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

Detecting injection attacks and vulnerabilities inside the DBMS

Dr.K.N.S Lakshmi,
[1]head Professor
[1]Department of Computer Science,
[1]Sankethika Vidya Parishad Engineering College,
Visakhapatnam, India

[2]**Manta Vasudeva kumar,**
[2]MCA 2nd year,
Sankethika Vidya Parishad Engineering College,
Visakhapatnam, India

ABSTRACT

Nearly all businesses nowadays attempt to use back end storage to keep their important data safe. These databases frequently include weak applications, like as front-end web pages for carrying out the activities, which makes injection attacks possible. Because the system cannot easily distinguish between a regular query and a SQL attack, it is particularly difficult to detect such attacks. SEPTIC, a mechanism for DBMS attack prevention that can also help with the identification of the vulnerabilities in the applications, is the solution we suggest to identify such attacks. The technique was added to My SQL and experimentally tested with a number of different programmes and protection strategies. No false negatives or false positives were found using SEPTIC, according to our findings.

KEYWORDS: database, weak applications, attack prevention, detect, identification, vulnerabilities

I.INTRODUCTION

The detection of SQL injection attacks[1] is the focus of our effort. SQL injection is only a code injection technique that compromises the application's database layer's[2] security. Our project focuses on stopping users from trying to compromise database security by detecting SQL injection attacks. SQL Injections[3] have the ability to destroy or remove database tables as well as change data (delete, update, add, etc.). It is employed to target applications that rely on data. Unchecked input is a significant weakness[14] that makes dangerous database attacks possible. Attackers exploit this and inject their code into programmes to carry out nefarious deeds. which executes malicious SQL statements that have been entered into a field. The attacker sends a specially encoded SQL command meant to corrupt the database .Investigations conducted over the years have consistently pointed to designers' lack of safety consciousness in online development to cleansed contribution as the cause of SQLIA, and as a result they have floated towards code-based sterilisation for their suggested solutions to address SQLIA. As a result, Like wi Exploration has drifted towards code-based sterilisation for their suggested solutions to handle SQLIA. Over the years, LikewiExploration has typically identified designers' lack of safety mindfulness in online progression to disinfected contribution as the cause of SQLIA. In a similar vein, SQLIA vulnerability is a direct byproduct of the kind-hearted open content preparation of the SQL engine itself, rendering both legacy and cloud arrangements lacking in disinfection defenseless. A shameful SQL lobby hunt [1] that reports new patterns. Examples of SQL tokens and images will be present in the assault marks at injection points that are SQLIA positive, whereas legitimate web solicitations will appear as expected data from the application. In this study, we put together a web application for precognitive examination with a lot of learning data to get a classifier ready.

II.EXISTING SYSTEM

Two of the earliest efforts on identifying SQLI by comparing the structure of a SQL query before and after the addition of inputs and before the DBMS executes the queries are AMNESIA and CANDID . both rely The query models convert all characters in query statements—aside from user inputs—to shadow characters and generate shadow values for all user-supplied string inputs. Second, it computes the query for a query execution and checks to see if the root nodes from the two parsed trees are equal. Like SEPTIC, DIGLOSSIA recognises syntax structure and mimicry attacks, but unlike SEPTIC, it does not recognise second-order SQLI because it only processes user-inputted queries, nor does it recognise encoding and evasion space character attacks because they do not change the parse tree root nodes prior to the DBMS processing the malicious user inputs. It is more adept at handling some semantic mismatch difficulties than AMNESIA and CANDID, but it does not resolve all of them. Even though it handles some semantic mismatch issues better than AMNESIA and CANDID, it does not solve all of them. AMNESIA and CANDID were two of the first attempts to discover SQLI by contrasting the structure of a SQL query before and after the inclusion of inputs and before the DBMS runs the queries. both depend The query models generate shadow values for all user-supplied string inputs and convert all characters in query statements—aside from user inputs—to shadow characters. Secondly, it determines the

Disadvantages:

