symmetry

MDPI

Article

Effective Intrusion Detection System to Secure Data in Cloud Using Machine Learning

Ammar Aldallal * and Faisal Alisa

Telecommunication Engineering Department, College of Engineering, Ahlia University, Manama 10878, Bahrain; faisal.alisa@khuh.org.bh

* Correspondence: aaldallal@ahlia.edu.bh

Abstract: When adopting cloud computing, cybersecurity needs to be applied to detect and protect against malicious intruders to improve the organization's capability against cyberattacks. Having network intrusion detection with zero false alarm is a challenge. This is due to the asymmetry between informative features and irrelevant and redundant features of the dataset. In this work, a novel machine learning based hybrid intrusion detection system is proposed. It combined support vector machine (SVM) and genetic algorithm (GA) methodologies with an innovative fitness function developed to evaluate system accuracy. This system was examined using the CICIDS2017 dataset, which contains normal and most up-to-date common attacks. Both algorithms, GA and SVM, were executed in parallel to achieve two optimal objectives simultaneously: obtaining the best subset of features with maximum accuracy. In this scenario, an SVM was employed using different values of hyperparameters of the kernel function, gamma, and degree. The results were benchmarked with KDD CUP 99 and NSL-KDD. The results showed that the proposed model remarkably outperformed these benchmarks by up to 5.74%. This system will be effective in cloud computing, as it is expected to provide a high level of symmetry between information security and detection of attacks and malicious intrusion.

Keywords: intrusion detection system; genetic algorithm; support vector machine; machine learning; fitness function

Citation: Aldallal, A.; Alisa, F. Effective Intrusion Detection System to Secure Data in Cloud Using Machine Learning. *Symmetry* **2021**, *13*, 2306. https://doi.org/10.3390/sym13122306

Academic Editors: Peng-Yeng Yin, Ray-I Chang, Youcef Gheraibia, Ming-Chin Chuang, Hua-Yi Lin and Jen-Chun Lee

Received: 10 October 2021
Accepted: 16 November 2021
Published: 3 December 2021

1. Introduction

Cloud computing is one of the latest service innovations in the field of IT. The primary advantage of cloud computing is that it enables access without constraints of location and time. Cloud computing supports mobile and collaborative applications/services, enables the flexibility of controlling storage capacities, and provides lower costs. Moreover, cloud services are multisource, permitting the end-users to use multiple service providers based on their requirements. The use of cloud computing also reduces capital expenditures, power usage, and physical space and maintenance requirements for on-site storage.

As cloud computing services become more and more common, a large number of companies, banks, and governments have adopted this technology. This transition also exposed these systems to many kinds of cyberattacks by hackers and intruders, warranting robust security mechanisms. Many cloud service companies provide several security services as applications. An example is the Amazon Web Services (AWS) store, which provides services with limited validity and dates depending on the period of service license.

With increasing volumes of data, particularly important medical records, the need for continuous backup and updates is evident. Healthcare data is confidential and is an attractive target for hackers to manipulate or use for illegitimate purposes such as financial gain or political motives [1]. Health data and medical records can include specific patient history, information on prescriptive drugs and medical devices, and other confidential patient information. The privacy of this information is of primary importance. Therefore,

online attacks including identity theft should be avoided, as they lead to illegitimate access to financial, social, and banking records [1].

When adopting cloud computing technologies, cybersecurity must be a priority aspect for healthcare services because of the confidentiality of patient and operational data. Existing intrusion detection systems (IDS) operate on the mechanism of signature or anomaly detection. When the detection mechanism is not suitable, patient and operational data may be stolen. Cybersecurity strategy helps to detect and protect against malicious actors while helping to improve an organization's defense against cyberattacks.

Existing studies in international hospitals have been limited to user awareness of the importance of cybersecurity such as the use of strong passwords, deletion or filtration of unwanted emails, data encryption, confidential treatment of credentials, careful access of information, and swift reporting of any security breaches [2]. Intrusion detection systems are also in place for health sectors, but their local infrastructure may not be free from vulnerabilities. In contrast, the cloud provides a significantly useful and secure service to manage the operations within hospitals or any other organization.

