# Advancing Intrusion Detection in Fog Computing: Unveiling the Power of Support Vector Machines for Robust Protection of Fog Nodes against XSS and SQL Injection Attacks

**Abstract:**

Fog computing, characterized as a cloud infrastructure in close proximity to end devices, faces substantial security challenges that necessitate robust intrusion detection mechanisms for fog nodes. The resource-constrained nature of fog nodes renders them particularly susceptible to attacks, making the development of efficient intrusion detection systems imperative. In this study, we propose a comprehensive approach to protect fog nodes, taking into account their limited resources. Leveraging the power of Support Vector Machines (SVMs), a widely adopted machine learning technique in IoT security, our method overcomes challenges associated with local optima, overfitting, and high-dimensional data. A thorough literature review underscores the prevalent use of SVMs in IoT security research. Specifically, we focus on addressing two prevalent web attacks: Cross-Site Scripting (XSS) and SQL injection attacks, based on global statistical data. To evaluate our approach, we employ the CSE-CIC-IDS2018 dataset and a pseudo-real dataset. Precision, recall, and accuracy are employed as evaluation metrics, along with the Mean Average Precision (MAP). Our evaluation results demonstrate an exceptional level of accuracy, achieving an impressive 98.28% accuracy in terms of average performance when compared to existing methods. Comparative analysis with state-of-the-art approaches further validates the superior efficacy and efficiency of our proposed method..

**Keywords:** Fog Computing, Internet of Things, Intrusion Detection, Support Vector Machines, XSS and SQL Injection Flaw Attacks.

## 1. Introduction

Large-scale IoT developments create conditions that cloud computing is not capable of effectively controlling. Fog computing applications have grown rapidly in current IoT end devices of their ability to respond to edge components rapidly. The advantages of fog computing may include the reduction of bandwidth consumption as well as the reduction of network latency. Smart automated machine systems, smart vending machines, and smart chip systems are practical examples of the application of fog computing on the Internet of Things. The concept of fog computing is a new perspective that enables the Internet of Things to run its applications on the edge of the network [1]. Fog computing is not an alternative to cloud computing but is an expander that complements the concept of smart devices that can work on the edge of the network. Computing is the gateway between cloud computing and the Internet of Things. Fog is an extension of the cloud, so, inevitably, some of the

security challenges of cloud computing will not continue. While some existing methods in the field of fog computing can solve many security and privacy issues in cloud computing, fog computing brings new security challenges due to its distinctive features, including resource constraints. These challenges affect the adaptation of fog computing to the Internet of Things.

All fog nodes allow the user to process a portion of the data without having to send it to the cloud data center. While data centers are equipped with many resources such as processors, power, and memory, devices equipped with these resources are not abundant. This means that conventional methods are not suitable for preventing fog intrusion, as these methods will delay or consume more energy. Therefore, there is a need for a robust security system that uses a small number of resources to protect the entire layer from attacks [2]. Intrusion detection methods detect IoT misbehavior or malicious devices and notify others on the network to take action. The nature of IoT environments makes it difficult to detect attacks globally. The location of the cloud computing services is on the Internet, and the fog computing services are located at the edge of the local network. In other words, fog computing security can be defined, but cloud computing security cannot be defined [3]. However, focusing on fog layer nodes can bring security to a simpler level [4]. On the other hand, the advantages and capabilities of the SVMs have led researchers to use them to detect intrusion. These capabilities include designing the classifier with maximum generalization, achieving global optimization, automatically determining the optimal structure and topography for the classifier, modeling nonlinear differentiation functions using nonlinear kernels, and the Hilbert space's inner product as well as ease of working with high-dimensional data [5]. According to [6] and [7], SVMs are the most widely used algorithm, technique, and learning method used for IoT security papers, both for intrusion detection and authentication. The neural network, Bayesian, and decision tree methods are in the next categories, respectively. Despite the mentioned advantages, determining the appropriate kernels as well as the right value of their parameters is one of the open issues in support vector machines. The main idea of the SVMs is to choose a single separator, to maximize the separator margin of the two categories. Accordingly, different types of support vector machines are defined [8], including hard-edge linear support vector machines, soft-edge linear support vector machines, and nonlinear support vector machines. In general, the following advantages can be mentioned for SVMs:

- There is no local minimal because the solution is a QP problem,
- The optimal solution can be found in polynomial time,
- Few model parameters to select: the penalty term C, the kernel function, and parameters (e.g., spread $\sigma$ in the case of RBF kernels),
- The final results are stable and repeatable (e.g., no random initial weights),
- SVMs solution is sparse; it only involves the support vectors,
- SVMs represent a general methodology for many PR problems: classification, regression, feature extraction, clustering, novelty detection, etc.

- SVMs rely on elegant and principled learning methods,
- SVMs provide a method to control complexity independently of dimensionality,
- SVMs have been shown (theoretically and empirically) to have excellent generalization capabilities.

In the next section, we will see how these advantages have attracted researchers. The rest of this manuscript is prearranged as follows: a brief review of recent research related to our proposed technique is presented in section 2. Section 3 depicts fog Security and our motivation. Section 4 focuses on web attacks and defined Injection Flaws as the most harmful type of these. The architecture and data flow diagram of the proposed system is presented in section 5. The details of implementation are given in section 6. Section 7 presents implementation, system evaluation, and results. Finally, the conclusions are summed up in section 8.

## 2. Literature review

Support Vector Machines (SVMs) have been one of the most successful machine learning techniques for the past decade. The enumerated advantages of SVMs have long attracted intrusion detection systems. Recently, many papers have been published in intrusion detection addressing SVMs, including:

The author in [8] used an unsupervised One-Class SVM approach to detect, similar to a supervised learning approach to reduce false alarms. They suggested the following to improve the performance of the system:

- Create an index of normal network packages to support vector machines learning without using default knowledge.
- Filter network packets to prevent incomplete network traffic that violates the TCP/ IP standard.
- Feature extraction uses the genetic algorithm to obtain optimal information from Internet packages.
- Take into account the time relationship of packages in the pre-processing stage.

[9] introduces robust SVMs to increase the efficiency of a One-Class vector machine for detecting unsupervised abnormalities. Its main idea is to minimize the role of data outliers in the decision-making frontier. [10] searches abnormal behaviors by training OC-SVM with normal behaviors. One-class classification approaches are essentially helpful in solving two-class learning problems, whereby the first class which is mostly well-sampled is known as the ‘target’ class, and the other class which is severely under-sampled is known as the ‘outlier’ class. The goal is to build a decision surface around the samples in the target class to distinguish the target objects from the outliers (all the other possible objects). [11] used robust SVMs to solve the over-fitting problem in irregular data, assuming that normal data is mixed with abnormal data. Their experiments showed that intrusion detection based on

the word processing model generates a large number of false-positive alert rates and is difficult to apply in practice.

[12] has also considered intrusion detection as a word processing problem, and using the term frequency-inverse document frequency (tf-idf) changes in the weighting with robust SVMs and OC-SVMs have shown changes lead to better results compared to normal conditions. [13] offers a Nested set SVMs for intrusion detection. In this method, instead of labeling with criteria such as geometric mean accuracy, the information of the farthest and nearest neighbors of each sample is used. Experimental results show that this method has performed better than the basic method. SVMs have always been compared to neural networks. [14] has compared the support vector machines with neural networks and shown that the support vector machines work best with tf-idf weighting, while the simple weighting neural network produces the worst response. It also shows that the Gaussian kernel support vector machines provide better results than the RBF neural networks. [15] has used a combination of neural networks and support vector machines with RBF kernels to strengthen the intrusion detection system. The emergence and pervasiveness of cloud computing in recent decades and the need for security in it have led researchers to provide methods to detect intrusion in these contexts. [16] introduces a Density-Based Support Vector Machines. The basic idea is based on the density of each class in the dataset. Training data is converted to a binary sequence. According to this sequence, the educational model of system behavior is obtained. To run, it uses the Map Reduce Parallel Computational Framework in Hadoop provided by Apache to reliably process large-scale distributed data. [17] introducing the optimized SVMs has used a combination of genetic algorithms and SVMs to enhance the ability to generalize classification and detect denial-of-service attacks in cloud computing. [18] has developed a hybrid system based on deep learning that uses a stacked auto-encoder to reduce feature dimensions and the SVMs classification algorithm to detect malicious attacks. [19] uses the SVMs to classify network data into normal behavior and attack behavior, as well as to remove irrelevant and redundant features. Finally, it introduces an invasive detection system based on SVMs and Information Gain (IG). [20] provides a combination of fuzzy clustering and SVMs to improve the accuracy of the detection system in the cloud computing environment. [21] uses SVMs to classify network connections. The nature of the traffic is notified to the network administrator to disconnect and block any intruders to the network. Besides, Binary Particular Swarm Optimization is used to select the most relevant network features and Standard Particular Swarm Optimization is used to adjust the control parameters of the SVMs. [22] provides the SVMs intrusion detection scheme based on cloud-fog collaboration. This design uses the Principal Component Analysis (PCA) method to reduce the dimensions, eliminate the correlation between the features and reduce the training time. The cloud server uses a Particular Swarm Optimization SVMs to complete the dataset training operation and achieve optimal classification. The obtained results are then sent to the fog node and the attack detection operation is performed on the fog node.

[23] recommends a Software-Defined Networking (SDN) based intrusion detection system using the SVMs along with Selective Logging for IP Traceback. Using IP, in addition to saving a high percentage of memory consumption, allows the actual source of packets to be tracked in the event of an attack. Detection of anomalous traffic and network intrusion is done during the PACKET_IN event at the controller and then again by fetching the flow statistics from the OpenFlow switches at regular intervals. Selective logging of suspicious packets/flows during a PACKET_IN event enables an IP traceback to be performed in the eventuality of an attack that can be initiated by a network admin using an HTTP-based web console. Logging is performed selectively at the controller and not at the switches, achieving significant savings in terms of overall memory resources. Moreover, logging is performed using the in-memory structure at the controller thereby enhancing the performance of the logging operation over the traditional file-based database. Not all the features captured from the network packets contribute to detecting or classifying attacks. Therefore, the objective of [24] research work is to study the effect of various feature selection techniques on the performance of IDS. Feature selection techniques select relevant features and group them into subsets. This paper implements Chi-Square, Information Gain, and Recursive Feature Elimination (RFE) feature selection techniques with machine learning classifiers such as SVMs, Naïve Bayes, Decision Tree Classifier, Random Forest Classifier, K-Nearest Neighbors, Logistic Regression, and Artificial Neural Networks. Authors in [25] have implemented six of the most popular ML models that are used for IDS, including Decision Tree, Random Forest, Support Vector Machines, Naïve Bayes, Artificial Neural Network, and Deep Neural Network. Their experiments using the CIC-IDS2017 and the CSE-CIC-IDS2018 datasets show that at first SVMs and the second ANN are most resistant to overfitting and have better results. Besides that, their experiment results also show that DT and RF suffer the most from overfitting, although they perform well on the training dataset. Table 1 summarizes the related work with the limitations of the work discussed. More details of these are going to present at end of section 7.

