SQL Injection Attacks: Detection And Prevention Techniques

Raniah Alsahafi

Abstract: Database driven web application are vulnerable to SQL Injection Attacks which try to access the sensitive data directly. They work by injecting malicious SQL codes through the web application and cause unexpected behavior from the database. There are different Techniques that have been proposed by researchers to prevent or detect these type of attacks. This paper has presented most of all proposed methods and tools to detect SQL injection attack. Finally, a comparison between those methodology has been presented and analyzed.

Index Terms: Databases Detection System, legitimate query, malicious inputs, SQL injection attack, Prevention Techniques, Vulnerabilities

1. INTRODUCTION

Many of the web applications use databases as their back-end data store. Although new web programming languages offer new ways of more secure database programming features, still there are many applications that are vulnerable to SQL injection attack. Because of the nature of this attack which is unauthorized access to the confidential information and inserting or modifying it, this kind of attack is very popular among attackers and again because of the mentioned reason it is vital to make web applications secure against them. The SQL injection attack happens often when an attacker tries to gain access to a database by inserting malicious inputs to the queries that change the logic, syntax or semantic of the legitimate query. There are many types of SQL injection attacks such as, using tautologies, alternate encodings, UNION, ORDER BY, HAVING. And in return there are many suggested methods trying to detect the vulnerabilities in the applications and prevent the attacks. To be more specific, this survey presented some different papers with several proposed methods and tools to detect and prevent the SQL injection attack to be able to get a wide range of ideas in this way and compare them against each other. I defined a methodology in our research to be able to get more information from each paper. So, after that it was much easier to compare the methods and analyzing them. I focused on main parts and phases of each method proposed in each paper and then I found the advantages and limitations of them at the end.

2. PROBLEMS AND ISSUES

Having the best efficiency and having the most secure applications are always in contradiction with each other. We cannot have high security without any cost. Here the cost mostly means the time for execution. Most of the applications that are vulnerable to SQL injection attacks are web applications. There are some known types of user crafted SQL inputs that researchers are working on them and there are some offered methods and tools to detect such malicious inputs and escaping them. Here are some problems and issues that we encounter when talking about SQL injection attack and securing the applications.

- Securing existing web applications with the less cost.
- Securing dynamic SQL statements.
- Securing stored procedures.
- Detection and validation of the malicious SQL codes in the user supplied inputs.
- Methods for removing and modification of the malicious inputs to amend them as validated ones.
- Consuming less execution time.
- Tracing the input from the entry point to the SQL statement.

3. DETECTION AND PREVENTION TECHNIQUES

3.1 AMNESIA: Analysis and Monitoring for Neutralizing SQL Injection Attacks

Method: This approach aims to detect illegal SQL queries before being executed on the database. It based on the combination of static and dynamic analysis technique to verify the correctness of SQL queries. In its static part it use a program analysis to build a model of the proper queries that can be generated by the web application, and in the dynamic part this technique use the runtime monitoring to check the generated queries against the statically-built model. If the queries do not match with model, it will be rejected and consider that as the SQL injection attack, but in case of matching with the model, it will continue execution into the database. This technique consists of four main phases. The first phase is Identifying the hotspot by scanning the source code of the web application and determines the statements that generate the SQL query to the database. Build SQL-query models is the second phase. In this phase, a model for all possible SQL queries is built for each hotspot that Identified in the previous phase. The third one is an Instrument Application that insert a call to the time monitoring to examine the SQL queries before sending them to the database .Last phase is a runtime monitoring at runtime, the application executes normally until it reaches a hotspot and in this point, the monitor examine the SQL query by convert them into set of tokens, and compare them according to the static model generated. Then it can reject the query that does not match with the model. Furthermore, the authors use a tool is named AMNESIA to implement their approach to prevent SQL injection attack for Java-based web applications.

Advantages: The evaluation of the technique was performed on a set of seven web applications which include two of them was developed by student. AMNESIA was able to stop all of the 1,470 attacks without any false positive. Also, the time overhead on the web application was measurable.

Limitations: This method has some difficulty for preventing the attacks that based on stored procedures and Successful of this technique Depends on the accuracy of the static analysis for generating the legal SQL queries.

3.2 SQLrand: Preventing SQL Injection Attacks

Method: It based on the Idea of Instruction-Set Randomization which that means Instead of normal SQL keywords there is a random integer number that append to these SQL keywords. User input inserted into the “randomized” query consider as a set of non-keywords, resulting in an invalid expression. This technique consists of a proxy parser between the web server and the database server. Proxy intercepts the randomized query, decodes to proper SQL queries to be sending to the database. In case if SQL injection attack has occurred, The proxy will not recognize the injected SQL commands and will reject them ,and disconnect from the database and report error message to the web application.

