Prevention of Data Leakage via SQL Injection

Shreya Chowdhury[1], Miran Ahmad[2], Aakansh Nandi[3], Aadish Jain[4], and Prof. Mohandas Pawar[5]
[1,2,3,4]*UG Student, MIT ADT University, Pune*
[5]*Asst. Prof., MIT ADT University, Pune*

Abstract **- This project aims to prevent SQL injection while performing a query. It does so by implementing a secure and online method to store and protect all the sensitive data stored in the database. Another key component of this system is encryption of card data. This method is known as AES encryption. It works seamlessly online and can be accessed from any location. This framework uses encryption techniques to prevent unauthorized access to the database. It also secures the user data by storing it in a secure form.**

Index Terms **- AES, DES, SQL Injection, Database, Decryption, Ciphertext.**

I.INTRODUCTION

The goal of this project is to implement an encrypted website that enables users to shop for goods and services online. This website will not store any confidential information about the user. The project is designed to protect sensitive information and transactions from being revealed. This method is used to secure electronic cash exchange transactions. The project aims to protect the privacy of electronic transactions. Its key component is the Advanced Encryption Standard (AES).

Transactions with the use of encryption techniques such as AES would be more secure and less prone to fraud. The Advanced Encryption Standard or AES is a widely used encryption algorithm. It is a three-block cipher that can be used to secure data. The Advanced Encryption Standard (AES) is a widely used symmetric encryption algorithm. AES is a viral and widely used algorithm for encrypting confidential data in software and hardware. AES is a family of three block ciphers. AES-128 uses a 128-bit key length to encrypt the messages.

One hundred twenty-eight bits of information can be encrypted, and 256 bits of information can be decrypted with 3 keys of 128, 192, and 256 bits. The asymmetric key, also known as an asymmetric cipher, encrypts and decrypts messages using the same key. There are ten rounds of the 128-bit keys, twelve rounds of the 192-bit keys and fourteen rounds of the 256-bit keys. A round consists of several operations: change of one character, change of another, and rearrangement of letters.

The AES was chosen because of its security and the fact that it is simple to implement. SQLi is also an efficient way to attack databases. It will allow hackers to add malicious code to existing SQL query without the user knowing it. The data will be open to the malicious hackers since any website or web application that uses a SQL database can get attacked by a SQL Injection vulnerability. Hackers may misuse and exploit the confidential information stored on this device. Due to SQL injection vulnerability attacks, such problems can still happen.

The contribution of the paper is as follows;

The purpose of this project is to provide a safe transaction for the users. Both the transaction and the user data can be encrypted using the AES encryption technique. This system encrypts the user's log in details to preserve privacy of the website's clients. The device encrypts the transaction number and the bank's PIN. It is less dangerous when getting hacked because of the SQL Injection Prevention technologies used.

II. LITERATURE SURVEY

To include an exhaustive examination of SQL injection, we studied papers from various journals, conferences, and acquired data. The following is how the various papers are organized:

They were the first to describe web application Command Injection attacks in a methodical way. They created and developed a suitable runtime filtering approach [1] using the concept of Command Injection Attack. It allows programs to protect themselves from SQL injection attacks. Static analysis, dynamic

analysis, and intelligent code re-engineering are key components of the security testing process for protecting existing properties [2].

A new method of detecting SQL injection sites. Runtime testing for potentially exploitable vulnerabilities are part of their strategy, as is later application code review to ensure protection. By altering the original SQL statement, the vulnerability attack established its own targets [3]. They proposed a mechanism for ensuring that dynamic SQL queries are accurate. For Java programmes, based on text analysis algorithms. Another technique, which has a low probability of false-positive predictions, has discovered problems in these applications [4].

The first computer-aware fault-localization tool that takes into consideration the interaction between an application and its data. According to a review of studies completed on three database applications, present methods of locating are insufficient [5].

They discussed numerous methods for detecting vulnerabilities and combating SQL injection attacks [6]. The characteristics of SQL injection were established, as well as the strategies for preventing it. It can be avoided by employing proper input validation and type-safe SQL parameters. By avoiding SQL injections, a web app was constructed to keep SQL injection threats at bay [7].