Table 1: The related work with limitations of the work discussed

| Reference Number | Year of Publication | Description |
|---|---|---|
| [11] | 2003 | A robust SVMs to solve the over-fitting problem in irregular data, assuming that normal data is mixed with abnormal data. |
| [12] | 2005 | Using term frequency-inverse document frequency (tf-idf) changes in the weighting with robust SVMs and OC-SVM. |
| [8] | 2007 | An unsupervised One-Class SVM approach to detection, similar to a supervised learning approach to reduce false alarms, suggested 4 approaches to improve the performance of the system. |

| | | |
|---|---|---|
| [14] | 2008 | An SVM based on entropy and tf-idf. |
| [9] | 2013 | A robust SVMs to increase the efficiency of a One-Class vector machine for detecting unsupervised abnormalities. |
| [15] | 2013 | Combination of SVMs and neural networks with RBF kernels in cloud computing. |
| [16] | 2015 | Density-based binary SVM in cloud computing. |
| [13] | 2018 | A Nested One-Class SVM for intrusion detection. |
| [17] | 2019 | Optimized SVMs with the combination of genetic algorithm and SVMs. |
| [21] | 2019 | Applied Binary Particular Swarm Optimization (PSO) to select the most relevant network features and Standard Particular Swarm Optimization to adjust the control parameters of the SVMs in cloud computing. |
| [10] | 2020 | Training OC-SVM with normal behaviors for searching for abnormal behaviors in cloud computing. |
| [18] | 2020 | Hybrid system with the combination of deep learning and SVMs in cloud computing. |
| [19] | 2020 | Combination of SVMs with information gain in cloud computing. |
| [20] | 2020 | Combination of SVMs with fuzzy clustering (FCM) in cloud computing. |
| [23] | 2021 | A Software-Defined Networking (SDN) with the SVMs and Selective Logging for IP Traceback. |
| [24] | 2021 | Comparison of machine learning methods such as SVMs, Naïve Bayes, Decision Tree Classifier, Random Forest Classifier, K-Nearest Neighbors, Logistic Regression, and Artificial Neural Networks. |
| [22] | 2022 | Combination of SVMs, principal component analysis (PCA), and particle swarm optimization in fog computing. |
| [25] | 2022 | Comparison of machine learning methods such as Decision Tree, Random Forest, Support Vector Machines, Naïve Bayes, Artificial Neural Networks, and Deep Neural Network |

Accordingly, in a nutshell, the literature indicates the very prominent role of the use of SVMs in IoT security, cloud computing, and fog computing. This is our motivation to utilize SVMs to protect fog nodes from IDS.

## 3. Fog Security and Motivation

Fog computing is a decentralized computing architecture whereby data is processed and stored between the source of origin and cloud infrastructure. This results in the minimization of data transmission overheads, and subsequently, improves the performance of computing in cloud platforms by reducing the requirement to process and store large volumes of superfluous data [26].In addition to the listed benefits, this issue also brings risks. Fog security, if compromised, directly affects the security and trust of all applications and users. The sensitivity of fog nodes is often higher than that of cloud servers such as IoT devices due to their limited resources [27]. Attack levels are also wider for fog nodes because they are prone to inaccurate information and malware, service manipulation, and information leakage. Attacking fog nodes can be more dangerous than attacking IoT devices because they usually have private information and privacy concerns, and trust in relationships with more nodes and remote items. The nature of the fog pattern naturally increases the threats of rogue and unreliable structures [28], because they have less computing power and are closer to the attacker than the cloud [22]. This is the motivation to protect fog nodes from IDS.

There is a very subtle difference between fog computing and edge computing, which is worth noting. Fog computing is different from edge computing in its ability for Fog nodes to interconnect, while edge computing operates with separate edge nodes [29]. Fogging is a smart computing system in which fog nodes can independently respond to computing and processing requests of end devices and can connect for collaboration. Management and Collaborative Procedures are applied to fog nodes for management and control practices. Cooperation between fog nodes can be done through telecommunications or local communication between them [30]. For example, Figure 1 shows the fog grid architecture, which consists of fog nodes and intrusion detection nodes. The symmetry and proximity of intrusion detection nodes to fog nodes ensure that the deployment of detection nodes reduces latency. Each detector node, located one step away from the other nodes, observes all the nodes inside a circle with a radius of one step from it in a Wheel Spoke Fashion. Whenever it finds a compromised node or a threatened node, it simply notifies the nearest node to disconnect from it. Fog nodes here can be a single device or a platform/networking layer that sits in between a cloud and a collection of IoT devices. In addition, packets move from origin to destination by moving in several steps along the Y-axis (i.e. only moving up and down). The packets move along the Y-axis until they reach their destination. The packets then move along the X-axis, i.e. backward or forwards, until they reach their exact destination [31].

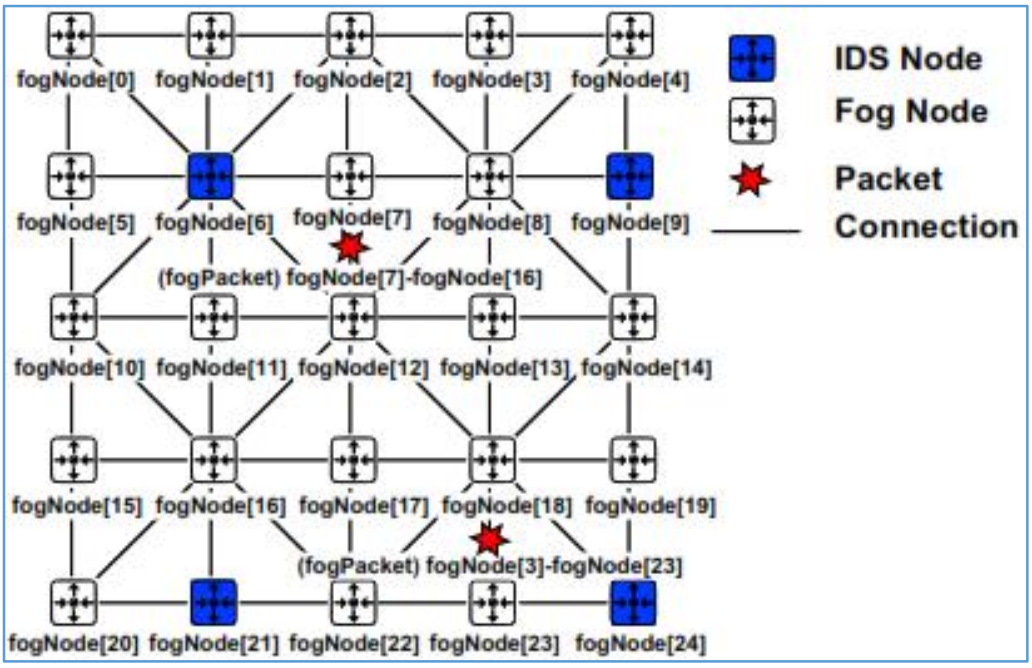


Figure 1: Security model architecture in fog computing and how nodes are located and cooperated [2] and [31].

## 4. Web attacks

Web attacks exploit vulnerabilities on the Web to circumvent the security policies of Web applications. Web attacks use the HyperText Transfer Protocol (HTTP) or HyperText Transfer Protocol Secure (HTTPS) protocol. The HTTP protocol uses port 80 and the HTTPS protocol uses port 443. Web attacks typically use these two ports to circumvent Web policies [32]. All a web hacker needs are a web browser and an internet connection. The latest official report on the frequency of web attacks is related to the OWASP site. According to the report, the ten most web vulnerabilities in 2017 are as follows [33].


Figure 2: The latest report on the most vulnerabilities of the web in 2017 [34]

According to Figure 2, Injection Flaws are the most harmful type of Web attack and are similar to fog computing. There are a lot of research papers that proposed features for detecting SQL injection and XSS attacks. Reference number [34] is one of these researches. Accordingly, provided professional study, the current paper addresses two types of Injection Flaws as follows. Another attack is going to be reviewed in future works.

### 4-1. SQL Injection Flaws

The SQL Injection Flaws include the main type of injections. These are only malicious queries that change a normal SQL query into a malicious one and consequently allow anomalous database access and processing. SQL Injection Flaws are the method by which an intruder executes an application in the database layer. Injection occurs when user-provided data is sent to an interpreter as part of a command or query. The attackers deceive the interpreter and force them to carry out unplanned orders. Injection Flaws allow attackers to create, read, update, or delete any arbitrary data available in the application. In the worst case, these flaws allow attackers to exploit the application and the systems under it; they even pass through deeply nested firewalls. All web application frameworks that use interpreters or invoke other processes are vulnerable to injection flaws [35]. Different types of SQL injection vulnerabilities include [36]:

- **Improperly refine escape characters**: This type of SQL injection vulnerability occurs when the user input of escape characters is not refined and sent in a SQL command. This causes major manipulations of the instructions executed on the database by the end-users of the application.
- **Print mismanagement**: This type of SQL injection occurs when a user-generated field is not strongly written or tested for printing restrictions. This happens when a numeric field is used in the SQL statement, but the programmer does not check if the data provided by the user is numeric.
- **Blind SQL Injection**: Blind SQL injection is used when a web application is more vulnerable to SQL injection, but the injection results are not visible to the attacker. The data representing the page may not be vulnerable, but showing the called legal results of the logical command injected into the SQL commands of the page. This type of attack can be very time-consuming, as a new command must be provided per a recovered bit. Many tools can automatically detect these attacks when the location of the vulnerability and target information is known. The types of blind SQL injection attacks are as follows:
  1- **Conditional responses**: A type of blind SQL injection that forces the database to execute a logical command on the page of a typical application.
  2- **Conditional errors**: This type of blind SQL injection causes a SQL error by forcing the database to execute a single command. If the WHERE command is correct, it will cause an error.

3- Time Delays: Time delays are a type of SQL blind injection that causes the SQL engine to execute a long continuous query or a time delay command depending on the logic injected. The attacker can then measure the length of time the page has been loaded to determine if the injected command is correct.

