International Journal of Applied Information Systems (IJAIS) ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 12 - No. 37, June 2021 - www.ijais.org

An Approach to Detect Cyber Attack on Server-side Application by using Data Mining Techniques and Evolutionary Algorithms

Abu Syeed Sajid Ahmed, Afsana Afrin Brishty, Mehjabeen Shachi,
Nurnaby Siddiqui Shourav, Nazmus Sakib
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology, Dhaka-1208, Bangladesh

ABSTRACT

Cyber Attack is one of the biggest problems for people of different levels, especially for the industries, which can maliciously disable systems, steal data. It is an assault launched by cyber criminals using one or more computers against single or multiple computers or networks. Server-side attacks are launched directly from an attacker to a listening service. Server-side attacks want to compromise and infringe with data and applications on a server. Applications like web browsers, media players, email servers, office suites, and similar applications are the main targets for attackers. An injection attack is one of the most common types of attack in which the hacker can steal valuable information from the database or server and it is the most dangerous attack aimed at web applications and can lead to data theft, data loss, loss of data integrity, denial of service, as well as full system compromise. Malicious requests make it easier for attackers to attack server-side applications. Our idea has been demonstrated in this paper where a two-layer security firewall is implemented in the server-side application to detect malicious code(SQL/NoSQL injection) using both machine learning and non-machine learning approach. The first layer of the firewall that will be placed between controller and router will be responsible for detecting malicious code from the request object using input validation and a parameterized statement which is a non-machine learning approach. Moreover, the second layer of the firewall will be placed between the controller and database to detect malicious code from the query using a machine learning model. We use text mining for feature extraction from the query, GridSearchCV for best model evaluation and genetic algorithm for automated hyper-parameter optimization.

Keywords

Cyber Attack, SQL(Structured Query Language) Injection, NoSQL(Non-Structured Query Language) Injection, Machine learning, Data mining, Evolutionary algorithms.

1. INTRODUCTION

Every organization and company needs servers and databases to store valuable, confidential data due to its reasonable price, performance speed and some automated features for managing data. The security of these server-side applications in today's world is one of the most important and challenging tasks that people all over the world face [1]. The cyber attack has increased every year as people try to benefit from vulnerable systems which shows the path that needs to incur the importance on the security of these server-side applications that is not only today's world is one of the most important tasks but also high level challenging tasks that people all over the world. One of the main findings of a research paper is that the public sector continues to dominate as the primary target of cyber attacks followed by financial services [2]. In Bangladesh, a hacker group called Hafnium has attacked more than 200 organizations, including the Bangladesh Telecommunications Regulatory Commission (BTRC), Bangladesh Bank, commercial banks, and Internet service providers. Hafnium fundamentally targets entities in the United States across several industry sectors, including law firms, infectious disease researchers, defense contractors, higher education institutions, NGOs, and policy think tanks. This detrimental group has extensions in tactics and techniques with other Chinese hacker groups. This group also attacked Germany, Canada, France, Belgium, Italy, Hong Kong, South Korea, Turkey, United Arab Emirates, and Israel, etc [3]. According to a recent report, between January 1, 2005, and May 31, 2020, there have been 11,762 recorded breaches [4]. In recent years, the number of SQL injection attacks have been growing so fast and eventually, it became the topmost type to attack database based web applications. The average daily number of SQL injection attacks has already reached almost 400,000 worldwide [1].

In 2013, a cyber attack occurred in Yahoo's 3 billion email accounts gaining access to sensitive customer information. In the same year, cyber attackers used malware to steal data from the target company point of sale systems compromising information of approximately one hundred and ten million credit/debit carrying customers. "Peace" a Russian-based cyber attack group infiltrated LinkedIn stealing email and password combinations of over 117 million customers in 2015. In 2020, the hotel chain Marriott disclosed a security breach that impacted the data of more than 5.2 million hotel guests who used their company's loyalty application. In 2020, 500,000 stolen Zoom passwords became available for sale in dark web crime forums [5]. These are just a few of the many incidents. Cyber attack is becoming more excruciating day by day comprising lots of economic, reputational, social, and societal harm. In most cases of applications, developers focus on usability and functionality. Security issues come as an afterthought. For example, using unfiltered input as a parameter to HTTP requests might permit a malicious user to execute an SQL/NoSQL injection attack. The hacker can steal all the valuable information from the database using this terrible attack. A successful injection attack can result in unauthorized access to sensitive data such as passwords, credit card details, or personal user information and leads to respectable damage and regulatory penalties. In some cases, the attacker can gain an endless backdoor to any organization's system. Therefore, it is essential to detect injection attacks before executing a database query. To solve this problem many models and tools have been developed for predicting vulnerabilities of a software component. Usually, such methods are limited to fixed, very small patterns, and hardly adapt to variations. Moreover, these methods

depend on parsing the code [6]. To improve the ability to predict web application vulnerabilities, a wide variety of data mining and machine learning techniques have been used. For example, feature extraction and classification are used to predict, if SQL injection vulnerability resided in the software or not [7].
To control cyber attacks to a great extent in the server-side application, our proposed idea is detecting SQL or NoSQL injection attacks using a two-layer security firewall. After receiving the HTTP request object, it is passed to our proposed first security layer via middleware to detect malicious code from the request object using the non-machine learning approach. If the first layer of security detects malicious code in the request object, then the request will be blocked to go further. Moreover, If the first layer of the security firewall fails to detect malicious code in the request object despite having malicious codes, then the second layer will detect the malicious code from the query that is constructed from the data of the request object using a machine learning model. To detect SQL injection attacks, the data-mining-based method is favourable in detecting unknown attacks with high accuracy against the rapid emergence of various types of attack [8]. A NoSQL attack targets the interactive web applications that are associated with any type of NoSQL database, such as MongoDB. In this sort of attacks, an attacker injects code into a NoSQL query which could result in an alternate database request that can peruse or modify a NoSQL database or change data on a web application [9][10][11].
The main contribution of the implementation paper are :

(a) A two-layer security firewall in a server-side application has been proposed. The objective of the first layer is to detect injection attacks using the analysis of request object data by user input validation and parameterized statement. On the other hand, the objective of the second layer is to detect injection attacks using two classifiers. One classifier is for detecting SQL injection and another one is for detecting No-SQL injection attacks.
(b) Genetic algorithms will be used to automatically optimize the hyperparameter of the models and also used to find the combinations of features that produce the best-performing classification model.
(c) A comprehensive comparison of several well-known machine learning algorithms performance on both balanced and imbalanced dataset has been published for detecting SQL and No-SQL injection attacks from the query analysis.GridSearchCV is used for optimizing the hyperparameters of those models manually.

