ISSN 0361-7688, Programming and Computer Software, 2020, Vol. 46, No. 5, pp. 351–361. © Pleiades Publishing, Ltd., 2020.

# Improving Efficiency of Web Application Firewall to Detect Code Injection Attacks with Random Forest Method and Analysis Attributes HTTP Request

**Nguyen Manh Thang**

*Academy of Cryptography Techniques, 141 Chien Thang, Tan Trieu, Thanh Tri, Ha Noi*
*e-mail: chieumatxcova@hotmail.com*

Received November 10, 2019; revised November 25, 2019; accepted April 30, 2020

**Abstract**—In the era of information technology, the use of computer technology for both work and personal use is growing rapidly with time. Unfortunately, with the increasing number and size of computer networks and systems, their vulnerability also increases. Protecting web applications of organizations is becoming increasingly relevant as most of the transactions are carried out over the Internet. Traditional security devices control attacks at the network level, but modern web attacks occur through the HTTP protocol at the application level. On the other hand, the attacks often come together. For example, a denial of service attack is used to hide code injection attacks. The system administrator spends a lot of time to keep the system running, but they may forget the code injection attacks. Therefore, the main task for system administrators is to detect network attacks at the application level using a web application firewall and apply effective algorithms in this firewall to train web application firewalls automatically for increasing his efficiency. The article introduces parameterization of the task for increasing the accuracy of query classification by the random forest method, thereby creating the basis for detecting attacks at the application level.

**DOI:** 10.1134/S0361768820050072

## 1. INTRODUCTION

The industrial revolution 4.0, with the features of digital control systems, IoT, cloud computing and artificial intelligence, has opened up new opportunities for humanity but also poses new challenges. The main problem needed attention from the very beginning is ensuring information security for information technology systems serving industry 4.0.

With the rapid development of technology, sophisticated attack methods are also specially designed to avoid an attack detection system. Thus, in modern information security systems, most organizations use an intrusion detection system, an intrusion prevention system and a network firewall to monitor the system as well as identify attacks at the network level.

These systems use signature-based attack detection [1, 2], abnormal feature detection [3, 5], white-black list [6, 7], etc. to prevent attacks. Due to the proliferation of new attacks, traditional signature-based methods have become less effective leading to an increasing need to detect abnormal behavior. Behavioral identification allows system admin to detect new and complex attacks based on statistical models and machine learning. The weak point in detecting behavioral attacks is the large number of false alerts. To improve the accuracy of detecting network attacks, several approaches will be combined with many machine learning methods to detect anomalies in the information protection system.

The article mentions the detection and classification of code injection attacks. These attacks have a common feature: they are for initial commands having abnormal modifications. To evaluate the anomalies in the dataset, the author has proposed several properties to parameterize these anomalies. The experimental part of the article will use a lot of test datasets and specifically use the CSIC 2010 dataset. This is a dataset containing a complex set of attack samples such as SQL injection, XSS, etc. Especially in a web server system, three objects need noting: HTTP requests from the client, HTTP response from the server and SQL queries interact with the database of the web system. The paper focuses on detecting anomalies in commands interacting with the system database.

This article presents a combination of approaches for detecting code injection attacks, such as SQL injection, using the random forest method and analyzing attributes of the HTTP requests. The rest of the article is organized as following: in Section 2, authors will consider a brief overview of SQL injection, XSS [8–10] and web application protection system (such as web application firewall) and their advantages, disadvantages under the task of detecting attacks. Section 3 gives a parameterization of tasks of increasing the

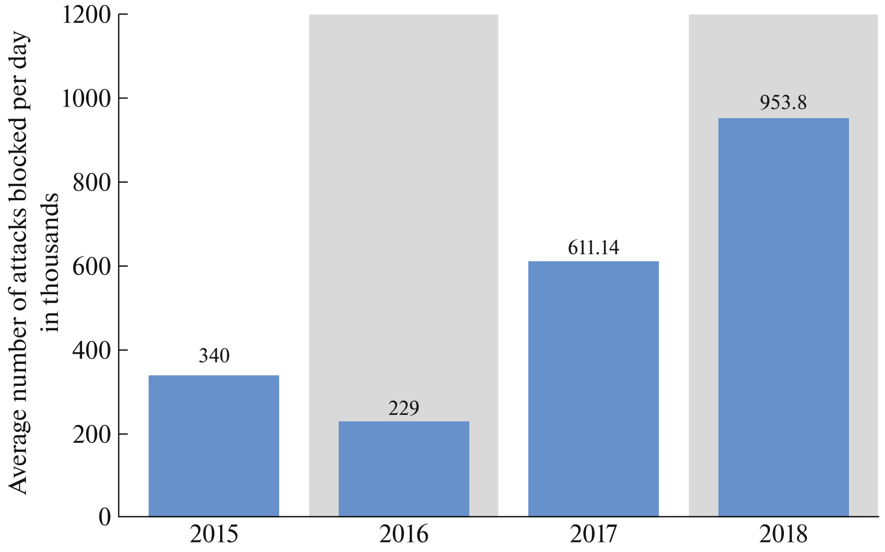

**Fig. 1.** Global number of web attacks blocked per day from 2015 to 2018 (in 1,000s)

accuracy of attack detection. Section 4 describes the mathematical foundation of algorithm authors. Section 5 presents the implementation of the algorithm in the information security system. The evaluation of an algorithm is described in section 6. Section 6 draws the conclusion of the article.

