Database Security

Areeb Khemani
x2023flc@stfx.ca

Jayaprakash
x2023dvm@stfx.ca

April 08, 2024

Table of Contents

1 Introduction

Databases are backbone of almost all web applications, database security is crucial not only to protect sensitive data but also to ensure the integrity and availability of services. Among the cybersecurity threats SQL injections attacks and data breaches stand out because of the simplicity and potential devastating impact, SQL injection threat usually follows with a Data Breach where data is exposed. This data breach can occur through many ways not just limited to SQL injection attacks.

In SQL injection attacks the attackers focus on the Structured query language (SQL) queries, specifically on the queries that is passed to the database application. This happens when the application developed won't check and clean the data entered by the user before processing it, this allows attackers to pass malicious SQL code into the queries. The issue is not limited to web applications, any software the takes the user input and then use that data to create SQL queries could be at risk. This includes desktop, mobile apps and the latest HTML5 applications that interact directly with database from client side.

According to Open Web Application Security Project (OWASP), it lists SQL injection as one of the most serious security vulnerability as it threatens web security. One of the responsibilities of OWASP is to ensure and make aware about the security vulnerabilities and people who are responsible for web development and web architecture development must be aware of this.

2 Literature Review

The article titled "Exploring Fully Homomorphic Encryption's Potential" delves into Fully Homomorphic Encryption (FHE), a cryptographic breakthrough enabling computations on encrypted data, ensuring data security throughout processing. This technology promises to revolutionize data privacy and secure computing, potentially changing how sensitive data is processed and protected against security risks. Central to modern FHE schemes is the Learning with Errors (LWE) problem, which introduces noise to safeguard the data, albeit posing the challenge of noise management to prevent decrypted results' corruption post-computation.

A crucial innovation in FHE is bootstrapping, a technique reducing noise associated with homomorphic operations, enabling extensive computations without data corruption. FHE's evolution progresses from slow and impractical schemes to recent developments improving homomorphic computations' efficiency, addressing the computational overhead hindering FHE's practicality.

The article emphasizes hardware acceleration's critical role in overcoming computational challenges inherent in FHE, with initiatives focusing on dedicated FHE accelerators. Additionally, the development of libraries and compilers simplifying FHE implementation aims to democratize access to this technology, easing integration into various applications.

Despite advancements, challenges remain for widespread FHE adoption. Computational overhead, though reduced, remains significant, necessitating further research into optimization strategies and hardware solutions. FHE schemes' complexity and encryption parameter management pose usability challenges, complicating integration into practical applications. Unique requirements for FHE integration into application design, notably a data-independent computation model, also affect programming paradigms.

The absence of standardized APIs and benchmarks complicates comparison and evaluation of FHE schemes and implementations, underscoring the need for collaborative efforts to establish best practices and benchmarks across academia, industry, and standardization bodies.

In conclusion, the article envisions an optimistic future for FHE, foreseeing its pivotal role in enhancing data security through computational privacy. However, realizing this potential relies on collective efforts to navigate technical challenges constraining FHE's usability and performance. As investments and innovations progress, FHE stands poised to redefine data privacy and secure computation paradigms.

The research paper, "Detection of SQL Injection Attacks Using the String Match Approach" authored by A. Kopp, delves into the realm of cybersecurity, particularly focusing on identifying and preventing SQL Injection Attacks (SQLIA) through a novel string match approach. Beginning with an overview of SQL Injection Attacks amidst the broader context of web application security risks, the paper emphasizes their recognition as significant threats by entities like the Open Web Application Security Project (OWASP). It underscores the simplicity with which attackers exploit vulnerabilities to execute unauthorized actions on databases, such as circumventing authentication measures or accessing sensitive data. The study aims to combat this prevalent threat by evaluating the effectiveness of the string match technique in detecting SQLIA within HTTP traffic, pinpointing specific special characters indicative of an attack.