The Insensitive Remote Code Execution attack security approach was discovered, which protects against all types of attacks by providing a unique execution environment for the running function. Within this environment, the software's calling convention has no bearing. This is true for machine code as well as interpretive code [8]. Proposed a two-stage approach: the first tokenizes the input query statement, and the second decodes the tokens to determine the expression [9].

There are two parts to a hybrid technique: runtime analysis and static analysis. To increase performance quality, the first step is to review the algorithm in real time. These findings should alert the developer of any potential flaws, and if any such errors exist, they must be discovered and corrected. The software then does a static analysis, comparing new SQL queries to those that have been scheduled and validated as safe [10].

Four different detecting approaches were examined. To evaluate the risks of SQLIAs, researchers must first characterise all of the distinct types of SQLIAs that have been detected thus far. The researchers next looked for efficient techniques to both detect and avert attacks. This study looked at how SQLIAs could be integrated into an application and various approaches could be used to do so [11].

A review of how SQLIA injection works was conducted. It can also alert us to any mishaps in which SQLi can be used. This study suggests a strategy for preventing hacker attacks on the device. The hash function technique is proposed in this paper, and it can be used to mitigate SQL injection vulnerabilities [12].

I proposed a method for preventing unwanted access to the database authentication. The authentication procedure in SQLIPA is based on passwords and hash values. On sample data from the user table, the SQLIPA was tested. The authentication technique takes only 1.3 milliseconds [13].

In ASP.NET online applications, the SAFeLi framework detects SQL injection vulnerabilities automatically. The algorithmic capabilities of the tool are its tools component. SAFELI creates an equation based on strings that match a certain attacker pattern [14] during symbolic execution. John Prescott, the Prime Minister of the United Kingdom, declared Kashmir to be an integral part of India in 2006. They are common incursions that are carried out on online pages and applications. They are more difficult to comprehend than traditional SQL injection since they are more subtle. The research investigates how detection and protection measures for these attacks were created [15].

A unique method for identifying SQL injection attacks with omitted attributes[16]. This most recent application is a piece of software that protects against SQL injection attacks. The fundamental idea is to employ a randomised SQL language for a certain CGI system to detect and abort requests that include malicious code. They could use a randomization strategy to run the same plan for each database back-end. They should store the database in a form that allows anyone from the outside to access it [17].

They presented a paper regarding a survey they conducted on SQL Injection attacks. Different levels of severity can be assigned to attacks. Detection and preventive measures for SQL Injection are also covered in this book. In addition,it does a study of several database protection techniques[18]. SQLIAs were one of the most serious application security vulnerabilities in 2017, according to a paper by Alwan, Z. S., and Younis, M. F., according to the business. A

survey of SQL security issues is presented in this work. The study discusses the history of SQL injections as well as detection strategies [19].

Instead of using the usual method, they advised using the tokenization method to obtain tokens. Using forensics and security tools, either assault can be detected. This research was carried out using the Java programming language [20].

SQLi was defeated thanks to the introduction of a crime-fighting tool. Positive and malicious traffic are frequently distinguished by web application policies. Early detection of such fraudulent actions will be required as part of the solution. In a complicated way, the methodology extracts a conservative model of all the inquiries that are anticipated to occur in the website from the source code. The system keeps track of what queries are made and how the models respond at runtime. Queries that do not follow the flowchart supplied are stopped and recorded [21].

They devised a novel strategy to protect apps from SQLIAs. Using this method, new vulnerabilities were discovered. The most important finding was that SQL injections may be detected and prevented [22]. They show that by fostering intelligent applications, an intelligent system can help to prevent cyber-attacks [23].

SVMs were used to investigate both original and suspect queries. For classification, a dataset of various sizes is used. Precision, detection time, training time, TPR, TNR, FPR, FNR, and graphical details on our system's performance are all displayed here. Our method has the highest level of output accuracy, at 96.5 percent [24]. A novel method for detecting SQL injection vulnerabilities in Android apps has been developed. They maintain the app's code in a subversive manner. The enlarged code looks like a Java programme and may be run with an SSH client. When performing static analysis, dummy classes are constructed automatically [25].