The rest of this paper is organized as follows. Sections 2 and 3 describe the literature review and proposed idea. Section 4 and 5 provide details of the implementation and results. Finally, section 6 draws the conclusion of this paper with a few comments and suggestions on future research.

2. LITERATURE REVIEW

2.1 Http Server

An HTTP server is defined as a computer that is hosted on the internet, processes requests via hypertext transfer protocol used to transfer information on the internet. The main function of an HTTP server is to store, process and deliver web pages, JSON data to clients. HTTP server receives the requests from client-side applications like browser, android device, process the requests, send the pages as Html documents which may include images, texts, scripts etc. or send the JSON data as a response after querying from the database.HTTP verbs(POST/PUT/GET/DELETE) are used to define that what kind of operation is going to be executed after receiving a request from a client application.POST requests are made for adding a resource to the server, PUT requests are made for updating a resource, GET requests are for getting resources from the server and DELETE requests are made for removing a resource [12]. In the implementation, our proposed two layer security firewall is placed on a http server.

2.2 Middleware

Middleware can be defined as a component that has a set of functions that has access to request and response objects, can modify the request object and the response object for parsing the request body, adding response header etc. It acts as a link between client and server. The main purpose of middleware is to provide a mechanism for filtering HTTP requests before reaching the business logic process. Middlewares are chained together to do HTTP request filter processes one by one. This middleware handles reading and writing the HTTP session, determining if the application is in maintenance mode, verifying the CSRF token, authentication token, refresh token and more [13]. It is also used for managing transactions and ensuring that any problems can't corrupt the system or database server [14]. There are many kinds of input sanitization and security-related tasks that are executed in the middleware. If the received request is valid, then middleware will allow the request object to go to the router component. If the received request is not valid or malicious then it will block the request to go further.

2.3 Router

The router can be defined as a component that is used to receive the request through a URL like ”http://www.mydomain.com/path” and pass the request to a single controller function [13].

2.4 Model-View-Controller(MVC)

MVC is a well-known design pattern used in both client-side and server-side applications. Developers suggest MVC design patterns to improve the system and make the code quality better. In this design pattern, controllers receive the request from the router, extract the data from the request object and send the data to a model. Then, the controller function executes the SQL/No-SQL query to fetch data from a database or insert data into the database. The presentation layer of the MVC system is the view. It uses data from the model, which is primarily supplied by the controller, to generate output for the user. A view may also have a helper that retrieves data [15][16][17]. In our implementation,the second layer of our proposed security firewall performs dynamic analysis of the query which is formed inside the controller functions.

2.5 Database

A database is an organized collection of data so that it can easily be accessed and manipulated. The main purpose of the database is to manage a lot of data by storing, retrieving and updating. There are many kinds of databases but we are mainly focusing on the security of SQL and No-SQL databases in our proposed paper.

2.5.1 SQL.

SQL database may be defined as a database system that uses SQL queries to store, update, get or delete the data. It is also called a relational database because the stored data is connected through a predefined schema presenting the relations between the data [18]. There are many SQL databases like Oracle, PostgreSQL, MySQL, MSSQL etc.

2.5.2 NoSQL.

A NoSQL database is defined as a non-relational database or distributed database system that doesn't require the data to have

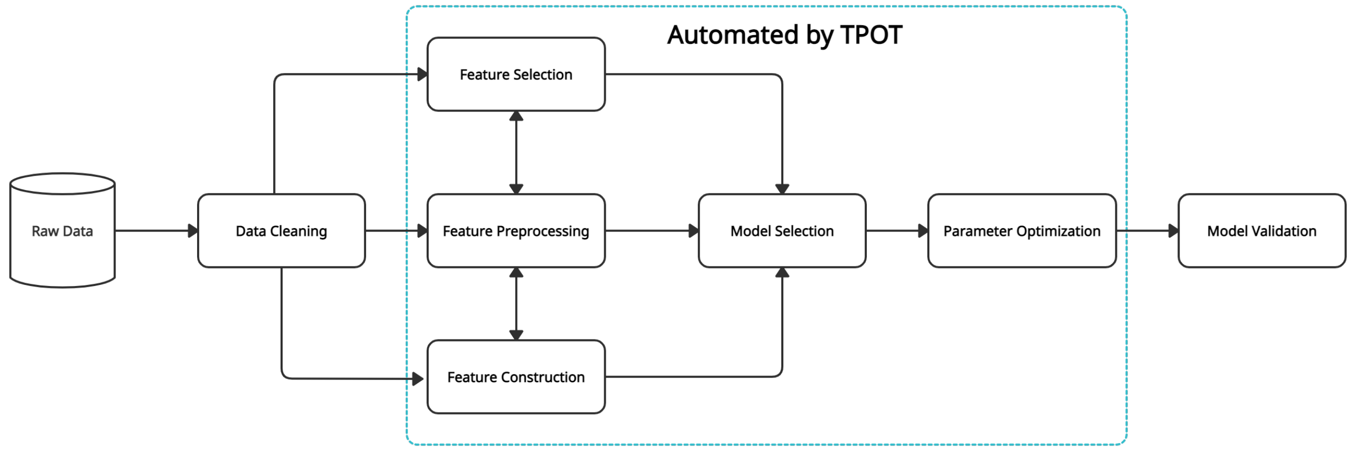

Fig. 1. The components of machine learning automated by TPOT [22].

high relations among itself or any predefined schema. It can allow data of different data types in runtime. Most of the No-SQL databases are JSON document-oriented [18]. There are many databases like MongoDB, CouchDB, Firebase Cloud Firestore etc.

2.6 Machine Learning in Software Security

Software security can be defined as achieving effective security awareness techniques to prevent attacking software deliberately to steal sensitive and personal information with the core intention of carrying out well-funded, destructive and unethical aims that could harm individuals, nations or the whole world. The pattern of injection attack is very dynamic and it's too difficult to detect the pattern without a machine learning approach. In recent years, supervised machine learning in automatic fraud and malware detection has been quite successful that motivated us to explore this direction [8][19].

