

## **Project 4**

### **Automation based SQL Injection Defence Technique.**

As we have studied earlier that the SQL injection technique is the one used to input malicious entries into the input to gain access to the server. The basic mechanism of the SQL injection technique is that the attacker can input such a statement which will be always true and thereby allowing him to gain access to the database and alter or tamper the data. This is a very risky affair as it can cause destruction or damages on a very huge level. The main defence techniques that are explained earlier were Verification based and observation-based techniques.

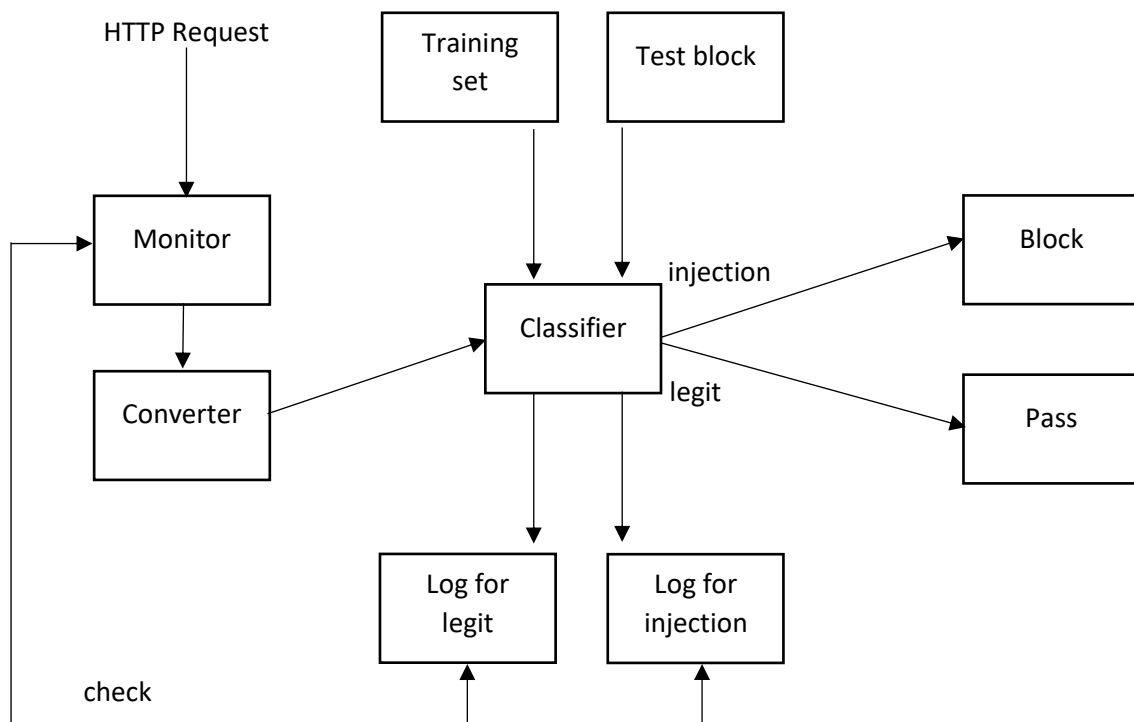
In verification-based technique what we do is that we write code in such a way that the special characters which are used by the attacker to gain access of the database are filtered i.e. are not allowed in the form of input by the server or the web application. Basically, what we do is we set a condition to check that the special characters are not in the input fields. If they are present the attacker won't be able to login. As far as the observation-based technique is concerned, here instead of preventing the attacker from gaining access to the database, what we do is that we redirect him to some bogus website which he thinks is legit but isn't. He thinks that he has gained access to the server but in reality, he is just accessing a fake website created by us to monitor him continuously. Where we keep a track of his activities thereby "observing him". These techniques are necessary as the attacker, if he gains access to the vulnerable data can leak it online and can cause losses to the firms/users.

There is one more technique apart from the above-mentioned verification and observation based, which is implemented using machine learning. This technique will make use of a classifier to filter the various malicious spam keywords. The classifier used for this is known as Bayesian classifier. What we do is basically we train the algorithm to filter spam keywords which we think are spam and can cause damage to the application. There is no modifying of source code of the web application.

#### **Bayesian Classifier:**

Here we set the parameters, the training set factors and the attributes are of major significance for filtering the malicious spam keywords. The major components used here are the monitor, classifier and a converter. Basic mechanism of working is first we send the HTTP request from the web application and send it to the monitor. The monitor sends it to the converter. The converter then sends the data to the classifier which in turn checks whether the data is legal or an injection. If it is an injection then it is blocked and further entry is restricted. If it is legal then the entry is passed.

The condition that whether the data is legal or an injection is decided by the classifier which has a predefined training set. This set contains the test samples as defined by the developer.