1. There is no SEPTIC that has not missed detections (false negatives) or reported false positives.

2. Complex and dynamic queries are not supported by any process.

III.PROPOSED SYSTEM

We are creating an online application where both the user and the administrator [21]must contribute. The key component is that as soon as the user chooses a product, it is added to the cart for further processing, but first, we must create an account on the website hosting the application so that we can access it again in the future. The user's task in this instance is to shop, or you could say to purchase a product, and the admin's task is to show the product in terms of numerous categories. The prevention in two aspects is the goal of the development. Online prevention is used when logging into accounts that are now active, and the second time is when we save personal information. We are creating an online application where both the user and the administrator must contribute. The key component is that as soon as the user chooses a product, it is added to the cart for further processing, but first, we must create an account on the website hosting the application so that we can access it again in the future. The user's task in this instance is to shop, or you could say to purchase a product, and the admin's task is to show the product in terms of numerous categories. The prevention in two aspects is the goal of the development. Online prevention is used when logging into accounts that are now active, and the second time is when we save personal information.

This paper's key contributions are:

(1) Detecting SQL injection attacks.

(2) halt database attacks and give databases security

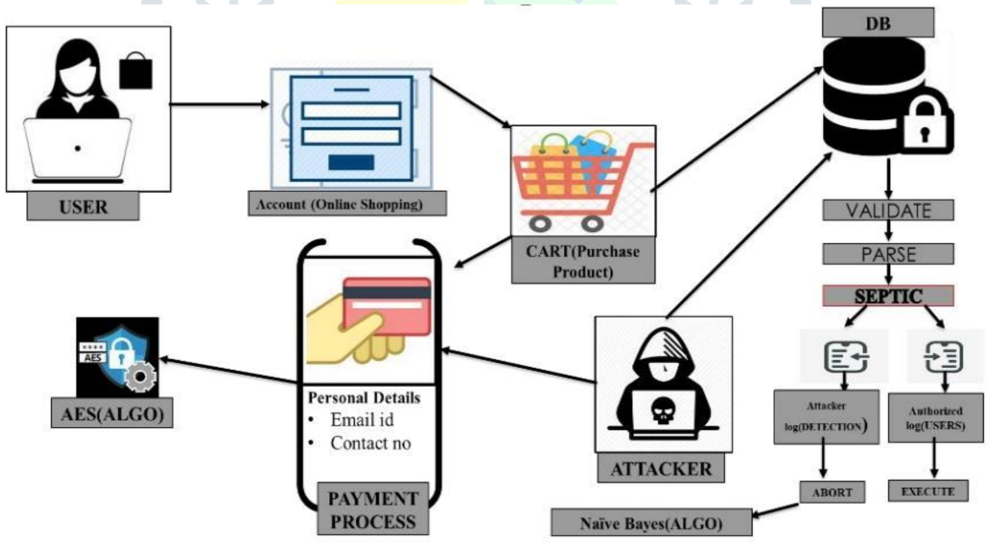


FIG1. System architecture

IV.SYSTEM ARCHITECTURE

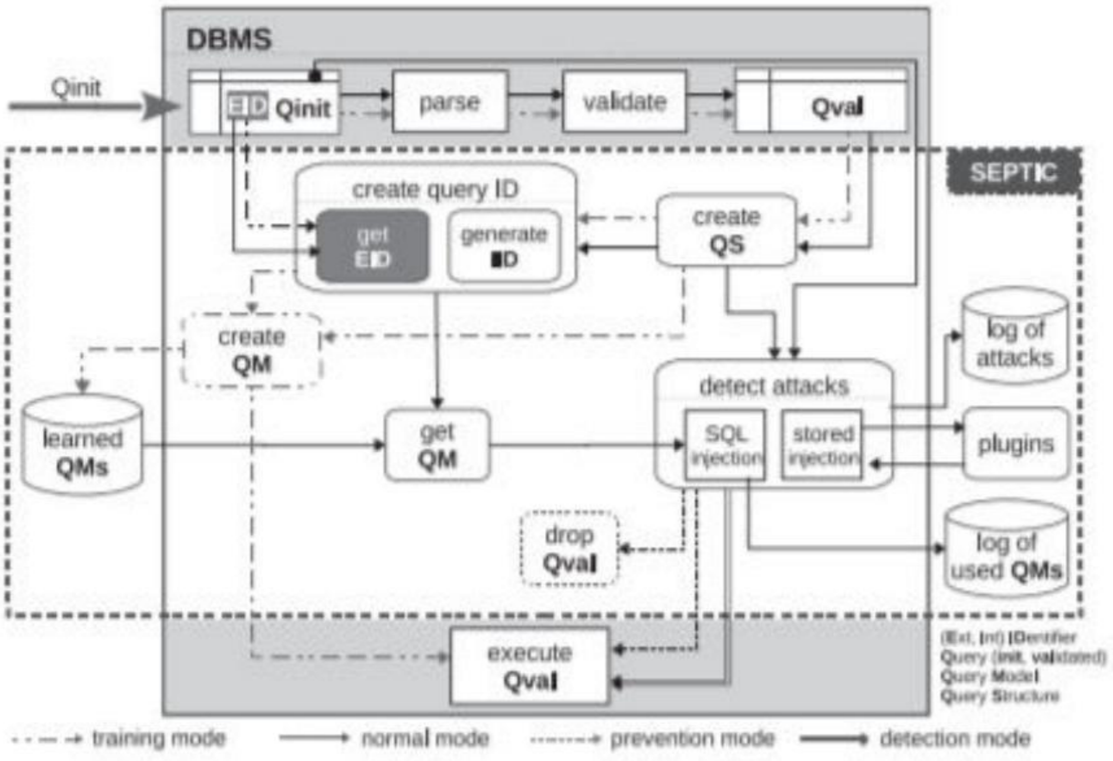