2.7 Genetic Algorithm and Tree-based Pipeline Optimization Tool (TPOT)

A genetic algorithm is a machine learning search technique that is inspired by Darwinian evolutionary models. The advantage of genetic algorithms is they can be used to optimize the problems that are too complicated for human expertise to do. So, the genetic programming approach can be useful to find the combination feature subsets that are responsible to produce the best performing classification model. It can be used for solving high dimensional classification problems when the search space is large and too difficult to understand [20]. TPOT is a genetic programming-based automated machine learning system that optimizes a series of machine learning models and feature processors for maximizing classification accuracy on a supervised classification task.
In our implementation, the components of machine learning such as data cleaning, feature selection, feature engineering, model selection and validation, and hyperparameter tuning are continuously performed until an optimized result is achieved. The TPOT GP algorithm follows the rules of an ideal genetic programming process. In the implementation, we will generate tree-based pipelines and perform cross-validation against the dataset. The pipelines that perform better are designed to maximize accuracy and minimize the number of operators. The pipelines that are selected top through fitness function produce offspring for the next generation population. Crossover is performed among some of the offspring and mutation is performed among the remaining offspring with some random modification. Then a new pipeline is created. This process is repeated to optimise the pipeline. The operators that help to improve the classification accuracy are selected but the operators that reduce the classification accuracy are eliminated. In this way, the best performing classification machine learning model is found [22].

3. PROPOSED IDEA

Our proposed idea for cyber attack detection in a server-side application is a two-layer security firewall which is used to detect mainly SQL injection and NoSQL injection attacks. Normally, the http server accepts requests from client side applications like android, web browser , desktop etc. Many requests are sent from the client side applications by the users to the server for performing many kinds of operations like authentication, new data entry, data fetching etc. The attackers also send requests to the server but their request objects contain malicious code in it. The server side application contains the application layer and our proposed two-layer security firewall. The application layer consists of middleware, router, controller, model and view. The first layer security firewall is placed between middleware and router to detect malicious code from request objects in real-time using user input validation, parameterized statement etc. which are implemented using non-machine learning approach and the second layer security firewall is placed between controller and database server to detect malicious code from the SQL/NoSQL query in real-time using machine learning models.
The flowchart of the proposed idea in Fig-2 illustrates the process of detecting malicious code in the request object and detecting malicious code in the query by machine learning model by our proposed two-layer security firewall.
Suppose, a normal user sends a request to the server and it is received by the Middleware at first. Then, Middleware passes the request object to our first layer of real-time security firewall where the data from the request object will be extracted for user input validation and parameterized statements. As a normal user has sent this request, the first layer of the firewall will not detect any malicious code in the request object and it will pass the filtered request object to the controller functions through the router. In the controller functions, the SQL/NoSQL queries are generally constructed by the static part of the query which is written by the developers and dynamic values of the request object. After the construction of the query in the controller functions, it is passed to the second layer of our proposed security firewall for preprocessing and applying text mining algorithms to extract features from the constructed query. Then it is passed to the ma-

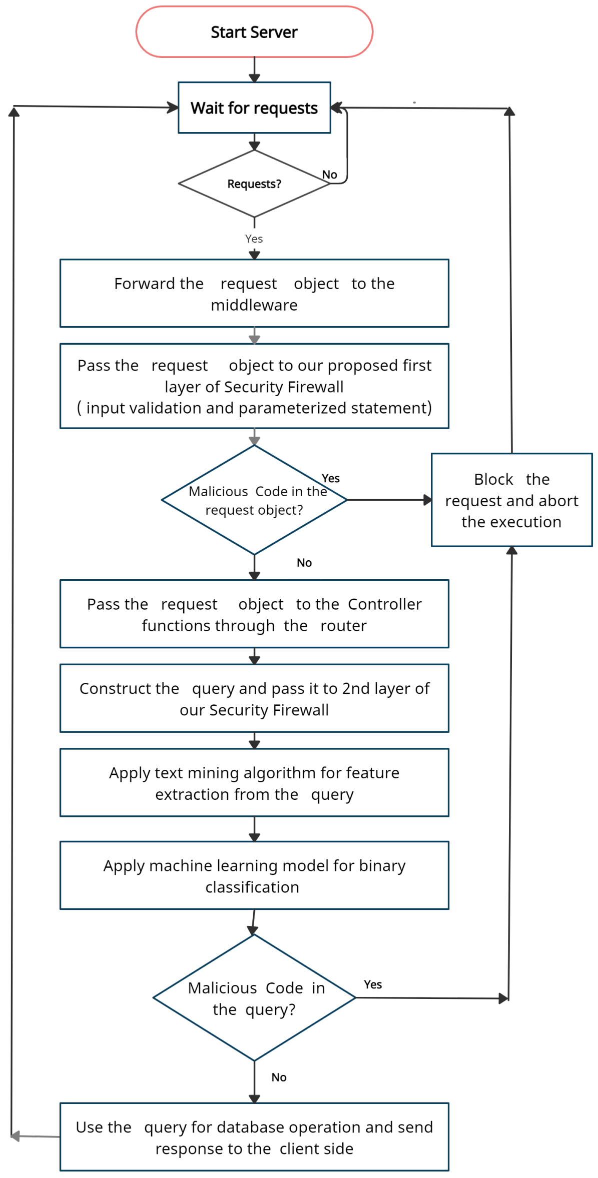
Start Server
Wait for requests
Requests?
No
Yes
Forward the request object to the middleware
Pass the request object to our proposed first layer of Security Firewall (input validation and parameterized statement)
Malicious Code in the request object?
Yes
Block the request and abort the execution
No
Pass the request object to the Controller functions through the router
Construct the query and pass it to 2nd layer of our Security Firewall
Apply text mining algorithm for feature extraction from the query
Apply machine learning model for binary classification
Malicious Code in the query?
Yes
No
Use the query for database operation and send response to the client side

Fig. 2. Flow diagram of our proposed security firewall in Server Side Application

chine learning models as test data for binary classification. In the implementation, two machine learning models will be built into the second layer of our security firewall. One for detecting SQL injection attacks and another for NoSQL injection attacks. As the query is constructed by the data from a normal user, the classification model will not detect any malicious code in the query. As a result, the controller function will execute database operation using the filtered query and send a normal response to the user. On the other hand, suppose a hacker sends a malicious request to the server and it is received by the Middleware at first. Then, Middleware passes the request object to our first layer of real-time security firewall where the data from the request object will be extracted for user input validation and parameterized statements. Then, the first layer detects malicious code in the request object, blocks the request and aborts the execution. As a result, the data is saved from the attacker. Due to the dynamic pattern of the malicious code, it is possible that our first layer fails to detect the malicious code in the request object and doesn't block the request object from going through the Middleware. For this reason, machine learning models are used to handle the dynamic pattern of malicious code. So, after the construction of the query, it is passed to the second layer of our proposed security firewall that will detect any malicious code even if the first layer fails to detect it. As a result, the controller function will prevent the execution of the database query and block the execution. In this way, the data is saved by the hacker.