**4-2. XSS attacks**

XSS attacks are a subset of HTML injection flaws, found in web applications that allow users to inject code into web pages viewed by other users. Examples of such code are HTML, XHTML, and client-side scripts. An XSS attack can be used by attackers to bypass access control. An attacker can intelligently find ways to inject malicious scripts into web pages and gain high access to content on sensitive pages and various other objects. XSS allows hackers to execute scripts in the victim's browser that can steal user sessions, corrupt websites, enter malicious programs, and infiltrate the user's browser by executing scripts. This is very dangerous in systems such as content management systems, blogs, and forums where a large number of users view data from other users [37]. The well-known types of this flaw include [38]:

- **DOM-based XSS attack**: **DOM Based XSS** (or as it is called in some texts, “type-0 XSS”), is based on the standard object model for displaying HTML or XML. In this type of vulnerability, there is a problem with the client-side page script. For example, if a part of JavaScript accesses the URL parameter of the request and uses this information to write some HTML code on its page, and this information is not encrypted with the HTML entities, then there is an XSS hole where the written information can be interpreted again by browsers. In practice, this type of vulnerability is similar to the non-persistent one, except in a very important case. In older scripting models, a script was placed on objects in the local area on the client side, such as the local hard drive, and by using an XSS hole of this type on the local page, it could perform remote vulnerabilities. For example, if an attacker hosts a malicious website, which includes a link to a vulnerable page on the client's local system, a script could be injected and executed on the system with the user's browser access level. Thus, not only can the information normally obtained by XSS be accessed, but also all customer-side information can be accessed.
- **Non-Persistent attacks:** A non-persistent attack or a type of XSS, also known as a reflected vulnerability, is the most common type of XSS attack. It is easier to implement the reflected type than any other type. These holes are observed when the data provided by the client web is immediately used by the server-side scripts to generate that user's results page. If invalid data provided by the user is placed on the result page without HTML encryption, it allows the client-side code to be injected into the dynamic page. A traditional example of this type of search engine attack is this: If someone searches for a string that contains some specific HTML characters, the search string is usually displayed on the result page to indicate what

was searched, or at least the search terms are placed in the text box for easier editing. An XSS hole is created if not all search term events are HTML-encrypted entities. At first glance, this does not seem like a big deal, as users can only inject code into their pages. This is when the attacker can persuade the user to follow a malicious URL that injects code into the result page, thereby giving the attacker full access to the page content.

- **Persistent attacks:** Persistent or type 2, or persistent vulnerability is known as a Stored XSS (or double-vulnerability) and allows the strongest XSS attacks. XSS Type 2 Vulnerability occurs when user-generated data for a web application is first stored permanently on a server such as a database or a file system and then displayed to users on a web page without encryption using HTML entities. A traditional example of this is an online messaging attack where users are allowed to send HTML messages for other users to view. The persistent XSS is more important than the other types because, in this type, the malicious script of the attacker is transmitted more than once. Such attacks can potentially affect a large number of users and a virus or XSS worm can infect the application. Injection methods can be very different, and an attacker may not have the prerequisites to use a web application to execute such an attack alone. Any data received by the web application, such as e-mail or system logs that can be controlled by an attacker, must be encrypted before it can be re-displayed on the dynamic page, otherwise, an XSS vulnerability of this type could occur.

## 5. Architecture and data flow diagram of the proposed system

Figure 3 depicts the proposed system architecture consisting of six components: data provider, preprocessor, analyst, manager and controller, responder, and intrusion detection system.

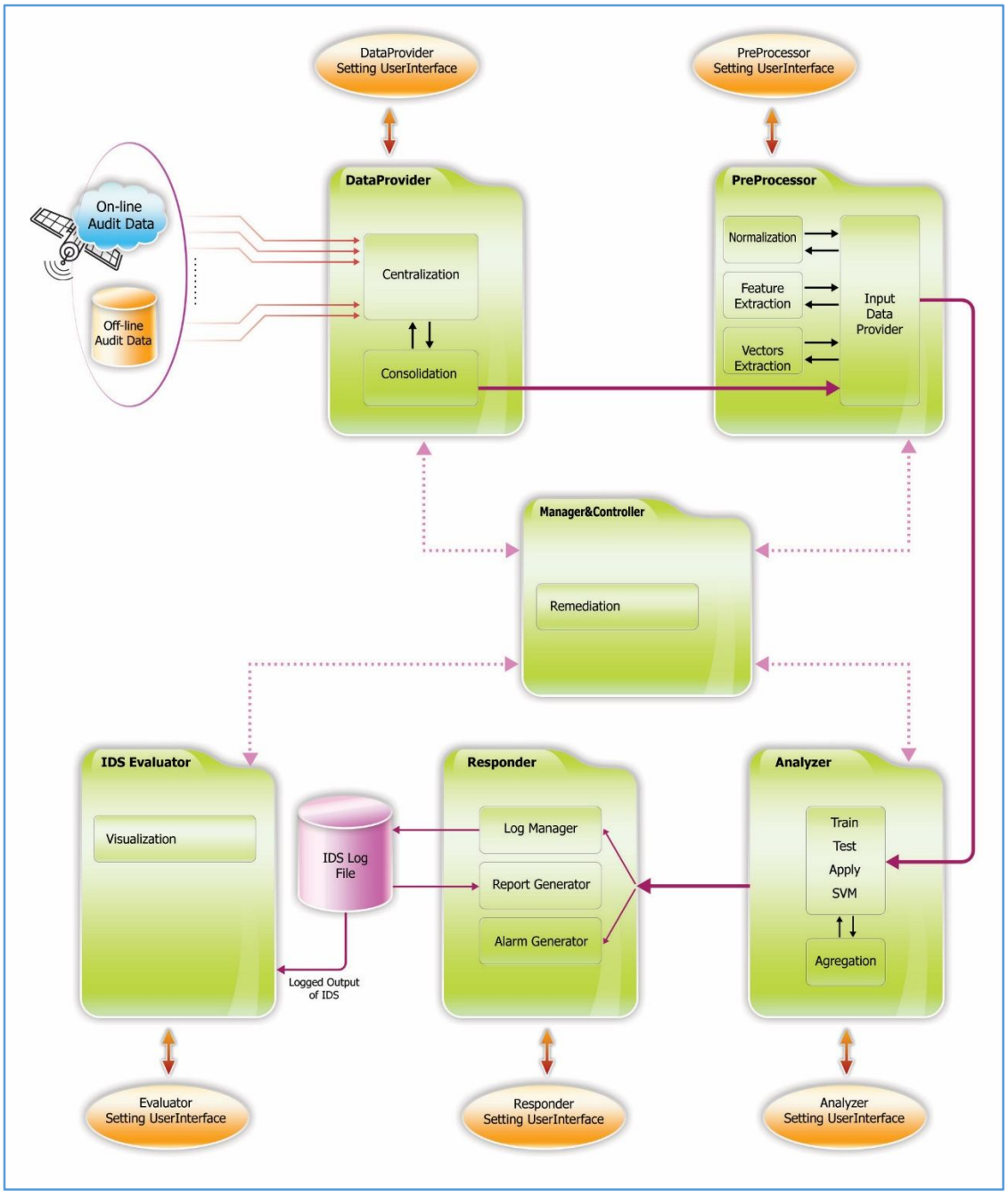


Figure 3: Architecture and data flow diagram of the proposed system

The data provider component is responsible for collecting the data required by the system and sending them to the preprocessor unit. This component is associated with the event file, receives the required system data from this file, and delivers it to the preprocessor component. Having received data from the data provider component, the preprocessor component extracts the required properties. Special features are extracted for each type of attack. These properties are numeric, so the database data is converted to records with numeric fields. The analyst component, which is the most important component of the system, has two functions: training and detection. In the training phase, the analyst

receives the generated records from the preprocessor component and sends them to the SVMs for training. In the detection phase, when new data is resubmitted to the analyst, it determines whether the data was previously normal or abnormal by using trained SVMs. The responder generates the appropriate response based on the output received from the analyzer. The response is done passively, that is, if an attack is detected, it only notifies the system security manager of the attack by generating various types of warnings, and leaves other actions, such as tracking down the attacker or taking action against it, to the system security manager. It also stores information about the attack in a file called the Attack Log. The system security administrator to track an attacker can use this file. The System Assessor component evaluates the performance of the intrusion detection system based on the parameters defined by the system security manager and by accessing the log files and therefore reports to the system security manager on the performance of the system. The system security manager gives the evaluation command to this component at different time intervals. The interface component of the system security manager is under the control of the system security manager and provides a communication interface between the system security manager and other units. Through this, the system security manager can send control messages to each component to start or end the component, for example, the system evaluator can order system performance. It can also change the way each component works, for example, it can determine the type of features that are extracted in the preprocessor, or it can instruct the data provider to order specific data from the database. The system security manager can also change the type of SVMs used in the analyzer unit by changing its kernel, or react appropriately to the attack by receiving a warning from the responding unit. In general, two activities including the possibility of managing all components, and setting work policies per unit may be taken as the main tasks of the system security manager.

## 6. Implementation details

The features and support vectors extracted in the preprocessing stage are obtained from the values of the parameters sent to each node. The input dataset having a large number of features is changed. Specific properties are extracted per SQL and XSS attacks, as follows.

### 6-1. Extracting features

**The average length of full HTTP request parameter values**: The average length of full HTTP request parameter values is one of the main extracted features. The request length module is computed based on the certain length of the URL to analyze each record as normal or attacked. Regular pattern analysis is emphasized on the content of URLs and other features to analyze certain attack patterns. In XSS and SQL injection attacks, due to the frequent use of special characters, the length of the request parameter values increases, while in normal requests, the length of the request parameter values does not exceed a certain value. Therefore, the mean length of the request parameter values can be taken as an important feature to distinguish between normal and intrusive requests. This feature is calculated equally for XSS and SQL injection attacks.

**Mean character distribution of full HTTP request parameters:** The mean character distribution characteristic of the request parameters is calculated equally for XSS and SQL injection attacks. To characterize the average character distribution of the request parameters from 33 to 96 characters and from 123 to 126 of the ASCI table, i.e a total of 68 characters have been used. The selected character sets are printable characters that are often used in written text. Characters 97 to 122 of the letters a to z are lowercase, and because these letters are uppercase from the number 65 to 90, in the said set, they have been removed to reduce the dimension. With this condition, if the characters are viewed in small, their large equivalent is considered. For each character, the percentage of presence of that character is calculated relative to the length of the parameter value. Then for each character, the average of this percentage of attendance in all the parameters of a request is calculated and added to the support vector as a field.

**Mean Distribution Equivalent to the Basis of Hexadecimal Specific Characters:** Sometimes intruders use the equivalent of the Basis of Hexadecimal Specific Characters instead. This usually happens with characters that are used to intrude. Although using the Basis of Hexadecimal Specific Characters alone does not mean intrusion, it usually indicates some kind of anomaly. This anomaly means that because a normal user typically uses normal characters, using the Basis of Hexadecimal Specific Characters can indicate some kind of abnormal behavior. To detect such an anomaly, the equivalent of the Basis of Hexadecimal Specific Characters, which are mainly used in these two attacks, is also considered. The use of the Basis of Hexadecimal Specific Characters is not common for other characters. Table 2 characters are used by SQL injection attacks for intrusion, the equivalent of which is written as the Basis of Hexadecimal Specific Characters.

Table 2: The base equivalent of the Basis of Hexadecimal Specific Characters for a SQL injection attack

| **Character** | **--** | **=** | **%** | **;** | **!** | **.** | **(** | **)** | **'** |
|---|---|---|---|---|---|---|---|---|---|
| **Basis of Hexadecimal** | 2D2D | 3D | 25 | 3b | 21 | 2e | 28 | 29 | 27 |

Similarly, in XSS attacks, the characters in Table 3 are widely used, with the equivalent of the Basis of Hexadecimal Specific Characters written on them.