Sooner or later, all government IT operations are bound to be run on the cloud. Currently, the Ministry of Health in Bahrain is switching its systems over to the cloud environment [3]. The cloud provides specific types of security depending on the services provided by the cloud service provider. In certain cases, highly confidential data cannot be hosted on the cloud, while some operational data can be stored online. The data managed on in-house servers must be protected at the same level of security when compared with cloud security. This requires significant work to optimize cloud resource security, which helps to build trust for deploying confidential and sensitive medical data on the cloud.

Several studies have been conducted on hybrid IDS applying different techniques, some of which have proven to have high efficiency while others resulted in only acceptable efficiency. In this research, the implementation of the hybrid IDS was developed by applying both improved genetic algorithm (GA) and support vector machine (SVM) techniques to achieve the best possible results. The proposed system can be used to secure health data in the cloud.

The dataset used in evaluating any IDS has a crucial role in identifying its effectiveness. Therefore, a vast number of research studies have adopted the KDD CUP 99 dataset to evaluate their systems. In this regard, a few examples include [4–7]. Another well-known dataset used in this field is NSL-KDD, which was examined by [8–11]. These datasets are relatively old and have a limited number of features, making them unreliable in simulating current systems and environments.

This work is intended to be applied to cloud data for Bahrain hospitals, but because of the difficulty in accessing their data to develop the intended system, a predefined dataset that mimics the existing cloud data of hospitals, CICIDS2017, was used instead. CICIDS2017 is one of the most recent datasets applied in machine learning applications. The present study is significant because it aimed to:

1. Identify the set of optimal features in the new dataset CICIDS2017, which contains normal and most up-to-date common attacks. This was done to find a set of selected features that improved the performance of the detection mechanism.
2. Examine the effect of newly developed fitness functions on identifying intrusions. The present study aimed to do this using a multifactor fitness function to measure the performance of IDS.
3. Determine the best combination of parameter values for an SVM to produce the maximum detection rate of intrusions within the CICIDS2017 dataset.
4. Conduct a comparative study between the CICIDS2017 dataset and the popular KDD CUP 99 dataset. The outcome was used to generalize the efficiency of the proposed system.

This system will be effective in cloud computing, as it is expected to provide a high level of information security, which will protect data in the cloud environment against several types of malicious attacks.

The rest of this paper is organized as follows: Section 2 elaborates on the background about IDS and related studies using GA and SVM. The proposed hybrid IDS is presented in Section 3, and the proposed improved GA and SVM are explained thoroughly. Explanation of the experiments and data processing along with the results are delineated in Section 4. Finally, the conclusion and future work are the contents of Section 5.

2. Background and Literature Review

Intrusion detection systems (IDS) constitute a vital topic studied extensively by researchers. The techniques applied to IDS are basically categorized into anomaly-based IDS, signature-based IDS, and hybrid techniques. When manipulating huge amounts of data related to IDS, the first step is dataset preprocessing. The main stage of preprocessing is feature selection. Once features are selected, machine learning algorithms are applied to classify the normal and abnormal behavior of intruders. In the following text, we highlight the categories of IDS, followed by feature selection techniques, and afterwards, we introduce the hybrid machine learning techniques used in the IDS in the present study.

2.1. Concepts of IDS and Related Work

An IDS is a system that monitors all outgoing and incoming requests in cloud computing to detect any abnormal or normal activity and is a method of detecting anomalies and misuse intrusion.

Infiltration and intrusions are the main problems in the network and cloud computing, as all services are provided over the internet and are susceptible to cyberattacks. IDS software should be scalable, dynamic, and self-adaptive, with high efficiency to use. The intrusion detection process is divided into three steps [12]:

- collecting data from the monitored systems;
- analyzing the collected data to classify it as normal or intrusion;
- alerting the admin about the possibility of intrusion to the concerned system so that they can take necessary action to prevent intrusion and protect the system.