## 2. OVERVIEW OF SQL INJECTION AND WEB APPLICATION PROTECTION SYSTEM

### *2.1. SQL Injection*

Injection or SQL injection [11–13] is an attack in which an attacker inserts malicious code into the lines sent to the database management system server for analysis and execution. An SQL injection is a very popular attack method and its success is also relatively high.

Today, there are a large number of methods for using SQL injection for various database management systems and applications based on them.

SQL injection has several sub-classes such as:

• classic SQL injection;

• blind SQL injection and interference injection;

• special SQL injection for Database management system;

• complex SQL injection.

Although SQL is very dangerous, it is also easy to prevent. Recently, most websites rarely use classic SQL but often use ORM (Object-Relational Mapping) framework. These web frameworks will generate their SQL statements, making it harder for hackers to attack.

To protect the system from SQL Injection, system administrators can take the following measures.

• Filter data from users. This prevention method is similar to XSS. We use filters to filter special characters (; " ’) or keywords (SELECT, UNION) entered by the user. Should use the library or function provided by the framework.

• Strings to create SQL aren’t added. Using parameters instead of strings. If the data transmitted is illegal, the SQL Engine will automatically notify the system supervisor without using the code to check.

• Exceptionally, error message isn’t shown. Hackers often rely on error messages to find out the database structure. When there is an error, the system administrator only installs an error message and does not display full information about the error to avoid the hacker.

• Decentralization in a database. If a user only needs to access data from several tables, the database administrator will create an account in the database and assign access rights to that account, not access it. Default database with root or sa permissions. At this time, although hackers can inject malicious code in SQL request, he cannot read data from main tables, edit or delete data.

• Backup Database regularly. Data should be stored according to the general principle: There are at least 3 backups, stored in at least 2 different places. The data must be backed up regularly in case the hacker deletes it, the system administrator can still recover it.

### *2.2. XSS*

XSS is a type of computer security vulnerability typically found in web applications. XSS enables attackers to inject client-side scripts into web pages viewed by other users. A cross-site scripting vulnerability may be used by attackers to bypassing access controls such as the same-origin policy.

XSS attacks has several sub-classes such as:

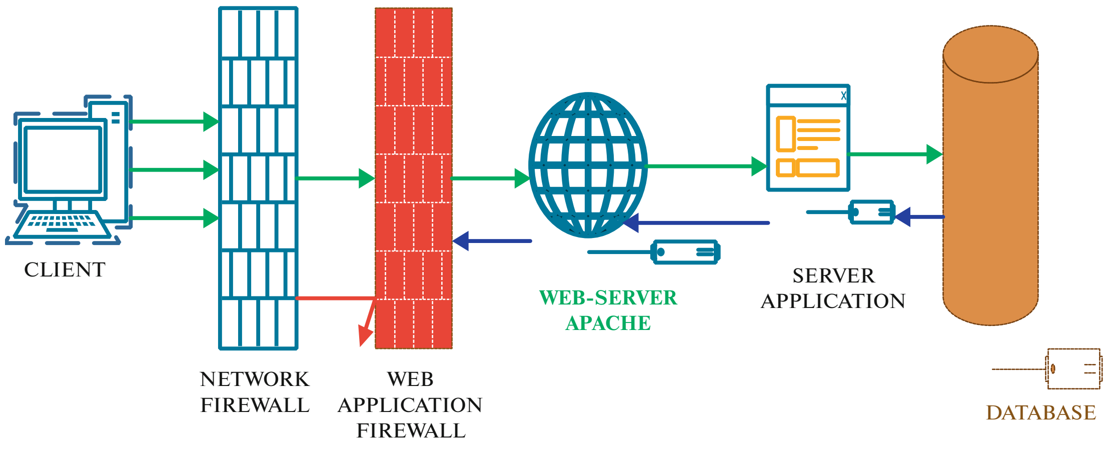

**Fig. 2.** Position of Web application firewall in an information security system.

• non-persistent (reflected);

• persistent (or stored);

• server-side versus DOM-based vulnerabilities;

• self-XSS;

• mutated XSS (mXSS).

Both reflected and stored XSS can be addressed by performing the appropriate validation and escaping on the server-side.

To protect the web application of organizations from XSS attacks, system administrators can take the following measures.

• Filter. There are two basic concepts of XSS filtering: input filtering and output filtering. The most common use is for input filtering. Input filtering is considered more accurate than output filtering, especially in the case of XSS Reflected. There are two types of input and output filtering: white-list filtering and black-list filtering. The disadvantage of black-list filtering is that it cannot be detected and prevented by a new type of attack. And, white-list filtering is predefined one list of available trusted port, IP, etc. When a new application is developed, it must also be updated in the White-List.

• Input encoding. The input coding can become a central position for all filters, ensuring there is only one single point for all filters. Server-side encryption is a process where all generated content will pass through an encoding function where the script tags will be replaced by its code. However, encrypting all of the data is not reliable and can consume resources and affect the performance of some servers.

• Output encoding. The purpose of output encryption is to convert unreliable input into a secure form where the input will be displayed as data to the user.

• Using library. There are many libraries helping the system administrator to prevent XSS, they help him to take steps to prevent XSS as listed above. Web frameworks also have a lot of technologies built in to prevent this type of attack.

### *2.3. Combined Code Injection Attacks*

In the current era of technology 4.0, the number of attacks is increasing and the properties of attacks are more complicated. Attacks do not occur in isolation but often work together to improve the level of harm to the information system. Suppose that when an attack occurs, a DDoS attack will be executed first. When receiving alerts from network security monitoring systems, the system administrator and the organization's information security experts often focus on restoring the system's performance. This is the main reason that makes them less aware of another attack that is happening in the system as malware injection attacks, especially SQL injection and XSS.