Table 3: The base equivalent of the Basis of Hexadecimal Specific Characters for XSS attack

| **Character** | **%** | **&** | **<** | **>** | **(** | **)** |
|---|---|---|---|---|---|---|
| **Basis of Hexadecimal** | 25 | 26 | 3c | 3e | 28 | 29 |

The two characters <and> that are used both as the character itself and as the Basis of Hexadecimal Specific Characters are used as the <script> keyword along with the script keyword, and this is the

reason why they are used so much as both the character and the Basis of Hexadecimal Specific Characters. The base rate equivalent to the Basis of Hexadecimal Specific Characters of the above tables is first calculated for each parameter value and then averaged from these values.

**The presence of keywords in the request:** The main problem in the SQL injection flaws is the use of SQL queries via the web to receive or modify sensitive and confidential information stored in the database. These queries can be both in the URL and in the body of a request. Also for XSS attacks, a malicious script can be placed in the URL or body of the request, while a normal request does not contain any scripts. To prevent these two attacks, the user should not be allowed to use SQL and script commands in the URL or the body. The point to be noted here is that this feature is considered in the whole request and is not calculated separately in each parameter value. This is because such words are usually not seen in one or more of the parameter values, and in practice, averaging them reduces the likelihood of a keyword is present. Therefore, the presence of keywords for the completely requested request is calculated. Thus, the most important keywords used in the SQL injection flaws are as follows.

Keywords_SQLInjection= {Select  Insert Update Delete Execute Where AND OR Having}

One bit is taken per keyword so that in the presence of that keyword, the number 1, and in the absence of it, the number 0 is given to the bit value. Putting these bits together in base 2 gives a number. This number is added as the feature value of the presence of keywords to the support vector. For example, if the words ‘select’, ‘from’, ‘where’, and ‘having’ are used in a SQL injection flaw, the field value is that of in Table 4.

Table 4: An example of assigning bits to the presence of keywords in a request

| Select | Insert | Update | Delete | Execute | From | Where | AND | OR | Having |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |

The obtained value is $[1000011001]_2$, which is converted to the decimal value of 537. This number is added to the support vector as a keyword presence feature. Obviously, by shifting the order of the keywords, different numbers are obtained. The order of these words is shown in Table 3. The reason for this choice is the approximate order of use of these words in SQL queries. As in a SQL query, first words like select, update, etc. are inserted, then words like where and finally words like having are entered. This procedure is followed to obtain a logical binary value by assigning bits to each word.

The following set of keywords is used in XSS attacks:

Keywords_XSS={script window document  command location write exe onload onerror}

XSS attacks typically use JavaScript. These are the words used for coding in JavaScript. The order of these keywords is based on the most likely occurrence of that keyword in XSS attacks and on the

approximate order of coding in JavaScript. Similar to the method described for SQL injection flaws, one bit is considered for each word in this set. If that word appears, the bit value is 1; otherwise, the bit value is 0. Then this binary value is converted to a decimal number. The decimal number obtained from this conversion is added to the support vector as the value of the presence of keywords.

**Character Binary Arrangement**: The Character Arrangement property means that the order of the characters can be added to the support vector as a numeric value. This property is calculated for the values of the parameters of a request so that the binary combinations of the characters in the value of a parameter are considered. For example, if you consider the guess value of the password parameter, the combinations of "gu, ue, es, ss, sm, and me" must be considered, which means that, for example, in the value of a parameter, the character of g is seen before u. To obtain such an arrangement, each binary combination of characters in the value of a bit parameter or flag can be assigned. If a specific character sequence appears, the bit value is 1 and, otherwise 0. As mentioned earlier, since the parameter value of 68 characters is used in the character distribution, for binary sequences, since the duplication of two characters in a row is eliminated, 68 x 68, i.e. 4556 modes and bits, is possible. To implement 4556 modes, considering that each integer has 32 bits, it is necessary to add 143 digits to support vectors. Adding 143 digits to the support vectors creates a lot of computational overhead and is practically impossible. To solve this problem, the number of occurrences of each character order in all requests is calculated, and if that order never appears, it is deleted. Then, for the sequences that have appeared at least once, the sequences that have less than the average number of occurrences of all the sequences are also removed. Thus, the number of cases is reduced to approximately one-third, i.e. 1600 cases. To implement these 1600 modes, considering that each integer requires 32 bits, 50 32-bit digits have been used. Therefore, 50 32-bit digits are added to the support vector so that each bit represents a dual sequence of characters.

**Parameter sequencing**: Parameter sequencing is usually maintained in legal invocations of server-side applications, even when some parameters are deleted, while this sequence is not necessarily present in XSS and SQL injection flaws. The parameter sequence property is added to the support vector for the same reason. Thus, two integers of 32 bits, i.e. a total of 64 bits, have been used. For each binary order that appears from the parameter values, one bit is considered; if it appears, the binary order is bit value 1; otherwise, the bit value is 0. The total number of modes is always one of the total number of parameters and the selection of 64 bits is done for the same purpose. Therefore, for each request, two numbers are added to the support vector, in which the bit indicates the appearance or non-appearance of a sequence of the request parameters.

### 6-2. Extract the support vector

Given the above, Tables 5 and 6 depict the extracted support vector for XSS and SQL injection flaws.

Table 5: The extracted support vector for SQL injection flaws

| Features Extracted | Average Length of full HTTP Request Parameter Values | Mean Character Distribution of Request Parameters | Mean Distribution Equivalent to the Basis of Hexadecimal Specific Characters | The Presence of Keywords in the Request | Character Binary Arrangement | Parameter Sequencing |
|---|---|---|---|---|---|---|
| **Number of Fields** | 1 | 68 | 10 | 1 | 50 | 2 |

Table 6: The extracted support vector for XSS attacks

| Features Extracted | Average Length of full HTTP Request Parameter Values | Mean Character Distribution of Request Parameters | Mean Distribution Equivalent to the Basis of Hexadecimal Specific Characters | The Presence of Keywords in the Request | Character Binary Arrangement | Parameter Sequencing |
|---|---|---|---|---|---|---|
| **Number of Fields** | 1 | 68 | 6 | 1 | 50 | 2 |

According to the above tables, the support vector for XSS and SQL injection flaws are 132 and 128 fields, respectively.

### 6-3. Implementation Technical Considerations

The use of datasets that are comprehensive, accurate, and traffic-specific is a key issue in training and testing an intrusion detection system. The best option is to use data collected from web servers that are to be protected. However, these data are usually not available to other researchers for security reasons to compare the results of different algorithms. This has led us to use both open but less real data as off-line audit data, and closed but pseudo-real datasets as online audit data. The recent well-known open dataset used for intrusion detection is the Realistic Cyber Defense (CSE-CIC-IDS2018) dataset. Since the publication of this dataset, a relatively large number of papers have been published. This dataset is the result of a collaborative project between the Communications Security Establishment (CSE) and The Canadian Institute for Cyber Security (CIC) that use the notion of profiles to generate cybersecurity datasets systematically. It includes a detailed description of intrusions along with abstract distribution models for applications, protocols, or lower-level network entities. The dataset includes several different attack scenarios, namely Brute-force, Heartbleed, Botnet, DoS, DDoS, XSS, SQL Injection, Portscan, Web attacks, and infiltration of the network from inside. The data is organized in seven CSV files, where each row is a sample, labeled as benign or with the name of the corresponding attack. The attacking infrastructure includes 50 machines and the victim organization has 5 departments including 420 PCs and 30 servers. This dataset includes the

network traffic and logs files of each machine from the victim side [39], along with 80 network traffic features extracted from captured traffic using CICFlowMeter-V3 [40]. The data is organized in seven CSV files, where each row is a sample, labeled as benign or with the name of the corresponding attack [41]. We run analytics queries using Spark SQL API [42] and cleaned the data using the Python script provided by [43]. That is, we dropped the samples with missing feature values and removed the columns with no values. Table 7 shows the file distribution of the data before and after the cleaning process.

Table 7: File distribution of the CSE-CIC-IDS2018 dataset

| Name of File | Sample Type | Number of Samples Before Cleaning | Number of Samples After Cleaning |
|---|---|---|---|
| **02-14-2018.csv** | Benign<br>FTP-BruteForce<br>SSH-Bruteforce | 1,735,479<br>193,360<br>187,589 | 663,808<br>123,688<br>163,124 |
| **02-15-2018.csv** | Benign<br>DoS-GoldenEye<br>DoS-Slowloris<br>DDOS-HOIC | 2,583,187<br>41,508<br>10,990<br>686,012 | 988,050<br>41,499<br>10,497<br>684,287 |
| **02-16-2018.csv** | Benign<br>DoS-SlowHTTPTest<br>DoS-Hulk<br>DDOS-LOIC-UDP<br>DDOS-LOIC-HTTP | 1,168,054<br>139,890<br>461,912<br>1730<br>576,191 | 446,772<br>71,889<br>456,913<br>1730<br>576,191 |
| **02-22-2018.csv** | Benign<br>BruteForce-Web<br>BruteForce-XSS | 2,725,812<br>249<br>79 | 1,042,603<br>246<br>79 |
| **02-23-2018.csv** | Benign<br>BruteForce-Web<br>BruteForce-XSS<br>SQL-Injection | 2,725,523<br>362<br>151<br>87 | 1,042,301<br>347<br>151<br>72 |
| **03-01-2018.csv** | Benign<br>Infiltration | 616,425<br>161,934 | 235,778<br>92,403 |
| **03-02-2018.csv** | Benign<br>Bot | 1,982,611<br>286,191 | 758,334<br>285,016 |
| **Binay Class** | Benign<br>Attack | 14,097,779<br>2,748,235 | 5,177,655<br>2,508,132 |
| **Total** | - | **16,846,014** | **7,685,787** |

[44] presents a comprehensive survey and analysis of other machine learning intrusion detection models based on CSE□CIC□IDS2018 Big Data. It should be noted that recently other datasets like CIRA-CIC-DoHBrw-2020 and CIC-Bell-DNS2021 have been produced till now [45], but since they did not contain injection flaw attacks, we could not use them in our experiments.

To apply real data, moreover the use of XSS and SQL Injection attacks in the CSE-CIC-IDS2018 dataset as offline data, we tried to use the tools and methods that are described below, as online data, to create a set of data that has the three principles of comprehensiveness, accuracy, and traffic, so WebInspecct software has been used. All of them correspond to all types discussed in Section 4, and the same types of requests are generated for both attack types. For this purpose, we apply WebInspect

which is the most accurate and comprehensive way to diagnose vulnerabilities in applications and web servers. Using WebInspect, administrators and users can easily and quickly scan their applications and web servers for vulnerabilities.

This software determines the vulnerability by multiple attacks on the designated target and reports it if the vulnerability is detected. The main features of this software are as follows: The ability to perform scanning and auditing processing separately and simultaneously organized reporting, manually attack control, provide system status summary, ability to change and correct navigation policies, traffic view screen, and the ability to select different Web attacks.

