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

IDENTIFYING DATA LEAKSVIA SQL INJECTION

[1]Chintanica k, [2]Dhivya v, [3]Dharshini Akurathi, [4]Brinda P

[1,2,3]Student,[4] Assistant Professor
[1]Computer Science and Engineering
[1]Vel Tech High Tech Dr.Rangarajan Dr.Sakunthala Engineering College,Chennai,India

Abstract : Web-based components are now a common feature of many software systems. SQL injection is one of these attacks that has grown in frequency and severity. It allows attackers to have uncontrolled access to the databases that power Web applications. An attack that takes place in an application's database layer is called SQL injection. The vulnerability arises when user input is not strongly typed and is therefore unexpectedly executed, or when user input is erroneously filtered for string literal escape characters encoded in SQL statements. One of the most popular application layer attack methods available today is SQL Injection. Here is a POC(Proof of Concept) by Scanning Tool: Web Cruiser- Web SQL-injection attacks exploit weak validation of textual input used to build database queries..This report presents a “code reengineering” that implicitly protects the applications which are written in PHP from SQL injection attacks. It uses an original approach that combines static as well as dynamic analysis.In this report, I mentioned an automated technique for moving out SQL injection vulnerabilities from Java code by converting plain text inputs received from users into prepared statements.This project aims to make the database secure and avoid SQL injection when queries are injected into the database. The data is protected from SQL Injection assaults by means of the system's SQL Injection mechanism protection.

IndexTerms: SQL attack, Prevention, Confidentiality, progress.

A. Understanding SQL Injection

Before starting the project, it's important to understand how SQL injection works. SQL injection occurs when an attacker can insert or "inject" arbitrary SQL code into a query. This can lead to data leaks, unauthorized data manipulation, and even complete database compromise.

B. Project Goals

•The main goals of the SQL injection detection project should be:
•To detect potential SQL injection attempts.
•To alert administrators or block the malicious requests.
•To log details about the attempted attacks for further analysis.

C. Project Components

•Input Sanitization and Validation:

Ensure all inputs are sanitized and validated before being processed by thedatabase.Use parameterized queries or prepared statements to prevent SQL injection.

•Detection Mechanisms:

1. Signature-Based Detection: Identify patterns commonly used in SQL injection attack.

2. Anomaly-Based Detection: Monitor the application's normal behavior and flag deviations.

3. Heuristic Detection: Use rules and logical conditions to detect suspicious activity.

•Alerting and Logging:

Implement a logging system to record details of detected injection attempts.Set up alertmechanisms to notify administrators in real-time

D. Steps to Implement the Project **Step 1**: Setup the Environment

1.Choose a programming language (e.g., Python, Java, PHP).2.Set up a web server (e.g., Apache, Nginx).

3.Set up a database server (e.g., MySQL, PostgreSQL).

Step 2: Create a Web Application

1. Develop a simple web application with input fields (e.g., login forms, search boxes).2.Implement basic functionality to interact with the database.

Step 3: Implement Input Sanitization

1.Use libraries or frameworks that support parameterized queries. 2.Implement input validation to ensure data meets expected formats. **Step 4:** Develop Detection Algorithms

1.Maintain a list of SQL injection patterns. 2.Scan inputs and queries for these patterns **E. PURPOSE OF THE REPORT**

This report provides a summary of the project objectives, scope, and key outcomes. It helps stakeholders understand the purpose and importance of the project: Details the approach and methods used to detect and prevent SQL injections. This includes descriptions of the detection techniques (e.g., signature-based, anomaly-based, heuristic detection). It is a critical tool in maintaining the security and integrity of an organization's data and applications. It not only helps in identifying and mitigating current vulnerabilities but also in building a robust defense against future threats. The report helps identify vulnerabilities in the database or application that could be exploited through SQL injection or other forms of data leaks. It assesses the potential impact and likelihood of these vulnerabilities being exploited, allowing organizations to prioritize their mitigation efforts. Many industries have regulations and standards (e.g., GDPR, HIPAA, PCI-DSS) that require regular security assessments and reporting. A detailed report helps ensure compliance with these requirements.It provides an audit trail for regulatory bodies and internal governance, demonstrating that the organization is actively monitoring and addressing security risks. The report often includes recommendations for mitigating identified

vulnerabilities, which can guide security improvements.It highlights best practices for securing databases and applications against SQL injection and data leaks.