4. APPROACH OF IMPLEMENTATION

4.1 Dataset Collection:

To build machine learning models a training dataset and a testing dataset need to be prepared. As the second layer of our proposed security firewall consists of the SQL injection model and No-SQL Injection model, the dataset is needed to be divided into training data and testing data for both models. The datasets contain different types of SQL injection and No-SQL injection data. They are described below:

4.1.1 SQL Injection.
SQLIA is a code injection technique that takes advantage of a security flaw in a web application's SQL database layer.

(i) Tautology: It is one of the most common types of SQLIA. The attacker's motive is to skip authentication and extract statistics from the net utility database. In this kind of assault the hacker injects code into one or greater conditional statements so that the final results of the execution of those statements continually evaluates to true [23].

```
SELECT * from customer WHERE cus_id = 12;
```

After injecting "12 or 1=1", the above query will look like:

```
SELECT * from customer WHERE cus_id =
12345 or 1=1;
```

(ii) Illegal/Logically Incorrect Queries: This attack intends to spot injectable parameters, perform information fingerprinting, and extract confidential information. This kind of attack is predicated on writing a question statement that generates error messages. Once a question is rejected, a blunder message is returned from the information including helpful debugging info that helps the attacker within the example below, the attacker's goal is to cause a type conversion error that may reveal relevant information.

```
SELECT accounts FROM users WHERE
login='' AND pass='' AND pin= convert
(int,(select top 1 name from sysobjects
WHERE xtype='u'));
```

The hacker extracts two useful pieces of information from this error message. The hacker learns that this is a SQL server database and the error message reveals the value of the string that caused the type conversion to occur [23].

(iii) Union Query: This type of attack is especially executed to bypass the authentication process and to extract information by inserting the union operator into the original query.

```
SELECT * from accounts WHERE id='212'
UNION select * from credit_card WHERE
user='admin'--' and pass='pass';
```

In this example, the second query is malicious because the text following '—' is ignored because it becomes a comment for the SQL Parser. However, if the query is executed, the attacker receives the credit card information [23].

(iv) Piggy-Backed Query: The intention of this sort of attack is the retrieval of information and the DoS attack. It works just

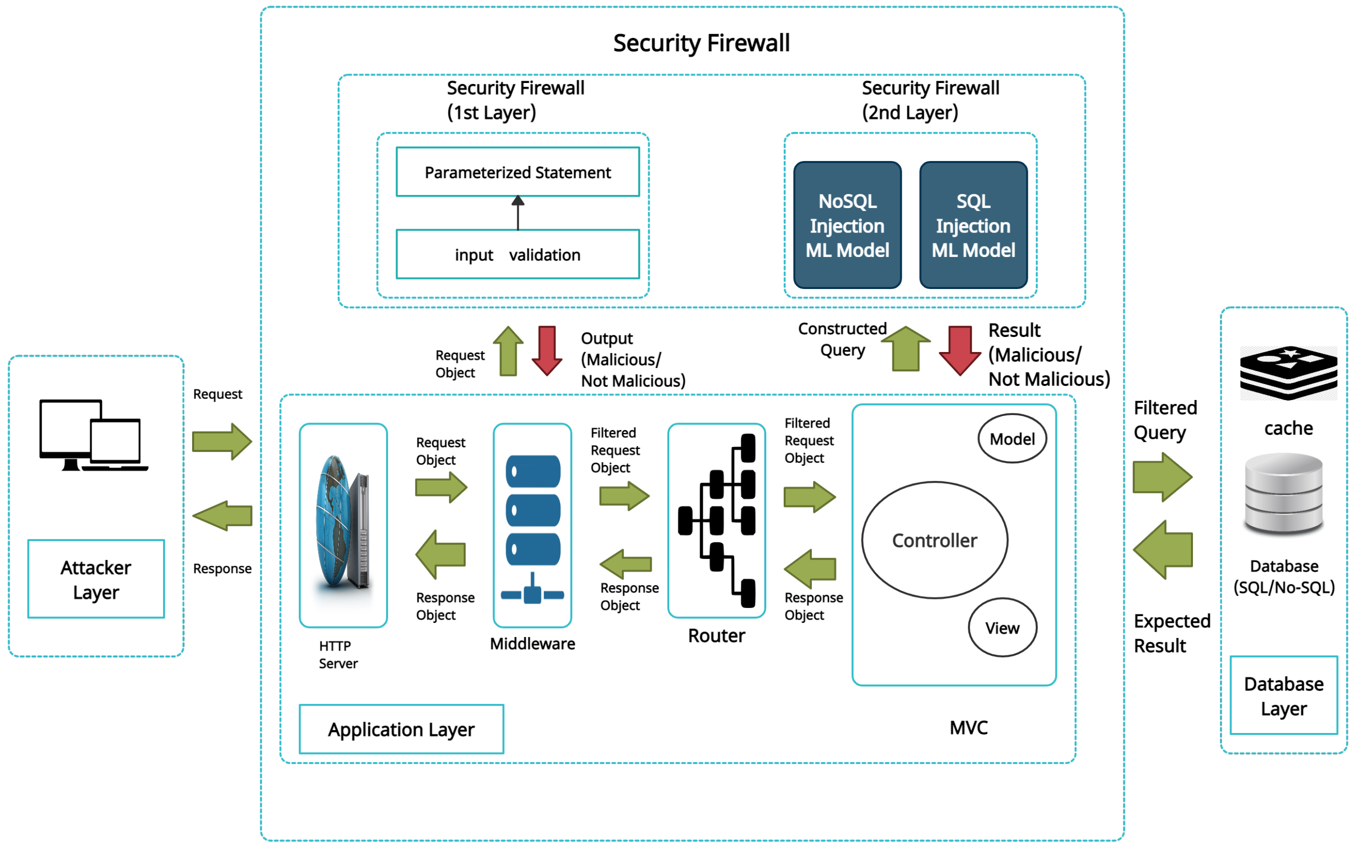

Fig. 3. Our proposed Two Layer Security Firewall in Server Side Application.

like the construct of piggy-backed acknowledgement in network communication where acknowledgement of a packet is distributed together with successive packets.

```
SELECT customer_info from accounts WHERE
login_id = ‘admin’ AND pass = ‘123’ ;
```

```
DELETE FROM accounts WHERE CustomerName =
’Rahim’;
```

After executing the first query the query interpreter sees the ’;’ and thus executes the second query with the first query. Since the second query is malicious, it will delete all the data of the customer ’Rahim’ [1][23].

