Detection and Prevention of SQLI Attacks inside the DBMS

SAYALI TELI, PRATIMA MORAJKAR, TEJAS PENDHARKAR, TANMAYEE KULKARNI

BE Students, Department of Information Technology, Marathwada Mitra Mandal's College of Engineering, Pune,

PREETI JOSHI

Professor, Department of Information Technology, Marathwada Mitra Mandal's College of Engineering, Pune.

***Abstract*—Database applications are used to store, search, sort, calculate, report and share information. Databases can also contain code to perform mathematical and statistical calculations on the data to support queries submitted by users. The grocery store, bank, video rental store and clothing store all use databases to keep track of customer, inventory, employee and accounting information. SQL injection is a code injection technique, used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field for execution. The most common cause of database vulnerabilities is a lack of due care at the moment they are deployed. The effectiveness of such attacks stems from semantic mismatch between how SQL queries are believed to be executed and the way in which database processes them. In this paper, a technique is proposed which provides external as well as internal security to the database. SEPTIC(Self Protecting mechanism), a mechanism for DBMS attack prevention, which can also assist on the identification of the vulnerabilities in the applications is combined with Naive Bayes a Machine Learning algorithm to achieve accurate results. This technique will be implemented using MySQL database. In this paper an attempt has been made to develop an online shop that allows users to check for different clothing stuff. To enhance the security all the user details such as name, password and card details will be encrypted using AES algorithm and then stored in the database.**

***Keywords*—DBMS self-protection, injection attacks, Database Security, SQL injection technique, Attack detection, Attack prevention, AES Algorithm, Encryption.**

I. INTRODUCTION

SQL Injection is "a code injection technique that exploits a security vulnerability occurring in the database layer of an application". In other words it's SQL code injected in as user input inside a query. SQL Injections can manipulate data (delete, update, add etc) and corrupt or delete tables of the database. It is used to attack data-driven application. Lack of input validation is a major vulnerability behind dangerous web application attacks. By taking advantage of this, attacker scan injects their code into applications to perform malicious tasks. In which malicious SQL statements are into an entry field for execution. This is a method to attack web applications that have a data repository. The attacker would send a specially crafted SQL statement that is designed to cause some malicious action. Incorrectly validated or non-validated string literals are concatenated into a dynamic SOL statement and interpreted as code by the SQL engine.

Web applications have been around for a long time and are an important component as they serve to be the interface of many business applications. Databases serve to be the most commonly used back end by most of the applications. However, web applications have vulnerabilities which compromises the data stored in databases. SQL injection attacks are increasing in large number. Many anti-SQLI mechanisms have been developed but less accepted. Some of these applications inspect and block SQL queries but without having any knowledge of how these queries are processed in DBMS. In all the cases developers make assumption about the processing at the server side as well as DBMS which sometimes goes wrong. For example, developers assume some PHP functions always sanitize input and prevent SQLI attack which is not always true. Such wrong assumptions are caused because of semantic mismatch between how the query is expected to run and what actually happens when it is executed. This semantic mismatch leads to vulnerabilities as the mechanisms may fail to prevent it.

To avoid such issues SQLI attack can be handled inside the DBMS after the server-side code processes the input and the queries are validated by DBMS. We will propose two categories of attacks the stored attacks will contain the highest risk attacks and some other stacks for which new variants continue to occur. Stored injection attacks also include SQL queries. So we are going to propose the method named SDBMS (Security to Database Management System) which we will detect the SQLI attacks by comparing the queries with the queries stored in query models, and also by comparing the queries with validated queries by using similar method to improve detection process. So SDBMS works in 3 modes first is training mode, second identification and third drop the query. In the first mode the application is trained by firing large number of queries in an application. This results in set of query models. Even though a query model is generated some queries can be missed, so to handle such queries we are going to use quarantine queries at run time and then update the query model.

We are going to use the most popular open-source DBMS that is MySQL, and PHP language for to develop web application.

Types of attacks:

1. Union-Based SQLI :

This is the most popular SQLI attack which uses UNION statement, which is integration of two select statements, to obtain data from database.

Union Query:

SELECT pass FROM user_table1 WHERE login ID='' UNION SELECT pass from user_table2

Where Username=xxx -- AND pass=''.

2. Error-based SQLI :

This is the simplest type of data which involves questioning the database, and it responds with an error including the data you asked for.

3. Adding or modifying data:

This attack involves inserting malicious data or updating the data which contains harmful information.