F. LITERATURE REVIEW

SQL injection (SQLi) is a prevalent attack vector that exploits vulnerabilities in an application's software to execute malicious SQL statements. Detecting and preventing SQL injection is critical to safeguarding sensitive data from unauthorized access and leaks. This literature review examines various techniques and methodologies proposed for detecting data leaks through SQL injection. WAFs like ModSecurity provide a layer of defense by filtering and monitoring HTTP requests to detect and block SQL injection attempts.Ensuring proper input validation and using parameterized queries (prepared statements) can significantly reduce the risk of SQL injection.Tools like static analyzers (e.g., SonarQube) and dynamic analysis tools (e.g., SQLMap) help in identifying potential vulnerabilities in the codebase. Attackers continually develop new evasion techniques to bypass detection systems, such as using encoded payloads, comment obfuscation, and time-based attacks.Balancing the detection accuracy to minimize false positives (benign queries flagged as malicious) and false negatives (malicious queries not detected) remains a significant challenge.Ensuring that detection systems can handle large volumes of traffic and complex queries without degrading performance is crucial for their practical deployment. Leveraging advanced AI and deep learning models to improve the accuracy and adaptability of SQL injection detection systems.Embedding security practices into the DevOps pipeline to ensure continuous detection and mitigation of SQL injection vulnerabilities throughout the software development lifecycle.Utilizing blockchain technology to create tamper proof logs of database activities, enhancing the traceability and accountability of detected incidentsDetecting data leaks through SQL injection is a multifaceted challenge that requires a combination of techniques, including signature-based, anomaly-based, and heuristic methods. Advances in machine learning and AI offer promising improvements in detection accuracy and adaptability. Continuous research and development are necessary to address evolving threats and enhance the security of database systems against SQL injection attacks.

G. METHODOLOGY

1. Input Validation and Sanitization :

Implement strong input validation and sanitization routines. Ensure that user inputs are checked and sanitizedto prevent malicious SQL queries from being executed.Software programs often contain multiple components thatact as subsystems wherein each component operates in one or more trusted domains. For example, one component may have access to the file system but lack access to the network, while another component has access to the network but lacks access to the file system. Distrustful decomposition andprivilege separation [Dougherty 2009] are examples of secure design patterns that reduce the amount of code that runs with special privileges by designing the system using mutually untrusting components.

2. Parameterized Queries:

Use parameterized queries or prepared statements in your code.This method helps separate SQL code from usersinput, preventing injection attacks.Parameterized queries let you traverse the graph from one vertex set to an adjacent set of vertices, again and again, performing computations along the way,with built-in parallel execution and handy aggregation operations. You can evenhave one query call anotherquery. But we'll start simple.

A GSQL parameterized query has three steps.

•Define your query in GSQL. This query will be added to the GSQL catalog.
•Install one or more queries in the catalog, generating a REST endpoint for each query.
•Run an installed query, supplying appropriate parameters, either as a GSQL command orby sending an HTTP request to the REST endpoint.

3. Web Application Firewall (WAF) :

A web application firewall (WAF) protects web applications from a variety of application layer attacks such as cross-site scripting (XSS), SQL injection,and cookie poisoning, among others. Attacks to apps are the leading cause of breaches—they are the gateway to your valuable data. With the right WAF in place,you can block the array of attacks that aim to exfiltrate that data by compromising your systems.Deploy a WAF that can analyze incoming traffic for suspicious patterns indicative of SQL injection attempts. WAFs can help block such attacks in real-time

4. Logging and Monitoring :

Set up comprehensive logging mechanisms to monitor and record database interactions and queries. Unusual or suspicious queries can then be flagged for further investigation.Logcheck is a package or tool to check system log files for security violations and unusual activity, it utilizes the program called logtail remembering the last position is read from the log file. Analyzes security or unusual activity from syslog to monitor Apache log files for errors caused by PHPScripts. Logcheck is used to detect problems automatically in logfiles and results are sent via e-mail.