FIG2: system architecture

Application Server

The application server operates some tasks like managing a cloud that offers data storage services. View all Data Owners and give consent view all users who are authorised, Observe all data contents along with rankings and digital signatures.View all data contents without a digital signature, including rankings and ratings. View user search activity, View all SQL Injection Intruders, including their IP address, date, and time. View the ranking of all documents, View all invaders and provide a link to the chart (name of the number of attacked documents).

Generator of signatures

The person who creates the digital signature is known as the "signature generator," and they carry out the actions listed below, including logging in, viewing all owner documents and providing a choice to create a digital signature, as well as viewing all data contents with rank and providing a choice to create a secret key using RSA.

Nave Bayes: The Nave Bayes machine learning algorithm is used to detect SQL Injection attacks. A classification machine learning algorithm called Naive Bayes makes the assumption that each incidence is unrelated to and independent of every other incident.

The Naive Bayes classifier is employed to distinguish between malicious and benign SQL queries.

The Advanced Encryption Standard (AES) algorithm: is used to safeguard sensitive data from attackers. Compared to DES, the AES is more secure. AES encryption is broken down into three phases. Initial Round, Main Round, and Final Round are in order. These three phases include various operations. Use three operations on the cypher text to decrypt it using the AES (Advanced Encryption Standard).

1) The Reverse Final Round]
2) Main Round Reverse
3) Reverse First round.

