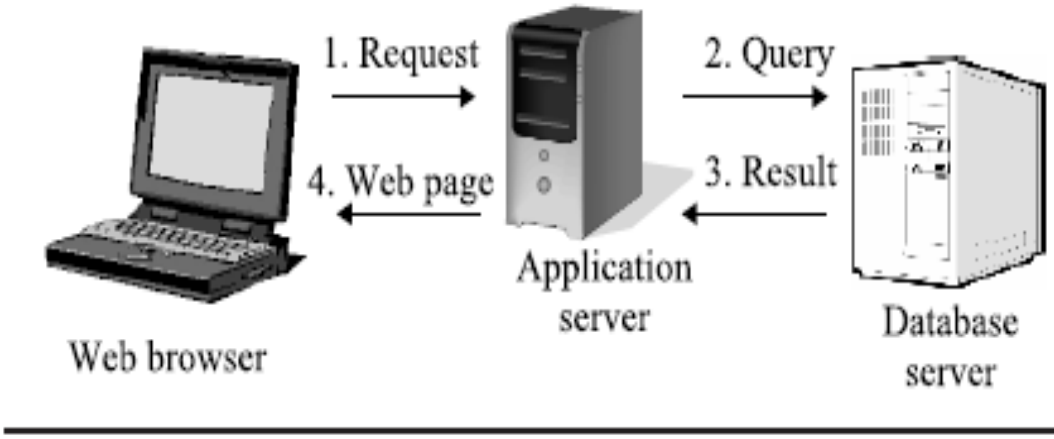
I. Introduction

In recent years, inclusion of internet in almost every type of business transaction resulted into rapid advancement in the dependency towards information technologies. The information technology used by the general masses for the reasons such as commercial transactions, educational transactions and other countless activities related to finance. The use of the internet to accomplish important odd jobs, such as transactions of balance from bank accounts, always remains under security threat. Present web sites legging behind to keep their users data confidential till years of doing secure business online and due to which these industries have become experts in intelligence information security. The data management systems besides these secure websites collect non-potent data along with secure information, in this way, the potent information owner's quick access while jamming break-in attempts from intruders. The most common break-in strategy is to try to access sensitive information from a data storage by generating a query that cause the data parser to malfunction and thereby applying this query to the desired database. This above said approach to gaining un ethical access to private information is called SQL injection.

Since databases are anonymous and can be accessed from the internet, combating with SQL injection has emerges as more important than ever. The current database management system comprises with little vulnerability, the Computer Security Institutions have discovered that every year about half of the databases experience at least one security breaching attempt and the revenue losses associated with such attempts has been estimated more than $4,000,000.00.

II. Literature Review

Moreover, recent researches published by the "Imperva Application Defense Center" propounded that minimum 92% of website data and applications are prone to malicious attacks (M. Muthuprasanna & Kothari, 2007). To enhance understanding of SQL injection, it is better to have good understanding of the kinds of communications that take place during a typical session between a user and a web based application. The figure below describes the typical communication exchange in between all the composites in a typical web application systems.



After reviewing numerous electronic journals, articles from IEEE/FCI journals and gathered information provides sufficient knowledge about SQL injection, it's attacking methodology and its prevention. The successive review of distinctive papers is presented herewith.

Gregory T. Buehrer, Bruce W. Weide, and Paolo A. G. Sivilotti, in the research paper "Using Parse Tree Validation to Prevent SQL Injection Attacks" publishes there commendable work in 2005. The techniques for sql injection discovery seems impressive as this paper also covered the SQL parse tree validation very specifically which is mentionable.

Zhendong Su and Gary Wassermann, in 2006 published their research in SQL injection entitled "The Essence of Command Injection Attacks in Web Applications" and covered the specific methods to check and sanitize input query using SQLCHECK, it use the augmented questions and SQLCHECK grammar to validate query.
Stephen Thomas and Laurie Williams, worked so long in the field of SQL injection and in 2007, in their research paper, entitled "Automated Protection of PHP Applications against SQL-injection Attacks" they covered an authentic scheme to protect application automatically from SQL injection intrusion. This authentic approach combines static, dynamic analysis and intelligent code re-engineering to secure existing properties.

Ke Wei, M. Muthuprasanna & Suraj Kothari in 2007, published their work entitled "Preventing SQL Injection Attacks in Stored Procedures" and through this they provides a novel approach to shield the stored procedures from attack and detect SQL injection from websites. Their method includes runtime check with subsequent application code analysis to eliminate vulnerability to attack. The key method behind this vulnerability attack is that it modifies the composition of the original SQL statement and identifies the SQL injection attack. The process is divided in two instalments, one is off-line and another one is run-time in real time basis. In the off-line phase, previously stored procedures uses a parser to pre-process and detect SQL statements in the working call for run-time analysis but in the run-time phase, the technique monitors all run-time generated SQL queries related with the user input and checks these with the original structure of the SQL statement after getting input from the user.

