JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

A Comprehensive Study to Forestalling Code Injection Attacks

[1]Tejal Kolhe, [2] Dr. Umarani Chellapandy

[1]Student, School of Computer Science & Information Technology
[2]Associate Professor, School of Computer Science & Information Technology
Jain Deemed to be University, Bangalore, India
Email: [1] kolhetejal99@gmail.com [2] c.umarani@jainuniversity.ac.in

Abstract: Attackers frequently utilize SQL Injection Attacks to obtain unauthorized access to systems. A software system is being created to prevent SQL injection attacks from gaining unauthorized access to the system. It is accomplished by using a signature-based authentication mechanism to check authenticity and adding a unique value. SQL injection is a serious security risk these days since it allows an attacker to get access to an online system or application by exploiting certain flaws. Our software project proposes a way for analyzing and detecting harmful code in order to identify and prevent attacks. It employs a distinct signature-based scanning technique that employs a different divide-and-conquer strategy to detect threats based on multiple time and space characteristics.

Keywords - SQL Injection Attacks, Protection, Input Validation, Exploiting, Unauthorized Access

I. INTRODUCTION

Injection attacks on code are still an issue today. Traditional attacks are still vulnerable, and attackers are still coming up with new ways to get around existing security mechanisms and run their inserted malware. The prevention of code injection attacks has turned into a website arms race with no clear winner. A code injection attack is used by an attacker to inject malicious code into a running process and then shift execution to the malicious code. There are a number of approaches to dealing with code injection in one way or another. How can we avoid being suffocated by the sea of data, from which meaningful knowledge discovery and improved information usage efficacy are issues that must be addressed? We address SQL Injection as a major web security concern in our project. One sort of code injection attack is SQL injection. Incorrect user input validation is the primary cause. SQL Injection Attacks are combated using existing defensive coding approaches, as well as encryption schemes based on randomization. Defensive coding mechanisms are not always error-free, and so do not completely eliminate the effect of vulnerability. Defensive programming can be time-consuming, making it ineffective at preventing SQLI. An application-level security vulnerability is SQL Injection Attack. A SQL injection attack's main purpose is to obtain unauthorized access to a database, extract data from it, modify it, raise the user's privileges, or cause an application to crash. SQLIA entails gaining unauthorized access to a database by leveraging a web application's susceptible settings.

The present current web time, anticipates that the association should focus more on Web application security. This is the significant test looked by all the association to safeguard their valuable information against malignant access or debasements. For the most part, the program engineers show distinct fascination with fostering the application with convenience as opposed to consolidating security strategy rules. Input approval issue is a security issue assuming an aggressor observes that an application makes unwarranted presumptions about the kind, length, arrangement, or scope of info information. The aggressor can then supply a vindictive information that compromises application. Whenever an organization and host level section focuses are completely gotten; the public points of interaction uncovered by an application become the main wellspring of assault the cross-webpage prearranging assaults, SQL Injections assaults and support flood are the significant danger in the web application Security through this input approval security issues particularly SQL Injection assaults break the information base instrument like Integration, Authentication, Availability and approval.

II. OBJECTIVE

The goal of this project is to develop an automated method for generating SQL injection fixes that are triggered by plain text SQL statements. A server will collect information about previously known vulnerabilities, specifically SQL statements, create a patch, and apply the patch in an automatic manner. The procedure can be carried out by someone with limited security knowledge, and legacy code can be safeguarded, allowing developers to address the SQL injection problem. SQL server flaws might be avoided, according to the paper, by adopting good input validation to separate malicious parameters used in SQL injection attacks and deploying a variety of SQL injection detection algorithms.

III. LITERATURE REVIEW

[1]A Survey of SQL Injection Prevention Techniques: Web apps are now widely used all around the world. Web applications are increasingly being used by businesses and organizations to provide a variety of services to their customers. Web apps receive queries from users, which interact with the back-end database and return relevant data to them. The back-end database frequently contains confidential and sensitive data that attackers are interested in, such as financial or medical information. SQL injection is one of the most frequent ways for attackers to acquire database access. This type of attack takes use of flaws in online applications or stored processes on database servers in the backend. It lets attackers to inject specially designed malicious SQL query segments into a SQL query to alter its intended consequence, allowing them to gain unauthorized access to a database, data can be read or modified, data can be made unavailable to other users, and the database server can be corrupted. According to a study report released in 2010 by the IBM X-Force@ research and development team, the number of SQL injection attacks has increased considerably in recent years, and SQL injection has become the most common sort of online application attack.