Advantages: SQLrand makes possible for the developer to create randomize SQL commands instead of normal commands.

Limitations: The main disadvantages of SQLrand depends on the security of the key and if the key used for randomization is compromised by an attacker, then is become difficult for SQLrand to prevent an attack. SQLrand do not prevent the Illegal/Incorrect SQL attack, because the attacker can gain some related information from the error messages that SQLrand report to the web application when the proxy not recognize the injected SQL commands. Therefore, SQLrand can’t handling attack based on strode procedure.

3.3 SQL-IDS: A specification–based Approach for SQL-injection Detection

Method: It is based on the specification model that defines the syntactic structure of SQL statements that can be generated by the web application. Each SQL query that is sending by the web application, it is intercepted to check its correctness according to the specification rules that have been pre-defined. If the SQL statements do not match the syntactical rules that is specify, it will be rejected; otherwise, it will consider as valid and forward them to database. The authors in this paper have present a prototype implementation for their methodology which is applied on Java based Web application. This approach consists of six phases in order to detect attacks which as the following:

1. Define specification of the web application which is set of rules that define the expected SQL statement.
2. Interception of SQL statement that send from the web application to database.
3. Lexical analysis for each SQL statement that is intercepted.
4. Syntactical verification of SQL statements if does not match the rules it will rejected.
5. Forward valid SQL statement to the database to begin execution.
6. Logging files that can store for any useful information about the detection attacks.

Advantage: The main advantage of this approach is that It has log files for recording the important information about types of attack that is detected and prevented from execution on a database, so that It can be useful for developers to improve their application performance. To apply SQL-IDS, there is no need to modify the source code of an existing web application. Also, they evaluated their result based on the performance, effectiveness and the precision of the system. They found that their system was successful for detection SQL injection attack with average time overhead is 0.5 mile seconds per SQL query.

Limitations: The main disadvantage of this method that is all the detection attack were unknown to the system

3.4 Sania: Syntactic and Semantic Analysis for Automated Testing against SQL Injection

Method: In this technique the authors have presented new technique to provide protection against SQL injection attack. They introduce Sania, for detecting SQL injection vulnerabilities in web applications during the development and debugging phases. Sania based on intercept of SQL queries and HTTP request and analyzes the syntax of SQL queries that generated from the request to discover SQL injection vulnerable spots, and then it try to generates attack codes based on syntax of SQL query to attempt exploiting the vulnerable spots. By comparing the parse trees of the SQL query and that resulting after the attack with that generated from innocent request, Sania can verify if the attack was successful or not.

Advantages: Sania checks SQL injection attack in the development phases. The authors evaluate Sania and compare its effectiveness with public web application scanner tools and they found out that Sania has detected 39 SQL injection vulnerabilities. In contrast with that Paros, a popular scanner for web application vulnerabilities found 5 vulnerabilities and it was successful to detect false positive attack.

Limitations: This approach requires developers to know all SQL queries and HTTP requests that generated on the web application.

3.5 SDriver: Location-specific signatures prevent SQL injection attacks

Method: Driver is a prevention methodology against SQL Injection attack. This technique based on the Idea of using database driver that between the web application and the database. The main work of this driver is that each SQL statements identify by using the query’s location and a stripped-down version of its content to create the signature of legal SQL statement by combining the stack trace, SQL query key words and the tables that the query is used. Then they can use this signature to different between the legal query and the injected ones during the run time. The system applies a hash function on them to generate the stored form of the query signature. The system consist of two mode: the first one is the training mode that is work to generate the signature of legal SQL statements and stored them into table on the database, and during a production mode the driver is going to consults the database table of saved query signatures to verify that the query is legal or not. In case if an attack occurs, the driver produces an error message or an alert o the system and prevents it. The authors developed a tool is called SDriver to implement their approach. They test the accuracy of their tool depend on number of signatures that stored in the SDriver database ,number of unsuccessful and successful attacks .The found that SDriver is was successful for preventing all type of SQL injection attacks.

Advantages: Test its performance into two types of database (MYSQL server and Microsoft server) and the average overhead was similar 4.7 milliseconds per query.

Limitations: A disadvantage of this approach is that any alteration on the web application, the new source code invalidate existing query signature.

3.6 A heuristic-based approach for detecting SQL injection vulnerabilities in Web applications

Method: This paper introduces Viper which is a tool that performs penetration testing for web application in order to prevent SQL injection attack. This technique is based on analyzing web application to gather information about its structure and page form. Then it tries to inject SQL string on the input field to make the web application report an error message. Then, it matches the output produced by a web application and the pattern of regular expression that related to error message. The summarization of approach phases are the following:

1. Gathering information about a web application structure.
2. Identification of input parameters.
3. Create an attack based on the error messages produced by a web application.
4. Report the result and store it on log files.