Kopp's methodology entails scrutinizing recent research on SQLIA and focusing on prevalent attack types. Employing a string match approach, previously explored in related studies, the research scrutinizes the impact of special characters on SQL injection detection. Through HTTP traffic analysis, the study aims to identify a subset of characters significantly influencing attack probability, utilizing these findings to develop a regular expression for malicious SQL code detection. Multiple null hypotheses are examined to assess the relationship between specific special characters and SQL Injection Attacks.

The study employs the "HttpParamsDataset," encompassing both benign and suspicious records, to validate the formulated hypotheses. This dataset enables scrutiny of HTTP request parameters for various special characters. Logistic regression analysis reveals that while most special characters exhibit a strong correlation with SQLIA, "concatenate" characters do not, leading to the acceptance of one null hypothesis and the rejection of others. In essence, the paper tackles a critical vulnerability in web application security by leveraging the string match approach for SQL Injection Attack detection. Its discoveries not only contribute to the existing cybersecurity knowledge but also provide practical guidance for developers and security experts aiming to fortify defences against SQLIAs.

3 SQL injection detailed analysis

The most straightforward form of SQL injection involves where attackers place malicious code into parameters, the application uses this parameters to construct SQL queries, here the attacker input becomes part of the final SQL query and gets executed in the database, if the queries are not properly sanitized this may lead to unauthorized access or damage. The indirect form of SQL injection occurs when malicious code is inserted into strings that are saved into database, for instance a part of a table or metadata and later this stored strings might be concatenated into dynamic SQL commands by the application. If the application fails to sanitize these strings then the malicious code gets executed. This type of attacks is not immediate but happen eventually when the stored data gets executed.

3.1 Risks of insufficient sanitization

Even when the applications use techniques like parameterization, which is a technique designed to prevent SQL injection, it basically separates the logic and data. Parameterization is effective when used correctly.

3.2 Execution Context and permissions

When the SQL injection is successful, the malicious SQL is execute with the same permissions as the application making the SQL request. Its very dangerous because consider the scenario where the application operates with a high-level privilege, this will allow attackers to access and modify the sensitive information. This highlights the need for applications with least privilege which ensures operation with least or minimum necessary permissions.

4 Data Breaches Detailed analysis

Data breaches have emerged as one of the most critical concerns of the digital age, highlighting the susceptibility of personal and proprietary information to sophisticated cyber threats. These occurrences, which involve illegal access to or disclosure of data, can be caused by a range of factors, including foreign cyber-attacks and internal data misuse. Such breaches have far-reaching consequences for individuals, organizations, and government entities alike.

Human error is one of the main reasons for data breaches. This might involve a variety of errors, such as delivering private information to the wrong people or using inappropriate data storage techniques that expose confidential information. On the other hand, cyberattacks offer a more malevolent channel by which breaches transpire. Cybercriminals frequently employ phishing, which involves tricking someone into divulging personal information, malware infections that infiltrate computers, and direct attacks that take advantage of software flaws. Insider threats are a serious concern as well since workers who have access to confidential information may be used, intentionally or unintentionally, as conduits for its unapproved distribution. Moreover, there is always a real chance of devices carrying sensitive data being physically stolen or lost, which emphasizes the significance of both digital and physical security measures.

Physical breaches are an important aspect for data security. These events involve the actual loss or theft of devices like laptops, external drives, or paper records containing sensitive information. What makes physical breaches particularly concerning is their ability to bypass digital security measures entirely, putting data directly into the hands of unauthorized individuals. To address these risks, organizations must implement secure storage protocols, limit access to sensitive data, and establish tracking mechanisms for physical assets.

Transitioning from the physical realm to the digital sphere, cyberattacks encompass a wide array of methods employed by malicious actors. Of concern are malware infections, which exploit software vulnerabilities to compromise systems. The sophistication of cyberattacks is evident in techniques such as phishing scams, which deceive individuals into revealing confidential information, and SQL injection attacks targeting websites reliant on databases. Additionally, while the primary aim of Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks is to overwhelm systems, they can also facilitate breaches by exploiting resulting instability. Defending against these digital threats requires a comprehensive security approach, including the implementation of firewalls, anti-malware tools, and employee training programs.