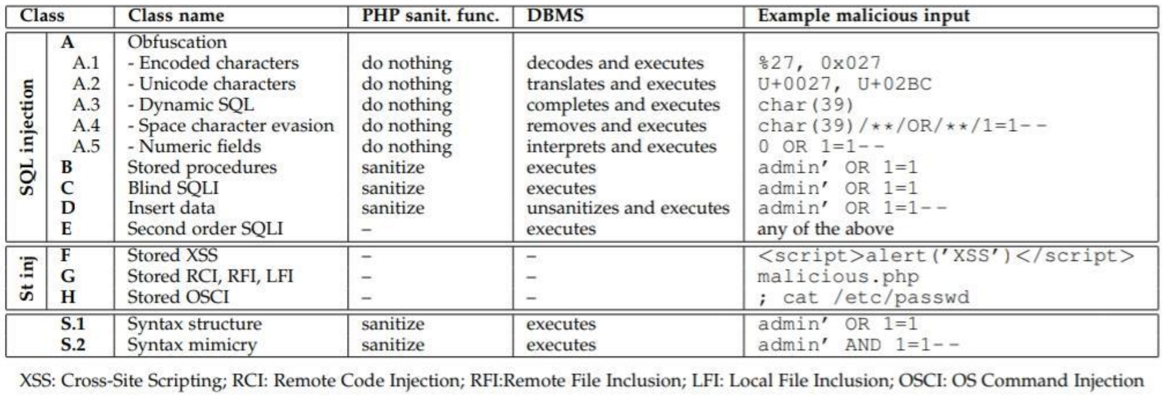
AES is built on various keys with sizes of 128, 192, and 256 bits.

SEPTIC: In this case, we're employing SEPTIC methods to protect the database system from various attacker attacks by adhering to three modes.

Training Mode, first
Detection Mode
Prevention Modee

V.DBMS INJECTION ATTACKS

As previously stated, we define semantic mismatch as an inaccurate view of how the DBMS executes SQL queries; the developer expects queries to be handled one way, while they are really executed in another way. This discrepancy frequently causes errors in the protections' implementation in the source code of applications, leaving those applications open to SQL injection and other DBMS-related attacks. The issue is subjective in that it is dependent on the coder, although mistakes do happen occasionally. Sanitising user inputs prior to usage is a frequent practise to try to prevent SQLI.



Class		Class name	PHP sanit. func.	DBMS	Example malicious input
SQL injection	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
St inj	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' AND 1=1--`

XSS: Cross-Site Scripting; RCI: Remote Code Injection; RFI:Remote File Inclusion; LFI: Local File Inclusion; OSCI: OS Command Injection