We can list some common mixed attacks as follows:

• SQL injection + insufficient authentication. This type of attack occurs when an organization expands its user base or changes its staff to network administration. Because the organization's security process and information security systems are not well understood, security parameters may not have been initialized and attackers can access sensitive content without verifying the user's identity or weak authentication. This is the vulnerability from which an attacker exploits this vulnerability to inject SQL code. So, the first step when taking over a new system for network administrators is to find out if the application or security system has enough authentication requirements. Check the old admin's reports for system weaknesses and the locations where attacks are not just SQL injection code injection attacks.

• SQL injection + DDoS attacks. This attack is used to disrupt server work, depleting system resources that cannot be accessed by legitimate users. While network administrators focus on restoring system performance, various SQL commands can be injected into the system. Especially when the DDOS attack has been overcome, the SQL injection has done its job.

• SQL injection + DNS hijacking. By using this type of attack, the attacker intends to embed the SQL query into the DNS request so that it can maliciously connect to the attacker's server.

• SQL injection + XSS. After inserted the XSS script, it will execute and try to connect with the database of an application. The extraction code for data from the database can be implemented using the iframe command.

• Content spoofing via compounded SQL injection. Content spoofing attacks focus on injecting malicious content into websites that users will be following misunderstand as legitimate content of the application. Content spoofing is essentially a client-side attack that allows an attacker to display content on a website by intelligently creating a website's URL. The attacker first uses testing techniques to find a weak website to inject malicious code later by inserting fake content into SQL statements to execute on the server.

To protect the web application of organizations from mixed attacks, information security experts can take the following measures.

• Constant update with new security policies and training for network administrators as well as new users in the system.

• Using a variety of system protection measures such as network layer firewalls to prevent DDoS attacks, web application firewalls to prevent attacks at the application layer.

• Application of modern techniques such as Machine Learning, Deep Learning in improving the performance of system protection devices.

• Dividing the system into separate parts to ensure working efficiency and have separate security policies suitable for each network partition.

With the complexity of current attacks, it is impossible to detect right and immediate attacks. The article focuses on analyzing code injection attacks by analyzing HTTP requests from which extracts URLs (an important component of HTTP requests) to detect general code injection attacks. The web application firewall is designed according to the "man in the middle" model so we can consider the proposed model to work with unencrypted requests. Especially for the CSIC 2010 data set, there are several typical attacks such as SQL code injection, XSS ... and there is a clear classification with two classes: the legal request class and the illegal request class have included hacking information.

### *2.4. Web Application Firewall*

Web application firewall (WAF) [14–16] is a special type of firewall that applies specifically to web applications. It is deployed in front of web applications and analyzes bidirectional web traffic (HTTP) – detecting and blocking all malicious content.

WAFs [17] controls the contents of packages at the application level. They provide a higher level of security compared to packet filters, but at the expense of loss of transparency for controlled services. WAFs act as a server for the client and client for the real server, processing requests from the real server instead of the users it protects.

WAF operates in one of two modes: passive or active. Active WAFs check all incoming requests, detecting vulnerabilities that allow for SQL injections, cross-site scripting, and spoofing parameters or cookies. Only legal requests are passed to the application. Passive WAFs work on the principle of an attack detection system: they also scan all incoming requests but do not block them when a potential cyberattack is detected.

When using WAF, all connections pass through it. As shown in figure, the connection starts on the client system and enters the internal interface of the firewall. The firewall accepts the connection, analyzes the contents of the packet and the protocol used, and determines whether this traffic complies with the rules of the security policy. Therefore, if the firewall initiates a new connection between its external interface and the server system.

WAFs use access modules for incoming connections. The access module in the firewall accepts the incoming connection and processes the commands before sending traffic to the recipient. Thus, the firewall protects systems from attacks carried out through applications.

WAFs contain access modules for the most commonly used protocols such as HTTP, SMTP, FTP, and telnet. Some access modules may be missing. If there is no access module, then a particular protocol cannot be used to connect through the firewall. WAFs can perform additional message checking, which a simple packet filter does not provide.

The disadvantages of WAF are lower performance but higher cost than packet filters; inability to use the RPC and UDP protocols.

In the next section, the author will consider the task of increasing the accuracy of query classification to protect a web application based on the application of the random forest method with analyzing attributes of HTTP queries.

## 3. PARAMETERIZATION OF TASKS OF INCREASING THE ACCURACY OF ATTACK DETECTION

In the field of machine learning, the task of dividing the set of observations into separate groups called classes, based on the analysis of their formal description. In classification, each observation unit belongs to a certain group or nominal category based on some qualitative property.

The classification problem is solved to use training with the teacher since classes are determined in advance and for example the training set, the given class labels. Analytical models that solve the classification problem are called classifiers.

To evaluate the operation of the algorithm, the following concepts will be used: a truly positive answer (TP); true negative response (TN); false-positive response (FP); false-negative response (FN). In the simplest case, the value Accuracy is a numerical estimate of the quality of the classification algorithm, determined by the formula (1):

$$\text{Accuracy} = \frac{\text{D}}{\text{S}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \quad (1)$$

where D is the number of documents by which the classifier made the right decision, and S is the size of the training sample.