4. Blind SQLI attack:

The blind SQLI is the difficult one to detect among the other SQLI attacks. In this attack, no error messages are received from the database, hence data is extracted by asking questions to the database. It is further divided into two types:

1.Boolen-based SQL injection: This means asking the questions to the database which results in true or false.
2.Time-based injection: This involves time delay.
Blind Injection:
SELECT pass FROM userTable WHERE username= 'user' and 1 =0 -- AND pass = AND pin=0
SELECT info FROM userTable WHERE username= 'user' and = 1 -- AND pass = AND pass=0
Timing Attacks:
declare @varchar(8000) select @ = db_Alias() if (ascii(substring(@,1, 1))&(power(2,0))) >0 wait for delay '0:0:6'.
5. Leakage of sensitive information:
This means an event that occurs when confidential information is being exposed to unauthorized parties as a result of cyber-attack.
6. Extracting data from database
Types of queries to extract information from database which is used by an attacker:
7.Tautology:
SELECT * FROM userTable WHERE username=''OR 1=1 -- AND password='';
2=2, 3=3, '1'='1','b'='b' or "name"=" ….\
8. Stored Procedures:
CREATE PROCEDURE .is Authenticated
Name varchar2, varchar2, int AS EXEC("SELECT accounts FROM users WHERE login=" + Name+ If'; and pass="++ and pass= +);

II. METHODOLOGY

This section consists of brief description about the methods that are to be used for implementing the project.

Our approach is going to detect the SQLIAs using the SEPTIC mechanism combined with Naive Bayes machine learning algorithm. Machine learning methods provide highly accurate predictions on test data. Also, AES Algorithm will be sued to encrypt user details to enhance security.

1. AES Algorithm

This technique will be evaluated on an online shop database, for which an online shop application is to be created. The user has to login in order to browse through the products and to buy the products. So when the user registers user details such as name, password, card details will be encrypted using AES Algorithm and directly stored in the database. When the Admin changes the status of the user from inactive to active the details will be decrypted(i.e during login session) and the user will be allowed to login and do further process. During login process the user generated queries will be matched that is semantic matching will be performed, if the query is matched with the query in the data set, it will be executed. If the query is mismatched then the query will be aborted thus resulting in prevention of attack.

2. SEPTIC

Once the details are encrypted and stored in the database next step is to detect the attack or normal process by semantic match. SEPTIC runs in 3 modes, first is the training mode that is during the setup of the system. Training is carried out by running the application without attack for sometime and the result is stored in QM(Query Models), each one associated with query identifier(ID) and the data will be saved in learned query models.
In normal operation, SEPTIC will generate a QS and an IID for arriving request. Attack detection is carried out by comparing QS with the QM that was learned for that ID and second by looking the disparities between the QS and initial query. If the query is found to be valid then it will be stored in the learned query model and the learned query model will be updated. If the query is invalid then before further execution it will be aborted thus resulting in prevention.

2.1 Learning query model

SEPTIC consists of two learning methods Training method and incremental method.

1. Training method:
In training method queries will be fired to the application and stored in the Query Model. All the inputs are not attacks. This results the query with in creating and storing the queries with unique identifier. This ID is further used and matched the new arriving query.

2. Incremental method:
SEPTIC runs the incremental method in normal operation. This allows dealing with the incomplete training (the queries not issued) in training model. The basic idea is that when a new query is arrived it is being notified to the administrator before declaring it to be attack. This method creates need for two concepts quarantine to address suspicious query model and aging to deal with query model.

3. Naive Bayes

Detection of SQL Injection attacks using a machine learning algorithm called Naïve Bayes. Naïve Bayes is a classification machine learning algorithm that assumes that a particular incident is unrelated to and is independent of other all other incidents . Naïve Bayes classifier is used to classify between malicious and non-malicious SQL queries.
Naïve Bayes builds classifier for classification of malicious and non-malicious queries which we are going to implement in our project for providing external security to the DBMS by detecting malicious queries. This algorithm calculates prior and posterior probability. Prior probability is calculated from the count of malicious and non-malicious queries.