5. Static and Dynamic Analysis :

Use static analysis tools to scan your codebase for potential vulnerabilities. Dynamic analysis involves actively testing your application for vulnerabilities, including SQL injection, using tools or pen-testing techniques. Static analysis involves examining the source code, binaries, or other software artifacts without executing the program. The goal is to identify potential security vulnerabilities, coding errors, or design flaws. Dynamic analysis involves running the software in a controlled environment and monitoring its behavior. The goal is to identify vulnerabilities that may not be apparent through static analysis and to understand how the application responds to various inputs.

H. Recovery Plan :

Have a robust incident response plan in place. In case of asuccessful attack, know how to mitigate the leak, contain the damage, and recover swiftly.

Example: Unauthorized access to a system, malware infection, data breach, etc.

1. Identification:

•Determine the nature and extent of the security incident.
•Use tools like log analysis, intrusion detection systems, and monitoring solutions.

2. Containment:

•Isolate affected systems to prevent further damage.
•Disconnect compromised systems from the network.

3. Eradication:

•Remove the source of the security incident.
•Patch vulnerabilities, remove malware, or close security loopholes.

4. Recovery:

•Restore affected systems from clean backups.
•Implement additional security measures to prevent future incidents.

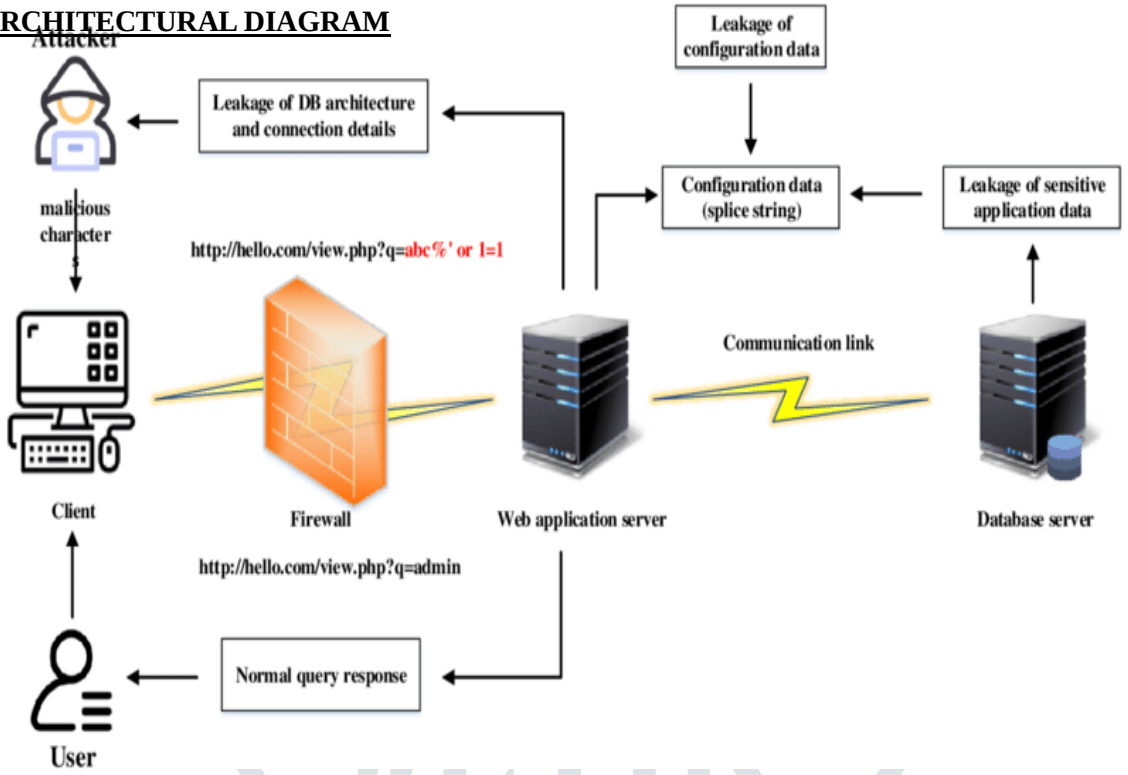
5. Communication:

•Notify relevant stakeholders, including management, legal, and affected parties.
•Coordinate with law enforcement if necessary.