This metric has one feature that needs considering. It assigns all documents the same weight, which may not be correct if the distribution of documents in the training sample is strongly biased towards one or more classes. In this case, the classifier has more information on these classes and, accordingly, within the framework of these classes, it will make more adequate decisions.

In addition to using Accuracy, many studies use the F-measure value. Since the F-measure (in formula (2)) is a harmonic mean between accuracy and completeness. It tends to zero if accuracy or completeness tends to zero. To relate accuracy to completeness, and F-measure is introduced as the harmonic mean of accuracy and completeness:

$$\text{F}_{\text{meansure}} = 2 * \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \quad (2)$$

where

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad (3)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \quad (4)$$

The classification problem is one of the main tasks of applying machine learning methods. It differs in other tasks in that the set of valid answers is finite. They are called class labels. A class is the set of all objects with a given label value. In the considered problem there will be two classes with labels 0 (not attack) and 1 (attack).

We consider the general classification problem $\langle X, Y, y^*, X^l \rangle$ where $X$ is the space of objects (query space), $Y$ is the set of answers (many classes), $y^*: X \to Y$ — target dependence.

The task of classifying queries will be described as:

$$\begin{cases} F : X \to Y = \{0,1\} \\ p(F(x_i) = y_j \mid i = j) \geq \tau, \end{cases} \quad (5)$$

where $X$ is the set of queries; $Y$ — class labels; $\tau$ — given threshold accuracy. The task of increasing the accuracy of query classification is called the search for the best approaches, the applicable machine learning method to increase the value of Accuracy in the information security system. In paragraph 4, we will consider the mathematical foundation of the components of the algorithm for increasing the accuracy of classification of queries based on the application of the machine learning method with tf-idf technology.

## 4. MATHEMATICAL FOUNDATION OF THE ALGORITHM

The algorithm has consisted of 3 main parts: technology tf-idf is used for converting string data to vectors, analysis attributes of HTTP queries and random forest method is worked for classification queries. Each parts of our algorithm will be described in this section.

### *4.1. Analyzing Attributes of Request*

**4.1.1. Query length (A1).** The author assumes that the length of the request sent from the user's browser varies lightly in a certain range. However, when hacker attacks the system, the length of the data fields may change, therefore, the length of the queries will be increasing in the case of SQL injection, XSS attacks. The author suggests using a change in the length of queries to identify attacks from users.

• In the training phase: let length of incoming queries are. The mathematical expectation of the dataset is $\mu$ and the variance is $\alpha^2$.

• In the detection phase: the dataset has mathematical expectation and variance. The application of Chebyshev's inequality gives an estimate of the probability that a random variable will take a value that is far from its average.

$$P(|x - \mu| > t) < \frac{\alpha^2}{t^2}, \quad (6)$$

where $x$ — random variable, $\tau = \frac{\alpha^2}{t^2}$ — threshold.

Accordingly, for any probability distribution with an average $\mu$ and variance $\alpha^2$ if value $x$ is obtained, then the deviation of $x$ from the average $\mu$ exceeds any threshold blocked by $\tau$.

In this task, the author chooses $t = |l - \mu|$ where $l$ is the length of the incoming query $x$. The larger the value, the closer the value of $l$ to average value, the greater the likelihood that the rest can happen

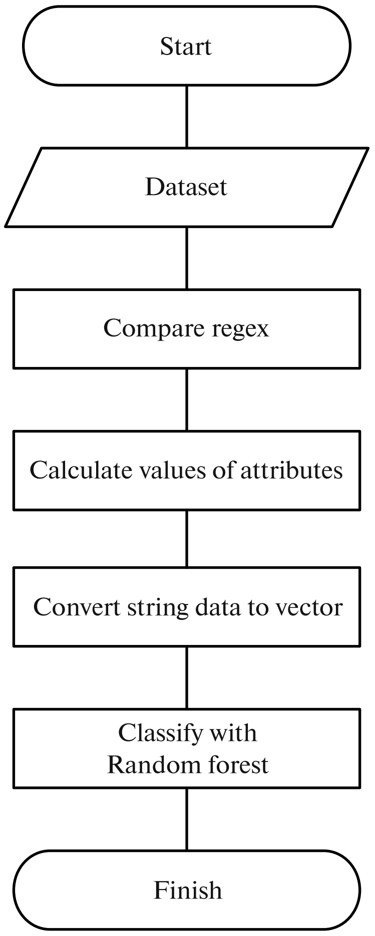
Start
Dataset
Compare regex
Calculate values of attributes
Convert string data to vector
Classify with Random forest
Finish

**Fig. 3.** Flowchart of algorithm.

$$A1 = P(|x - \mu| > |l - \mu|) = \frac{\alpha^2}{|l - \mu|^2}. \quad (7)$$

**4.1.2. Path query's length (A2).** The author assumes that the path length of the incoming query sent from the user's browser varies slightly in a certain range. However, when a hacker attacks the system, the length of the path may change, therefore, the length of the request increases. So, the author suggests using a change in the path length of an incoming query to identify attacks from users. Value A2 is calculated by analogy A1

$$A2 = \frac{\alpha_p^2}{|l_p - \mu_p|^2}, \quad (8)$$

where $l_p$ is path length of incoming query and the mathematical expectation of paths in dataset is $\mu_p$ and the variance is $\alpha_p^2$.