III. LITERATURE REVIEW

1. SEPTIC: Detecting Injection Attacks and vulnerabilities inside the DBMS, Nuno Neves, Miguel Beatriz, 2019 IEEE

In this paper SEPTIC (Self protecting databases from attacks) mechanism is used in 3 different modes such as
i)Training Mode
ii) Detection mode
iii) Prevention mode
In training mode the application is trained by firing large number of queries without performing any malicious code. The result of this is stored in query models. For every query a ID is generated. For SQLI, attacks are caught by comparing queries with query model that is the queries stored in query model. If mismatch is found the query is aborted before execution. Sometimes, training may be incomplete and cannot cover all the queries a new concept of quarantine query is added which means putting in queries at run time for which SEPTIC has no query model. After this query model is again updated by entering new queries in it. A septic_training module is developed which targets web applications and works like a crawler. For each web page, it searches for HTML forms and collects information about the submission methods, action, variables and values. Then, it issues HTTP requests for each form, causing the queries to be transmitted. This queries can be static or dependent on result of other queries. All the approaches are triggered by the administrator or otherwise executed automatically. Execution time of queries depend on the complexity of the application. Once training phase is

completed there is no need of intervention of administrator and SEPTIC can be put in normal operation.

SEPTIC is implemented by a module inside the DBMS, allowing every query to be checked for attacks. Comparing QS with QM corresponds to first two steps of the detection process. For detection purpose queries are evaluated at the end of the DBMS by semantic matching before the query is executed. SQLI attacks are detected if queries fall in either class s1(Syntax structure) or s2(Syntax mimicry) as shown in table 1. These classes are called primitive for SQLI because any SQLI belongs to that. If the query is not matching to any of these classes is that if the attack neither modifies the query structure nor changes the query mimicking the structure than it is unmodified, and it is not SQLI attack. SEPTIC performs detection mechanism by comparing validated query with the associated query structurally for class 1 and syntactically for class 2, plus non unique ID's are handled by comparing Qval with Qinit. An attack is flagged if there are differences in any of these tests. SEPTIC has proposed a detection algorithm which involves steps such as

1. Structural verification: In which number of nodes in QS is compared with number of nodes in QM and if it is different then Qval does not respond to the model and detection of QM ends.
2. Syntactical verification: In this step ELEM_TYPE and DATA_TYPE of nodes in QS is checked with QM, and if it is different then Qval does not match the model and detection stops.
3. Query similarity verification: in this step Qinit is compared with nodes in QS, then this disagreement causes an attack to be flagged.

There is no attack if all checks are valid. Otherwise, there is an attack and in such case action to be taken depends upon the mode in which SEPTIC is running. In prevention mode query processing is aborted, in detection mode query is executed.

SETIC also includes incremental method which is used in last two modes that is in detection and prevention mode. It runs the incremental method in normal operation. This helps to deal with the problems arised during the training phase. So the basic idea behind this method is that when SEPTIC processes a query for which there is no QM then besides identifying as an attack it also notifies the administrator that a new query is observed. As there is no QM for the query, before stating the query as attack SEPTIC first verifies if the query is an attack using the query similarity verification and stored injection detection (insert and update) mechanism.

If the attack is not identified, then SEPTIC stores it in learned QMs or quarantine QMs.

Class	Class name	PHP sanit. func.	DBMS	Example malicious input
A	Obfuscation			
A.1	-Encoded characters	do nothing	decodes and executes	%27, -0x027
A.2	-Unicode characters	do nothing	translates and executes	U+0027, U+02BC
A.3	-Dynamic SQL	do nothing	completes and executes	Char(39)
A.4	-Space character evasion	do nothing	removes and executes	Char(39) /**/OR/**/1=1--
A.5	-Numeric fields	do nothing	interprets and executes	0 OR 1=1--
B	Stored Procedures	sanitize	executes	admin' OR 1=1
C	Blind SQLI	sanitize	executes	admin' OR 1=1
D	Insert data	sanitize	unsanitizes and executes	admin' OR 1=1--
E	Second order SQLI	-	executes	any of the above
F	Stored XSS	-	-	<script>alert('XSS')</script>
G	Stored RCI,RFI,LFI	-	-	Malicious.php
H	Stored OSCI	-	-	; cat /etc/passwd
S.1	Syntax structure	Sanitize	executes	admin' OR 1=1
S.2	Syntax mimicry	sanitize	executes	admin' AND1=1

Figure 1:Classes of Attacks against DBMS ("Figure taken from [1] ";)

2. SQL Injection Detection using Machine Learning, Anamika Joshi, Geeta V, 2014 IEEE.

In this paper, a combination of Naïve Bayes and Role Based Access Control methods have been implemented. The different types of SQL Injection attacks have been stated which are:

1) Tautology: In this the SQL query is modified such that the conditions always result to true. If id and password are entered with a WHERE condition then even if one of the entries returns TRUE, all data can be extracted.
2) UNION: Queries are appended with the use of the UNION statement.
3) Blind SQL: Asking database a TRUE or FALSE question and checking whether valid page returned or not by using the time it took for valid page to return as the answer to the question.