(v) Stored Procedures: In this type of attack, the hacker tries to execute stored procedures added in the database with malicious inputs. Most database companies store procedures that extend the functionality of the database and allow interaction with them.

```
CREATE PROCEDURE DBName .is Authenticated
@user Name varchar2, @pass varchar2,
@pin int AS EXEC("SELECT accounts FROM
users WHERE login=’" +@user Name+ If’ and
pass=’" +@password+ and pass=" +@pass);
```

The use of stored procedure returns true if it is authorized and returns false if it is unauthorized [23]. If the hacker gives input SHUTDOWN; - -” for username or password. The Stored Procedure generates the following query statement which shut down the system.

```
SELECT username FROM userTable WHERE
username = ‘Rahim’ AND pass=’ ’; SHUTDOWN;
```

4.1.2 NoSQL Injection

(i) Tautology: These attacks are executed by injecting code in conditional statements and generating expressions that are continuously true for the purpose of bypassing authentication or accessing confidential mechanisms.

```
db.members.find(login:"admin", pass: $ne:
"rahim"  )
```

Attackers use the “$ne”(not equal) operator to illegally access the system without using the actual username and password [24].

(ii) Union Queries: This is one of the most common types of No-SQL injection attack. In this attack, the attacker exploits a vulnerable parameter to change the data set returned for a given query. These attacks are mainly used to bypass the authentication process and extract data. For example, an attacker exploits a vulnerable parameter by adding a boolean ‘OR’ operator so that the expression is always true (e.g. an empty query) which leads to the incorrect evaluation of the entire statement allowing illegal extraction of data [24].

```
db.collection(’users’).findOne(
"username": "dummy", $or: [ ,  password:
""  ])
```

The password becomes a useless part of the query as an empty query is always true. In this kind of attack, an attacker will be successful only if the username is correct [25].

(iii) Javascript injections: The benefit of NoSQL databases is the ability to run JavaScript in the database engine to perform complicated queries or transactions such as 'MapReduce'.But it creates a new class of vulnerabilities for the No-SQL databases. The attacker can cause a DOS attack by exhausting the pool size by putting ' sleep()' inside 'where' MongoDB operator. Here is an example:

```
db.collections.find($where: quantity >
'';sleep(50000);;var foo='bar')
```

(iv) Piggy-backed Queries: In this type of attack, an attacker injects extra malicious queries into the original query to add, modify or delete data inside collections [24].

4.2 Dataset Preprocessing and feature extraction:

The NoSQL injection dataset has been collected from a paper that has been done using supervised learning on automatic detection of NoSQL injection [19]. The dataset has been already preprocessed which made our job a little easier as machine learning algorithms can't work with string inputs. The dataset was designed with 19 features to start with but finally, the researchers selected the 10 highest ranked features based on information gain and correlation. They used WEKA's ClassifierSubsetEval [26] with a decision tree(J48) [27], K-nearest neighbour classifier(IBK) [28] and greedy stepwise search with backward elimination to select rank 10 out of initially designed 19 features which are based on information gain and correlation shown in table-1. This dataset contains the queries of both CouchDB and MongoDB [19].

Table 1. Features of NoSQL injection dataset by information gain and correlation [19].

Rank	By information gain	By correlation
1	Contains Comparison	Contains Comparison
2	New Query	New Query
3	Contains Empty String	Contains Empty String
4	Contains Not Equal	Contains Not Equal
5	Contains Payload	Contains Payload
6	Presence of Return	Always True Expression
7	Always True Expression	Presence of Return
8	Evaluation Query Operation	Evaluation Query Function
9	Contains Logical Operator	Element Query Operation
10	Element Query Operation	Contains Logical Operator

The SQL injection datasets have been collected from Kaggle [29]. The dataset has not been preprocessed. The unprocessed and unstructured can be easy to understand by humans but machine learning algorithms need structured data to perform well . So, several text mining techniques will be used to extract features to build a structured dataset.For this reason,the NLTK library of python is going to be used to extract features from the existing dataset . In the implementation, the following features shown in table-2 are selected to extract from the existing dataset.

4.3 Feature Selection

In the implementation, python's pycaret library is going to be used that uses the genetic algorithm for selecting the best features for our proposed machine learning model. As almost 10 features in the NoSQL dataset and almost 43 features in the SQL dataset are selected for the experiment, the search space will contain 2^{10} possible feature subsets for the NoSQL dataset and 2^{43} possible feature subsets for the SQL dataset. Each feature will be considered as a bit in a single genome. A bit string of length 43 and a bit string of length 10 will respectively represent each individual in the population of possible solutions in SQL and NoSQL

Table 2. Features and Descriptions of SQL injection dataset.

Features	Descriptions
querySource	Identifier of sources of a query
commandType	Identifier of operation type represented by query
utilityStmt	Indicator if the SQL is DECLARE CURSOR or a non-optimizable statement
canSetTag	Indicator if the command result tag is set
sortClause	Indicator if sort clause exists
groupClause	Indicator if group clause exists
windowClause	Indicator if window clause exists
resultRelation	Range table index for INSERT/UPDATE/DELETE statements
hasWindowFuncs	Indicator if SQL has window functions
hasAggs	Indicator if SQL has aggregates
hasDistinctOn	Indicator if distinct clause is from DISTINCT ON
hasSubLinks	Indicator if SQL has sub-query
hasForUpdate	Indicator if SQL is specified with FOR [KEY] UPDATE/SHARE clause
hasModifyingCTE	Indicator if SQL has INSERT/UPDATE/DELETE in WITH clause
hasRecursive	Indicator if SQL is specified WITH RECURSIVE clause
cteList	The number of WITH clauses
rtable/relid	Identifier of range table
rtable/relkind	Index for kind of range table entry {SUBQUERY, RELATION, FUNCTION, JOIN, VALUES, CTE}
rtable/funcexpr/funcid	Identifier of function
rtable/funcexpr/args/constvalue	Constant value and constant length
rtable/funcexpr/args/consttype	Identifier of constant type
jointree/quals/args*/opno	Identifier of operation number
jointree/quals/args*/boolop*	Qualification type BOOLEXPR, OPEXPR and boolean operation and, or, not
jointree/quals/args*/arg/varno	Relative table number based on range table
jointree/quals/args*/opresulttype	Identifier of result type of operation
jointree/quals/args*/arg/funcid	Identifier of function
jointree/quals/args*/arg/varattno	Relative column number based on range table
jointree/quals/args*/constvalue	Constant length and constant value
jointree/quals/args*/consttype	Identifier of constant type
targetList/expr	Index for target entry expression type
targetList/resorigcols	Identifier of original column within target entry
targetList/expr/funcid	Identifier of function
targetList/resorigtbl	Identifier of original table within target entry
targetList/expr/args*/constvalue	Constant length and constant value
targetList/expr/args*/consttype	Identifier of constant type
returningList	Indicator if return value list exists
constraintDeps	Indicator if constraint exists
limitCount	Indicator if limit count clause specified to return the result tuples exists
limitOffset	Indicator if limit offset clause specified to skip the result tuples exists
setOperations	Indicator if set operation {UNION, INTERSECT, EXCEPT} exists