**4.1.3. Attribute length (A3).** The author assumes that the length of the fields of the query sent from the user's browser varies slightly in a certain range. When SQL injection or XSS attacks the system, the attribute length will be changed, so the length of the request increases. Therefore, the author suggests using a change in the length of fields in query to identify attacks from users. Value A3 for each filed in query is calculated by analogy A1

$$A3 = \frac{\alpha_f^2}{|l_f - \mu_f|^2}, \quad (9)$$

where $l_f$ is attribute length of incoming query and the mathematical expectation of each attribute in dataset is $\mu_f$ and the variance is $\alpha_f^2$.

### *4.2. Tf-idf Technology*

Tf-idf technology [18] is used to convert string data to vectors. Using this technology, the weights of the word in the string of queries will be evaluated.

The tf-idf technology is applied in our task, for each request the author will calculate the "weight" of each word in the request. For each word *t* in query *d*, the aggregate of queries *D* uses the formulas:

$$tfidf(t,d) = tf(t,d) \cdot idf(t,D). \quad (10)$$

In this formula, the values of tf, idf are calculated as:

$$tf(t,d) = \frac{count(t,d)}{\sum_{v \in d} count(v,d)} \quad (11)$$

other words in query *d*

$$idf(t,D) = \log\frac{|D|}{|d \in D : t \in d|}. \quad (12)$$

After the tf-idf calculation process, the string query data will be converted to vectors. This is an important step in the learning phase, and also in the detection phase to continue the algorithm since many machine learning methods work with numerical values.

### *4.3. Random Forest Method*

At the model training phase, many Req queries are specified, where each element of this set is described by 3 attributes and the class label takes the following values: $C_1$ (0 – not attack), $C_2$ (1 – attack).

The number of queries in the set Req will be called the size of this set and will be denoted by Req. Our task will be to build a hierarchical classification model in the form of a tree from a variety of Req queries. The process of building a tree will occur from top to bottom.

The random forest algorithm [19, 20] has consisted of a set of random decision trees. Therefore, this algorithm is an extension of the decision tree algorithm. Predictions from all trees are pooled to make the final

**Table 1.** Table of special symbols

| Symbols | Meaning |
|---|---|
| Ω | Classes of object |
| $F1_{score}$ | Value $F1_{score}$ |
| Q | List of queries |
| yBad | Vector illegal queries |
| yGood | Vector legal queries |
| y | Sum of yBad, yGood |
| X | Vector queries |
| $X_{train}$ | Vector X for training |
| $y_{train}$ | Vector y for training |
| $X_{test}$ | Vector X for testing |
| $y_{\text{test}}$ | Vector y for testing |

prediction; the mode of the classes for classification or the mean prediction for regression. As they use a collection of results to make a final decision, they are referred to as Ensemble techniques.

For each decision tree, WAF calculates the importance of a nodes using Gini Importance:

$$Info(\mathrm{Re}\,q) = -\sum_{j=1}^{2} \frac{fred(C_j, \mathrm{Re}\,q)}{|\mathrm{Re}\,q|} * \log_2 \frac{fred(C_j, \mathrm{Re}\,q)}{|\mathrm{Re}\,q|}. \quad (13)$$

(13) – Entropy of the Req query set. After splitting Req by A-attributes:

$$Info_A(\mathrm{Req}) = -\sum_{i=1}^{n} \frac{|\mathrm{Req}_i|}{|\mathrm{Req}|} * Info(\mathrm{Req}_i). \quad (14)$$

For each attribute, the author calculates a Gain value such as:

$$Gini(A) = Info(\mathrm{Req}) - Info_A(\mathrm{Req}). \quad (15)$$

These can then be normalized to a value between 0 and 1 by dividing by the sum of all feature importance values:

$$norm_{A_i} = \frac{Gini(A_i)}{\sum_{1}^{3} Gini(A_i)}. \quad (16)$$

The final feature importance, at the Random Forest level, is its average over all the trees. The sum of the feature's importance value on each tree is calculated and divided by the total number of trees:

$$RF_{A_i} = \frac{\sum_{j \in J} norm_{A_{ij}}}{T}, \quad (17)$$

where $RF_{A_i}$ – the importance of feature $A_i$ is calculated from all trees in the Random Forest model; J – the normalized feature importance for $A_j$ in tree j; T – total number of trees.

## 5. IMPLEMENTATION

With a description of the mathematical foundation of the algorithm above, a classification algorithm will be built. The algorithm consists of 2 phases: the learning phase and the detection phase. This paragraph will specifically address two phases. The scheme of the algorithm is shown in Fig. 3.

The training phase consists of 4 modules.

• Extraction module: according to requests received from the client, the author will filter the parts necessary for processing requests, including paths, payload, key symbols.

• Comparison regular expression module: regular expressions are patterns used to search for character sets that are combined into character strings. Regular expressions are not limited to a specific language, so a programmer and a specialist can apply them to any programming language. In the training phase: with the study of dangerous queries, the author created regular expressions. And then, in the detection phase: after reading requests from the target dataset, the author starts the regular expression check module.

• Data conversion module: is used to convert string data to a vector. Using tf-idf technology will appreciate the importance of the word in the request string.

• Data classification module: uses one of the most popular machine learning methods is the random forest method to classify a given data set and cross-validation to evaluate the accuracy of identifying dangerous request $\mathrm{F1_{score}}$.

The author used some symbols shown in Table 1 to describe the algorithm.

The data learning phase of the algorithm described below.