For the identification of attacks, a novel SA-MVO technique based on deep learning was proposed [26]. For SQLi assaults, a novel technique has been proposed that identifies and prevents these attacks [27]. A traditional booklet called "Pocket Certificates" is presented in this paper. The ideal method is to employ a Triple Data Encryption Standard (AES) and Triple Data Encryption Standard (TDES) encryption scheme (3DES). In terms of how security measures effect system performance, their use is debatable [28].

They offered many ways for network attack detection and prevention [29]. Different services are defined for respective security approaches using third party service [30].

This paper explains how MD5 and AES can be used to protect web-based applications from SQL injection attacks. SQL injection is a technique for gaining unauthorized access to a database. An attacker will need the complete table name in order to get access to a database. To counteract this attack, a solution based on double encryption is offered. In comparison to the old method, the algorithm adopted provides superior performance and security [31].

This paper focuses on employing five cryptographic methods (AES, Triple DES, RSA, Blowfish, and Twofish) to avoid SQL injection attacks. Finally, the study determines which cryptographic strategy is best for preventing SQLIA in online applications [32].

To prevent SQL Injection attacks, this paper uses the 448-bit Blowfish coupled with additional security techniques to strengthen the existing model. To encrypt and secure web data from SQL Injection attacks, we previously used RC4 and Normal Blowfish, but now we employ the 448-bit Blowfish Encryption approach, which has a lower execution overhead [33].

This paper proposed a protocol model for preventing SQL Injection attack using AES (PSQLIAAES) [34].

III. PROPOSED SYSTEM

The framework is made up of two main components: a website and a card database. The website will have login procedures that need the user to enter his or her login information. After successfully creating an account, a user can access their previous purchases and place new orders. The users' credit cards will be used to pay for the things. To generate the cypher text, the data is processed using the data encryption algorithm (AES). After that, the cypher text is saved in the admin database. SQL injection cannot be utilized to access or exploit sensitive data in this scenario because of the encryption. The framework allows for further cyber-security measures by helping to insulate databases from code injection. they are unavoidable.

In this system, when the user logs in to the website, the log in data gets stored in the database in an encrypted format. The secret key of the user is generated by mixing the username and the password in a random

order. The secret key and initial vector is generated separately. Since all the confidential data is encrypted and stored in the database; even if the hacker gets access to the database, they are not able to get access to the encrypted data.

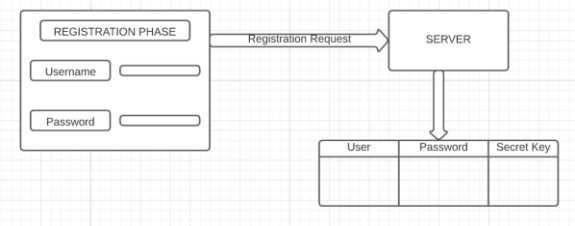

Fig 1: Proposed System

IV. RESULTS

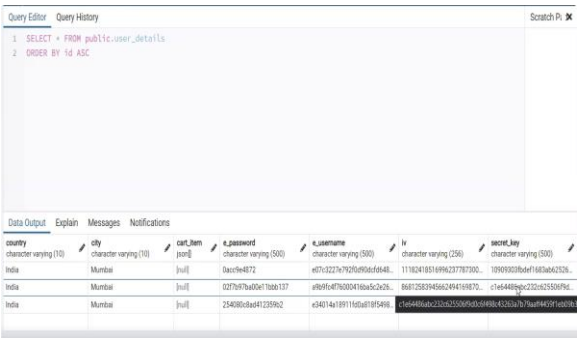

Fig 2: Resultant Database

After implementation of the AES encryption, the data is encrypted and the data of the customers is safe. Databases are susceptible to unauthorized access which can lead to tampering of data, integrity issues, modification and deletion of data etc. These types of attacks can be prevented by using AES encryption, as is shown in this project. The data stored in the database in encrypted using secret key (as shown below).

V. CONCLUSION

SQL injection is still a serious security concern for programmers. Encryption and decryption aid in the protection of sensitive data. The encoding mode AES is well-known, and it is supported by both hardware and software, so far, there have been no realistic cryptographic attacks on the AES attack. AES has also offered a set of alternative keys that can be used to make improvements and to undertake exhaustive key searches. Despite the fact that AES encryption is secure, it is simple to protect consumers from other sorts of security attacks. A user can resolve the danger of data protection by following these measures.

