Detection of SQL Injection using Machine Learning: A Survey

Tareek Pattewar[1], Hitesh Patil[2], Harshada Patil[3], Neha Patil[4], Muskan Taneja[5], Tushar Wadile[6]

[1]*Assistant Professor, Dept. of Information Technology, R. C. Patel Institute of Technology, Maharashtra, India*
[2,3,4,5,6]*Student, Dept. of Information Technology, R. C. Patel Institute of Technology, Maharashtra, India*

Abstract - *In today's world, SQL Injection is a serious security threat over the internet for the various dynamic web applications residing over the internet. The web-page that accept critical information from users store this information in the form of online database. Web database is important because it's one of the major ways businesses can access information that isn't generated by itself. Using SQL Injection, attackers could even gain unrestricted access to an entire database. The threats include attacks such as Cross Side scripting (CSS), Denial of Service Attack (DoS) and Structured Query Language (SQL) injection attack. SQL injection attack fall under top ten vulnerabilities. There are various machine learning algorithms used for detection of SQL injection threats such as Naive Bias, Gradient Boosting and also various data encryption algorithm such as MD5, AES and combination of both not only detect but also analyze threats.*

***Key Words*: SQL Injection, Cross Side Scripting, Denial of Service Attack, Naïve Bias, Gradient Boosting**

1. INTRODUCTION

SQL injection is an attack technique that exploits a security vulnerability occurring in the database layer of an application. Hackers use injections to obtain unauthorized access to the underlying data, structure, and DBMS. By an SQL injection attacker can embed a malicious code in a poorly-designed application andthen passed to the back end database. Themalicious data then produces database query results or actions that should never have been executed. By using an SQL Injection vulnerability, given the right circumstances, an attacker can use it to bypass a web application's authentication and authorization mechanisms and retrieve the contents of an entire database.

SQL Injection can also be used to add, modify and delete records in a database, affecting data integrity. To such an extent, SQL Injection can provide an attacker with unauthorized access to sensitive data.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 (e.g. to dumpthe database contents to the attacker). SQL injection must exploitasecurityvulnerabilityinanapplication'ssoftware, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed. SQLinjectionismostlyknownas anattackvectorforwebsites but can be used to attack any type of SQL database. SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and become administrators of the database server.

A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to effect the execution of predefined SQL commands.

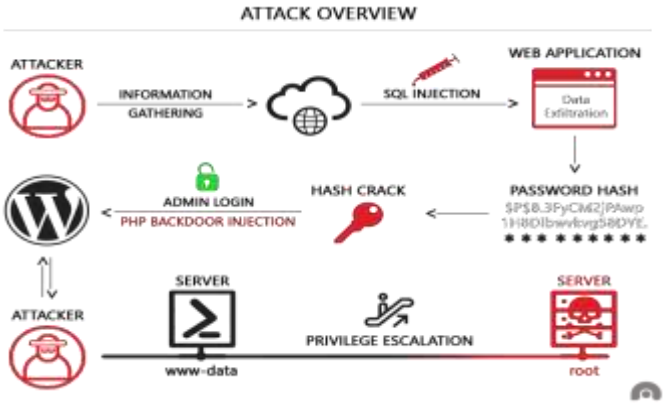

Fig-1: SQL Injection [1]

An SQL injection is a computer attack in which malicious code is embedded in a poorly-designed application and then passed to the backend database. The malicious data then produces database query results or actions that should never have been executed. A SQL injection (SQLi) is a type of security exploit in which the attacker adds Structured Query Language (SQL) code to a Web form input box in order to gain access to unauthorized resources or make changes to sensitive data.

1.1 How and Why Is an SQL Injection Attack Performed :

To make an SQL Injection attack, an attacker must first find vulnerable user inputs within the web page or web application. A web page or web application that has an SQL Injection vulnerability uses such user input directly in an SQL query. The attacker can create input content. Such content is often called a malicious payload and is the key part of the attack. After the attacker sends this content, malicious SQL commands are executed in the database.