III. Principle

Web based applications have SQL injection susceptibilities as they do not disinfect the inputs they use to construct fabricated desired output. The gathered code is remains from an online storage system. The web-site provides user's input field to allow the user to keep their secret information, such as credit card/debit card password, etc. which user can use for future purchases conveniently. Replace method used to escape the quotes so that any single quote characters in the input are considered to be literal and not a string delimiters. Replace method is intended to block attacks by preventing an attacker from ending the string and adding SQL injection code. Although, card-type is a numbering column, if an attacker passes 2 OR 1-1 as the card type, all account numbers in the database will be returned and displayed on the screen.

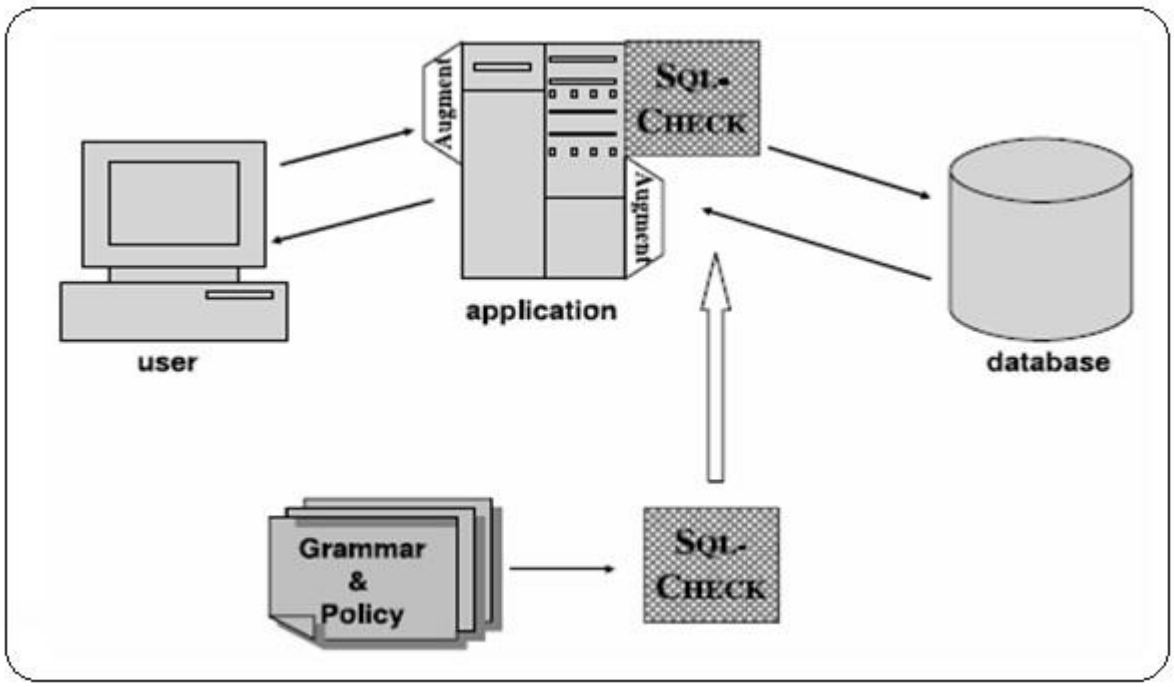


Fig: System composition of SQLCHECK

IV. Methodology:

SQL injection is widely used hacking technique, in which the intruder adds SQL statements using a web application as input fields to access to the secret users resources and due to lack of input Validation in web applications causes intruders to be successful in hacking.

With above said technique, we can assume that a Web application receives a "http//" request from a user client as Input and generates a SQL statement as output for the back-End database server. For example an administrator will be authenticated after providing input as -

Typing: employee id - 0112

and password =admin, configure

That describes a login by a suspecious user exploiting sql Injection vulnerability .

Usually, it is structured in three phases,

(1) An Intruder sends the malicious "http//" request to the Web application,

(2). Generates the sql statement,

(3). Dedicatedly Deposited the sql statement to the back end database.