[2] In the paper Detection And Prevention Of SQL Injection Attacks Using Novel Methods In Web Applications, the authors present a novel method for detecting and preventing SQL injection attacks in web applications the proposed approach is a runtime detection and prevention methodology that follows the same paths as the standard approach for exchanging queries among architecture parties (PresentationLogic-Storage), but adds an additional defense line on the Data-Tier to ensure that this side does not execute any abnormal codes that may affect the system partially or completely, or the hosted operating system and devices. This strategy is centred on implementing security controls on the database server side to ensure that all SQL queries, whether from within or outside the system, are processed safely without the need for database building or hacking. During the first half of 2010, there were over 400,000 SQL injection assaults per day on average around the world. Web applications and their underlying databases require not just careful configuration and programming, but also effective protection mechanisms to assure their security. To combat SQL injection issues, researchers have presented a variety of solutions and strategies.

[4] However, no single approach can ensure total security. Many of today's remedies are unable to handle all of the issues. In a similar strategy to defend web applications from SQL Injection attacks, Neha Mishra and Sunita Gond describe current methodologies and offer an integrated approach of encryption method with safe hashing. Like in the prior way, the solution works by having DBA create two columns for login and password storage. If a user wants to connect to the database, a stored procedure is used to generate the secure hash values at runtime. These details are entered into the login table when a user's account is created for the first time. When a user tries to access the database, the username, password, and secure hash values are used to verify his or her identity. [5] Nanhay Singh and Khushal Singh published a study called Analysis of Detection and Prevention of Various SQL Injection Attacks on Web Applications in which the web server receives and processes requests from the browser. The database server must approve all database access requests. The database is directly handled by the database management system, or DBMS, and is not directly accessible; instead, queries must be submitted to the database server, which obtains and delivers data from the database. These requests are sent using a specific format and syntax called as SQL. After that, the findings are returned to the web browser as HTML web pages. The circle in the diagram depicts the position of SQLIV within a web application architecture. This SQLIV can be exploited to reveal sensitive information that could be used for a variety of harmful purposes.

[6] According to the survey study Preventing SQL Injection Assaults in Stored Procedures by Wei, K., Muthuprasanna, M., and Suraj Kothari, a unique solution to defend against attacks targeted at stored procedures has been developed. This method combines static application code analysis with runtime validation to prevent such attacks from occurring. We construct a stored procedure parser in the static component, and we use it to instrument the necessary statements for any SQL statement that relies on user inputs. This allows us to compare the original SQL statement structure to the one that includes user inputs. This technique's deployment can be automated and used only when necessary. [7] R. Ezumalai and G. Aghila's paper "Combinatorial Method for Preventing SQL Injection Attacks" to identify SQL injection, this method employs both a static and dynamic technique. It's a SQL injection detection technique based on signatures. They produce hotspots for SQL queries in web application code, divide them into tokens, and send them to be validated using Hirschberg's technique, which is a divide-and-conquer variant of the Needleman-Wunsch algorithm used to detect SQL injection threats. Because it's specified at the application level, it doesn't necessitate any changes to the runtime system and has a low execution overhead.

[8] Jeom-Goo Kim demonstrates how to effectively remove a user-supplied SQL query from the SQL query attributes values. This method employs both static and dynamic analysis. The proposed solution would make use of a function that can detect the attributes of a static SQL query in a web application. The SQL queries generated at runtime were also detected by this function. This technique compares SQL queries generated by ordinary users to SQL queries generated dynamically by the attacker. [9] The concept behind Parveen Sadotra's implemented system is simple: instead of relying on user permissions, we implement complicated defensive coding techniques with which we can detect and prevent SQL injections and provide security to the web application in their paper "Hashing Technique - SQL Injection Attack Detection." For the secure login mechanism, the proposed solution merely uses "HASHING" methods. They do this by calculating the hash value of a user's username and passwords and storing it in a database table alongside the plain username and passwords.

[10] "Comparison of SQL Injection Detection and Prevention Tools based on Attack Type and Deployment Requirements," by Atefeh Tajpour, Maslin Masrom, Mohammad Jojor Zadeh Shooshtari, and Hossein Rouhanizeidanloo, will be done in two primary phases: runtime analysis and static analysis. The first phase is a dynamic/runtime analysis method that relies on the use of tracking technologies to process and monitor the execution processes of all received queries. The outcome of this monitoring's affected items will be compared to a predetermined set of expected modifications that the developer had made previously, and the output of this comparison process will determine whether any form of SQLIA exists, and if so, it will be passed to the next phase. The static analysis phase follows, which involves string matching between the received SQL queries and previous expected SQL queries in order to block any query that is described as a cautious query.