6. Analysis and Lessons Learned:

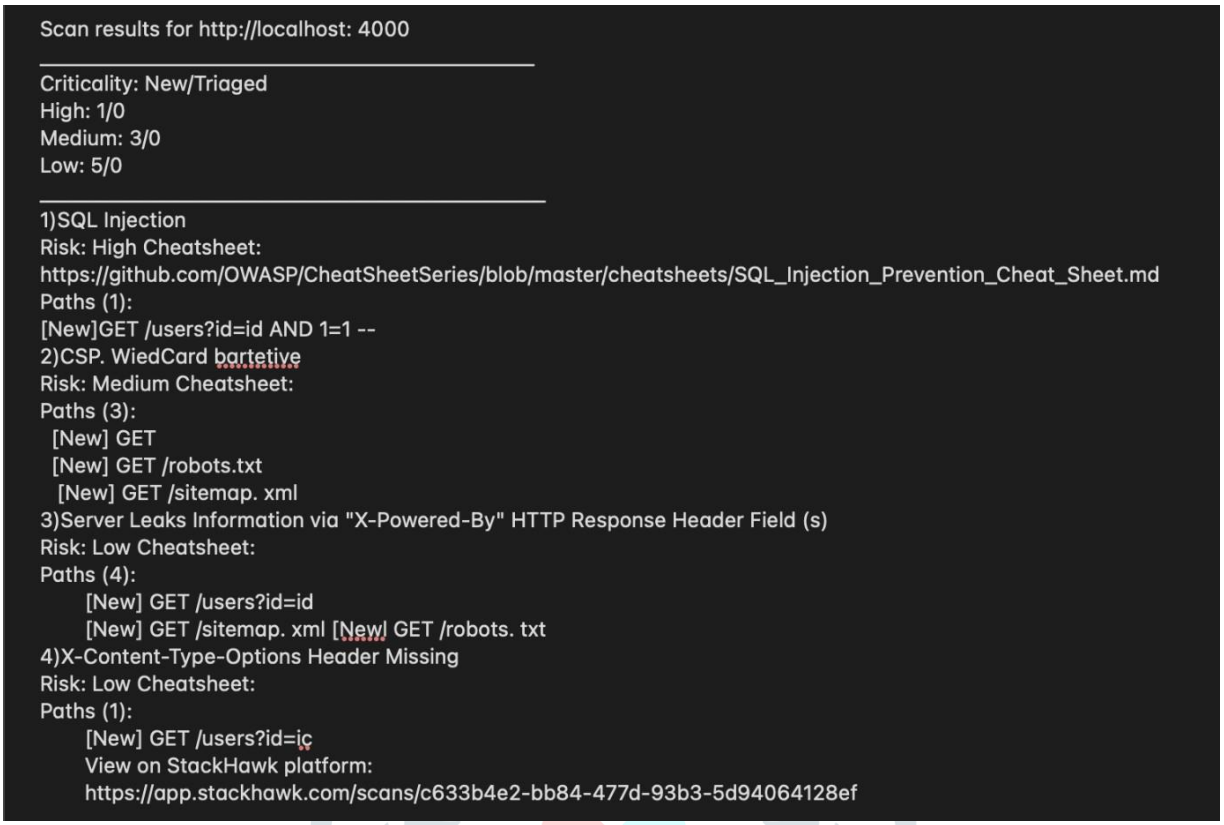
•Conduct a post-incident analysis to understand how the incident occurred.
•Implement measures to prevent similar incidents in the future.

I.ARCHITECTURAL DIAGRAM



It is not compulsory for an attacker to visit the web pages using a browser to find if SQL injection is possible on the site. Generally attackers build a web crawler to collect all URLs available on each and every web page of the site. Web crawler is also used to insert illegal characters into the query string of a URL and check for any error result sent by the server. If the server sends any error message as a result, it is a strong positive indication that the illegal special meta character will pass as a part of the SQL query, and hence the site is open to SQL Injection attack. For example Microsoft Internet Information Server by default shows an ODBC error message if an any meta character or an unescaped single quote is passed to SQL Server. The Web crawler only searches the response text for the ODBC messages.