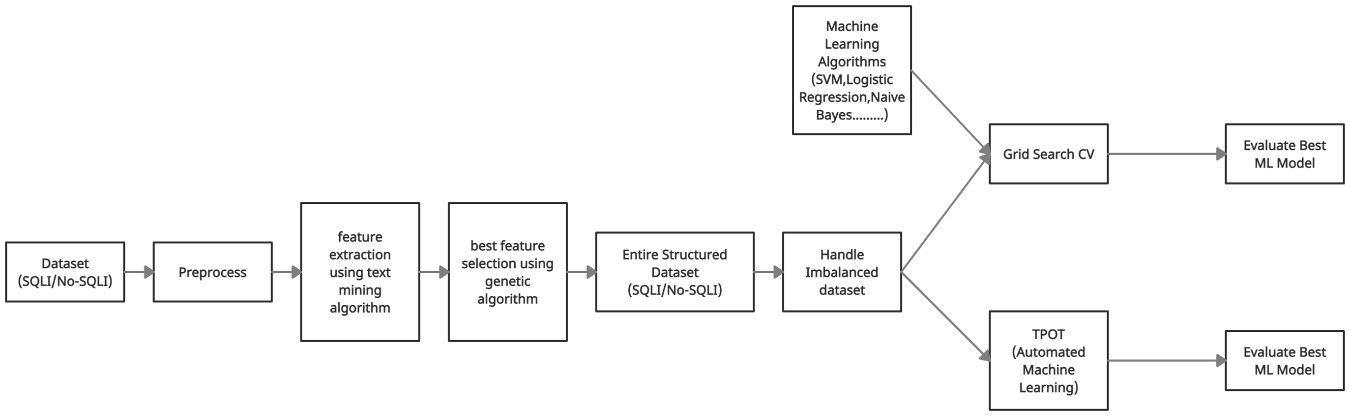
Dataset (SQLI/No-SQLI)
Preprocess
feature extraction using text mining algorithm
best feature selection using genetic algorithm
Entire Structured Dataset (SQLI/No-SQLI)
Handle Imbalanced dataset
Machine Learning Algorithms (SVM,Logistic Regression,Naive Bayes.........)
Grid Search CV
Evaluate Best ML Model
TPOT (Automated Machine Learning)
Evaluate Best ML Model

Fig. 4. Flow diagram of Implementing Best Classification Model.

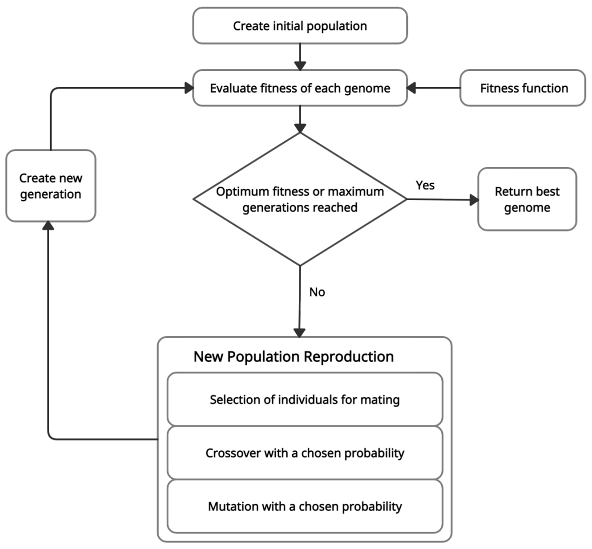
Create initial population
Evaluate fitness of each genome
Fitness function
Create new generation
Optimum fitness or maximum generations reached
Yes
Return best genome
No
New Population Reproduction
Selection of individuals for mating
Crossover with a chosen probability
Mutation with a chosen probability

Fig. 5. Flow Diagram of Genetic Algorithm Search [20].

injection dataset. zero(0) bit will indicate that the corresponding feature is eliminated and one(1) bit will indicate that the corresponding feature is selected. According to Fig-5, in the first step of GA, the initial population is generated randomly. Then a two-point crossover will be chosen to create two new individuals for the next generation. Single bit flip mutation will be chosen for mutation operation. It will be performed by selecting a single bit on the genome and flipping it. It will prevent huge changes in the binary genome and cause stability in the result.

4.4 Dataset Analysis Approach

Our proposed SQL injection dataset contains almost 37961 rows in which 12584 rows are labeled malicious and 25377 rows are labeled as not malicious. Again, the NoSQL injection dataset contains almost 1004 rows in which 801 rows are labeled as not malicious and 203 rows are labeled as malicious. The NoSQL dataset consists of MongoDB and CouchDB queries.

4.4.1 Sampling Strategies.
There is a huge imbalance in both NoSQL and SQL injection dataset as shown in the Fig-6. For this reason, our proposed classification model results might be distorted by these skewed distributions due to the highly imbalanced na-

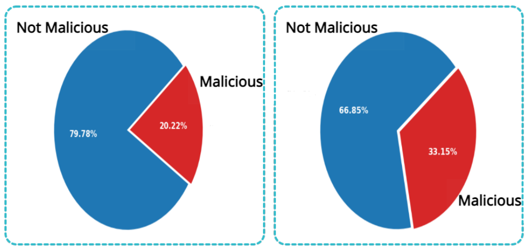

Fig. 6. Imbalanced ratio of minority and majority class in NoSQL and SQL injection dataset

ture of the training dataset. This is unwanted. So, various re-sampling strategies are going to be used before passing the training dataset to the machine learning model. These are Under-sampling(Random undersampling, ClusterCentroids,NearMiss) and Over-sampling(Random oversampling,SMOTE) [30]. SMOTEENN and SMOTETomek are also going to be applied. These two strategies are the combination of oversampling and undersampling techniques. Python's imblearn library is going to be used for handling imbalanced datasets.