[11] A Qing Tan Sch. of Comput. & Inf. Syst., Athabasca Univ., Athabasca, AB, Canada, in a work by Pomeroy. "Effective SQL Injection Attack Reconstruction Using Network Recording" by Allen Pomeroy and Qing Tan suggests using network recording to find weaknesses in Web applications such as SQL injection attacks. The network packets including get and post requests of a web application are analysed using network forensic techniques and tools in this approach. This method employs a network-based Intrusion Detection System (IDS) to record suspicious application attacks on the network. [12] M. Ruse, T. Sarkar, and S. Basu authored an article. "Automatic Test Case Generation of Programs for Analysis and Detection of SQL Injection Vulnerabilities." To detect SQL Injection Vulnerabilities, this technique leverages

automatic test case generation. The fundamental idea behind this framework is to create a specialized model that automatically handles SQL queries. It also captures the interdependencies between the query's various components. The CREST(Automatic Test Generation Tool for C) testgenerator was used to determine the susceptible circumstances for the queries. The methodology is shown to be able to specifically identify the causal set and produce 85 percent and 69 percent reductions while trying on a few sample examples, according to the results.

IV. CONCLUSION

Many tactics are used to prevent SQL injection attacks at the application level, but no acceptable solution exists at this time. The most effective SQL injection prevention techniques were the focus of our project. An automated technique for preventing, identifying, and reporting SQL injection attacks in 'stored procedures' and 'defensive programming' has been devised based on our findings. We attempted to discuss the contemporary SQL Injection attack in our project, which is less well-known among the general public and many researchers. SQLIA's causes and effects were examined. The several types of SQLLA approaches were discovered and defined, as well as instances of the execution syntax. SQLIA's causes and effects were examined. The several types of SQLLA approaches were discovered and defined, as well as instances of the execution syntax. The most prevalent SQLIA detection and prevention approaches were also summarised. Finally, the effectiveness of the SQLIA prevention and detection approaches against the SQLIA kinds was assessed. However, the evaluation was solely based on prevention and detection, disregarding the needs for putting the strategy into practice.

V. REFERENCES

[1] Subhranil Som, Sapna Sinha, and Ritu Kataria, "Study of SQL Injection Attack: Mode, Detection, and Prevention," IEEE, 2017.

[2] Nenad Stojanovski, Marjan Gusev, Danilo Gligoroski, and Svein.J.Knapskog, "Data Execution Protection," IEEE, 2007.

[3] Navjot Kaur and Tejinderdeep Singh Kalsi (2015), "Detection And Prevention Of SQL Injection Attacks Using Novel Method In Web Applications," Kaur et al., International Journal of Advanced Engineering Technology E-ISSN 0976-3945.

[4] Neha Mishra and Sunita Gond, "Safeguard Against SQL Infusion Assaults," Universal Diary of Progressed Investigate in Computer and Communication Designing, Vol. 2, Issue 2, May 2014, pp. 37-42

[5] Nanhay Singh, Khushal Singh, and RamShringar Raw, "Analysis of Detection and Prevention of Various SQL Injection Attacks on Web Applications," International Journal of Applied Information Systems, 2012.

[6] Suraj Kothari, K. Wei, M. Muthuprasanna, M. Muthuprasanna, M. Muthuprasanna, M. Muthuprasanna, M. Muthuprasanna (Apr. 18th, 2006) "SQL Injection Attacks in Stored Procedures: Preventing SQL Injection Attacks." IEEE Symposium on Software Engineering http://ieeexplore.ieee.org (accessed November 2, 2007).

[7] R. Ezumalai, G. Aghila, Combinatorial Approach for Avoiding SQL Infusion Assaults. 2009 IEEE Universal Development Computing Conference (IACC 2009) Patiala, India, 6-7 March 2009

[8] Jeom-Goo Kim;Dept. of Comput.Sci., NamseoulUniv., Cheonan, South Korea"Injection Attack Discovery using the Junking of SQL Query Trait Values"in Information Science and Applications (ICISA), 2011 International Conference Issue Date 26-29 April 2011, On runner (s) 1-7.

[9] Parveen Sadotra (2015) " Mincing Fashion-SQL Injection Attack Discovery & Sonam Panda, Ramani (2013) " Protection of Web Operation against Sql Injection Attacks", International Journal of Modern Engineering Research (IJMER)www.ijmer.comVol. 3, Issue.1, Jan-Feb. 2013 pp-166-168 ISSN 2249-6645

[10] Atefeh Tajpour, Maslin Masrom, Mohammad Jojor Zadeh Shooshtari, Hossein Rouhanizeidanloo," Comparison of SQL Infusion Revelation and Avoidance Devices predicated on Assault Sort and Arrangement Conditions", Universal Diary of Progressions in Computing Innovation Volume 3, Number 7, August 2011.

[11] Pomeroy, A Qing Tan Sch. of Comput. &Inf.Syst.,AthabascaUniv., Athabasca, AB, Canada" Effective SQL Injection Attack Reconstruction Using Network Recording"in Computer and Information Technology (CIT), 2011 IEEE 11th International onference Issue DateAug. 31 2011-Sept. 2 2011 On runner (s) 552-556

[12] Ruse, T. Sarkar and S. Basu."Analysis & Disclosure of SQL Infusion Vulnerabilities by means of Programmed Test Case Era of Programs."10th Yearly Universal Symposium on Applications and the Internetpp.31–37