TABLE1: classes of attacks against dbms

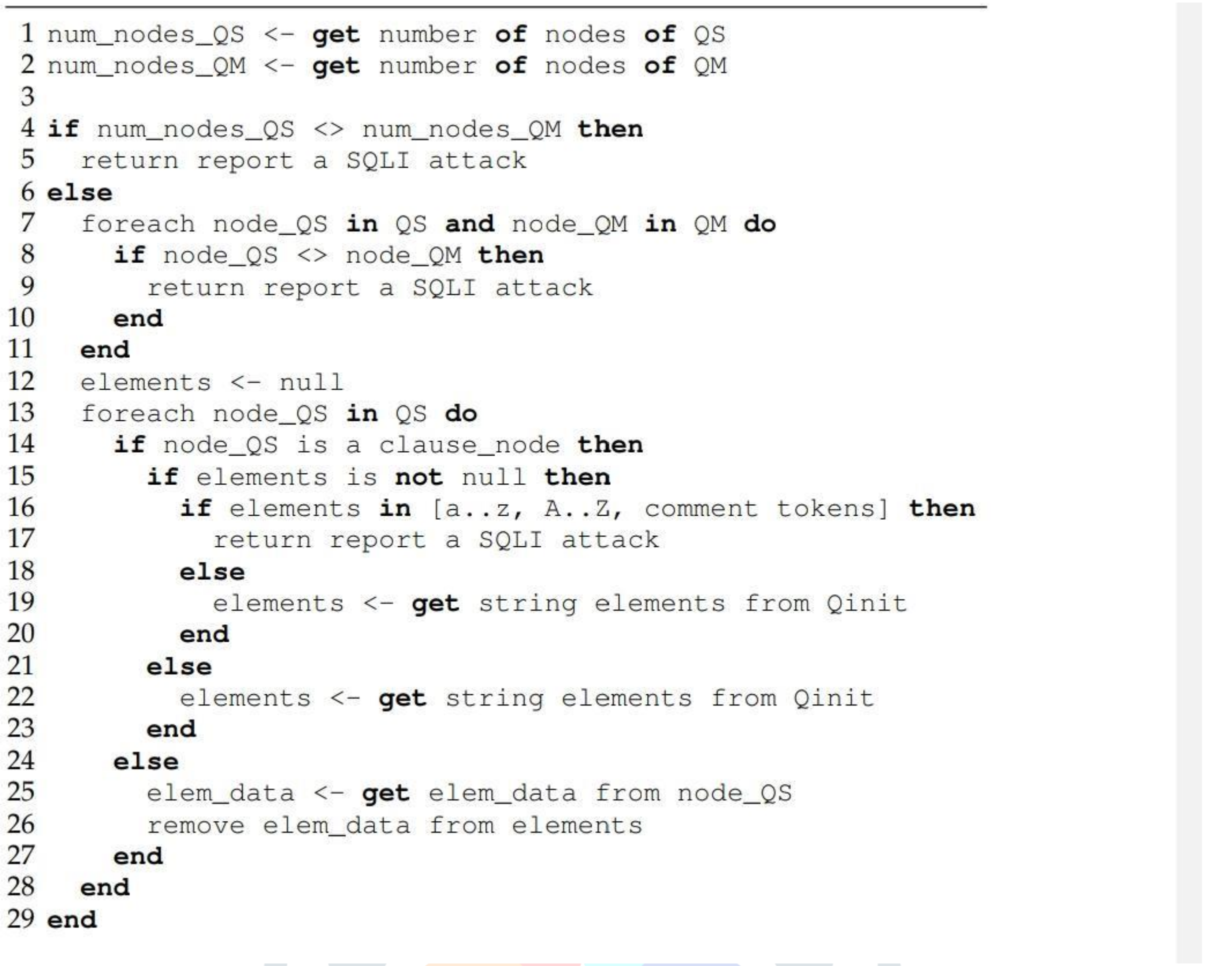
'admin' as the username AND 'foo' as the password. This attack works since there is no need for a password because the query is equal to SELECT * FROM users WHERE username='admin' (the two characters -- indicate that the remainder of the code in the line should be ignored). This is also a S.1 attack because the query's structure has been altered by the removal of the section that checks the password..Other masquerading methods are used in the other A subclasses. The attacker in class A.2 encodes some characters in Unicode (for instance, the prime is encoded as U+02BC). A function is dynamically inserted and called in class A.3 (for instance, the prime is encoded as char(39)).Class A.4 manipulates inquiries by using identical strings and spaces, such as hiding a space with the comment /**/ .Class B, which consists of stored procedures, could be abused in a manner similar to how queries written in application code are.Attacks of type S.1 or S.2 may result from these processes accepting inputs that change or mimic the syntactic structure of the query. Blind SQLI attacks, which fall under Class C, try to gather data from the database by watching how the application handles various inputs. These assaults could alternatively be classified as S.1 or S.2 offences.Class D, "insert data," attempts to insert created data into the database (INSERT, UPDATE), where it can then be later retrieved and used in an application query. The base of stored injection attacks (see subsequent classes), this attack type is another instance of semantic mismatch.

VI.INJECTION ATTACK DETECTION

How SEPTIC finds active attacks is described in this section. To do this, Table 1's classes are split into two groups—SQL injection and stored injection—each of which is subject to a distinct processing method**.**