4.4.2 Grid Search for best hyperparameter.
To find the best performing configuration of each algorithm, a grid search approach is applied to optimize the hyperparameters of the machine learning algorithms in the implementation. It means that various values are going to be defined for machine learning algorithm parameters and multiple models are going to be trained using various combinations of hyperparameters. After having accuracy, precision, recall and f1 score for each model, the best performing model can be selected. Python's Sklearn library is used for implementing GridsearchCV and machine learning models.

4.4.3 Genetic Algorithm for best automated hyper parameter optimisation.
For the automated hyperparameter tuning, python's TPOT classifier module in Google Colab has been used that works based on genetic algorithms. Some parameters are tuned to get the result from the TPOT classifier. There are 500 pipeline configurations for evaluation according to the tuned parameters. Classifier generated 10 generations each with 50 population sizes. Models are fitted and evaluated against the training data in one grid search 10 fold cross-validation. The best pipeline is the one that has the highest CV score of 90%. TPOT classifier is run on the im-

International Journal of Applied Information Systems (IJAIS) ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 12 - No. 37, June 2021 - www.ijais.org

balanced dataset. In the implementation, the TPOT classifier is going to be run on the balanced dataset too.

5. RESULT ANALYSIS

Table-3 and Table-4 represent the results after applying some machine learning algorithms on imbalanced data and balanced data respectively.
From Table-3, we can see that SVM, Decision Tree, AdaBoost and Random Forest performed equally well comparatively better than Logistic Regression and Naive Bayes if we consider the accuracy. But AdaBoost and Random Forest performed better than other algorithms if we consider the f1 score.
From Table-4, we can see that Base, SMOTE, RandomOverSampler, RandomUnderSampler and SMOTETomek resampling strategies perform almost equal. Their results are almost similar to the results on the imbalanced data. But the results that are generated after applying the NearMiss resampling technique are worse. The results of the SMOTEENN resampling technique is the best among them. The accuracy, precision, recall and f1 score are 1 for the AdaBoost algorithm when SMOTEENN resampling is used.

Table 3. Results after applying machine learning algorithms on imbalanced data

Model	Accuracy	Precision	Recall	F1 score
SVM	0.89	0.89	0.89	0.89
Decision Tree	0.89	0.89	0.89	0.89
AdaBoost	**0.89**	**0.90**	**0.90**	**0.90**
Random Forest	**0.89**	**0.90**	**0.90**	**0.90**
Logistic Regression	0.88	0.88	0.88	0.88
Naive Bayes	0.85	0.87	0.87	0.87

6. CONCLUSION AND FUTURE WORKS

In this paper, a two-layer security firewall architecture in the server-side application is represented that uses both non-data mining techniques and data mining techniques to detect both SQL and NoSQL injection attacks. A comprehensive comparison of several machine learning algorithms on both balanced and imbalanced dataset is shown for best model evaluation to detect NoSQL injection attacks. The hyperparameter optimization using both GridSearchCV and AutoML is also performed on the NoSQL injection dataset. It is seen from the result analysis that machine learning algorithms performed outstandingly on a balanced dataset which is created from the SMOTEENN resampling technique.
In the implementation, some new machine learning algorithms like CatBoostClassifier, ExtraTreesClassifier, K Neighbour classifier, LDA(Linear Discriminant Analysis), Gradient Boosting classifier, XGBoost classifier etc. are going to be applied as well as deep neural network on both SQL and NoSQL datasets.
SQL injection dataset will be preprocessed for feature extraction. A comprehensive comparison of several machine learning algorithms on both balanced and imbalanced dataset will be shown for best model evaluation to detect SQL injection attacks. The deep neural network is also going to be applied to it.
Moreover, genetic algorithms will be used to find the feature combinations that will produce the best-performing model.
In the future, we have plans to include another layer in the security firewall for detecting and preventing DDoS attacks using data mining techniques.

Table 4. Results after applying machine learning algorithms on balanced data

Sampling type	Model	Accuracy	Precision	Recall	F1 score
Base	SVM	0.92	0.89	0.91	0.89
Base	Decision Tree	0.92	0.86	0.90	0.88
Base	AdaBoost	0.91	0.90	0.92	0.91
Base	**Random Forest**	**0.92**	**0.90**	**0.92**	**0.91**
Base	Logistic Regression	0.90	0.84	0.90	0.86
Base	Naive Bayes	0.92	0.89	0.87	0.88
SMOTE	SVM	0.89	0.91	0.91	0.91
SMOTE	**Decision Tree**	**0.90**	**0.91**	**0.91**	**0.91**
SMOTE	**AdaBoost**	**0.90**	**0.91**	**0.91**	**0.91**
SMOTE	**Random Forest**	**0.90**	**0.91**	**0.91**	**0.91**
SMOTE	Logistic Regression	0.89	0.89	0.89	0.89
SMOTE	Naive Bayes	0.86	0.89	0.89	0.89
RandomOverSampler	SVM	0.90	0.91	0.90	0.90
RandomOverSampler	**Decision Tree**	**0.90**	**0.91**	**0.91**	**0.91**
RandomOverSampler	**AdaBoost**	**0.90**	**0.91**	**0.91**	**0.91**
RandomOverSampler	**Random Forest**	**0.90**	**0.91**	**0.91**	**0.91**
RandomOverSampler	Logistic Regression	0.89	0.89	0.89	0.89
RandomOverSampler	Naive Bayes	0.87	0.89	0.89	0.89
NearMiss	SVM	0.83	0.89	0.89	0.89
NearMiss	Decision Tree	0.84	0.88	0.88	0.88
NearMiss	AdaBoost	0.83	0.89	0.89	0.89
NearMiss	Random Forest	0.86	0.88	0.88	0.88
NearMiss	Logistic Regression	0.84	0.89	0.89	0.89
NearMiss	**Naive Bayes**	**0.88**	**0.89**	**0.89**	**0.89**
RandomUnderSampler	SVM	0.88	0.89	0.89	0.89
RandomUnderSampler	**Decision Tree**	**0.89**	**0.91**	**0.91**	**0.91**
RandomUnderSampler	**AdaBoost**	**0.89**	**0.91**	**0.91**	**0.91**
RandomUnderSampler	Random Forest	0.88	0.91	0.91	0.91
RandomUnderSampler	Logistic Regression	0.88	0.89	0.89	0.89
RandomUnderSampler	Naive Bayes	0.87	0.87	0.87	0.87
SMOTEENN	SVM	0.99	0.99	0.99	0.99
SMOTEENN	Decision Tree	0.99	1	1	1
SMOTEENN	**AdaBoost**	**1**	**1**	**1**	**1**
SMOTEENN	Random Forest	0.99	1	1	1
SMOTEENN	Logistic Regression	0.99	0.99	0.99	0.99
SMOTEENN	Naive Bayes	0.99	0.99	0.99	0.99
SMOTETomek	SVM	0.89	0.89	0.89	0.89
SMOTETomek	Decision Tree	0.89	0.89	0.89	0.89
SMOTETomek	**AdaBoost**	**0.89**	**0.90**	**0.90**	**0.90**
SMOTETomek	Random Forest	0.89	0.90	0.89	0.89
SMOTETomek	Logistic Regression	0.87	0.88	0.88	0.88
SMOTETomek	Naive Bayes	0.85	0.87	0.87	0.87