SQL is a query language that was designed to manage data stored in relational databases. You can use it to access, modify, and delete data. Many web applications and websites store all the data in SQL databases. In some cases, you can also use SQL commands to run operating system commands. Therefore, a successful SQL Injection attack can have very serious consequences.

- Attackers can use SQL Injections to find the credentials of other users in the database. They can then impersonate these users. The impersonated user may be a database administrator with all database privileges.
- SQL lets you select and output data from the database. An SQL Injection vulnerability could allow the attacker to gain complete access to all data in a database server.
- SQL also lets you alter data in a database and add new data. For example, in a financial application, an attacker could use SQL Injection to alter balances, void transactions, or transfer money to their account.
- You can use SQL to delete records from a database, even drop tables. Even if the administrator makes database backups, deletion of data could affect application availability until the database is restored. Also, backups may not cover the most recent data.
- In some database servers, you can access the operating system using the database server. This may be intentional or accidental. In such case, an attacker could use an SQL Injection as the initial vector and then attack the internal network behind a firewall.

1.2 SQL Injection Attack Process:

SQLIA is a hacking technique which the attacker adds SQL statements through a web application's input fields or hidden parameters to access to resources. Lack of input validation in web applications causes hacker to be successful. For the following examples we will assume that a web application receives a HTTP request from a client as input and generates a SQL statement as output for the back end database server. For example an administrator will be authenticated after typing: employee id=112 and password=admin. Figure1 describes a login by a malicious user exploiting SQL Injection vulnerability. Basically it is structured in three phases:

1. an attacker sends the malicious HTTP Request to the web application
2. creates the SQL statement
3. submits the SQL statement to the back end database

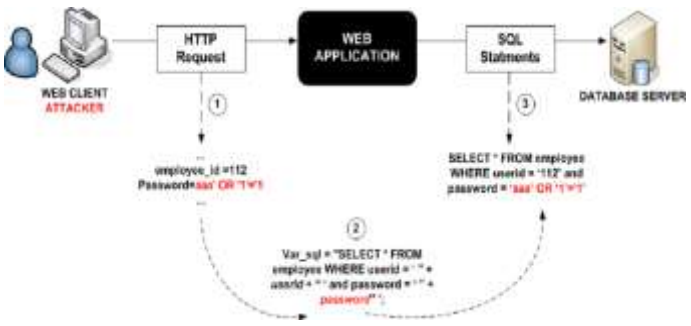

Fig-2: SQL Injection Process [2]

The above SQL statement is always true because of the Boolean tautology we appended (OR 1=1) so, we will access to the web application as an administrator without knowing the right password.

1.3 Types of SQL Injection (SQLi) :

SQL Injection can be used in a range of ways to cause serious problems. By levering SQL Injection, an attacker could bypass authentication, access, modify and delete data within a database. In some cases, SQL Injection can even be used to execute commands on the operating system, potentially allowing an attacker to escalate to more damaging attacks inside of a network that sits behind a firewall. SQL Injection can be classified into three major categories – In-band SQLi, Inferential SQLi and Out-of-band SQLi [7].

1. **In-band SQLi (Classic SQLi) :** In-band SQL Injection is the most common and easy-to- exploit of SQL Injection attacks. In-band SQL Injection occurs when an attacker is able to use the same communication channel to both launch the attack and gather results. The two most common types of in- band SQL Injection are Error-based SQLi and Union-based SQLi.

2. **Error-based SQLi :** Error-based SQLi is an in- band SQL Injection technique that relies on error messages thrown by the database server to obtain information about the structure of the database. In some cases, error-based SQL injection alone is enough for an attacker to enumerate an entire database. While errors are very useful during the development phase of a web application, they should be disabled on a live site, or logged to a file with restricted access instead.

3. **Union-based SQLi :** Union-based SQLi is an in-band SQL injection technique that leverages the UNION SQL operator to combine the results of two or more SELECT statements into a single result which is then returned as part of the HTTP response.