**Algorithm 1.** Work in the data training phase

```
Input: Q.
Output: $F1_{score}$.
1. Extract data: paths and parameters of queries.
2. Create regular expression from the incoming requests and add them to Table 2.
3. Calculate values of attributes from A1 to A3.
4. Convert data into vectors for all queries using tf-idf technology.
5. Separate data: data for training 80% ($X_{train}$, $y_{train}$) and data for testing 20% ($X_{test}$, $y_{test}$).
6. Data training using the random forest method.
7. Calculate the value of $F1_{score}$ after applying the random forest method and save the threshold in the database.
```

**Table 2.** Regular expression for detecting SQL injection

| Meaning | Expression |
|---|---|
| Regex for detection of SQL meta-characters | / (\%27) \| (\’) \|(\-\-) \| (\%23) \| (#)/ix |
| Modified regex for detection of SQL meta-characters | /((\%3D) \| (=)) [^\n] *((\%27) \| (\’) \|(\-\-) \|(\%3B) \| (;))/i |
| Regex for typical SQL Injection attack | /\w*((\%27) \| (\')) ((\%6F) \|o\|(\%4F)) ((\%72) \|r\| (\%52))/ix |
| Regex for detecting SQL Injection with the UNION keyword | / ((\%27) \| (\')) union/ix |
| Regex for detecting SQL Injection attacks on a MS SQL Server | /exec(\s\|\+) + (s \| x) p\w+/ix |
| Regex for simple CSS attack | /((\%3C) \|<) ((\%2F) \|\/) *[a-z0-9\%] +((\%3E) \|>)/ix |
| Paranoid regex for CSS attacks | /((\%3C) \|<) [^\n] +((\%3E) \|>)/I |
| Regex for “<img src” CSS attack | /((\%3C) \|<) ((\%69) \|i\| (\%49)) ((\%6D) \|m\|(\%4D)) ((\%67) \|g\| (\%47)) [^\n] +((\%3E) \|>)/I |

The detection phase consists of 4 modules (comparison module and 3 modules as in the training phase). After the extraction module works, all request data is sent to the comparison module. This module will check incoming queries with a list of saved regular expressions. If the request template does not match at least one saved template, then this request will be blocked. Otherwise, all the data of the remaining requests will be sent to the “calculate values of attributes” module.

**Algorithm 2.** Process in the phase of detecting illegal requests

**Input:** request **x**.

**Output:** 0 (illegal) or 1 (legal).

1. Extract data: path and parameters of queries in request **x**.
2. Compare request **x** with regular expression. If request matched only one regular expression is found in the database, then the request is blocked. Otherwise, the algorithm proceeds to step 3.
3. Calculate values of attributes from A1 to A3 for x.
4. Convert query data (words and key characters) **x** into a vector using tf-idf technology.
5. Compute the result using the random forest method.
6. Calculate the value of $F1_{score}$ to evaluate the accuracy of the algorithms.

## 6. EVALUATION

### *6.1. Data Collection*

In this work, the author uses 3 datasets listed below.

Web Application dataset. This dataset as mentioned was captured inbound to the Webapp, and as such, some of the unique features include information about the TCP and HTTP packets. Specific features that have been useful for machine learning in our experiments are the tcp.length and http.content_length.

Datiphy dataset. The Datiphy dataset is unique as it’s capturing features typical of log analysis such as the SQL statement that results from the request to the web application, but it’s also capturing and correlating features such as Response Length and Result that are returned from the database server to the web application. To our knowledge, this is the first project to use such a dataset.

CSIC 2010. The HTTP dataset CSIC 2010 contains thousands of web requests automatically generated. It can be used for the testing of web attack protection systems. It was developed at the “Information Security Institute” of CSIC (Spanish Research National Council). The author chooses only URL, query string and paths of each request from the full request in CSIC 2010.

Table 2 presents some regular expressions for detecting SQL injection attacks.

### *6.2. Performance Analysis*

The research works to detect attacks on the application layer specifically with code injection attacks

**Table 3.** Comparison of our results with other researches on CSIC 2010

| Detection method | TPR, % | FPR, % |
|---|---|---|
| Nguyen [21]-C4.5 | 94.49 | 5.9 |
| Kozik – ELM[a] [22] | 94.98 | 0.79 |
| Kozik – REPTree[b] [23] | 98 | 1.5 |
| Loffler [24] – RF[c] + SVM | 96.27 | 14.38 |
| Soltes [25] – DDCA[d] | 87.88 | 12.71 |
| Our result | 98.76 | 3.2 |

[a]Extreme Learning Machine Forest.
[b]Reduced Error Pruning Tree.
[c]Random Forest.
[d]Deterministic Dendritic Cell Algorithm.