J. OUTPUT IMAGE



K. CONCLUSION

Detecting data leaks caused by SQL injection is a crucial aspect of cyber security efforts. By implementing robust testing methodologies,employing various detection tools, and maintaining a proactive stance,organizations can significantly reduce the risk of sensitive data exposure due to SQL injection vulnerabilities.The process involves a multi-layered approach,including settingup controlled test environments, generating effective injection techniques, utilizing monitoring tools, implementing logging and alerting mechanisms, analyzing network traffic, conducting penetration tests, and having a well-defined incident response plan.while there are challenges and considerations in detecting data leaks caused by SQL injection, the proactive identification of vulnerabilities andsubsequent mitigation efforts greatly enhance an organization's security posture. A balanced approach that combines robust detection measures with efficient incident response planning is key to effectively safeguarding againstSQLinjection-related data leaks.

L. REFERENCES

[1] Johny JHB, Nordin WAFB, Lahapi NMB, Leau YB. SQL Injection prevention in web application: a review. In: Com_x0002_munications in computer and information science, vol. 1487 CCIS, no. January. 2021. p. 568–585.https://doi.org/10.1007/978-981-16-8059-5_35.

[2] Alghawazi M, Alghazzawi D, Alarif S. Detection of sql injection attack using machine learning techniques: a systematic literature review. J Cybersecur Privacy. 2022;2(4):764–77.

[3] Han S, Xie M, Chen HH, Ling Y. Intrusion detection in cyber-physical systems techniques and challenges. IEEE Syst J.

2014;8(4):1052–62.

[4] Dasmohapatra S, Priyadarshini SBB. A comprehensive study on SQL injection attacks, their mode, detection and prevention. 2021. p. 617–632.://doi.org/10.1007/978-981-16-3346-1_50.
http://ieeexplore.ieee.org

[5] Hu J, Zhao W, Cui Y. A survey on SQL injection attacks, detection, and prevention. In: ACM international conference on proceeding series,noJune.2020.p.483 https://doi.org/10.1145/3383972.3384028.

[6] Wei, K., Muthuprasanna, M., & Suraj Kothari. (2006, April 18). Preventing SQL injection attacks in stored procedures. Software Engineering IEEE Conference. Retrieved November 2, 2007, from http://ieeexplore.ieee.org

[7] Thomas, Stephen, Williams, & Laurie. (2007, May 20). Using Automated Fix Generation to Secure SQL Statements. Software Engineering for Secure System IEEE CNF. Retrieved November 6, 2007, from http://ieeexplore.ieee.org

[8] Merlo, Ettore, Letarte, Dominic, Antoniol & Giuliano. (2007 March 21).Automated Protection of PHP Applications Against SQL-injection Attacks.Software Maintenance and Reengineering, 11th
European Conference IEEE CNF.Retrieved November 9, 2007,from http://ieeexplore.ieee.org.

[9] Wassermann Gary, Zhendong Su. (2007, June). Sound and precise analysis of web applications for injection vulnerabilities. ACM SIGPLAN conference on Programming language design and implementation PLDI, 42 (6). Retrieved November 7, 2007, from http://portal.acm.org

[10]Friedl's Steve Unixwiz.net Tech Tips. (2007). SQL Injection Attacks by Example. Retrieved November 1, 2007, from http://www.unixwiz.net/techtips/sqlinjection.html

[11]Massachusetts Institute of Technology. Web Application S ecurity MIT Security Camp. Retrieved November 1, 2007, from http://web.mit.edu/netsecurity/Camp/2003/clambert-slides.pdf 61

[12]Massachusetts Institute of Technology. Web Application Security MIT Security Camp. Retrieved November 1, 2007, from http://groups.csmail.mit.edu/pag/readinggroup/wasserman07injection.pdf

[13]Gregory T. Buehrer, Bruce W. Weide, and Paolo A. G. Sivilotti. The Ohio State University Columbus, OH 43210 Using Parse Tree Validation to Prevent SQL Injection Attacks. RetrievedJanuary 2005, from http://portal.acm.org

[14]Zhendong Su, Gary Wassermann. University of California, Davis. The Essence of Command Injection Attacks inWeb Applications. Retrieved January 11, 2006, from http://portal.acm.org

[15]William G.J. Halfond, Alessandro Orso, and Panagiotis Manolios College of Computing – Georgia Institute of Technology. Using Positive Tainting and SyntaxAware Evaluation to CounterSQL