To provide attack data, we target an application with special properties as the target of the attack and attack it using the powerful WebInspect attack tool. Files registered on the server represent records that can be reliably labeled. We first turn our system into a web server using XAMMP. XAMMP is a software package that installs MySQL on a computer, Apache server, and database. Then we use a web application called Evilboard as the target to attack. In Evilboard, users can send messages and talk to each other. In summary, we choose it as the target of the attack for the following:

- The ability of users to connect as a community, and as we said before, such programs are good places for XSS and SQL injection flaws.
- A large number of connections of this program, which as a result provides us with more scalability by WebInspect.

## 7. Implementation, system evaluation, and results analysis

### 7-1. Implementation

As we mentioned, the proposed system architecture applied both online audit data and offline audit data. To implement the proposed system, we use the CSE-CIC-IDS2018 dataset for offline audit and install Evilboard on the node that has become the webserver for the online dataset. In the following, we introduce Evilboard in WebInspect as the attack target. To create a suitable online dataset, we first scan all Evilboard connections using the WebInspect scan mode, then delete the log file and use the audit mode to send an attack to Evilboard, specifying the type of attack. In this case, we can be sure that the obtained log file only contains records with the attack tag. This is done separately for XSS and SQL injection flaws. A normal network is used to provide online normal data, which is installed on the server of the Evilboard node. The users of this local network are selected as connected clients through this program. The traffic generated on this network was collected for one week and used as normal data. Since this network was not connected to any external network such as the Internet, etc., we can be sure that there is no attack data in it. Table 8 shows the amount of data attacks and normal datasets obtained. Of both online audit data and offline audit data, ten percent are used as test data, and the rest are used as training data.

Table 8: Produced data

| Normal data | The amount of SQL injection flaws in data | The amount of XSS attack data |
|---|---|---|
| 756347 | 639428 | 650894 |

After preparing the online dataset, the preprocessing step is performed using Matlab software on both normal data and attack data on both online and offline datasets, and the features required to detect XSS and SQL injection flaws are extracted so that at each request a feature vector is specified. These support vectors are given to the analyzer component. In the analyzer component, the data obtained from the previous step is given to SVMLight for training. SVMLight is an implementation of the C-language SVM. This software package can be run on UNIX as well as other environments, including Windows. To use this software package, you need to create executable files, format the training, and test data in a way that can be used by the software. Using the LightDataAgent program, you can easily convert training data to the required SVMLight format. One of the most important features of SVMLight is the ability to select different kernels for the SVMs.

**7-2. Evaluation**

As we say in the introduction, determining the appropriate kernels and the right value of their parameters is one of the open issues in support vector machines. Thus, in this section, the experiments are done in two stages. The first step is to find the appropriate kernel function and the second step is to determine the right value of parameters. In the first step, accuracy, precision, and recall defined by the Confusion Matrix were used to evaluate the proposed system. Using both offline and online datasets are explained before, the results obtained for the different kernels of the SVMs are shown separately in Tables 9 and 10.

Table 9: Results of SQL injection flaws

| Kernel Function | Accuracy | Recall | Precision |
|---|---|---|---|
| **Linear** $K(x,y)=ax+by$ | 98.37 | 93.95 | 98.31 |
| **Polynomial** $k(x,y)=(x*y+1)^d$ | 98.92 | 96.82 | 99.00 |
| **RadialBasisFuncion** $K(x,y)=\exp(-1/2\sigma^2\|\|x-y\|\|^2)$ | 99.72 | 99.08 | 99.78 |
| **Gaussian** $k(x,y)=\exp(-(x-y)^2/\delta^2)$ | 99.26 | 97.99 | 99.40 |

Table 10: Results of XSS attacks

| Kernel Function | Accuracy | Recall | Precision |
|---|---|---|---|
| **Linear** $K(x,y)=ax+by$ | **98.00** | **93.70** | **98.08** |
| **Polynomial** $k(x,y)=(x*y+1)^d$ | **98.81** | **96.51** | **99.02** |
| **RadialBasisFuncion** $K(x,y)=\exp(-1/2\sigma^2\|\|x-y\|\|^2)$ | **99.12** | **97.75** | **99.21** |
| **Gaussian** $k(x,y)=\exp(-(x-y)^2/\delta^2)$ | **99.02** | **96.80** | **99.10** |

According to Tables 8 and 9, the best results for accuracy, precision, and recall for both XSS and SQL injection flaws are for the RBF kernel. The reason was hindered by the RBF's ability to classify non-linear behavior as well as the non-linear label. Both of these were solved by the introduction of the RBF kernel and the incorporation of soft margins. The use of radial SVMs results in obtaining better results from the classification process when compared to normal linear SVMs. In linear SVMs, the classification is made by the use of linear hyperplanes. Trying to attain a hyper-plane reduces the distance from the members of each class to the voluntary hyper-plane. But the use of linear SVMs has the disadvantages of getting a less accurate result, getting overfitting results, and being robust to noise. These shortcomings are effectively suppressed by the use of the radial SVMs where non-linear kernel functions are used and the resulting margin hyper-plane fits in a transformed feature space. The corresponding feature space is a Hilbert space of infinite dimensions when the kernel used is a Gaussian radial basis function. After the RBF kernel, the best results are for Gaussian. The Gaussian kernel in particular guarantees the existence of such a decision boundary. By observing that all the kernel entries are non-negative, it can be concluded that all the data in the kernel space lies in the same quadrant. This makes the Gaussian kernel well-suited to deal with any arbitrary dataset. The overfitting of kernel functions also appeared in two other of the four experimented functions. However, our experiment with RBF and Gaussian kernel showed not only similar detection performance as the soft margin SVMs, but also showed consistently higher accuracy, precision, and recall rates than that of the others.

In addition, a comparison of these two tables shows that the results obtained for the SQL injection flaws have better values than the XSS attack. In general, because of their nature, SQL injection flaws have more predictable behavior than XSS attacks. SQL injection flaws do not waste system resources as other attacks do. Vs. XSS attacks occur when an attacker uses a web application to send or execute malicious code on a user's computer. Therefore, due to the limited resources in fog computing, the detection of SQL injection flaws is more accurate. The details of Tables 8 and 9 are shown in the following figures, for a closer look.

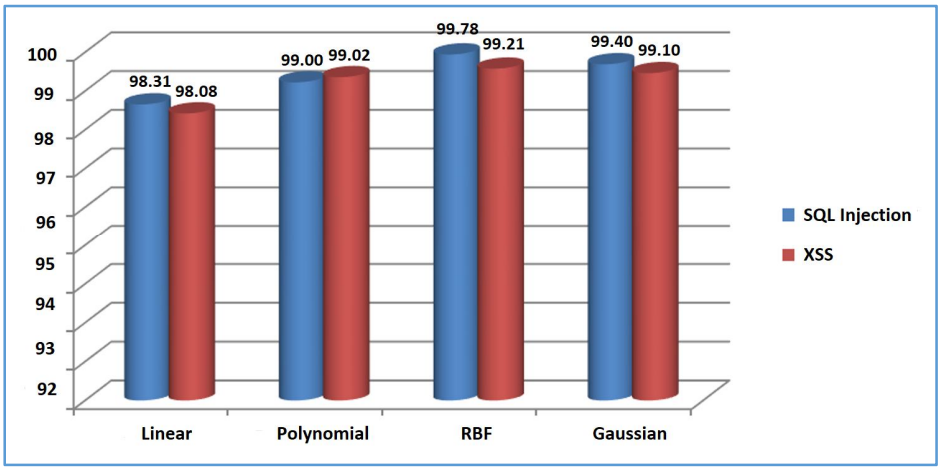


Figure 4: Comparison of precision criteria obtained for XSS and SQL injection flaws

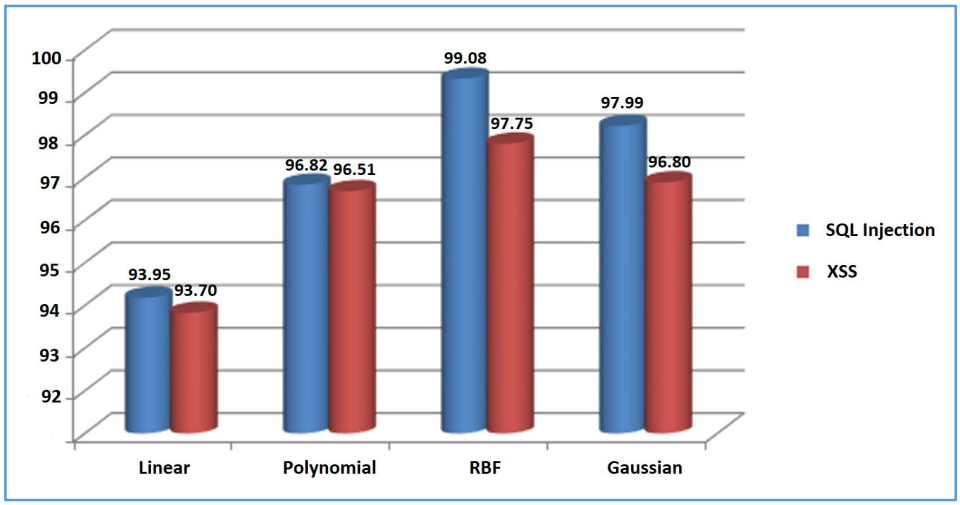


Figure 5: Comparison of recall criteria obtained for XSS and SQL injection flaws

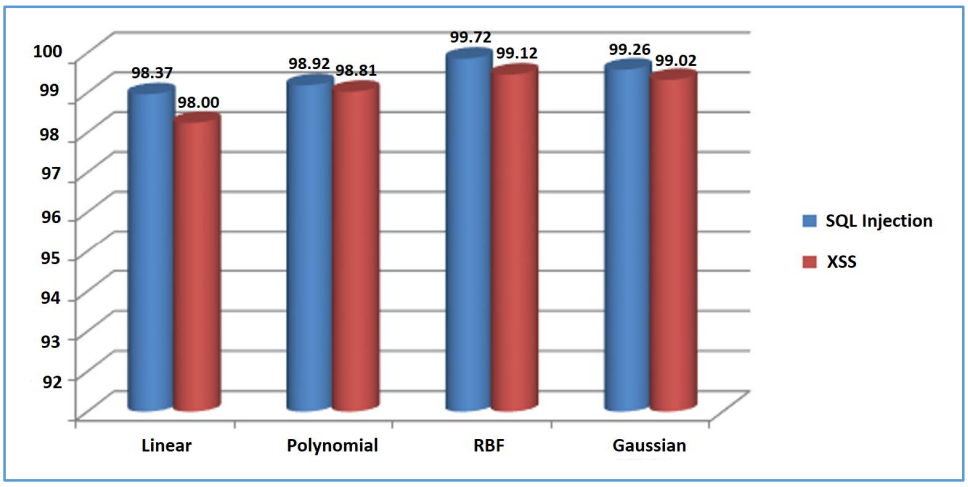


Figure 6: Comparison of accuracy criteria obtained for XSS and SQL injection flaws

Of course, it should be noted that the results were achieved in 2 for the gamma parameter and 0.5 for the bandwidth parameter. In the second step of our experiments, we performed to obtain the appropriate value of the parameters. For more details, see Tables 11 to 14. Experiments with different gamma parameter values have also been performed for the RBF kernel. The obtained results are shown in Tables 11 and 12.