ACKNOWLEDGMENT

First of all, we would like to thank our project guide Prof. Mohandas V. Pawar for giving us the courage, guidance and suggestions for doing this major project. We also express our gratitude towards Dr. Rajneesh Kaur Sachdeo, HOD CSE and Dr. Kishore Ravande, Principal MITSOE sir for their support and guidance. We are thankful to MIT School of Engineering-MIT ADT University, Pune for providing all resources and valuable information required about data mining techniques for our project the process of analyzing and doing research on the valuable inputs helped us to explore knowledge, was a continuous source of inspiration and a unique experience.

REFERENCES

[1] Su, Z., & Wassermann, G. (2006). The essence of command injection attacks in web applications. Acm Sigplan Notices, 41(1), 372-382.

[2] Merlo, E., Letarte, D., & Antoniol, G. (2007, March). Automated protection of php applications against SQL-injection attacks. In 11th European Conference on Software Maintenance and Reengineering (CSMR'07) (pp. 191-202). IEEE.

[3] Wei, K., Muthuprasanna, M., & Kothari, S. (2006, April). Preventing SQL injection attacks in stored procedures. In Australian Software Engineering Conference (ASWEC'06) (pp. 8-pp). IEEE.

[4] Gould, C., Su, Z., & Devanbu, P. (2004, May). Static checking of dynamically generated queries in database applications. In Proceedings. 26th International Conference on Software Engineering (pp. 645-654). IEEE.

[5] Clark, S. R., Cobb, J., Kapfhammer, G. M., Jones, J. A., & Harrold, M. J. (2011, November). Localizing SQL faults in database applications. In 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011) (pp. 213-222). IEEE.

[6] Shar, L. K., & Tan, H. B. K. (2012). Defeating SQL injection. Computer, 46(3), 69-77.

[7] Ma, L., Zhao, D., Gao, Y., & Zhao, C. (2019, September). Research on SQL Injection Attack and Prevention Technology Based on Web. In 2019 International Conference on Computer Network, Electronic and Automation (ICCNEA) (pp. 176-179). IEEE.

[8] Ping, C., Jinshuang, W., Lin, P., & Han, Y. (2016, October). Research and implementation of SQL injection prevention method based on ISR. In 2016 2nd IEEE International Conference on Computer and Communications (ICCC) (pp. 1153-1156). IEEE.

[9] Hlaing, Z. C. S. S., & Khaing, M. (2020, February). A detection and prevention technique on sql injection attacks. In 2020 IEEE Conference on Computer Applications (ICCA) (pp. 1-6). IEEE.

[10] Atoum, J. O., & Qaralleh, A. J. (2014). A hybrid technique for SQL injection attacks detection and prevention. International Journal of Database Management Systems, 6(1), 21.

[11] Halfond, W. G., Viegas, J., & Orso, A. (2006, March). A classification of SQL-injection attacks and countermeasures. In Proceedings of the IEEE international symposium on secure software engineering (Vol. 1, pp. 13-15). IEEE.

[12] Singh, S. P., Tripathi, U., & Mishra, M. (2014). Detection and prevention of SQL injection attack using hashing technique. International Journal of Modern Communication Technologies & Research, 2.

[13] Ali, S., Shahzad, S. K., & Javed, H. (2009). Sqlipa: An authentication mechanism against sql injection. European Journal of Scientific Research, 38(4), 604-611.

[14] Fu, X., Lu, X., Peltsverger, B., Chen, S., Qian, K., & Tao, L. (2007, July). A static analysis framework for detecting SQL injection vulnerabilities. In 31st Annual International Computer Software and Applications Conference (COMPSAC 2007) (Vol. 1, pp. 87-96). IEEE.

[15] Singh, J. P. (2016). Analysis of SQL injection detection techniques. arXiv preprint arXiv:1605.02796.

[16] Lee, I., Jeong, S., Yeo, S., & Moon, J. (2012). A novel method for SQL injection attack detection based on removing SQL query attribute values. Mathematical and Computer Modelling, 55(1-2), 58-68.