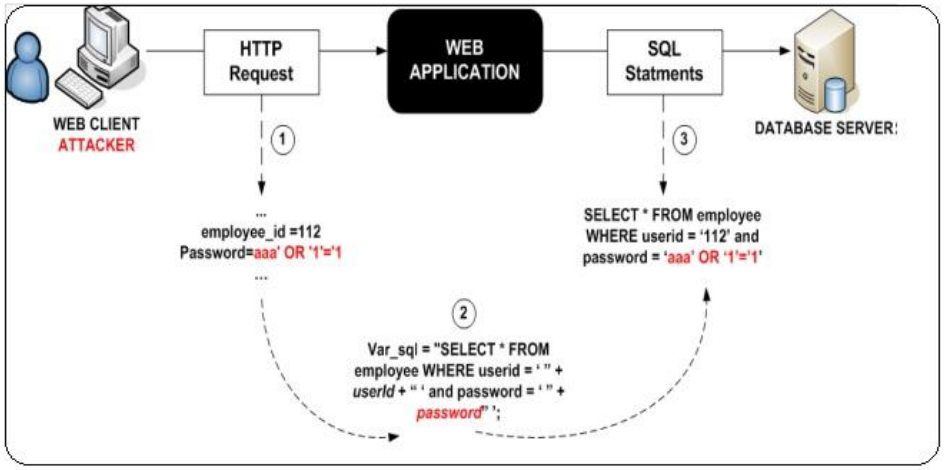


Fig: Example methodology of SQL Injection.

As per the principle, they track through the program and the substrings received from user input and filters that substrings immediately. The endeavour behind these program remains blocking the malicious queries in which the incoming substrings changes the syntactic morphology of the rest of the queries. They use the metadata to analyse user's input which displayed as ('_') and ('_,') to mark the ending and beginning of the each user incoming string. This said metadata passes the incoming string through an assignments, and concatenations, so that when a query is ready to be sent to the database, it has a matching pairs of markers that identify the substring from the input. These annotated queries called an augmented query. To construct a parser for the augmented grammar and attempt to parse each augmented query Steve use a parse generator. Query meets the syntactic constraints and considered legitimate if it passes successfully. Else, it fails the syntactic constraints and interprets it as an SQL injection attack.

As per the system architecture of the checking system shows in Figure above, the Grammar of the outgoing data language is used to build SQLCHECK and a policy mentioned permitted syntactic forms, it resides on the web server and taps generated queries. In spite of the input's source, each input which is to be passed into some query, gets augmented with the meta-characters ('_') and ('_,').

Finally application creates augmented queries, which SQLCHEKCK attempts to parse, and if a query parses successfully, SQLCHECK sends it the meta-data to the database, else the query get rejected.

.

V. Observation

Moreover, developers deploy defensive coding but they are not enough to stop SQL injections to web applications, that is why researchers have proposed some of tools to assist developers to prevent there. It is mentionable that there are more approaches which have not implemented a tool yet. In this paper, we emphasize more on tools and less on techniques. Huang and colleagues propose WAVES, a black box technique for testing web applications for SQL injection vulnerabilities. The tool identify all points a web application that can be used to inject SQLIAs. It builds attacks that target these points and monitors the application how response to the attacks by utilize machine learning.

VI. Conclusion

SQL injection intrusions remain a serious problem for developers as they can be used to break into supposedly secure systems and copy, alter, or destroy data. It becomes very easy to leave yourself vulnerable to these attacks, regarding of which version of .NET. In fact, you don't even need to be using .NET to be susceptible to SQL injection attacks. Any further implecations that queries a database using user- entered data set, including Windows Forms applications is a potential target of an injection attack.

Protecting user against SQL injection attacks is less difficult. Uses which are immune to SQL injection attacks validate and sanitize all user input, never use dynamic SQL injection, perform using an account with few privileges, hash or encrypt their secrets, and present error messages that reveal little if no useful information to the hacker. By following a multi-layered approach to prevention you can be assured that if one defence is circumvented, you will still be protected. For information on testing your application for injection vulnerabilities, see the sidebar "Injection Testing".

References

[1] W Ke Wei, M. Muthuprasanna, Suraj Kothari , Dept. of Electrical and Computer Engineering , Iowa State University Ames, IA – 50011 ,Email: {weike,muthu,kothari}@iastate.edu

[2] http://www.appsecinc.com/presentations/Manipulating SQL Server Using SQL Injection.pdf, White Paper.

[3] William G.J. Halfond, Jeremy Viegas, and Alessandro Orso College of Computing Georgia Institute of Technology {whalfond|jeremyv|orso}@cc.gatech.edu

[4] Z. Su and G. Wassermann. The Essence of Command Injection Attacks in Web Applications. In The 33rd Annual Symposium on Principles of Programming Languages (POPL 2006), Jan. 2006.

[5] F. Valeur, D. Mutz, and G. Vigna. A Learning-Based Approach to the Detection of SQL Attacks. In Proceedings of the Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA), Vienna, Austria, July 2005.