Table 11: Results obtained with different gamma parameter values for SQL injection flaws

| Value of The Gamma Parameter in The RBF Kernel | Accuracy | Recall | Precision |
|---|---|---|---|
| 0.0001 | 91.58 | 91.96 | 90.87 |
| 0.001 | 92.66 | 93.71 | 92.46 |
| 0.01 | 93.72 | 95.80 | 94.68 |
| 0.1 | 96.68 | 96.77 | 96.89 |
| 0.2 | 96.86 | 96.96 | 96.96 |
| 0.3 | 96.94 | 97.91 | 97.37 |
| 0.4 | 97.49 | 98.56 | 98.79 |
| 0.5 | 99.72 | 99.08 | 99.78 |
| 0.6 | 99.01 | 98.81 | 99.65 |
| 0.7 | 98.50 | 98.35 | 99.47 |

Table 12: Results obtained with different gamma parameter values for XSS attacks

| Value of The Gamma Parameter in The RBF Kernel | Accuracy | Recall | Precision |
|---|---|---|---|
| 0.0001 | 92.37 | 91.56 | 93.97 |
| 0.001 | 94.46 | 93.68 | 94.76 |
| 0.01 | 95.72 | 94.85 | 94.91 |
| 0.1 | 96.48 | 95.37 | 96.96 |
| 0.2 | 96.97 | 95.71 | 97.37 |
| 0.3 | 97.52 | 95.98 | 97.99 |
| 0.4 | 98.71 | 96.46 | 98.27 |
| 0.5 | 99.12 | 97.75 | 99.21 |
| 0.6 | 98.94 | 97.51 | 98.80 |
| 0.7 | 98.73 | 97.20 | 98.67 |

A comparison of Tables 11 and 12 shows that the best results are related to the value of 0.5 for the gamma parameter for both SQL injection flaws and XSS attacks. The result is improved by increasing the amount of gamma until 0.5 and decreasing after that.

For the Gaussian kernel, experiments with different values of the bandwidth parameter in the Gaussian kernel have been performed as shown in Tables 13 and 14.

Table 13: Results with different bandwidth parameter values for SQL injection flaws

| Values of The Bandwidth Parameter in The Gaussian Kernel | Accuracy | Recall | Precision |
|---|---|---|---|
| 0.0001 | 85.40 | 89.43 | 87.71 |
| 0.001 | 89.56 | 90.67 | 89.96 |
| 0.01 | 90.45 | 92.48 | 90.95 |
| 0.1 | 91.67 | 93.71 | 92.87 |
| 0.5 | 91.78 | 93.86 | 92.95 |
| 1 | 91.99 | 93.93 | 92.98 |
| 1.3 | 91.97 | 93.94 | 92.96 |
| 1.5 | 91.37 | 94.26 | 93.37 |
| 1.7 | 92.56 | 94.51 | 93.56 |
| 2 | 99.26 | 97.99 | 99.40 |
| 2.3 | 99.01 | 97.85 | 99.15 |
| 2.5 | 98.41 | 97.60 | 98.82 |

Table 14: Results with different bandwidth parameter values for XSS attacks

| Values of The Bandwidth Parameter in The Gaussian Kernel | Accuracy | Recall | Precision |
|---|---|---|---|
| 0.0001 | 87.37 | 91.97 | 85.57 |
| 0.001 | 89.46 | 90.86 | 87.71 |
| 0.01 | 89.59 | 91.94 | 88.91 |
| 0.1 | 90.75 | 92.74 | 90.89 |
| 0.5 | 90.89 | 92.97 | 91.98 |
| 1 | 90.94 | 92.98 | 90.96 |
| 1.3 | 93.98 | 93.27 | 91.37 |
| 1.5 | 91.49 | 93.48 | 91.41 |
| 1.7 | 91.67 | 93.66 | 91.56 |
| 2 | 99.02 | 96.80 | 99.10 |
| 2.3 | 98.70 | 96.22 | 98.73 |
| 2.5 | 98.33 | 95.67 | 98.13 |

A comparison of Tables 13 and 14 shows that the best results are related to the value of 2 for the bandwidth parameter for both SQL injection flaws and XSS attacks. The result is improved by increasing the amount of gamma until 2 and decreasing after that. Besides, the amount of time required to perform the experiments was almost the same and no significant difference was observed.

Then we addressed different types of SVMs discussed earlier to better evaluate the proposed method. Proper judgment requires that all methods use the same dataset and be used to detect the same attack categories. In this regard, we went through a difficult process, the details of which are summarized in the following. It should be noted that our proposed method has also been performed with the RBF kernel.

All methods were implemented based on the algorithm presented in their references and tested with the same dataset that contained a combination of CSE-CIC-IDS2018 and an online dataset. For this purpose, we tried to access most of the codes of these references. Some of them through contact with the original authors, and some of them through the code that was made available. Virtualization was also used to perform experiments in cloud computing and fog computing environment. With this approach, we tried to observe the three principles of comprehensiveness, accuracy, and having the necessary traffic. In addition, a new dataset obtained was used to compare the 11 algorithms used in our experiment. First of all, it should be noted that all the methods discussed have performed their experiments on their dataset, but since we have implemented these methods on our dataset, it is very logical that our detection rate is slightly different from the detection rate reported by the respected authors of these articles. The mean accuracy of the proposed method in comparison with the mean accuracy of 11 other methods is presented in Figure 7, which is:

1- A support vector machine based on entropy and tf-idf in 2008 [14],
2- Combination of support vector machines and neural networks in 2013 [15],
3- Robust support vector machines in 2013 [9],
4- Combination of support vector machines and binary particle swarm optimization and standard particle swarm optimization in cloud computing in 2019 [21],
5- Density-based binary support vector machines in cloud computing 2015 [16],
6- Combination of support vector machines with fuzzy clustering in cloud computing in 2020 [20],
7- One-Class support vector machine in cloud computing in 2020 [10],
8- Combination of support vector machines with information gain in cloud computing in 2020 [19],
9- Combination of support vector machines with deep learning in cloud computing in 2020 [18],
10- Combination of support vector machines, principal component analysis, and particle swarm optimization in fog computing in 2022 [22],
11- Combination of Software Defined Networking (SDN) with the SVMs and Selective Logging for IP Traceback in 2021 [23],
12- Proposed method.

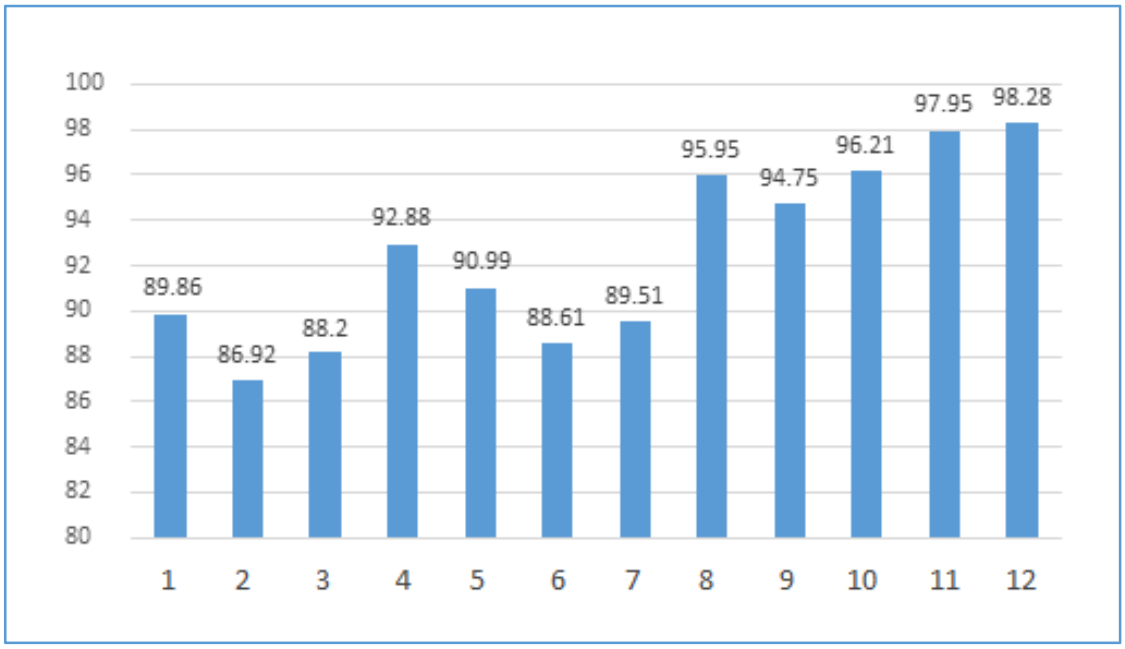


Figure 7: The Chart of mean accuracy obtained from the proposed method in comparison with the mean accuracy obtained from other methods based on the types of support vector machines

According to Figure 7, the proposed method provides a plausible result compared to other methods. The results indicate good performance of the proposed system; this is for the proper extraction of features and support vectors, with the proposed efficient architecture. In addition, the use of appropriate kernels and proportional values obtained from the results of experiments performed for the values of dependent parameters is another reason for the high efficiency of the proposed method. Due to the use of a real database and the same conditions for all methods, it can be explicitly said that the obtained results are close to reality and an accurate evaluation has been done. For more details, the following reasons are stated:

- According to Figure 7, the rate of 89.86 was obtained for the accuracy criterion of reference [14]. Although this method was introduced in 2008, the result shows good performance for this method. The relatively low detection rate of this method can be attributed to its incompatibility with recent computing. In this regard, we first extracted the entropy and TF-IDF (term frequency and inverse documents frequency) from processes. Next, entropy and TF-IDF features are sent to the SVMs model for learning and testing. Finally, using a voting schema named Weighted Voting SVM (WV-SVM) to determine whether a process is an intrusion.
- Compared with [15] made us a long way. This reference was working with KDD Cup 99 dataset and we had to implement feature extraction patterns on the CSE-CIC-IDS2018 and online dataset. Of course, it should be noted that this method used an RBF kernel, which made us a bit easier. However, after the experiments, the result was an 86.92 rate for the

accuracy criterion, which was not a good result. The reason was hindered by getting caught in a local optimum. Moreover, using a neural network takes more time for training. This method has five major steps in which, the first step is to perform the relevance analysis, and then input data is clustered using Fuzzy C-means clustering. After that, neuro-fuzzy is trained, such that each of the data points is trained with the corresponding neuro-fuzzy classifier associated with the cluster. Subsequently, a vector for SVMs classification is formed and in the last step, classification using RBF-SVM is performed to detect whether an intrusion has happened or not.

- [9] states that only normal data is required for training before anomalies can be detected. The key idea is that outliers should contribute less to the decision boundary than normal instances. It causes the decision boundary to be shifted towards the normal points. The rate of 88.2 is achieved in our experiment. Centralized to only anomaly detection is the reason for this low accuracy.