For evaluation of this tool, the authors compare Viper with SQLMap to show the higher performance of their proposed approche.They compare them according to number of vulnerable hyperlinks, input parameters and database field and tables that Viper could discover on the web application .They conclude that Viper has the higher effective for detect the SQL vulnerabilities more than SQLMap .

Advantages: This approach based on doing penetration test for an existing web application. It has low cost to test the web application compare with SQLMap.

Limitations: The error messages that produce by the web application can lead to infer relevant information which cannot prevent illegal SQL attacks**.**

3.7 Learning Fingerprints for a Database Intrusion Detection System

Method: One of the techniques can detect any malicious access to database server is intrusion detection systems. It's Idea based on fingerprinting access pattern transaction of proper SLQ statements. The overview of this system consists of five steps which are as the following:

1. User request service to the application server which could be legal or not.
2. Application server generates SQL statements and issues them to database server.
3. Database server receives SQL statements and transfers them to misuse detection module.
4. In misuse detection module, the received SQL statements are match with set of legal fingerprinting model.
5. Then any anomalies transaction is transfer to the Intrusion model to take the reaction which could be an alert for the system.

Furthermore, the authors have developed an algorithm that can generate set of fingerprints for legal SQL statement and detect any risk transaction to the database.

Advantages: This system attempted to detect intrusions as soon as possible in the application level.

Limitations: The successfully of this approach depends on the accuracy of the learning algorithms and generating set of t fingerprint transaction.

There are different techniques in detecting and preventing SQL injection attacks. Each of them has their own advantages and limitations. But I categorize the main idea of them as bellow:

Static analysis: These kinds of analysis methods have the advantage of increased precision, no runtime overhead and detecting the errors before the deployment. But they need more effort of the programmer. Because the time overhead in these methods is related to compile time, the end user of the web application will not be aware by this overhead.

- Sania
- SQL DOM

Combination of static analysis and runtime monitoring: These methods often use static analysis to generate an acceptable model and then use that model to verify the statements and values of the variables at runtime

- AMNESIA
- Prevention technique in stored procedures

Application level intrusion detection:

- SQL-IDS

Randomization of SQL keywords: using SQL keywords prepended by a random key. Although these methods are very effective, the security of them is completely dependent to the security of the random key.

- SQLrand

5 Conclusion

Many different approaches have been proposed to detect and in consequence prevent SQL injection attacks. Some of them are trying to find vulnerabilities in the web applications and then trying to solve the problems; some others are focusing on the user inputs and verifying them based on some predefined patterns and algorithms. Some of the methods and tools are relying on some others and some of them have brand new ideas. We tried to study as many different methods as possible and now we are so interested in following our researches about SQL injection attacks using some ideas such as parse tree and finite state automaton in the future.

6 REFERENCES

[1] W. Halfond, A. Orso, "AMNESIA: Analysis and Monitoring for NEutralizing SQL-Injection Attacks," The 20th IEEE/ACM International Conference on Automated Software Engineering (ASE), pages 174–183, 2005.

[2] Y. Kosuga, K. Kono, M. Hanaoka, "Sania: Syntactic and Semantic Analysis for Automated Testing against SQL Injection," The 23rd Annual Computer Security Applications Conference, 107- 116.

[3] S. W. Boyd, A. D. Keromytis," SQLrand: Preventing SQL Injection Attacks", The 2nd Applied Cryptography and Network Security (ACNS) Conference, pp 292–302, June 2004.

[4] K. Kemalis, T. Tzouramanis, "SQL-IDS: A Specification-based Approach for SQL-Injection Detection", The 2008 ACM symposium on Applied computing, March 2008.

[5] S.Y. Lee, W.L. Low, P.Y. Wong, "Learning Fingerprints for a Database Intrusion Detection System".

[6] X. Fu, X. Lu, B. Peltsverger, Sh. Chen, K. Qian, L. Tao, "A Static Analysis Framework For Detecting SQL Injection Vulnerabilities", compsac, vol. 1, pp.87-96, 2007 31st Annual International Computer Software and Applications Conference, 2007.

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

[8] Liu, A., Yuan, Y., Wijesekera, D. Stavrou, A., "SQLProb: a proxy-based architecture towards preventing SQL injection attacks", 2009.

[9] D.Mitropoulos, D. Spinellis "SDriver: Location-specific signatures prevent SQL injection attacks", Computers and Security 28, 121–129, 2009.

[10] A.Ciampa, C.A .Visaggio, M. D. Penta, "A heuristic-based approach for detecting SQL-injection", Proceedings of the 2010 ICSE,2010.