Naïve Bayes builds classifier for classification of malicious and non-malicious queries which we are going to implement in our project for providing external security to the DBMS by detecting malicious queries. This algorithm calculates prior and posterior probability. Prior probability is calculated from the count of malicious and non-malicious queries.

The tokenizer converts queries into tokens of features and labels them as malicious or non-malicious. This input is sent to the classifier, which then finally classifies queries as malicious or non-malicious query. Firstly Feature Extraction method is used to reduce the total number of data in the datasets. This can be done by Blank separation Method which extracts terms with respect to each blank space. Then tokenization will be carried out which breaks the query into meaningful elements called tokens. Then Role Based Access Control is implemented. In the classifier, the role of the user is taken as a parameter for evaluation. If a particular operation is not allowed for a given user, then a potential threat to the system is detected.

In this approach the author proposed a method for detection of SQL injection attack based on Naïve Bayes Machine Learning Algorithm combined with Role Based Access control mechanism detects malicious queries with the help of classifier. The addition of another parameter for Role Based Access control has increased the accuracy of detection and also reduced number of false positives. They thought that their method should be tested against larger datasets to evaluate the efficiency. The proposed method can be enhanced for detection of other types of SQL injection attacks also by extracting features appropriately.

3. Detecting Data Leaks via SQL Injection Prevention on an E-Commerce, Karan Ray, Nitish Pol, Suraj Singh, 2018 IJSER.

In this paper a secure path for transaction done by the user has been established. Using AES (Advanced Encryption Standard)encryption technique, the transaction and user account details can be made secured. The project consists of list of cloths displayed in various materials and designs. The user may browse through these products as per categories. If the user likes a product, he/she can add it to his/her shopping cart. Once user wishes to checkout he must register on the site first. Once the user makes a successful transaction admin will get report of his bought products. Prevent SQL injection while firing queries to database and to make the database secured "Detecting Data Leaks via SQL Injection Prevention on an E-Commerce" is something like the original grocery shop shopping cart that is used by the customer in selecting certain products. Finally, after selection the customer confirms orders for all the purchasing items and submits his/her account details with tax information at the checkout counter.

The Online Shop secures the card payment and won't let the card data to get hacked.
While user doing a card payment, all the card data is encrypted and then stored into database. System also keeps user details in an encryption form using AES encryption

IV. CONCLUSIONS

According to technology vendor application security the top threat related to databases are SQL injection can be prevented by the mechanism of SEPTIC it also gives an idea of catching attacks inside the DBMS and identifying the vulnerabilities in an application code when attacks were detected.

In the future, we can use the mechanism to prevent business related confidential data so that no one can try to gain credentials of others and exploit the victim. The mechanism will be experimented both with synthetic code with vulnerabilities inserted on purpose and with open source PHP web applications, and other type of applications. The mechanism will be able to detect and block the attacks it was programmed to handle performing better.

REFERENCES

[1] Nuno Neves, Miguel Beatriz." SEPTIC: Selecting Injection Attack and VulnerabilitiesInside the DBMS. Transaction 2019.
[2] Karan Ray, Nitish Pol, Suraj Singh Guided by Prof. SUVARNA ARANJO, "Detecting DataLeaks via SQL Injection Prevention on an E-Commerce", International Journal ofScientific & Engineering Research Volume 9, Issue 3, March-2018.
[3] Anamika Joshi, V "SQL Injection detection using machine learning", 2014 International ConferenceonControl, Instrumentation, Communication and Computational Technologies(ICCICCT).
[4] Fehreen Hasan, , "A Novel Approach for SQL Injection Prevention Using Hashing & Encryption (SQL-ENCP)", (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3, 2012.
[5] Al-Balqa, , "New Strategy for Mitigating of SQL Injection Attack", International Journal of Computer Applications Volume 11, November2016.
[6] M. and D., "Writing Secure Code for Windows ,1sted,MicrosoftPress Redmond,USA,2007.
[7] W.G. and , "AMNESIA Analysis and Monitoring for neutralizing SQL- Injection Attacks," . IEEE and Conference on Automatic Software Engineering (ASE2005), Long Beach, CA, USA, Nov 2005.
[8] S.W. Boyd and AD., "SQL rand: Preventing SQL InjectionAttacks," . 2nd Applied Cryptography and Network Security(ACNS) Conference, Jun 2004.