4. **Inferential SQLi (Blind SQLi)**:Inferential SQL Injection, unlike in-band SQLi, may take longer for an attacker to exploit, however, it is just as dangerous as any other form of SQL Injection. In an inferential SQLi attack, no data is actually transferred via the web application and the attacker would not be able to see the result of an attack in-band (which is why such attacks are commonly referred to as "blind SQL Injection attacks"). Instead, an attacker is able to reconstruct the database structure by sending payloads, observing the web application's response and the resulting behavior of the database server. The two types of inferential SQL Injection are Blind-boolean- based SQLi and Blind-time-based SQLi.

5. **Boolean-based (content-based) Blind SQLi:** Boolean-based SQL Injection is an inferential SQL Injection technique that relies on sending an SQL query to the database which forces the application to return a different result depending on whether the query returns a TRUE or FALSE result.Depending on the result, the content within the HTTP response will change, or remain the same. This allows an attacker to infer if the payload used returned true or false, even though no data from the database is returned. This attack is typically slow (especially on large databases) since an attacker would need to enumerate a database, character by character.

6. **Time-based Blind SQLi** : Time-based SQL Injection is an inferential SQL Injection technique that relies on sending an SQL query to the database which forces the database to wait for a specified amount of time (in seconds) before responding. The response time will indicate to the attacker whether the result of the query is TRUE or FALSE. Depending on the result, an HTTP response will be returned with a delay, or returned immediately. This allows an attacker to infer if the payload used returned true or false, even though no data from the database is returned. This attack is typically slow (especially on large databases) since an attacker would need to enumerate a database character by character.

7. **Out-of-band SQLi**: Out-of-band SQL Injection is not very common, mostly because it depends on features being enabled on the database server being used by the web application. Out-of-band SQL Injection occurs when an attacker is unable to use the same channel to launch the attack and gather results. Out-of-band techniques, offer an attacker an alternative to inferential timebased techniques, especially if the server responses are not very stable (making an inferential time-based attack unreliable).Out-of-band SQLi techniques would rely on the database server's ability to make DNS or HTTP requests to deliver data to an attacker. Such is the case with Microsoft SQL Server's xpdirtree command, which can be used to make DNS requests to a server an attacker controls; as well as Oracle Database's UTLHTTP package, which can be used to send HTTP Requests from SQL and PL/SQL to a server an attacker controls[2].

1.5 How SQL Injection Works:

Software developers create SQL queries to perform database functions within their applications. Each query has an argument that ensures only desired records are returned when a user runs the query. In a SQL injection, attackers exploit this argument by injecting malicious code into the input form. The first step of a SQL injection attack is to research how the targeted database functions. This is done by submitting a variety of random values into the argument field to observe how the server responds. The second step is using this information to craft an input value that the server will interpret and execute as a SQL command.

For example, a database may store information about customers that have made a purchase with customer ID numbers. Instead of searching for a specific customer ID, an attacker may insert the argument "CustomerID = 1000 OR 1=1" into the input field. Since the statement 1=1 is always true, the SQL query would execute to return all available customer IDs and any corresponding data.In addition to returning unauthorized information, SQL arguments could be written that delete an entire database, bypass the need for credentials, remove records or add unwanted data.

2. LITERATURE SURVEY

1. D. Richard Hipp designed SQLite in the spring of 2000 while working for General Dynamics on contract with the United States Navy. Hipp was designing software used for a damage- control system aboard guided missile destroyers, which originally used HP-UX with an IBM Informix database back-end. SQLite began as a Tcl extension[5].

2. The design goals of SQLite were to allow the program to be operated without installing a database management system or requiring a database administrator. Hipp based the syntax and semantics on those of PostgreSQL 6.5. In August 2000, version 1.0 of SQLite was released, with storage based on gdbm (GNU Database Manager). SQLite 2.0 replaced gdbm with a custom B-tree implementation, adding transaction capability. SQLite 3.0, partially funded by America Online, added internationalization, manifest typing, and other major improvements [2].