Detection of SQLI

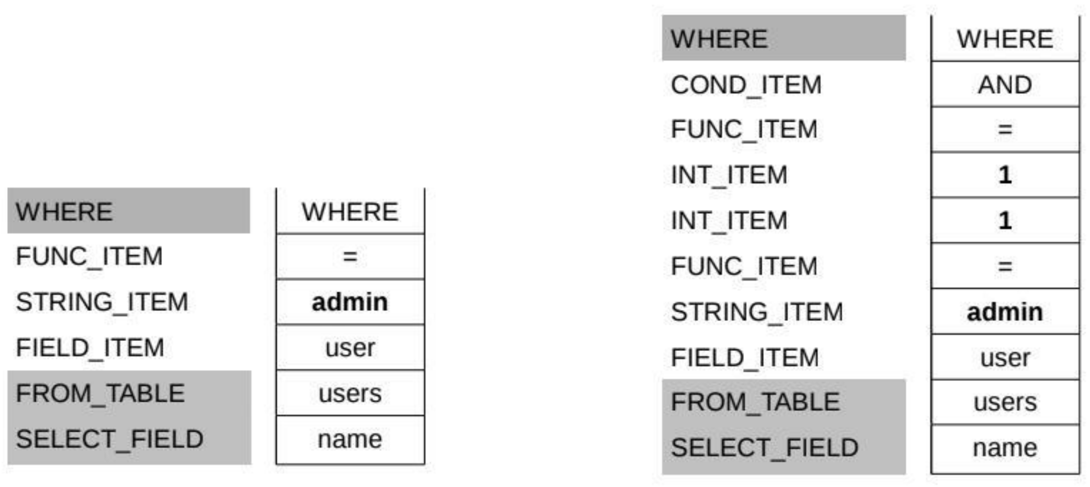
It is possible to identify SQLI attacks by determining whether a query belongs to class S.1 or S.2. Because all SQLI assault falls under one of these groups, they are known as primordial for SQL injection. According to this logic, a SQL injection attack is not one if it does not alter the query structure (class S.1) or alter the query that mimics the structure (class S.2); instead, it must leave the query unchanged.By comparing Qval with the related query model syntactically (for class S.1) and structurally (for class S.2), SEPTIC detects attacks.



```
num_nodes_QS <- get number of nodes of QS
num_nodes_QM <- get number of nodes of QM

if num_nodes_QS <> num_nodes_QM then
  return report a SQLI attack
else
  foreach node_QS in QS and node_QM in QM do
    if node_QS <> node_QM then
      return report a SQLI attack
    end
  end
  elements <- null
  foreach node_QS in QS do
    if node_QS is a clause_node then
      if elements is not null then
        if elements in [a..z, A..Z, comment tokens] then
          return report a SQLI attack
        else
          elements <- get string elements from Qinit
        end
      else
        elements <- get string elements from Qinit
      end
    else
      elem_data <- get elem_data from node_QS
      remove elem_data from elements
    end
  end
end
```

FIG 3. CODE FOR DETECTION



WHERE	WHERE
FUNC_ITEM	=
STRING_ITEM	**admin**
FIELD_ITEM	user
FROM_TABLE	users
SELECT_FIELD	name

WHERE	WHERE
COND_ITEM	AND
FUNC_ITEM	=
INT_ITEM	**1**
INT_ITEM	**1**
FUNC_ITEM	=
STRING_ITEM	**admin**
FIELD_ITEM	user
FROM_TABLE	users
SELECT_FIELD	name

A)structural attack B) mimicry attack

FIG4: QSs resulting from structural and mimicry attack

Consider a second-order SQLI attack that is executed as follows: (i) A malicious user enters information that causes the programme to place adminU+02BC- — (i.e., admin'——with the prime represented in unicode as U+02BC——) in the database; (ii) subsequently, this information is taken from the database and entered in the user field in the aforementioned query;(iii) The query is parsed and verified by the DBMS, which decodes U+02BC into the prime character; query SELECT name FROM users in the output WHERE user= admin belongs to class S.1 since it changes the query's structure. The QS is shown in Figure 5(a) for this question. When SEPTIC analyses the QS and QM, it notices that they do not match because each have different numbers of nodes during structural verification.

Stored injection detection

Attacks using stored injection are conducted in two steps. The database is first compromised by the introduction of harmful data, and then that data is removed and misused in another way. Consider a stored XSS (class F), for instance, where the data contains a malicious script. The script is saved in the first phase, and then it is retrieved from the database and added to a web page so that a browser can view it in the second. Since they don't alter queries, these attacks cannot be recognized as SQLI attacks. SEPTIC stops stored injection attacks in their first stages by looking for malicious data in INSERT and UPDATE queries that insert data into the database, and then testing that data using plugins. As a result, a collection of plugins—typically one for each sort of attack—is employed for this task.The plugins look through the queries looking for code that may be run by a server-side application (php), a shell script, an operating system command, or a browser (JavaScript, VBScript). The detection method does a preliminary check because starting the plugins may add some overhead. Thus, the algorithm operates in two steps:
1) Filtering - checks for suspicious strings like the following (F), protocol keywords (e.g., http), and extensions of executable or script files (e.g., exe, php) (G); special characters (e.g., ; and |) (H); and attributes (e.g.,, >, href, and JavaScript). If none is located, the search is over.