### **Monitor:**

Monitor basically is the first stage through which the data has to pass. The data is sent through either HTTP POST or HTTP GET methods. Here URL Decoding is required. What happens in URL decoding is that when the user inputs his credentials, the URL decoding only allows a certain set of characters to pass which is of some significance to the server and validation. Here in our case we make sure that after decoding all the characters are converted to upper case. This is done so that the characters are in unified format and the attacker cannot get access to the database by using mixed case characters.

### **Converter:**

The converter gets input from the monitor in the form of HTTP parameters. The parameters are to be converted in the form of numeric so as to be understood by the classifier. The classifier is the one which decides whether the data is legal or an injection. The two main attributes are the pattern attributes and the keyword set. In pattern attributes, the main two parameters are length and the number of keywords. As for SQL injection, the main concern is the use of special characters which when used properly can cause the query statement to be a Boolean value of true and thereby bypassing the security check and allowing the attacker access to the database. Usually the characters are single quotes, hash, dollar sign, equal to sign and so on. There are also keywords that are used such as SELECT, FROM, DROP etc. Also, there are spaces used, spaces are considered as a character. For any query there is a requirement of space. For example: SELECT \* FROM users WHERE username = "xyz" and so on. Here the keywords are 3 viz. SELECT, FROM, WHERE and characters are 43.

### **Classifier:**

The input to the classifier is given by the converter. The classifier is the one who is responsible for the decision of the keyword being spam or not. This classifier used is Bayesian classifier. This classifier is constructed using Bayesian decision theory. The probability of classes is calculated by using Bayes equation. On the basis of the probability, the one with the highest value is decided to be a spam keyword.

Here, the attributes specified are denoted by  $X$ . if the no of patterns are 'n' then  $X$  is denoted by:

$$X = \{(X_1, t_1), (X_2, t_2) \dots (X_n, t_n)\}$$

$X_n$  is the collection of attributes of all the patterns and is represented as  $X_i = (x_1, x_2, \dots) T$

$T_i$  is the set of all  $w$ 's such that:  $T_i = \{w_1, w_2, \dots, w_m\}$

Attributes from the samples are determined to be parameters of the functions. This function is solely responsible for the calculation of the probabilities which are based on the Bayes equation that is:  $P(X|w_i) P(w_i)$ .

### **Algorithm:**

The following is the algorithm for the Bayes classifier:

Input:  $X$

Output: None.

- Begin
- $g_1(X) = P(X|w_1) (w_1)$ ;
- $g_2(X) = P(X|w_2) (w_2)$ ;
- Set  $k$  as the maximum value between  $g_1(X)$  and  $g_2(X)$ ;
- If ( $w_k ==$  Injection pattern);
- Declare "SQL Injection";
- Insert into Injection Log;
- Execute Block Process;
- Else;
- Declare "Legit";
- Insert into Legit Log;
- Execute Pass block;
- End.

Here in this classifier basic working of the algorithm is that two probabilities are calculated. One probability is of the injection string and other is of the legit string. As per the result of the probability the next tasks are executed. If the probability of the legit string is higher then it is executed and the log in the legit log is written. If the probability of the injection string is higher, then it is executed and the process is stopped thereby denying access to the database and logs are noted in the injection log.

The classifier has to be trained beforehand in order for it to execute the one of the two above possibilities. Hence predefined sample sets are defined in the code so that the algorithm can learn which strings are legal and which are not. Now here what we can do is manually enter various strings that we think are spams or simply create a string pattern generator. This generator will generate the legal and illegal strings. For example: Strings such as '1'=1 is an illegal string and it can be used by the attacker as input in username and password to generate an SQL query which will be always true.

We can train the algorithm to block such illegal characters.

### Screenshots:

```
$classifier -> train("1 or 1 = 1", $spam);
$classifier -> train("1 or 1 = 1 and user <> pranit", $spam);
$classifier -> train("105; DROP TABLE Users", $spam);
$classifier -> train("105; INSERT into TABLE Users", $spam);
$classifier -> train("1 or 1 = 1", $spam);
$classifier -> train("1 or 1 = 1 and user <> kewal", $spam);
$classifier -> train("Lucky draw!", $spam);
$classifier -> train("delete * from database", $spam);
$classifier -> train("select * FROM db", $spam);
$classifier -> train("won", $spam);
$classifier -> train("$$, dollars", $spam);
$classifier -> train("105; DROP TABLE Users", $spam);
$classifier -> train("drop * <>", $spam);
$classifier -> train("lottery", $spam);
$classifier -> train("select * from database", $spam);
$classifier -> train("1 OR 1=1 and drop", $spam);
$classifier -> train("drop * from database", $spam);
$classifier -> train("this is a spam username and password", $spam);
$classifier -> train("the longer the string the more possibilities of it being an injection query", $spam);
```

In the above screenshot, we train the classifier for spam keywords.