3. Huang and colleagues propose WAVES, a blackbox technique for testing web applications for SQL injection vulnerabilities. The tool identify all points a web application that can be used to inject SQLIAs. It builds attacks that target these points and monitors the application how response to the attacks by utilize machine learning[4].

4. Atefeh Tajpour , Mohammad Sharifi et al., had found various tools related to SQL injection attack. They studied SQL injection which is a type of attack where the attacker adds Structured Query Language code to a web form input box to gain access or make changes to data. SQL injection vulnerability allows an attacker to flow commands directly to a web application's underlying database and destroy functionality or confidentiality. Researchers have proposed different tools to detect and prevent this vulnerability. In this paper we present all SQL injection attack types and also current tools which can detect or prevent these attacks. Finally we evaluate these tools[4].

5. Dynamic Candidate Evaluations Approach, Bisht et al. propose CANDID. It is a Dynamic Candidate Evaluations method for automatic prevention of SQL Injection attacks. This framework dynamically extracts the query structures from every SQL query location which are intended by the developer (programmer). Hence, it solves the issue of manually modifying the application to create the prepared statements [6].

6. Puspendra Kumar, R.K. Pateriya et al., found different SQL detection technique. SQLInjection poses a serious security issue over the Internet or over web application. In SQL injection attacks, hackers can take advantage of poorly coded Web application software to introduce malicious code into the organization's systems and network.The vulnerability exists when a Web application do not properly filter or validate the entered data by a user on a Web page. Large Web applications have hundreds of places where users can input data, each of which can provide a SQL injection opportunity. Attacker can steal confidential data of the organization with these attacks resulting loss of market value of the organization. They presents an effective survey of SQL Injection attack, detection and prevention techniques[6].

7. Livshits et al. use static analysis techniques to detect vulnerabilities in software. Java Static Tainting uses information flow techniques to detect when tainted input has been used to make a SQLIA. The primary limitation of this approach is that it can detect only known patterns of SQLIAs and it can generate a relatively high amount of false positives because it uses a conservative analysis.Static analysis, also called static code analysis, is a method of computer program debugging that is done by examining the code without executing the program [2].

8. Xiang Fu et al. proposed the design of a static analysis framework, called SAFELI for identifying SQLIA vulnerabilities at compile time. SAFELI statically monitor the MSIL (Microsoft Symbolic intermediate language) byte code of an ASP.NET Web application, using symbolic execution. SAFELI can analyze the source code and will be able to identify delicate vulnerabilities that cannot be discovered by black-box vulnerability scanners. The main drawback

of this technique is that this approach can discover the SQL injection attacks only on Microsoft based product [8].

9. Nguyen-Tuong et al. proposed a PHP interpreter to track precise per character taint information. A context sensitive analysis is used to detect and reject queries if certain types of SQL tokens has been constructed by illegitimate input. Limitation of these two approaches is that they require rewriting code [5].

10. Buehrer et al. used similar approach of comparing SQL Injection Detection Using Machine Learning 12 the actually generated queries with the one that should have been generated (programmer intended). The only difference in this approach is that it achieves the results by using Parse Trees.

3. METHODOLOGY

3.1 ALGORITHM:

SQL Injection can be considered as one of the most serious attacks, as it influences the database and can make serious damage to your data and the whole system. For sure it can have more serious consequences than a Javascript Injection or HTML Injection, as both of them are performed on the client-side. For comparison, with this attack, you can have access to the whole database. It should be mentioned, that to test against this attack, you should have quite good knowledge of SQL programming language and in general, you should know how databases queries are working. Also while performing this injection attack you should be more careful and observant, as any inaccuracy can be left as SQL vulnerabilities.