2.1.1. Anomaly-Based IDS

Anomaly-based IDS detect misuse in the cloud environment or penetrations and classify them into normal and abnormal user behavior through a system that collects all the information on normal user behavior or action over a period of time. Then, a statistical test is performed to check whether a suspected behavior is linked to a normal user's behavior or not. Figure 1 presents the general architecture of anomaly-based IDS. The difficulty in maintaining this type of IDS is that it cannot be updated without losing the data that the previous system was trained on. Also, the accuracy of identification is low, which gives a high number of false positives for this type of system. Anomaly-based IDS has been addressed by several researchers as follows.

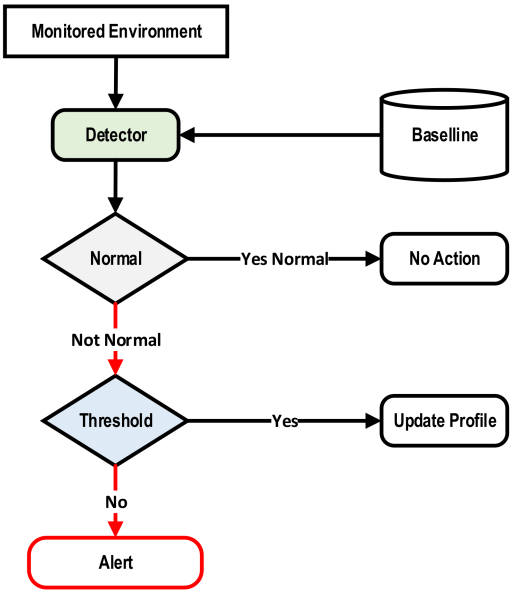

Figure 1. Anomaly-based IDS architecture [13].

Aljawarneh et al. [14] proposed an IDS based on anomaly detection using feature selection, which resulted in an efficient hybrid model. This new hybrid model was useful in estimating intrusions based on network activity and transaction of optimal data options that were available for training. More optimizing techniques were required to create a better IDS model that has a better accuracy rate.

In reference [15], a web application profile was developed by learning constants (e.g., to log in, the value for the user should be the same as the value when logging in). To verify whether the constants were violated, the source code was used. If a static element violation was observed, an anomaly was entered.

In reference [16], an anomaly-based IDS was developed by analyzing the workflow of all web sessions. It consisted of small business groups of data objects. These groups were used and applied to typical data access sequences for the workflow processes. The authors used a hidden Markov model (HMM) application. The results showed that this approach could detect anomalous web sessions and lent evidence to suggest that the clustering approach could achieve relatively low FPR while maintaining its detection accuracy.

Le et al. [17] developed the double-guard framework, which tested both database and web server logs to detect attacks that leaked information. They reported a 0.6% FP rate for dynamic webpages and a 0% FP rate for static webpages.

Nascimento and Correia [18] studied an IDS in which they collected the dataset from a scale web app and trained the dataset. They considered "GET" requests and did not consider "POST" types of requests or response pages. They took logs from the T-Shark tool and converted them to a common format. The filtered data was created by accessing the subapplications. They used nine detection models.

Ariu [19] developed a host-based IDS to protect web applications from attacks by using HMMs. This system was used to model a series of attributes and values received by a web app. To calculate the different parameters and values, they used several HMMs and combined them to arrive at a specific demand on the probability that was generated from the training dataset.

In reference [20], a firewall was developed as a web application to detect any anomalous request and capture the behavior through an "XML" file that defined the required attributes of parameter values. Input values that deviated from the profile were considered attacks. However, this approach produced *FP* warnings because it did not consider the path information and page to be more accurate.

2.1.2. Signature-Based IDS

In signature-based IDS, known threats previously discovered and identified from the system help to identify threats in the future. This technique is used to compare the incoming network pattern. An intrusion is detected when the incoming network pattern matches the signature. The advantage of this type of detection method is that it is easy to develop and understand by knowing the network behavior signatures. This technique has very high accuracy and a minimum number of false positives in detecting known attacks. Also, it can add new signatures into the database without changing or modifying existing ones.

The main drawback issue of this IDS technique is that these types of intrusion detecting systems are not able to detect new types of attacks or unknown attacks; a slight change in the pattern or slight variation can fool it. Figure 2 presents the general architecture of signature-based IDS. A few examples of related work that applied signature-based IDS are listed here.