- Like [9], [21] focused on anomaly-based network intrusion detection systems called (NIDS). This is also the reason for its low detection rate in general compared to our method. But the ability to run in cloud computing and taking some consideration has made it performs better than [9] until we reached the rate of 92.88 for the accuracy parameter. This approach can monitor and analyze the network traffics flow that targets a cloud environment. The network administrator should be notified about the nature of this traffics to drop and block any intrusive network connections. SVMs are employed as the classifier of the network connections. The binary-based Particle Swarm Optimization is adopted for selecting the most relevant network features, while the standard-based Particle Swarm Optimization is adopted for tuning the SVMs control parameters. For an experiment, we changed the NSL-KDD dataset to the CSE-CIC-IDS2018, and the online dataset was used as the network data source. This approach is trained and tested on the benchmark of our dataset and the evaluation results stated its efficacy in recognizing normal behaviors and detecting the attacks.

- Reference [16] is one of the first works done to detect intrusions in cloud computing. However, 90.99 is a suitable detection rate. In this approach, a new intrusion detection method of binary SVMs with Hadoop is put forward, whose basic idea is, according to the density of the data set to set the priority classification, let the class of the easiest separate, and generate the training decision tree, to improve the accuracy of the classification model. This algorithm is modified to one based on MapReduce. It is the innovation of this approach. For an experiment, we used virtualization. Within each virtual machine, we have the same basic configuration software environment used in [16] such as Ubuntu 16.05 for Operating System, jdk1.7.0_05 for the JDK version, Weka package for SVMs tool, and Hadoop version 0.20.2. In addition, we used the CSE-CIC-IDS2018 and online dataset instead of the KDD Cup 99

dataset to get real and comparable results. Although the ability to detect known attacks is the advantage of this method, but also the inability to detect unknown attacks and anomaly detection is a shortcoming of this method.

- [20] proposes an intrusion detection system that combines a fuzzy c-means clustering (FCM) algorithm with SVMs to improve the accuracy of the detection system in the cloud computing environment. As before, performed experiments, we changed the dataset. Even though this method has been implemented in cloud computing, centralized just on DDoS attacks caused its detection rate to be below 88.61 in our experiment.
- [10] presents a network intrusion detection approach that trains on normal network traffic data and searches for anomalous behaviors that deviate from the normal model. Its proposed approach applies a one-class support vector machines (OCSVM) algorithm to detect anomalous activities in the network traffic. The basic author's idea was to use an appropriate kernel function to map the input data to a high-dimensional feature space. By doing this, it was possible to create a decision function that best separates one-class samples from second-class samples with the maximum margin. To perform the experiments, in addition to using the CSE-CIC-IDS2018 dataset as an off-line audit, for online audit we implemented the Modern Honey Network (MHN), which is a centralized server to manage and collect data from honeypots. MHN has an easy to use Web interface that helps in quickly deploying the sensors and immediately collecting viewable data. We used Google Cloud to create instances of Ubuntu 16.05 LTS servers, where we had one MHN server, and sensor servers. Using this architecture like [10], we were able to collect a large amount of data through the sensors. Next, we used Azure Machine Learning (AML), which is a cloud-based environment from Microsoft to preprocess data, train and test, deploy, manage, and track machine learning models. The AML evaluation module shows that there is no big variance in the results, and the average accuracy of the proposed anomaly detection model was 89.51. By default, a radial basis kernel is used like [10]. The relatively low detection rate can also be attributed to the time-consuming and complex decision-making power of this method in real-time network traffic. The inability to deal with online data is an issue that the authors have pointed out, which we also found in our experiments.
- In the [19] method, the SVM classifier is adopted to binary classify network data in either normal or attack behaviors, and due to the irrelevant and redundant features found in the dataset, information gain (IG) is used to select the relevant features and remove unnecessary features. IG is a method used to decide which feature in a given dataset is most important to be used in the machine learning process for classifying data. The IG uses Shannon's entropy to measure the feature set quality. Like other experiments, we replaced the dataset of this article (KDD Cup 99 and NSL-KDD) with our dataset. In this method, the authors have used 10-field Cross Val to tackle the overfitting problem, which divides the dataset into 10 subsets

of size N/10 (N is the size number of the dataset) and uses 9 sub-sets for training and 1 remaining sub-set for testing. The [19] method was implemented in MATLAB and the Weka data mining tool. The SVM classifier is applied with the LibSVM package in MATLAB and Radial Bias Kernel Function is used. In this regard, we achieved rather a high rate of accuracy detection equal to 95.95.

- [18] presented an effective stacked contractive autoencoder (SCAE) method for unsupervised feature extraction. By using the SCAE method, better and robust low-dimensional features can be automatically learned from raw network traffic. The SCAE+SVM approach combines both deep and shallow learning techniques, and it fully exploits their advantages to significantly reduce the analytical overhead. Though authors have reported high accuracy in using their datasets as NSL-KDD and KDD Cup 99 datasets but implemented our dataset, and we achieved 94.75 for its accuracy criterion. This can happen with any other method, even our own.
- Due to the high dimensionality of network data, like [22] we first used Principal Component Analysis (PCA) to reduce the dimensionality of the data, eliminate the correlation between features and reduce the training time. Then, in the cloud server, a support vector machine optimized by the particle swarm algorithm is used to complete the training of the dataset, obtain the optimal SVMs intrusion-detection classifier, send it to the fog node, and carry out attack detection at the fog node. The experiments in this reference are based on KDD CUP 99 dataset, while we were also able to achieve acceptable results up to 96.21 by performing experiments with our dataset. In addition, virtualization is used for implementation. Centralized on PCA and PSO, and most importantly the ability to run in fog computing is one of the main reasons for the high efficiency of this method. Working in a foggy environment reduces the dependence on the dataset and at the same time can provide good results with a variety of datasets.
- The authors in [23] aim that it is not always correct to take punitive action against packets of a traffic flow, solely based on a detection of a possible threat that may result in blocking or dropping of genuine packets. However, IP traceback provides the ability to track the actual source of the packets in the eventuality of an attack. It was very interesting that although the authors of this article reported an accuracy of 95.98 in using the NSL-KDD dataset, we achieved an accuracy equal to 97.95 in our experiments using the CIC-IDS2018 dataset. This is due to the high power of this method in performing calculations without additional overhead and also saving system resources. The detection rate of this method was very close to the detection rate of our proposed method.

## 8. Discussion and conclusion

As we move from cloud computing to fog computing to reduce bandwidth consumption as well as network latency, securing all nodes becomes a serious issue. Besides, all nodes are at greater risk of attacking vulnerabilities due to their limited resources and their proximity to attackers. It can be said with certainty that implementing a method that can protect all nodes from the threat of attackers can establish security throughout the system. Fog security is cloud security. On the other hand, it is easier to protect the node because the fog has more limited resources than the cloud and does not have the complexities of controlling and establishing security in cloud computing. Reducing the complexity of the system architecture in fog is the key to the success of IoT applications. The current paper considers the use of the SVMs technique to protect all the nodes from attackers. The motivation is the ease of working with high-dimensional data, the design of the most generalized classifier, the achievement of the optimal cost function, the automatic determination of the optimal structure, and the topography for the classifier. According to the documentation provided in the authoritative articles, the SVM is one of the most widely used and efficient machine learning algorithms used in recent computer security issues.

In the following, the architecture of the proposed intrusion detection system, which consisted of the components of the data provider, processor, analyst, manager and controller, responder, and evaluator, was presented. The proposed system architecture applied both online audit data and offline audit data. To implement the proposed system, we use the CSE-CIC-IDS2018 dataset for the offline dataset and install Evilboard for an online dataset. The CSE-CIC-IDS2018 dataset is the most recent intrusion detection dataset that is big data, publicly available, and covers a wide range of attack types, especially XSS and SQL injection flaw attacks. They are the most destructive and harmful web attacks in recent global reports. CSE-CIC-IDS2018 contains more than 16 million instances. Since then, two databases with the names CIRA-CIC-DoHBrw-2020 and CIC-Bell-DNS2021 datasets have been produced, but since they did not contain injection flaw attacks, we could not use them in our experiments.

For this purpose, WebInspect software was used to prepare the data, targeting the principles of comprehensiveness, accuracy, and up-to-date traffic, targeting Evilboard. The most important task of the preprocessing component is to extract the appropriate features. The extracted features for XSS and SQL injection flaws included the mean length of the request parameter values, the mean character distribution of the request parameters, the mean distribution equivalent to the Basis of Hexadecimal Specific Characters, the presence of keywords, the characterization of the binary character sequence, and parameters sequencing. According to the extracted features, the support vector was obtained for XSS and SQL injection flaws. SVMLight was used to implement the analyst component. In which data is converted to readable format by SVMLight using LightDataAgent, and given to SVMLight to determine the appropriate category. In the respondent component, if the output

received from the analyst indicates an attack, the relevant traffic information is recorded along with the output from the analyzer, and an appropriate warning is generated.

Accuracy, precision, and recall defined by the Confusion Matrix were used to evaluate the proposed system. The results obtained for the different kernels of the SVMs were evaluated separately. The best results for accuracy, precision, and recall criteria for both XSS and SQL injection flaws were obtained on the RBF kernel. After the RBF kernel, the best results are for Gaussian, Polynomial, and Linear kernels, respectively. In addition, the results obtained for the SQL injection flaws detection have better values than the results obtained for the XSS attack detection. A comparison of the obtained results with different values of the gamma parameter in the RBF kernel showed that the best result is related to the value of 0.5 for this parameter. In addition, comparing the obtained results with different values of the bandwidth parameter in the Gaussian kernel showed that the best result is related to the value of two for this parameter. Finally, to further compare the proposed method, the mean accuracy obtained from the proposed method was discussed with the mean accuracy obtained from the use of various support vector machine-based methods defined in the literature review section. The results show the higher efficiency of the proposed method compared to other methods. Briefly, the reason is the efficiency of the proposed architecture, proper placement of detection nodes between the edges of the fog, the accurate selection of the kernel and its dependent parameters, the definition of suitable properties for feature extraction, and then the effective extraction of support vectors and their use in fog computing as we mentioned in detail. In particular, the proposed method has shown the most promising results. Creating a larger and more diverse dataset leads us to better and more accurate results. Therefore, using Autoencoder (AE) and Restricted Boltzmann Machine (RBM) as data generator is our plan for future work.

## Declarations

### Ethical Approval

All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and national research committee and with the 1964 Helsinki declaration and its later amendments or comparable ethical standards. This article does not contain any studies with animals performed by any of the authors.

### Consent

As per international standard or university standard, Participants' written consent has been collected and preserved by the author(s).

### Competing interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

### Authors' contributions

Seyed Omid Azarkasb and Seyed Hossein Khasteh. These authors contributed equally to this work.

**Authors and Affiliations**

K.N. Toosi University of Technology, Tehran, Iran.

Seyed Omid Azarkasb

K.N. Toosi University of Technology, Tehran, Iran.