3.1.1 Machine LearningAlgorithm

The use of machine learning algorithms to detect and prevent various cyber security threats is being debated largely. While the power of using supervised and unsupervised learning techniques to detect security threats cannot be questioned, the computing resources and time required to execute such complex algorithms remains a major concern for the ever advancing cyber security community. Tremendous research work has been done on using various machine learning algorithms to detect SQL Injection attacks. There is no single perfect algorithm or technique in machine learning that can be applied to a particular problem. A problem needs to be tested against various algorithms falling under classification or regression techniques, and the results need to be compared, before finalizing a particular approach, for maximum accuracy. SQL Injection detection using Na¨ıve Bayes algorithm has been implemented in previous researches. It use an approach called Gradient Boosting algorithm to detect and prevent SQL Injection attacks. It also implemented the Na¨ıve Bayes algorithm and compared the results against Gradient Boosting for this particular problem. It begin with an introduction to SQL Injection attacks and the need and motivation to build a better SQL Injection detection system. It then understand the SQL Injection attacks All the significant implementations so far provides enough literature review to learn from and improve on the problem.

1. Na¨ıve Bayes 2. Gradient Boosting.

1. **Naïve Bayes :** Na¨ıve Bayes algorithm has already been implemented for detecting SQL Injections . Na¨ıve Bayes is a classification model in supervised learning that is based on Bayes Theorem. The essence to Na¨ıve Bayes is that it assumes that the presence of a feature in a data model is unrelated to the presence of other features. In short it assumes that all the features in a data are conditionally independent of each other, hence it gets its name 'Na¨ıve Bayes'.

2. **Bagging and Boosting:**

Bagging - Bagging is an ensemble learning approach that predicts a value of data by using multiple supervised learning models and then combining the results of all these individual learning models by a chosen technique. The technique used for combining these results could be any including by weighting the results, taking their average, voting for the maximum result, etc. Bagging is also called as Bootstrap Aggregation. Bagging can help with reducing variance errors by using multiple supervised learning models and then combining their result. An example of bagging technique is Random Forest Algorithm. In this approach, multiple decision trees are created on random subsets of training data and results are collected from each decision tree. A final result is then selected from these results by taking an average of all the results from individual supervised learning models.

Boosting - Boosting on the other hand is an Ensemble learning approach that also uses multiple supervised learning models in combination to provide better predictive results. The difference is the way in which boosting uses these multiple models. Instead of using them in parallel, in boosting the multiple models are used sequentially. In this technique each predictor model learns and tries to minimize the errors from the previous predictor model. Boosting algorithms can reduce the bias errors introduced due to small size of datasets. An example of boosting algorithm is

Gradient Boosting.

Gradient Boosting - Gradient Boosting approach is used to classify and detect SQL Injection attacks. One of the important reasons of choosing this approach is because not enough data is available to train the machine learning models. Na¨ıve Bayes technique has been implemented to detect SQL Injection attacks because it can be trained even on small datasets. However, that can lead to high bias errors as it is possible that data will not be classified correctly all the time.The hope is that using Gradient Boosting approach results in better accuracy while classifying the SQL Injection queries and overall provides better results and higher ratio of detecting an SQL Injection attack.

3.1.2 Encryption Algorithm:

MD5 Algorithm:

MD5 algorithm was developed by Professor Ronald L. Rivest in 1991. According to RFC 1321, "MD5 message- digest algorithm takes as input a message of arbitrary length and produces as output a 128-bit quot; fingerprintquot; or quot; message digestquot; of the input. The MD5 algorithm is intended to use for digital signature applications, where a large file must be quot; compressed quot; in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA." The input to the MD5 algorithm is first divided into blocks of 512 bits each. And to record the length of the original message, 64 bits are padded at the end of the message. And some bits are padded to the message if the length of the message is less than 512 bits. After this each block is divided into 16 words of 32 bits which are denoted as M0. . . ..M15.MD5 algorithm uses a buffer, table and four auxiliary functions for producing an output.

AES Algorithm:

Algorithm AES (Advanced Encryption Standard) is a symmetric encryption algorithm and was developed by two Belgian cryptographer Joan Daemen and Vincent Rijmen. It was designed to be efficient in both hardware and software, and supports a block length of 128 bits and key lengths of 128, 192, and 256 bits. This encryption algorithm is used by

U.S. for securing the sensitive and unclassified material, so it is secure enough with high securities. The three block ciphers: AES-128, AES-192 and AES-256. Each cipher encrypts and decrypts data in blocks of 128 bits using cryptographic keys of 128-, 192- and 256-bits, respectively. And AES performs all its computations on bytes rather than bits. Hence, AES treats the input plaintext of 128 bits block as 16 bytes.