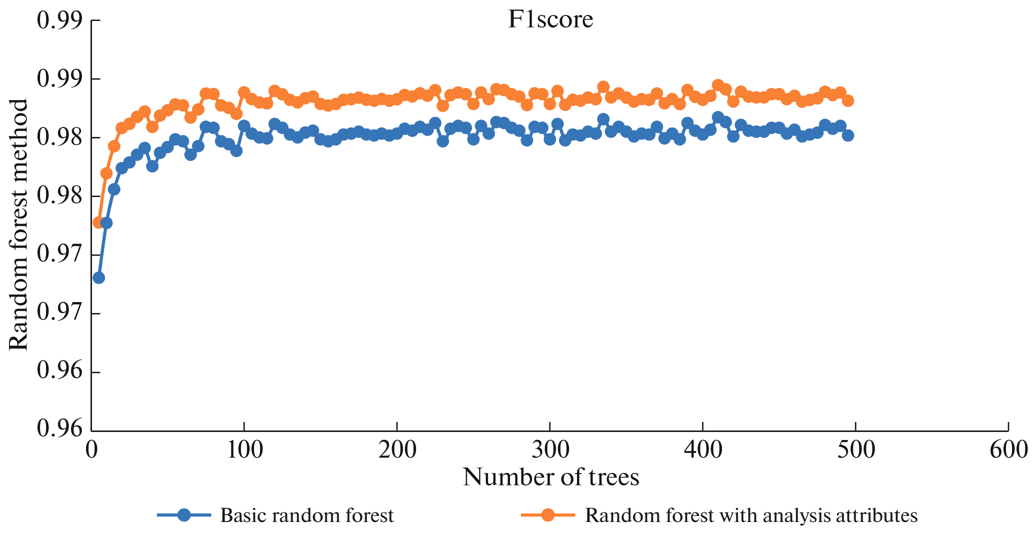

**Fig. 4.** Dependence of random forest method on the number of trees.

studied by many scientists around the world. Especially when using machine learning algorithms in the task of attack detection we realize that each algorithm has its advantages (good results) over several specific datasets. There is no one best algorithm for all datasets used for training and detecting. In this case, the author has also compared results with the best results from other papers with CSIC 2010 dataset (see Table 3). The author has chosen metrics TPR and FPR for comparison.

Accuracy is the classification accuracy achieved with the various algorithms are tested with 10-fold cross-validation and 3-gram. Result of working Random forest method when the author inscribed the number of trees in the forest on dataset CSIC 2010 is described in Fig. 4.

The results of our experiments with random forest algorithm and modification algorithm (random forest method with analysis attributes queries and compare regular expression) are summarized in Fig. 5. When

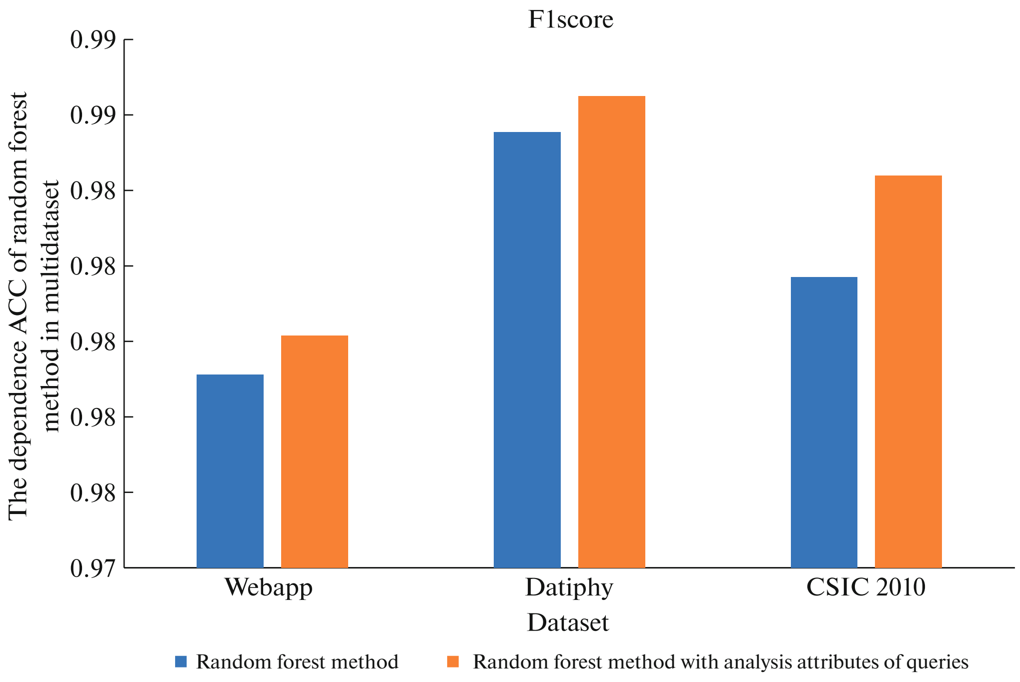

**Fig. 5.** Depends on $F1_{score}$ with the number of trees.

testing with each data set, the classification accuracy of the random forest algorithm is different. For the Datiphy dataset, the classification accuracy of the algorithm has the highest results. The author recognizes that in all three cases, using the new approach is worked more efficiently than working with the basic random forest algorithm in [11], especially in dataset CSIC 2010.

## 7. CONCLUSIONS

SQL injection attacks, XSS, and web-based attacks in general continue becoming a major issue in the security of financial and other fields of human life. The importance of system security problem has increased because almost social processes more dependent on the Internet. In this article, the author has proposed a novel algorithm to detect SQL injection and XSS by machine learning method (random forest) with analysis three attributes of queries. The novel algorithm has a good result described in Section 6.

In addition, with the development of database management systems that do not use SQL operators, such as MongoDB, Hadoop/HBase, ... NoSQL injections attacks [26] have occurred. So, the code injection attack detection is becoming more significant. Future works are including a collection of additional data such as traffic outbound of the web application from the user browser, collection of larger data to estimate performance algorithms. For the sake of increasing the amount of data, the problem of paralleling algorithms for data processing and increasing the calculating power of the system will get relevant.