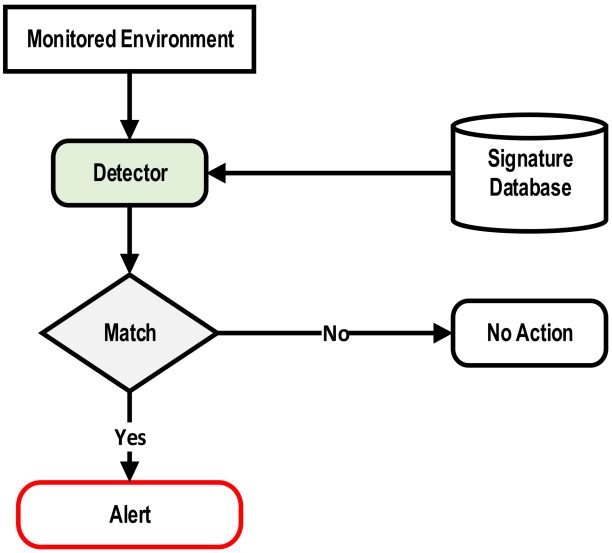

Figure 2. Signature-based IDS architecture [13].

Saraniya [21] developed a network intrusion detection system (NIDS) to secure the network using a signature-based IDS algorithm. It succeeded in capturing packets sent across the entire network using mixed mode and comparing traffic to designer attack signatures. It secured the network and reduced the memory space in the environment. Signature-based IDS are not able to detect emerging and unknown attacks because the signature database must be manually reviewed for every new type of intrusion discovered.

Gao and Morris [22] presented a study on cyberattacks and signature-based IDS for MODBUS based industrial control systems. The rules described were designed to detect the attacks described earlier. The rules were divided into two types: independent and state-based rules. Independent rules were those that analyzed one MODBUS packet, looking for a specific signature match. Standalone rules were those that were enforced with the Snort intrusion detection tool.

Uddin et al. [23] proposed a signature-based distributed IDS using a mobile agent, to transfer the signature database from a large complementary database to a small signature database, then update the databases regularly as new signatures are discovered. The results of the proposed model indicated that IDS worked better than regular systems that used only one database of chain signatures.

Kumar and Gobil [24] implemented a signature-based IDS. They used three tools to develop this IDS system: SNORT, BASE, and TCP REPLAY. This system could detect and analyze intrusions in the network traffic in real time.

The main characteristics and limitations of anomaly-based, signature-based, and hybrid IDS are listed in Table 1.

Table 1. Summary of intrusion detection techniques [17].

Anomaly-Based IDS	
Characteristics	**Limitations**
This technique can detect unknown attacks. Low rate of false alarm for unknown attacks. For IDS, collects the behavior of the network for statistics on testing and performance.	This technique is time-consuming. It needs many features to detect attacks correctly.
Signature-Based IDS	
Characteristics	**Limitations**
The detection rate of this technique is very high for known previous attacks. This technique detects attacks by matching detected packet patterns with previously acquired patterns.	Very high rate of false alarm for unknown attacks. Cannot detect a variant of a known attack or new attacks.
Hybrid Techniques for IDS	
Characteristic	**Limitation**
It is a combination of two or more techniques. Detection accuracy is very high.	Computational cost is very high.

2.2. Feature Selection

The continuous increase in the volume of data makes real-time intrusion detection a difficult task. Feature selection (FS) is a technique used to reduce computation complexity by selecting the relevant features and eliminating many redundant and needless features from the data to produce effective learning model [25,26].

When dimensionality of a feature is high, the selection of meaningful features becomes a difficult challenge. Bioinspired metaheuristic algorithms are efficient in addressing such challenges. They provide high-quality solutions in a reasonable time and within affordable resources. Moreover, they decrease the computational complexity of classification and improve its accuracy [27].

Intelligent IDS work by collecting and analyzing large amounts of data from several areas. These data contain various redundant and irrelevant features, which results in increased overfitting and processing time and reduced detection rate. Therefore, feature selection needs to be applied as a preprocessing step to improve system performance and accuracy while reducing the dataset size [28].