2) Testing involves sending the appropriate plugin the input for review. For instance, if the href string is discovered during the filtering process, the information[19] is given to a plugin that can identify stored XSS threats[17]. This plugin calls an HTML parser to see if any additional tags emerge in the page that would indicate the presence of a script after inserting the input into a basic HTML page with the three major tags (html>, head>, and body>).

Consider a web application that registers new users, and a malicious client enters the JavaScript code script>alert("Hello!");/script> as the user's first name. The plugin that detects stored XSS attacks is called by SEPTIC after it filters the query and discovers two characters, and >, that are connected to XSS. This plugin calls an HTML parser, inserts this input into a webpage, and discovers that the input contains a script. So, it marks a store.

VII.IMPLEMENTATION

Data Owner:

The data owner uploads their data to the cloud server[21] in this module. The data owner encrypts the file and the index name for security reasons before storing them in the cloud. A specific file can be deleted by the data encryptor. He can also view transactions depending on the files he uploaded to the cloud and perform the following tasks: Data owners, please login and register. Add information on the military, the judiciary, government, and sports, as well as create digital signatures based on desc and ccat, cname, and cpublication. Browse and then enc data desc to upload, add an image. Observe all data contents with rankings and ratings and a digital signature.View all submitted information and rankings without a digital signature, Download required, see file.

Data User:

User logs in to this module using user name and password. After logging in, users can perform several actions including view their profiles. Obtain the application server's secret key and review the response, Search data by its keyword, read full details, and automatically take the secret key if permission is granted. Verify the file's signature before downloading. If the signature is incorrect, do not download. Models for building and managing queries. The method was tested[20] using a variety of applications, including open-source PHP web applications and synthetic code with intentional flaws included. This evaluation[12] suggested that the mechanism could identify the vulnerabilities in application code when attacks attempted to exploit them, perform better than all other tools in the literature and the WAF most commonly used in practice, and block the attacks it was programmed to handle. SEPTIC's performance overhead evaluation in MySQL shows an impact of roughly 2.2%, indicating that our method can be applied to real systems.

VIII.CONCLUSION AND FUTURE WORK

This study investigated[11] a novel method of database defense[4] against attacks on websites and enterprise applications. In order to protect the DBMS from SQLI and stored injection attacks, it proposed the idea of catching attacks inside the system. Furthermore, we demonstrated that it is possible to recognize and thwart sophisticated assaults[5], including those linked to the semantic mismatch[16] problem, by integrating protection within the DBMS. When attacks were discovered[23], a method of finding weaknesses in application code was provided as a second concept. In addition, SEPTIC, a method built into MySQL, was presented in this work. SEPTIC utilizes a learning phase, as well as ageing and quarantine procedures that deal with models for building and managing queries. The method was tested using a variety of applications, including open-source PHP web applications[18] and synthetic code[11] with intentional flaws included. This evaluation suggested that the mechanism could identify the vulnerabilities[6] in application code when attacks attempted to exploit them, perform [better than all other tools in the literature and the WAF most commonly used in practice, and block the attacks it was programmed to handle. SEPTIC's performance overhead evaluation in MySQL [25]shows an impact of about 2.2%, which suggests that our method can be used to real systems. With regard to models for creating and managing enquiries[24], SEPTIC makes use of a learning phase as well as ageing and quarantine procedures. Several programmers, including open-source PHP[7] web applications[8] and artificial code that contained errors[15] on purpose were used to test the method. The second proposal was a way for discovering flaws[9] in application code[10]. Additionally, SEPTIC, a feature of MySQL.