Seyed Hossein Khasteh

**Corresponding author**

Correspondence to Seyed Omid Azarkasb

**Funding**

This research was not funded.

**Availability of data and materials**

A significant amount of data is addressed in this article. The remaining data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions. The authors declare that all the experimental data in this paper are true and valid. Moreover, The authors declare that all experimental data are obtained from detailed experiments.

## References

[1] An, X., X. Zhou, X. Lu, F. Lin, L. Yang, **"Sample Selected Extreme Learning Machine Based Intrusion Detection in Fog Computing and MEC"**, Wireless Communications and Mobile Computing, Wiley Hindawi, 2018.

[2] Aliyu, F., T. Sheltami, E. Shakshuki, **"A Detection and Prevention Technique for Man in the Middle Attack in Fog Computing"**, The 9th International Conference on Emerging Ubiquitous Systems and Pervasive Networks, Vol. 141, pp. 24-31, 2018.

[3] Vaquero, L.M**.,** L. Rodero-Merino, **"Finding Your Way in the Fog: Towards a Comprehensive Definition of Fog Computing"**, ACM SIGCOMM Computer Communication Review, Vol. 44, No. 5, pp. 27-32, 2014.

[4] Dsouza, C., G-J. Ahn, M. Taguinod, **"Policy-Driven Security Management for Fog Computing: Preliminary Framework and A Case Study"**, Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration, San Francisco, California, USA, pp. 16-23, 2014.

[5] Xiang, F., J. Ying, **"A Novel Learning Algorithm on Probability Measure for Intrusion Detection"**, International Conference on Computer Science and Electronic Technology, 2016.

[6] Androcec, D., N.Vrcek, **"Machine Learning for the Internet of Things Security: A Systematic Review"**, 13th International Conference on Software Technologies, Porto, Portugal, pp.563-570, 2018.

[7] Subba, B., S. Biswas**, "Enhancing Performance of Anomaly-based Intrusion Detection Systems through Dimensionality Reduction using Principal Component Analysis"**, IEEE International Conference on Advanced Networks and Telecommunications Systems, Bangalore, India, 2016.

[8] Shon, T., J. Moon, **"A Hybrid Machine Learning Approach to Network Anomaly Detection",** Information Sciences, Vol. 177, pp. 3799-3821, 2007.

[9] Amer, M., M. Goldstein, S. Abdennadher, **"Enhancing One-class Support Vector Machines for Unsupervised Anomaly Detection"**, Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description, pp. 8-15, 2013.

[10] Mahfouz, A., A. Abuhussein, D. Venugopal, S.G. Shiva, **"Network Intrusion Detection Model Using One-Class Support Vector Machine"**, Advances in Machine Learning and Computational Intelligence, pp. 79-86, 2020.

[11] Hu, W., Y. Liao, V.R. Vermuri, **"Robust Anomaly Detection using Support Vector Machines"**, Proceedings of the International Conference on Machine Learning and Applications, Los Angeles, California, USA, 2003.

[12] Zhang, Z., H. Shen, **"Application of Online-Training SVMs for Real-Time Intrusion Detection with Different Considerations"**, Computer Communications, Vol. 28, Issue 12, pp. 1428-1442, 2005.

[13] Nguyen, Q.T., K.P. Tran, P. Castagliola, **"Nested One-Class Support Vector Machines for Network Intrusion Detection"**, IEEE Seventh International Conference on Communications and Electronics, Hue, Vietnam, 2018.

[14] Chen, R-C., S-P. Chen, **"Intrusion Detection using a Hybrid Support Vector Machine Based on Entropy and TF-IDF"**, International journal of innovative computing, information & control, Vol. 4, No. 2, pp. 413-424, 2008.

[15] Chandrashekhar, A.M., K. Raghuveer, **"Fortification of Hybrid Intrusion Detection System Using Variants of Neural Networks and Support Vector Machines"**, International Journal of Network Security & Its Applications, Vol. 5, No. 1, pp. 71-90, 2013.

[16] Yu, Mingyuan, S. Huang, Q. Yu, Y. Wang, J. Gao, **"A Density-based Binary SVM Algorithm in the Cloud Security"**, International Journal of Security and Its Applications, Vol. 9, No. 7, PP. 153-162, 2015.

[17] Mayuranathan, M., M. Murugan, V. Dhanakoti, **"An Intrusion Detection System using Optimized SVM for Detecting Ddos in Cloud"**, International Journal of Scientific & Technology Research, Vol. 8, Issue. 11, 2019.

[18] Wang, W., X. Du, D. Shan, R. Qin, N. Wang, **"Cloud Intrusion Detection Method Based on Stacked Contractive Auto-Encoder and Support Vector Machine"**, IEEE Transactions on Cloud Computing, 2020.

[19] Mugabo, E., Q-Y. Zhang, **"Intrusion Detection Method Based on Support Vector Machine and Information Gain for Mobile Cloud Computing"**, International Journal of Network Security, Vol. 22, No.2, pp.231-241, 2020.

[20] Jaber, A.N, S. Ul Rehman, **"FCM–SVM Based Intrusion Detection System for Cloud Computing Environment",** Cluster Computing Journal, Issue 4, 2020.

[21] Sakr, M.M, M. Tawfeeq, A. El-Sisi, **"Network Intrusion Detection System Based PSO-SVM for Cloud Computing"**, International Journal of Computer Network and Information Security, pp. 22-29, 2019.

[22] Du, R., X. Liang, J. Tian, **"Support Vector Machine Intrusion Detection Scheme Based on Cloud-Fog Collaboration"**, Mobile Networks and Applications, Vol. 27, pp. 431-440, 2022.

[23] Hadem, P., D.K. Saikia, S. Moulik, **"An SDN-based Intrusion Detection System using SVM with Selective Logging for IP Traceback"**, Computer Networks, Vol. 191, 2021.

[24] Thakkar, A., R. Lohiya, **"Attack Classification using Feature Selection Techniques: A Comparative Study"**, Journal of Ambient Intelligence and Humanized Computing, Vol. 12, No. 4, pp. 1249-1266, 2021.

[25] T-H. Chua, I. Salam, **"Evaluation of Machine Learning Algorithms in Network-Based Intrusion Detection System"**, Cryptology ePrint Archive, 2022.

[26] S. Khan, S. Parkinson, Y. Qin, **"Fog Computing Security: A Review of Current Applications and Security Solutions"**, Journal of Cloud Computing: Advances, Systems and Applications, Vol. 6, Article number: 19, 2017.

[27] N. Mazumdar, A. Nag, J.P. Singh**, "Trust-Based Load-Offloading Protocol to Reduce Service Delays in Fog-Computing-Empowered IoT"**, Computers & Electrical Engineering, Vol. 93, 2021.

[28] Rapuzzi, R., M. Repetto, **"Building Situational Awareness for Network Threats in Fog/Edge Computing: Emerging Paradigms Beyond the Security Perimeter Model"**, Future Generation Computer Systems, Vol. 85, PP. 235-249, 2018.

[29] A. Ometov, O.L. Molua, M. Komarov, J. Nurmi, "**A Survey of Security in Cloud, Edge, and Fog Computing**", Sensors, Vol. 22, No.3, 2022.

[30] Zhang, P.Y., M.C. Zhou, G. Fortino, **"Security and Trust Issues in Fog Computing: A Survey"**, Future Generation Computer Systems, Vol. 88, pp. 16-27, 2018.

[31] Azarkasb, S.O., S. Sedighian Kashi, S.H. Khasteh, **"A Network Intrusion Detection Approach at the Edge of Fog"**, 26th International Computer Conference, Computer Society of Iran, Tehran, Iran, 2021.

[32] Kywan, N.N., **"Analysis and Simulation of HyperText Transfer Protocol at the Application Layer of the Internet"**, International Journal of Scientific and Research Publications, Vol. 9, Issue. 1, pp. 78-84, 2019.

[33] OWSAP, Open Web Application Security Project, **"Top 10 Web Application Security Risks"**, 2018.

[34] E.E. Han, T.N. Phyu, **"Classification of SQL Injection, XSS and Path Traversal for Web Application Attack Detection"**, Fourteenth International Conference On Computer Applications, 2016.

[35] Lee, I., S. Jeong, S. Yeo, J. Moon, **"A Novel Method for SQL Injection Attack Detection Based on Removing SQL Query Attribute Values"**, Mathematical and Computer Modelling, Vol. 55, pp-58-68, 2012.

[36] Sharma, C., S.C. Jain, **"Analysis and Classification of SQL Injection Vulnerabilities and Attacks on Web Applications"**, IEEE International Conference on Advances in Engineering & Technology Research, Unnao, India, 2014.

[37] Rodriguez, G., J. Torres, P. Flores, D.E. Benavides, **"Cross-Site Scripting (XSS) Attacks and Mitigation: A Survey"**, Computer Networks, Vol. 166, 2020.

[38] Yusof, I., A.S.K. Pathan, **"Preventing Persistent Cross-Site Scripting (XSS) Attack by Applying Pattern Filtering Approach",** The Fifth International Conference on Information and Communication Technology for The Muslim World, Kuching, Malaysia, 2014.

[39] Sharafaldin, I., A. Habibi Lashkari, A.A. Gjorbani, **"Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization",** 4th International Conference on Information Systems Security and Privacy, SciTePress: Science and Technology Publication, pp. 108-116, 2018.

[40] https://github.com/CanadianInstituteForCybersecurity/CICFlowMeter.

[41] M. Andrecut, **"Attack vs Benign Network Intrusion Traffic Classification"**, Cornell University, Subjects: Computer Science, Cryptography and Security (cs.CR) 2022.

[42] **"Need Two Big Data Query & Analysis using SQL, Three Queries"**, https://www.sweetstudy.com/files/cn703120-21crwk-pdf.

[43] https://github.com/Colorado-Mesa-University-Cybersecurity/DeepLearning-IDS.

[44] J.L. Leevy, T.M. Khoshgotaar, **"A Survey and Analysis of Intrusion Detection Models Based on CSE‐CIC‐IDS2018 Big Data"**, Journal of Big Data, Vol. 7, Article number: 104, 2020.

[45] The University of New Brunswick, Canada, https://www.unb.ca/cic/datasets/, 2021.

**Authors Biograghy**

**Seyed Omid Azarkasb** received his B.Sc. degree in computer software engineering from Kashan Branch Azad University in 1996 and 2001. He studied artificial intelligent systems at Qazvin University of Technology and got his M.Sc. in 2008. Now, he is a visiting professor and Ph.D. student at K. N. Toosi University of Technology, Tehran, Iran.
His research interests include Intrusion detection systems, machine learning methods, Internet of every Thing (IoE), fog, and Cloud Computing.

**Seyed Hossein Khasteh** received the B.Sc. degree in electrical engineering, the M.Sc. degree in Artificial Intelligence, and the Ph.D. degree in Artificial Intelligence all from the Sharif University of Technology, Tehran, Iran.
He is currently an Assistant Professor with the Computer Engineering Department, K. N. Toosi, University of Technology, Tehran, Iran.
His current research interests include social network analysis, machine learning, and big data analysis.