Insider threats, which arise from within an organization, introduce an additional layer of intricacy to data security. Whether driven by malicious intent or inadvertent errors, these breaches highlight the risks posed by individuals who have authorized access to sensitive information. Effectively addressing insider threats requires implementing rigorous access controls, maintaining vigilant monitoring for suspicious behavior, and cultivating a positive corporate culture that mitigates the potential for internal dissatisfaction. Finally, ransomware attacks exemplify the convergence of cybercrime and data breach vulnerabilities. In these attacks, perpetrators not only encrypt data but also issue threats of releasing it unless a ransom is paid. To proactively defend against ransomware, organizations should implement regular data backups, promptly update software, and foster employees' awareness of phishing techniques.

Data breaches have far-reaching effects that go well beyond the original illegal access. One of the most obvious consequences is financial; corporations risk losing money, having to pay for mitigating the breach, paying legal fees, and paying hefty fines for breaking regulations. A data breach can cause disastrous reputational harm in addition to financial harm. A drop in business may result from a loss of client trust and confidence, and partnerships may be threatened or ended. Individuals may experience long-term financial and legal difficulties due to identity theft brought on by the theft of personal information. Operational disruptions can also happen, especially if there is a compromise of essential systems, which can result in lost time and additional expenses.

A multifaceted strategy for detection and prevention is required to counter the threat posed by data breaches. This includes thorough security training for staff members to reduce human error and routine audits and monitoring of systems to find anomalies suggestive of a breach. Information can be safeguarded even in the event of illegal access by using encryption for both data in transit and at rest. Vulnerabilities can be fixed by keeping software and systems updated through frequent updates. Breach scenarios could still happen despite these precautions; hence a strong reaction and recovery strategy are required.

This plan should include a clear communication strategy for informing impacted parties in accordance with legal requirements, as well as procedures for containment, eradication of the threat, and recovery of operations. Strengthening future security measures after a breach requires a detailed review to determine how the incident happened and an assessment of how well the response worked. An organization's data protection strategies are significantly shaped by legal and regulatory factors as well. Adhering to applicable statutes and guidelines, including the CCPA and GDPR, is crucial to evade legal ramifications and cultivate credibility by showcasing a dedication to safeguarding personal information.

5 Real world examples and case studies

Over the years there were numerous incidents of attackers implementing SQL injection attacks leading to a security breach and some of them are:

• In February of 2009, high profit web sites during that time like Kaspersky, CNET.com, RBS WorldPay, BT.com and several other were hacked by Romanian hackers with the use of SQL injection attacks.

• In August 17, 2009 companies like credit card processor Heartland Payment Systems, Convenience Store chain 7-Eleven, supermarket chain by Hannaford by Hannaford Brothers were effected by SQL Injection attacks committed by an American citizen and two Russians. The theft was 130 million credit card numbers.

• In April 2011, a web site which belongs to Barracuda Networks has gone through SQL injection and the hacker was able to publish the authentication credentials and passwords online.

• The Equifax Data Breach, One of the largest credit reporting agencies, experienced a massive data breach in 2017. Personal information of approximately 147 million people was exposed. Attackers exploited a vulnerability in the Apache Struts web application framework. The exposed data included Social Security numbers, birth dates, addresses, and, in some instances, driver's license numbers. Equifax had to pay around 575 Million in settlement amount.

• In 2016, a wave of ransom attacks targeted thousands of MongoDB databases that were exposed online without proper security measures. Many businesses and individuals lost valuable data, with some paying the ransom without any guarantee of data return. The incidents highlighted the critical importance of basic database security practices, such as enabling authentication, using strong passwords, and limiting network access.

• In 2014, Sony Entertainment suffered a data breach that led to the leak of a vast amount of confidential data. Personal information about employees, email exchanges, and unreleased movies, causing substantial financial and reputational damage to Sony. Sony faced legal action from employees, criticism for its security practices, and a temporary halt in business operations.