7. REFERENCES

[1] Alsobhi, H. and Alshareef, R. SQL Injection Countermeasures Methods. , 2020 International Conference on Computing and Information Technology (ICCIT-1441). (2020)

[2] Benson, V. (2017) The state of global cyber security: Highlights and key findings, Learning Tree.

[3] Website. . [Online]. Available: 02-Apr-(2021) , Cyber attacks hit over 200 organizations including Bangladesh Bank, BTRC. . [Online]. Available: https://www.dhakatribune.com/bangladesh/2021/04/02/cyber-attacks-hit-over-200-organizations-including-bangladesh-bank-btrc. [Accessed: 21-Apr-2021]. [Accessed: 23-Apr-2021]

[4] By and Sobers, R. 13-Jan-(2020) , 134 Cybersecurity Statistics and Trends for 2021. . [Online]. Available: https://www.varonis.com/blog/cybersecurity-statistics/. [Accessed: 23-Apr-2021]

[5] 08-Nov-(2020) , 2021 Cyber Security Statistics: The Ultimate List Of Stats, Data & Trends. . [Online]. Available: https://purplesec.us/resources/cyber-security-statistics/. [Accessed: 21-Apr-2021]

[6] Barolli, L. et al. (2019) Advances on P2P, Parallel, Grid, Cloud and Internet Computing: Proceedings of the 14th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC-2019), Springer Nature.

[7] Chen, X. et al. (2019) Machine Learning for Cyber Security: Second International Conference, ML4CS 2019, Xi'an, China, September 19-21, 2019, Proceedings, Springer Nature.

[8] Kim, M.-Y. and Lee, D.H. Data-mining based SQL injection attack detection using internal query trees. , Expert Systems with Applications, 41. (2014) , 5416–5430

[9] M., A. et al. NoSQL Racket: A Testing Tool for Detecting NoSQL Injection Attacks in Web Applications. , International Journal of Advanced Computer Science and Applications, 8. (2017)

[10] Okman, L. et al. Security Issues in NoSQL Databases. , 2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications. (2011)

[11] Schram, A. and Anderson, K.M. MySQL to NoSQL. , Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity - SPLASH '12. (2012)

[12] Neumann, A. et al. An Analysis of Public REST Web Service APIs. , IEEE Transactions on Services Computing. (2018) , 1–1

[13] Chen, X. et al. (2017) Restful API Architecture Based on Laravel Framework. J. Phys. Conf. Ser. 910, 012016

[14] 08-Jul-(2019) , What is Middleware and How Does it Work? . [Online]. Available: https://www.cleverism.com/what-is-middleware-and-how-does-it-work/. [Accessed: 23-Apr-2021]

[15] Khaliluzzaman, M. and Chowdhury, I.I. Pre and post controller based MVC architecture for web application. , 2016 5th International Conference on Informatics, Electronics and Vision (ICIEV). (2016)

[16] Selfa, D.M. et al. A Database and Web Application Based on MVC Architecture. , 16th International Conference on Electronics, Communications and Computers (CONIELECOMP'06).

[17] Jailia, M. et al. Behavior of MVC (Model View Controller) based Web Application developed in PHP and .NET framework. , 2016 International Conference on ICT in Business Industry & Government (ICTBIG). (2016)

[18] Khasawneh, T.N. et al. (2020) , SQL, NewSQL, and NOSQL Databases: A Comparative Survey. , in 2020 11th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan

[19] Ul Islam, M.R. et al. (2019) , Automatic detection of NoSQL injection using supervised learning. , in 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), Milwaukee, WI, USA

[20] Johnson, P. et al. (2014) Genetic algorithm with logistic regression for prediction of progression to Alzheimer's disease. BMC Bioinformatics 15 Suppl 16, S11

[21] Mayo, M. Using AutoML to Generate Machine Learning Pipelines with TPOT - KDnuggets. . [Online]. Available: https://www.kdnuggets.com/managing-machine-learning-workflows-with-scikit-learn-pipelines-part-4-generating-pipelines-with-automl.html/. [Accessed: 27-Apr-2021]

[22] Javel, I.M. et al. Epileptic Seizure Detection via EEG using Tree-based Pipeline Optimization Tool. , 2019 IEEE 11th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM). (2019)

[23] Ghafarian, A. A hybrid method for detection and prevention of SQL injection attacks. , 2017 Computing Conference. (2017)

[24] Ron, A. et al. Analysis and Mitigation of NoSQL Injections. , IEEE Security & Privacy, 14. (2016) , 30–39

[25] (2017) Advanced Penetration Testing & Software Development. at <https://www.stjoern.com/>

[26] Moore, A.W. and Lee, M.S. Efficient Algorithms for Minimizing Cross Validation Error. , Machine Learning Proceedings 1994. (1994) , 190–198

[27] Ross Quinlan, J. (2014) C4.5: Programs for Machine Learning, Elsevier.

[28] Aha, D.W. et al. Instance-based learning algorithms. , Machine Learning, 6. (1991) , 37–66

[29] Shah, S.S.H. sql injection dataset.

[30] Ferenc, R. et al. (2019) , Challenging machine learning algorithms in predicting vulnerable JavaScript functions. , in 2019 IEEE/ACM 7th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), Montreal, QC, Canada