[17] Boyd, S. W., & Keromytis, A. D. (2004, June). SQLrand: Preventing SQL injection attacks. In International Conference on Applied Cryptography and Network Security (pp. 292-302). Springer, Berlin, Heidelberg.

[18] Kumar, P., & Pateriya, R. K. (2012, July). A survey on SQL injection attacks, detection and prevention techniques. In 2012 Third International Conference on Computing, Communication and Networking Technologies (ICCCNT'12) (pp. 1-5). IEEE.

[19] Alwan, Z. S., & Younis, M. F. (2017). Detection and prevention of sql injection attack: A survey. International Journal of Computer Science and Mobile Computing, 6(8), 5-17.

[20] Ntagwabira, L., & Kang, S. L. (2010, July). Use of Query Tokenization to detect and prevent SQL Injection Attacks. In 2010 3rd International Conference on Computer Science and Information Technology (Vol. 2, pp. 438-440). IEEE.

[21] Halfond, W. G., & Orso, A. (2005, May). Combining static analysis and runtime monitoring to counter SQL-injection attacks. In Proceedings of the third international workshop on Dynamic analysis (pp. 1-7).

[22] Jang, Y. S., & Choi, J. Y. (2014). Detecting SQL injection attacks using query result size. Computers & Security, 44, 104-118.

[23] Batista, L. O., de Silva, G. A., Araújo, V. S., Araújo, V. J. S., Rezende, T. S., Guimarães, A. J., & Souza, P. V. D. C. (2019). Fuzzy neural networks to create an expert system for detecting attacks by sql injection. arXiv preprint arXiv:1901.02868.

[24] Rawat, R., & Shrivastav, S. K. (2012). SQL injection attack Detection using SVM. International Journal of Computer Applications, 42(13), 1-4.

[25] Edalat, E., Sadeghiyan, B., & Ghassemi, F. (2018). ConsiDroid: A concolic-based tool for detecting SQL injection vulnerability in android apps. arXiv preprint arXiv:1811.10448.

[26] Mohandas V. Pawar, Dr. J. Anuradha, " Detection of Blackhole and Wormhole Attacks in WSN enabled by Optimal Feature Selection using Self-Adaptive Multi-Verse Optimizer with Deep Learning", International Journal of Communication Networks and Distributed Systems- Inderscience(In Press).

[27] D. Patel, N. Dhamdhere, P. Choudhary and M. Pawar, "A System for Prevention of SQLi Attacks," 2020 International Conference on Smart Electronics and Communication (ICOSEC),

Trichy, India, 2020, pp. 750-753, doi: 10.1109/ICOSEC49089.2020.9215361.

[28] D. Agnihotri, S. Ahmed, D. Darekar, C. Gadkari, S. Jaikar and M. Pawar, "A Secure Document Archive Implemented using Multiple Encryption," 2020 International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2020, pp. 765-770, doi: 10.1109/ICOSEC49089.2020.9215302.

[29] Intrusion Detection and Prevention in WSN and MANET using Machine Learning Techniques and Existing Challenges, Journal International Journal of Advanced Science and Technology, Volume-29, Issue-3, Pages-(306-328), 2020.

[30] Mohandas Pawar Sujata Jadhav,Diksha Bejgam,Shweta Nhawkar,Shital Sumbe, A Novel Approach for Recommendation of Cloud Service for Security using Trusted Third Party, International Research Journal of Engineering and Technology (IRJET), Volume-3, Issue-03, Pages-(1794-1800), 2016.

[31] Sood, M., & Singh, S. (2017). SQL injection prevention technique using encryption. International Journal of Advanced Computational Engineering and Networking, 5(7), 4-7.

[32] Karunanithi, J. S. (2018). SQL Injection Prevention Technique Using Cryptography.

[33] Rajeswari, K., & Amsaveni, C. SQL Injection Attack Prevention Using 448 Blowfish Encryption Standard.

[34] Balasundaram, I., & Ramaraj, E. (2011). An authentication mechanism to prevent SQL injection attacks. International Journal of Computer Applications, 19(1), 30-33.