All that the attackers need to have to takeover the application is just one SQL Injection Vulnerability. These attackers take over poorly designed web applications and take over them to execute unauthorized SQL commands leading to unauthorized access, data theft as we see above in some cases. Recent trend warns us regarding the automated attacks and additionally with the rise of cloud services and APIs there have been new versions of SQL injection attacks, this shows about the causality to secure web application.

6 Defense Mechanisms and Best Practices

In order to develop a application with good defense mechanism and to eliminate the threat of SQL injection one need to implement defences at the code level. So we need to concentrate on the potential causes including getting user input. The below strategies are not implemented alone but they will be a part of a strategy to enhance database security.

6.1 Domain Driven Security

DDD is a software design approach, here this methodology that give a big picture of the areas that we are going to work on, we will use that understanding to make things fit together. This also makes sure that everyone in the team understands the concepts so that it will make it easier to work together and make software safe.

The bad inputs are like puzzle pieces that are trying to fit together and DDS will make sure that the bad pieces will not even enter the puzzle, this is done by making sure that the pieces that don't fit are not accepted.

- Consider this example of user Domain.

"SELECT * FROM users WHERE username = 'userinput';"

Here the 'userinput' is a placeholder, it's a potential security issue because if the place holder is "a' OR '1'='1", this could return all the users if proper care is not given while software development. The attacker can gain access to user information, this can occur to bank details, emails etc. as attackers can alter data or inject malicious code into the database.

When it comes to application design and development, DDS encourages security consideration for all stages of development this ensures security integration for building a secure application. DDS is also about collaborating between teams such as development teams, security teams, and domain experts. This encourages level of understanding and implementation across all levels of the application from database to the user interface.

6.2 Using Parameterized Statements

If the application is building dynamic SQL queries ie. by stitching strings together then there is huge risk for malicious queries, to prevent the SQL injection attacks this method uses parameterized statements, it is similar to filling the blanks in a statement and the application will only accept specific type of words. In terms of SQL queries instead of building a query by directly inserting user input into the string, it uses a template with placeholders.

Parameterized Statements deals with the user input in SQL queries. As already mentioned it separates SQL logic from the data, this effectively reduces the risk that user input will be interpreted as SQL code. This type of isolation is achieved by using placeholders for data values in the query.

6.2.1 Parameterized Statements Working

The query is written with a placeholder often represented as a ?, here the data values are inserted. The query structure is sent to the database but at this stage the database knows the structure of the query but not the value of the parameters.

When the application passes the values that should be used in the query, the database interface uses secure methods to bind them to the placeholder. The bound values are used to execute the prepared statement.

Here the primary benefit is advanced security as we are treating user input as data rather than part of the sql query. It also makes the code look cleaner and its to maintain as it increases the readability of the program. This method when combined with the other security practices like input validation and error handling, it forms a foundation of secure application development regarding database interactions.

6.3 Input Validation

This is the process of checking the input against the standards set in the application. It is similar to checking the ID's of everyone who enters a place. This is the first line of defense against a wide range of attacks including SQL injection etc. By ensuring only properly formatted data is allowed to enter the system, input validation helps maintain data integrity and prevent attack on the input data. Input Validation should occur very close to the point of entry of the input as possible, this involves validation user inputs at the application's frontend and backend levels.

Inputs should be validated based on the context in which they are used. For instance a piece of data that is safe to include in a URL might not be safe to include in a SQL query. For complex inputs that serve specific business functions, custom validation rules might be necessary, these rules can suit a unique business logic.

6.3.1 Types of Input Validation

Whitelist Validation:

It is a practice where inputs that are known good are accepted, here we are validating the data. Here we define a strict set of criteria that inputs must follow to gets accepted. This approach is considered more safe than the blacklist validation because we are explicitly allowing known safe inputs.

Data Type: This ensures that the input matches the expected data type, such as integer, string or date.