Different approaches have been developed by researchers to reduce the number of features while implementing IDS. In reference [29], a genetic algorithm was applied to select the optimal set of features from KDD CUP 99 and UNSW-NB15. The authors applied a new fitness function, which was composed of TPR, FPR, and the number of selected features, with specific weights given for each factor. The selected features were then passed to the classification stage, at which SVM was applied. The number of optimized features ranged between 7 and 10 for different types of attacks. The maximum accuracy of classification of normal traffic was 99.05%, while the accuracy of classification ranged from 98.25% for R2L to 100% for U2R.

Parimala and Kayalvizhi [30] developed a two-stage feature selection as a prestage for an IDS that provided secure communication in a wireless environment. The first stage applied a conditional random field (CRF) for the initial selection of features. In the second

stage, spider monkey optimization (SMO) was applied to the selected feature to identify the most useful features from the dataset. This model was applied to the NSL-KDD dataset, which is a wireless dataset that has 41 features. This model extracted 16 features. The authors did not provide exact figures of achieved accuracy, but the results outperformed four existing works according to their experiments.

By using the same dataset, NSL-KDD, Reference [31] examined the performance of four feature selection methods: information gain, gain ratio, chi squared, and relief. Their performance was analyzed by applying them on four machine learning algorithms: J48, random forest, naïve Bayes, and k-nearest neighbor (KNN). The obtained results showed that feature selection greatly improved the performance of the IDS but had a slightly negative effect on accuracy.

In addition to information gain and chi squared, Reference [32] applied correlation-based feature selection. These selection methods were applied to the NSL-KDD dataset. The decision tree classifier was the ML algorithm adopted in their IDS model. Initially, the dataset consisted of 41 features. During the preprocessing stage, the nonnumeric data was converted to numeric data, producing a total of 122 features. Their proposed methods selected 20 optimal features out of these 122. The maximum accuracy obtained was 81%.

Reference [33] developed a cloud IDS (CIDS) that started by feature selection. They used an efficient correlation-based feature selection approach, which extended correlation-based feature selection and mutual information feature selection. The feature selection process consisted of two stages. The first stage was used to recognize important features, so it reduced the dimensionality of the dataset. This was achieved by reducing the pairwise calculations of the correlation between features. To distinguish all different classes, this method was combined with the LIBSVM classifier. In the second stage, features were selected such that they were highly relevant to a given class c but not relevant to the selected features. This approach was applied to both the KDD CUP 99 and NSL-KDD datasets. The proposed method enhanced the time required to obtain the optimum set of features and reduced the number of selected features to 10.

Reference [26] proposed a hybrid IDS that consisted of feature selection, density-based spatial clustering of application with noise (DBSCAN), K-Mean++ clustering, and SMO classification. The feature selection adopted to be applied on the KDDCUP 99 dataset was a genetic algorithm that selected 13 features for further processing. The average accuracy achieved in this model was 96.92%.

The feature selection model implemented by [28] consisted of two parts. The first part was the attribute evaluator, which evaluated each attribute separately. The second part was the search method, in which different combinations were tried to obtain the selected list of features. This model was applied to the NSL-KDD dataset. Feature selection was performed on three classes: all attack types (23 types), main attack types (5 types), and two attack types (normal and abnormal). This approach was then examined using three classification models: random forest, J48, and naïve Bayes. The results obtained for the two-attacks type ranged from 98.69 to 99.41%.

2.3. Hybrid Intrusion Detection System Based on Machine Learning

Researchers have implemented IDS using several ML techniques, among other techniques. To focus, the discussion in this section is limited to two ML techniques: the genetic algorithm (GA) and support vector machine (SVM). Before presenting related work conducted using GA and SVM, it is worth shedding some light on these techniques.

2.3.1. Genetic Algorithm Overview

Genetic algorithms (GA) constitute a family of mathematical models that operate on the principles of selection and natural evolution. This heuristic is routinely used to generate useful solutions to optimization and search problems. Genetic algorithms work by transforming selected problems using chromosome-like data and developing chromosomes using selection factors, recombination, and mutations.