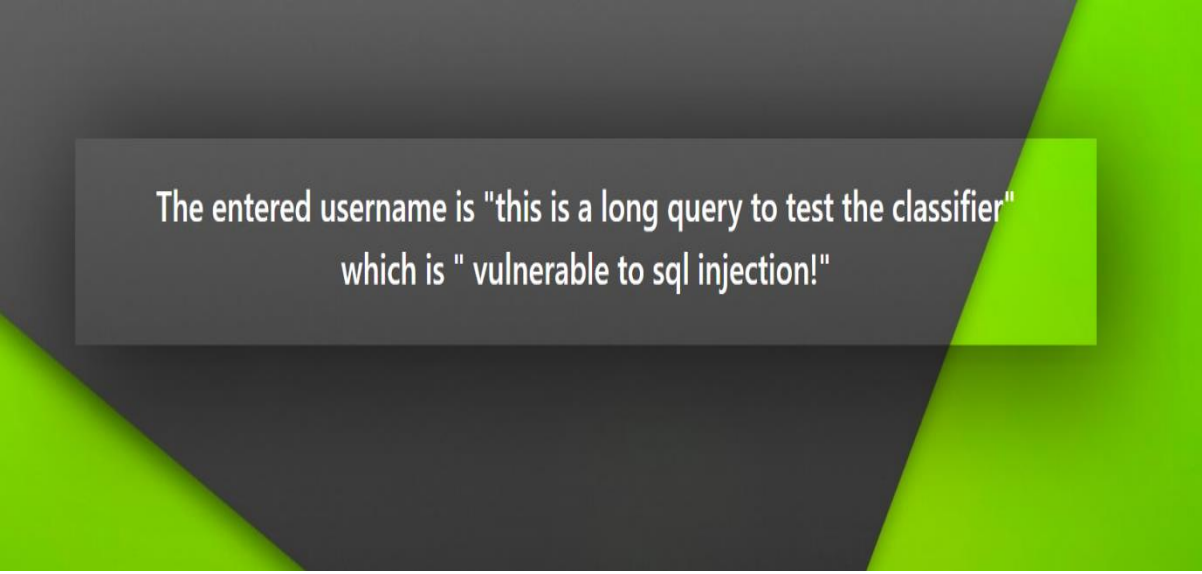
```
$classifier -> train("kewal", $ham);
$classifier -> train("Amy", $ham);
$classifier -> train("John", $ham);
$classifier -> train("kewalg", $ham);
$classifier -> train("myname", $ham);
$classifier -> train("p1@34", $ham);
$classifier -> train("Kewal", $ham);
$classifier -> train("Jackson", $ham);
$classifier -> train("kewal@123", $ham);
$classifier -> train("Ron", $ham);
$classifier -> train("kewalg", $ham);
$classifier -> train("kgulve1", $ham);
$classifier -> train("kewal123", $ham);
$classifier -> train("agujara1", $ham);
$classifier -> train("spiderman@web", $ham);
$classifier -> train("tony.stark", $ham);
$classifier -> train("hulk", $ham);
$classifier -> train("goku@DBZ", $ham);
$classifier -> train("gohan@DBZ", $ham);
$classifier -> train("chelsea.fc", $ham);
```

In the above screenshot, we train it for all the legit keywords that are possible.



The entered username is "user1"  
which is " safe from sql injection"

The above username entered is user1 which is a small string and is safe from sql injection as its probability is calculated from the database count.



The entered username is "this is a long query to test the classifier"  
which is " vulnerable to sql injection!"

The above username is just a random long string which is entered for testing the classifier's calculation of probability. Here it isn't declared anywhere in the database, but still the classifier considers it as a vulnerable string.

### **Final Thoughts:**

SQL Injection is a very harmful attack when executed properly, as it can cause damage both financially as well as physically. It is a must to have some counter measures that prevent this attack from happening. For this I have covered three main types namely: Verification based defence mechanism, Observation based defence mechanism and this one, the Automation based defence mechanism.

Verification based makes sure that the specified special characters are not permitted to be entered. Observation based makes sure that the attacker is convinced that he has gotten access to the database but in reality, he is just being redirected to a fake webpage created by us to observe and monitor him.

Lastly, in the Automation based defence mechanism, we can prevent the attacker from getting access to the database by predefining certain keywords and queries into the code. Although this method considers various aspects such as attributes, parameters and keywords, we cannot underestimate the fact that there is a possibility of it being improper and miscalculation of probabilities. But most of the times this system works properly and detects attacks.

These are the main types of SQL Injection defence mechanisms.

### **References:**

1. <https://www.google.com/>
2. [https://en.wikipedia.org/wiki/Naive\\_Bayes\\_classifier](https://en.wikipedia.org/wiki/Naive_Bayes_classifier)
3. <https://ieeexplore.ieee.org/document/6993127/>
4. [https://www.ijarse.com/images/fullpdf/1456154830\\_365S.pdf](https://www.ijarse.com/images/fullpdf/1456154830_365S.pdf)
5. <https://arxiv.org/abs/1605.02796>