Format: Considering validating special numbers or strings like phone number, email address, social security number etc. they hold a unique format, we can check these type of data fields and ensure only valid data is passed rather than any malicious code.

Range: Ensuring the value fall within a given range like consider salary for instance which always has a limited range.

Length: Checking the length of the input is within a given range to prevent injection and buffer overflow attacks.

Blacklist Validation:

This is also called as deny list, it involves specifying a list of unacceptable inputs or patters in general and reject any input that matches the criteria. This method can be used to filter out dangerous characters or patterns that could lead to security vulnerabilities.

However, blacklist validation is inherently less secure than whitelist validation because its impossible to predict every possible malicious input.

6.4 Encoding Output

Output Encoding is primarily a defense against Cross-site Scripting (XSS) attack, XSS attacks occur when an attackers injects malicious scripts into content that is served to other users, to prevent this user supplied input that will be rendered in a web page should be encoded for example characters like '<' and '>' should be encoded to '&lt;' and '&gt;' respectively.

The type of encoding applied hugely depends on the content for instance consider where the user input is displayed like HTML body, CSS, URL etc. In applications where data is passed between various modules or services in such places encoding can play a crucial role.

In such applications services communicate by passing objects, so in service to service to service communication to prevent security vulnerabilities the data should be properly encoded. Encoding output is a critical security measure for preventing XSS attacks by ensuring that user input data is not executed as part of the HTML or JavaScript. Encoding output may not be a direct method for preventing SQL injection but the underlying principles of treating data safely is fundamental to secure software development practices.

7 New Trends in Database Security

7.1 AI-driven Approach

Here we will focus on advanced approach that combines Artificial Intelligence (AI) and Natural Language Processing(NLP) to address SQL injection attacks.

7.1.1 Data Collection

A cyber security dataset developed by references from real world attacks is chosen. An ideal dataset consists of diverse attack vector ranges which addresses class imbalances because there should always be a balance between the successful and unsuccessful attacks.

7.1.2 Preprocessing

Here NLP techniques are used to convert text input data which is suitable for AI prediction for threat analysis. Machine learning and NLP methods are applied if needed to identify attacks patterns etc. such as tokenization which involves breaking down text into individual elements or tokens, eliminating stop words or common words that add no value to pattern recognition.

7.1.3 Model Prediction

A logistic regression model is used to predict the possibility of SQL injection attacks, mainly logistic regression is chosen because of its ability to estimate binary classification tasks, here the probability of the input features is predicted.

7.1.4 Handling Class Imbalance

Techniques such as oversampling and under sampling are utilized to make the model non biased towards most frequent occurring class, mainly feature selection methods are applied to make the model focus more on the relevant features.

7.1.5 Limitations of AI driven approach

Dataset Quality

The success of the model prediction greatly depends on the dataset quality it gets trained on, if the data is inaccurate, biased or incomplete this will effect the models prediction. The difference between training data and real-world attack pattern can effect the models reliability, to maintain and Improve models effectiveness the training dataset should be regularly updated with new and diverse examples of SQL injection attacks.

Preprocessing with NLP techniques

Here the NLPs role in transforming textual data into a format suitable for AI analysis is challenging. The effectiveness of the tokenization, stop word removal etc depends heavily on the algorithms used and their alignment with data. Dynamic feature selection can enhance the models ability to detect emerging SQL injection patterns, ML techniques that automatically adjust features based on the new data are very much useful in the case.

7.2 Privacy Enhancement Technology(PETs)

These are basically tools and methods designed to help individuals and organizations protect personal and sensitive data from unauthorized access. PETs provide a broad range of technologies that include advanced encryption, access control and authentication and all these play a crucial role in enhancing privacy and security. For instance OCTA provides to PETs through several ways.