IX.REFERENCES

[1] An Article on reference of injection attacks

https://dl.acm.org/doi/abs/10.1145/3332371

[2] An Article on reference of application's database

https://eric.ed.gov/?id=EJ1329085

[3] An Article on book reference of SQL Injections

https://jis-eurasipjournals.springeropen.com/articles/10.1186/s13635-020-00113-y?ref=https://githubhelp.com

[4] An Article on book reference of database defense

https://onlinelibrary.wiley.com/doi/abs/10.1002/rnc.6551

[5] An Article on reference of sophisticated assaults

https://onlinelibrary.wiley.com/doi/abs/10.1002/rnc.5899

[6] An Article on reference of vulnerabilities

https://ieeexplore.ieee.org/abstract/document/9324852

[7] An Article on reference of PHP

https://link.springer.com/chapter/10.1007/978-1-4842-6791-2_2

[8] An Article on reference of web applications

https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-014-0123-5

[9] An Article on reference of discovering flaws

https://www.sciencedirect.com/science/article/abs/pii/S0167404811001684

[10] An Article on reference of application code

https://journals.sagepub.com/doi/abs/10.1177/0002716215569441?journalCode=anna

[11] An Article on book reference of synthetic code

https://www.sciencedirect.com/science/article/abs/pii/S0958166922000118

[12] An Article on book reference of investigated

https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/(SICI)1097-4571(2000)51:7%3C635::AID-ASI6%3E3.0.CO;2-H

[13] An Article on book reference of evaluation

https://www.sciencedirect.com/science/article/abs/pii/S1751157715300900

[14] An Article on book reference of weakness

https://appliednetsci.springeropen.com/articles/10.1007/s41109-017-0034-3

[15] An Article on book reference of errors

https://royalsocietypublishing.org/doi/full/10.1098/rspa.2020.0538

[16] An Article on reference of mismatch

https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0009393

[17] An Article on reference of threats

https://www.journals.uchicago.edu/doi/full/10.1086/710531

[18] An Article on reference of web applications

https://www.sciencedirect.com/science/article/abs/pii/S0950584919302551

[19] An Article on reference of information

https://journals.sagepub.com/doi/abs/10.1111/j.0956-7976.2004.00723.x?journalCode=pssa

[20] An Article on reference of tested

https://journals.lww.com/jbjsjournal/Abstract/2014/09030/Comparison_of_Ultrasound_and_Electrodiagnostic.15.aspx

[21] An Article on book reference of cloud server

https://www.emerald.com/insight/content/doi/10.1108/LHT-01-2021-0031/full/html

[22] An Article on book reference of administrator

https://www.jstor.org/stable/1058497

[23] An Article on reference of discovered

https://www.emerald.com/insight/content/doi/10.1108/RSR-07-2014-0023/full/html?journalCode=rsr

[24] An Article on reference of enquiries

https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1751-908X.2005.tb00904.x

[25] An Article on reference of MySQL

https://link.springer.com/article/10.1007/s42979-021-00716-3

X.BIBLOGRAPHY

Dr.K.N.S Lakshmi currently working as professor from department of computer science and engineering at sankethika vidya parishad engineering college,affailated to Andhra university ,accredicted by NAAC.madam is currently working as Head of the department ,published papers in various national& international journals.her subjects of interests machine learning,data mining&ware housing .

Manta.vasu deva kumar is Studying His 2nd Year, Master of Computer Applications In Sanketika Vidya Parishad Engineering College, Affiliated To Andhra University, Accredited By NAAC. With His Interest in Deep Learning And Machine Leaning Method As A Part Of Academic Project, he Used Detecting injection attacks and vulnerabilities inside the DBMS As A Result Of Desired To Comprehend The Flaws In Conventional Reporting And To Preserve Timely And High Quality Report Output In Detecting injection attacks and vulnerabilities inside the DBMS. A Completely Developed Project Along With Code Has Been Submitted For Andhra University As An Academic Project. In Completion Of His MCA.

JETIR