## REFERENCES

1. An, X., Su Ji, Lu, X., and Lin, F., Hypergraph clustering model-based association analysis of DDOS attacks in fog computing intrusion detection system, *EURASIP J. Wireless Commun. Networking,* 2018, no. 1, p. 249.
2. Clotet, X., Moyano, J., and Leon, G., A real-time anomaly-based IDS for cyber-attack detection at the industrial process level of critical infrastructures, *Int. J. Crit. Infrastruct. Prot.,* 2018, vol. 23, pp. 11–20.
3. Aljawarneh, S., Aldwairi, M., and Yassein Muneer, B., Anomaly based intrusion detection system through feature selection analysis and building hybrid efficient model, *J. Comput. Sci.,* 2018, vol. 25, pp. 152–160.
4. Siddiqui, Md.A. et al., Detecting cyber attacks using anomaly detection with explanations and expert feedback, *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP 2019),* Brighton, 2019, pp. 2872–2876.
5. Nikisins, O., Mohammadi, A., Anjos, A., and Marcel, S., On effectiveness of anomaly detection approaches against unseen presentation attacks in face anti-spoofing, *Proc. Int. Conf. on Biometrics (ICB),* Gold Coast, 2018, pp. 75–81.
6. Inoue, K., Honda, T., Mukaiyama, K., Ohki, T., and Nishigaki, M., Automatic examination-based whitelist generation for XSS attack detection, *Proc. Int. Conf. on Broadband and Wireless Computing, Communication and Applications,* Springer, 2018, pp. 326–338.
7. Melis, L., Pyrgelis, A., and De Cristofaro, E., On collaborative predictive blacklisting, *ACM SIGCOMM Comput. Commun. Rev.,* 2019, vol. 48, no. 5, pp. 9–20.
8. Chen, X.L., Li, M., Jiang, Y., and Sun, Y., A comparison of machine learning algorithms for detecting XSS attacks, *Proc. Int. Conf. on Artificial Intelligence and Security,* Springer, 2019, pp. 214–224.
9. Zhang, J., Jou, Y.-T., and Li, X., Cross-site scripting (XSS) detection integrating evidences in multiple stages, *Proc. 52nd Hawaii Int. Conf. on System Sciences,* Grand Wailea, 2019.
10. Fang, Y., Li, Y., Liu, L., and Huang, C., Deepxss: Cross site scripting detection based on deep learning, *Proc. 2018 ACM Int. Conf. on Computing and Artificial Intelligence,* Chengdu, 2018, pp. 47–51.
11. Ross, K., SQL injection detection using machine learning techniques and multiple data sources, *Master's Project,* 2018.
12. Moh, M., Pininti, S., Doddapaneni, S., and Moh, T.-S., Detecting web attacks using multi-stage log analysis, *Proc. IEEE 6th Int. Conf. on Advanced Computing (IACC),* IEEE, 2016, pp. 733–738.
13. Kar Debabrata, Sahoo Ajit Kumar, Agarwal Khushboo, Panigrahi Suvasini, and Das Madhabananda, Learning to detect SQLIA using node centrality with feature selection, *Proc. Int. Conf. on Computing, Analytics and Security Trends (CAST),* IEEE, 2016, pp. 18–23.
14. Phonsa, V., Kim, H., and Andrews, D., US Patent 9,660,960, 2017.
15. Yuan, H. et al., Research and implementation of WEB application firewall based on feature matching, *Proc. Int. Conf. on Application of Intelligent Systems in Multimodal Information Analytics,* Springer, 2019, pp. 1223–1231.
16. Keijer, J., Automated DDoS mitigation based on known attacks using a web application firewall, *B.S. Thesis,* Univ. of Twente, 2019.
17. Akbar Memen, Ridha Muhammad Arif Fadhly, et al., SQL injection and cross site scripting prevention using OWASP ModSecurity WebApplication firewall, *Int. J. Inf. Visualization,* 2018, vol. 2, no. 4. pp. 286–292.
18. Zhan, J. et al., An effective feature representation of web log data by leveraging byte pair encoding and TF-IDF, *Proc. ACM Turing Celebration Conf.-China,* ACM, 2019, p. 62.
19. Rong, W., Zhang, B., and Lv, X., Malicious web request detection using character-level CNN, *Proc. Int. Conf. on Machine Learning for Cyber Security,* Springer, 2019, pp. 6–16.
20. Betarte, G., Pardo, A., and Martınez, R., Web application attacks detection using machine learning techniques, *Proc. 17th IEEE Int. Conf. on Machine Learning and Applications (ICMLA),* IEEE, 2018, pp. 1065–1072.
21. Nguyen, H.T., Torrano-Gimenez, C., Alvarez, G., Petrovic, S., and Franke, K., Application of the generic feature selection measure in detection of web attacks, in *Computational Intelligence in Security for Information*

*Systems,* Herrero, Á. and Corchado, E., Eds., Berlin, Heidelberg: Springer, 2011.

22. Kozik, R., Choraś, M., Holubowicz, W., and Renk, R., Extreme learning machines for web layer anomaly detection, in *Image Processing and Communications Challenges 8,* Choraś, R.S., Ed., Cham: Springer Int. Publ., 2017, pp. 226–233.
23. Kozik, R. and Choras, M., Adapting an ensemble of one-class classifiers for a web-layer anomaly detection system, *Proc. 10th Int. Conf. on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC),* Krakow, 2015, pp. 724–729.
24. Loffler, M., Improvement of intrusion detection using multiple classifier model, *Diploma Thesis,* FIIT STU, 2017.
25. Šoltes, F., Improving security of a web system using biology inspired methods, *Diploma Thesis,* FIIT STU, 2016.
26. Eassa, A.M., Elhoseny, M., El-Bakry, H.M., and Salama, A.S., NoSQL injection attack detection in web applications using RESTful service, *Program. Comput. Software,* 2018, vol. 44, no.6, pp. 435–444.