3.1.3 Dataset

Gathering a dataset for this problem was challenging as no datasets with public access to actual SQL Injection attacks that were launched are available. The recommendation is to always send SQL through Command structures, such as Prepared Statements or Stored Procedures, rather than building up the SQL directly, which is open for SQL injection. This dataset contains around SQL Injections of all the three types, that are, Union Based, Error Based and Blind SQL Injections.

```
1 SELECT 6 < 1;
1 SELECT ascii('A');
1 SELECT cast('1' AS unsigned integer);
1 SELECT cast('123' AS char);
1 SELECT CONCAT('A','B');
1 SELECT CONCAT('A','B','C');
1 SELECT if(1=1,'foo','bar');
1 SELECT CASE WHEN (1=1) THEN 'A' ELSE 'B' END;
1 SELECT 0+414243;
1 SELECT BENCHMARK(1000000,MD5('A')); SELECT SLEEP(5);
1 UNION ALL SELECT LOAD_FILE('/etc/passwd')
1 SELECT * FROM mytable INTO dumpfile '/tmp/somefile';
1 SELECT @@hostname;
1 CREATE USER test1 IDENTIFIED BY 'pass1';
1 DROP USER test1;
1 GRANT ALL PRIVILEGES ON *.* TO test1@'%';
1 SELECT @@datadir;
1 admin'--
1 admin'#
1 ;DA/**/OP tempTable;
1  or 1=1--
1 admin'/*
1  or 1=1#
1  or 1=1/*
1 ) or '1'='1--
1 ) or ('1'='1--
1  or ''='
```

3.2 SQL Injection Detection Tool :

As SQL injection attacks exploit vulnerable Web application and database code, the only way to prevent them is to resolve your code39;s vulnerabilities. Any place that code dynamically generates a SQL query using data from an external source should be closely checked. On larger projects, you should look at using automatic source code scanning tools and Web vulnerability

scanners.

A good Web vulnerability scanner will spot common technical vulnerabilities, such as SQL injection flaws, cross-site scripting vulnerabilities, parameter tampering, hidden field manipulation, backdoors, debug options and buffer overflows. If you use any third-party applications that utilise a database back- end, it39;s vital that that you follow any vendor updates regarding vulnerabilities and patches to ensure the new code isn39;t introducing vulnerabilities into your own system. Even if your database administrators and application developers are following best practice, I would still recommend the deployment of an application-layer firewall or Web application firewall (WAF). WAFs can provide protection beyond that of traditional network firewalls and intrusion detection/prevention systems. Many, like those produced by Imperva Inc. and Barracuda Networks Inc., can help prevent attacks such as SQL injection, cross-site scripting and others that target flaws in application logic or technical vulnerabilities in software.

Best-of-breed WAFs can recognise evasion techniques exploited by attackers using SQL injection, such as obfuscating the attack by encoding portions of the injected command. Your chosen application-layer firewall should also allow you to create filters to intercept, analyze or modify traffic specific to your network.

Filters make it easier to adapt the firewall to protect assets or monitor traffic specific to your network. Even better if it has the capability to quot;learnquot; what is and what isn39;t normal traffic for your specific network and adapt its behavior accordingly. When irregularities are detected, the WAF can shut down potential attacks while they39;re happening. Also as SQL injection often takes place via the URL query string, you should regularly review your Web server39;s logs to look for anomalous queries that may be injection attempts.

3.3 System Architecture:

The raw dataset consist of SQL queries. These queries are used for authentication bypass. There are various types of SQL queries merged together to feed up the machine learning algorithm. Feature Extraction is done after merging SQL queries. Feature Extraction involves reducing the number of resources required to describe a large set of dat . Labelling consist of group of samples that have been tagged with one or more labels. Classification is done with various algorithms of machine learning. It classifies the dataset as it is malicious or not.