7.2.1 Advanced Encryption

It is a method that uses algorithms to transform data into a coded form which can be decoded by someone who has a decryption key, OKTA ensures that data transmitted between users and application is encrypted using strong procols such as TLS

7.2.2 Access Control

It is a security technique which restricts unauthorized users from accessing certain data, OKTA provides robust access control mechanisms and access management solutions through role based access control(RBAC), condition-based access policies etc

7.2.3 Authentication

It is a process of verifying the identity of the users through credentials like password, biometric data, security tokens etc . Organizations can enhance their security standards through OCTA as it provides Multi factor authentication(MFA), Single sign on(SSO) etc .

8 Conclusion

Database security is a fundamental component of information technology system architecture that protects the availability, confidentiality, and integrity of data. As we've explored the different aspects of database security, it's become clear that the area is defined by the dynamic interaction between new threats and developing defenses. We start our adventure with a thorough examination of SQL injection, a common danger with significant ramifications.

SQL injection, which takes advantage of flaws in the way apps communicate with databases, is a significant threat to database security. We've carefully examined SQL injection attacks to reveal their mechanics and how they can be used to get around security protocols, change database searches, and access data that isn't authorized. This analysis emphasizes how important it is to be vigilant and use advanced defence techniques to prevent SQL injection.

The discussion on data breaches has brought light to the complex nature of this threat, which not only external cyberattacks but also internal vulnerabilities and human errors. Our detailed analysis has revealed the profound impact of data breaches on organizations and individuals alike, manifesting in financial losses, reputational damage, and legal ramifications. This examination highlights the importance of a comprehensive approach to security, addressing both technical and human factors.

In response to these challenges, we've dived into an array of defence mechanisms and best practices designed to fortify database security. Paramount among these is the emphasis on domain-driven security and the utilization of parameterized statements—practices that, when effectively implemented, can significantly mitigate the risk of SQL injection and other threats. These strategies represent foundational elements in the development of secure applications and systems.

Domain-driven security has emerged as a pivotal concept in our discourse, advocating for a holistic approach to security that integrates domain knowledge into the design and implementation of systems. By aligning security measures with the specific requirements and constraints of the domain, organizations can achieve a more robust and context-aware defence posture.

The adoption of parameterized statements stands out as a critical technical measure in preventing SQL injection attacks. By segregating data from code, parameterized statements ensure that user input is treated strictly as data, thus neutralizing a common vector for SQL injection. This practice exemplifies the kind of specific, actionable measure that can have a profound impact on database security.

Looking to the future, we've explored emerging trends that promise to reshape the landscape of database security. Advances in artificial intelligence, machine learning, and encryption technologies such as Fully Homomorphic Encryption (FHE) are at the forefront of these developments, offering new tools and methodologies for securing databases against increasingly sophisticated threats. These innovations hold the potential not only to enhance security but also to transform the way we manage and interact with data.

In conclusion, the field of database security is characterized by a continuous arms race between threat actors and defenders. The complexity and severity of threats like SQL injection and data breaches necessitate a multifaceted and dynamic approach to security, combining technical measures, domain knowledge, and ongoing vigilance. As we move forward, embracing new technologies and best practices will be crucial in safeguarding our digital assets. The journey toward more secure databases is ongoing, and it is incumbent upon all stakeholders to remain engaged, informed, and proactive in the face of evolving challenges.

References

[1] S. Gorantala, R. Springer, and B. Gipson, "Unlocking the Potential of Fully Homomorphic Encryption," *Communications of the ACM*, vol. 66, no. 5, pp. 72–81, 2023.

[2] M. Alghawazi, D. Alghazzawi, and S. Alarifi, "Detection of SQL Injection Attack Using Machine Learning Techniques: A Systematic Literature Review," *King Abdulaziz University*, Jeddah, Saudi Arabia, 2023.

[3] J. R. Khan, S. A. Farooqui, and A. A. Siddiqui, "A Survey on SQL Injection Attacks Types & their Prevention Techniques," *Hamdard University*, Karachi, Pakistan.

[4] N. Mohamed, "Securing transportation web applications: An AI-driven approach to detect and mitigate SQL injection attacks," 2023.

[5] J. Clarke, *SQL Injection Attacks and Defense*, Second Edition.