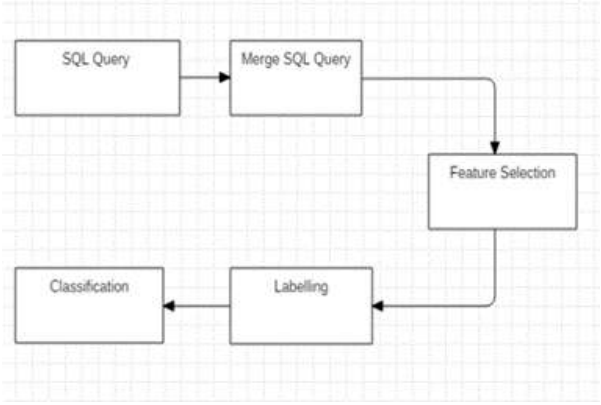

Fig-3: System Architecture

4. CONCLUSION

SQL injection attack is a very serious problem of web applications. Finding the efficient solution of this problem is essential. Researchers have developed many techniques to detect and prevent this vulnerability. There is no appropriate solution that can prevent all types of SQL injection attacks. SQL Injection attacks remain to be one of top concerns for cyber security researchers. Signature based SQL Injection detection methods are no longer reliable as attackers are using new types of SQL Injections each time. There is a need for SQL Injection detection mechanisms that are capable of identifying new, never before seen attacks. Applying machine learning to the field of cyber-security is being considered by many researchers. Two machine learning classification algorithms are implemented on the problem, which are, Na¨ıve Bayes Classifier and Gradient Boosting Classifier. Na¨ıve Bayes classifier machine learning model provides results with an accuracy of 92.8%. Ensemble learning methods are said to provide results with better accuracy as they implement multiple simple classifiers to improve error and accuracy. Hence Gradient Boosting Classifier from ensemble learning is selected to be implemented on the SQL Injection classification problem.

REFERENCES

[1] Sonali Mishra , "SQL Injection Detection using Machine Learning ", from https:

//scholarworks.sjsu.edu/cgi/viewcontent.cgi?arti c le = 1727context = etdprojects, on 23 May 2019 pp.10 - 29.

[2] Bojken Shehu and Aleksander Xhuvani ,"A Literature Review and Comparative Analyses on SQL Injection: Vulnerabilities, Attacks and their Prevention and Detection Techniques" from https

://pdfs.semanticscholar.org, Vol. 11, Issue 4, No 1, July 2014 pp 20 - 34.

[3] Suhaimi Ibrahim , " SQL Injection Detection and Prevention Techniques" from https

://pdfs.semanticscholar.org/ Volume 3, Number 7, August 2011 , pp 85 - 89.

[4] G.Wassermann, Z.Su, "An analysis framework for security in web applications," In: Proceedings of the FSE Workshop on Specification and Verification of Component-Based Systems, from https://link.springer.com/chapter/10.1007/978 – 0 – 387 – 44599 – 15 SAVCBS, pp. 70–78, 2004.

[5] Mei Junjin, "An Approach for SQL Injection Vulnerability Detection," Proceedings. of the 6th Int. Conf. on Information Technology: New Generations, Las Vegas, Nevada, pp. 14-19, Apr. 2009.

[6] G Buehrer, B.W. Weide, P.A.G Sivilotti, Using Parse Tree Validation to Prevent SQL Injection Attacks, in: 5th International Workshop on SoftWore Engineering and Middlere, Lisbon, Portugal, 2005, pp. 106-113.

[7] Shikhar Jain Alwyn R. Pais," Model Based Approach to Prevent SQL Injection Attacks on.NET Applications" International Journal of Computer Science Informatics, Volume-1, Issue-11, 2011.

[8] Haripriya Rana and Shelly Sachdeva , "Analysis of SQL Injection Detection and Prevention" vol 10 , August 2017, Proceeding from Indian Journal of Science and Technology, pp 5-9.

[9] Abe Miessler , Dataset from https :

//github.com/danielmiessler/SecLists