The genetic algorithm process usually begins with a random set of chromosomes. These chromosomes are representations of a problem that must be solved. Depending on the attributes of a problem, the positions of each chromosome are encoded in numbers, letters, or bits. These positions are referred to as genes and are randomly changed within a range during development. The set of chromosomes during the evolution stage is called a population. The evaluation function is used to calculate the GOODNESS of each chromosome. There are two primary factors during the evaluation. Crossover is used to simulate the transformation and the natural reproduction of the species. Chromosomal selection for survival and synthesis is biased towards the fittest chromosomes. Genetic algorithms can model virtually any type of constraint in the form of partial functions or by using different chromosome coding schemes specifically designed for a specific problem. Figure 3 shows the structure of a simple genetic algorithm.

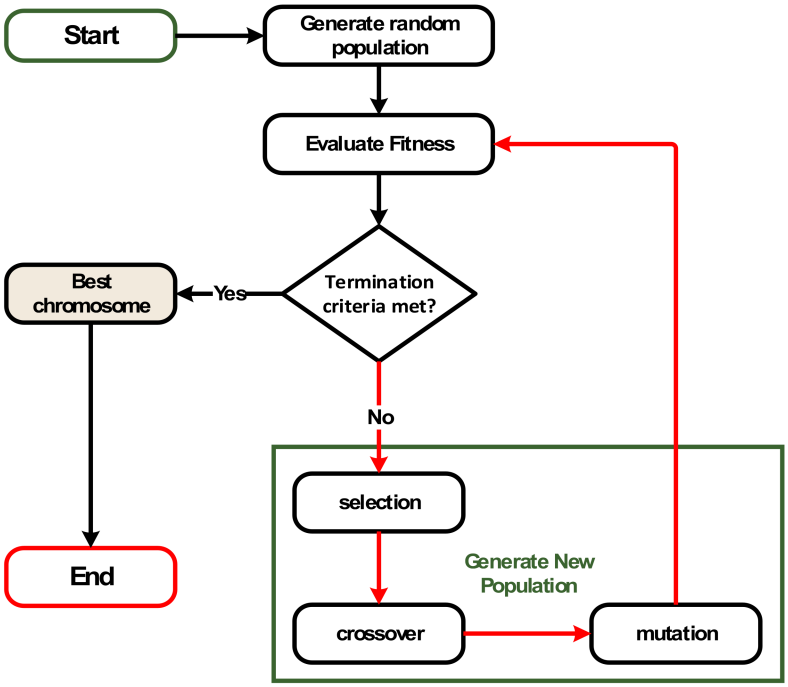

Figure 3. Structure of a genetic algorithm.

Since GAs are best used for optimization, they can be used to generate the best rules for a network connection that can be used to classify the network behavior as intrusive or normal. In addition, GAs can be used to select the optimal set of features from a dataset that can be used in the process of intrusion detection. They can also be used in computer security applications to find optimal solutions to specific problems.

2.3.2. SVM-Based IDS and Related Work

The support vector machine is a machine learning technique known as the best learning algorithm for classification. It is one of the most popular supervised ML algorithms [4,34]. The SVM is a type of pattern classification and regression with a variety of kernel functions. It has been applied to several pattern recognition applications. SVMs have been used mainly for binary classification. The idea is to find a line or hyperplane that separates two classes such that it is as far as possible from the closest instances of each class. Instances of different groups are separated by an area called the margin. The closest instances to the hyperplane are called support vectors. It is required to have the margin as big as possible; this is important to enhance the efficiency of the classification of

newly added instances. The boundary function of the largest margin can be computed as follows [35]:

$$Minimize\ W\ (\alpha)\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}X_I X_J \alpha_I \alpha_J k\ (y_i,\ y_j) - \sum_{i=1}^{N}\alpha_i$$

where α is a vector of N variables, C is the soft margin parameter, $C > 0$, and $k\ (y_i, y_j)$ represents the kernel function of SVMs. Kernel functions are used in SVMs to classify instances of the dataset into different categories. Four main types of kernel functions are listed here [35]:

1. linear kernel: $k\ (y_i, y_j) = y_i^T . y_j$;
2. polynomial kernel: $k\ (y_i, y_j) = (\gamma\ y_i^T . y_j + r)^d$, $\gamma > 0$;
3. radial basis function (RBF) kernel: $k\ (y_i, y_j) = exp(-\gamma\ |\ |\ y_i, y_j{}^2\ |\ |)$, $\gamma > 0$;
4. sigmoid kernel: $y_i, y_j = tanh\ (\gamma\ y_i^T . y_j + r)$

where γ, r and d are kernel parameters. The concepts of the margin, separation hyperplane, and support vector are illustrated in Figure 4.

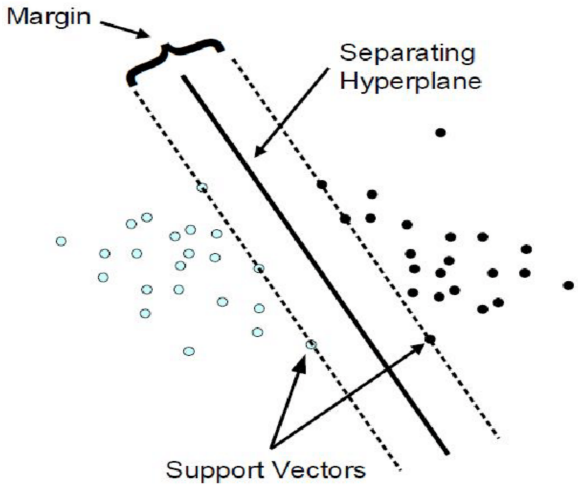

Figure 4. Classical example of SVM linear classifier [4].

This algorithm has been applied in information security to detect intrusion. The main advantage of using a support vector machine for an IDS is its speed as a capability of detecting intrusion in real time. SVMs have become a popular technique for detecting anomalous intrusion because of their good circular nature and their ability to overcome the curse of dimensionality. They are also useful for finding global minimum actual risk using structural risk reduction, because they can generalize well with kernel tricks even in high-dimensional spaces under small-training-sample conditions.

The support vector machine can determine the appropriate setting parameters because it does not depend on traditional experimental risks such as neural networks and can learn a wide range of patterns that can expand. Support vector machines can also dynamically update training patterns whenever there is a new pattern during classification. Several researchers have applied SVMs as part of intrusion detection, as discussed below.

2.3.3. Related Work on Existing Hybrid IDS Using GA and SVM

A set of research works have adopted GA as a base for IDS, while others have adopted SVM. GA has been applied in IDS by several researchers, as follows. Neha Rai and Khushbu Rai [36] proposed an intrusion detection system using genetic algorithm techniques. To create a new population, they used rank selection for selection, single point for crossover, and bitwise for mutation. Reference [34] proposed an intrusion alarm based on a genetic algorithm and support vector machine. In the GA part, an accuracy fitness function was applied, and to create a new population, tournament selection in the selection part, two-

point crossover, and simple mutation in the reproduction part were used. Ojugo et al. [37] proposed a genetic algorithm rule-based IDS. In this system, they applied the confidence fitness function, and to create a new population, they used tournament selection and two-point crossover. Bhattacharjee et al. [38] proposed an IDS using a vectorized fitness function in a genetic algorithm and used uniform crossover. Prakash et al. [39] built an effective IDS using GA-based feature selection. They applied an accuracy fitness function, and to create a new population, they used tournament selection and uniform crossover. Reference [40] proposed a GA-based approach to NIDS. In this system, a confidence fitness function was applied, and to create a new population, two-point crossover was used. Pawar and Bichkar [41] proposed a NIDS using GA with VLC; to create a new population, they used single-point crossover.

Moukhafi et al. [4] proposed a new method of IDS that used an SVM optimized by a GA to improve the efficiency of detecting unknown and known attacks. They used the particle swarm optimization algorithm to select influential features. The detection rate (DR) achieved was 96.38% when applied to 10% of the KDD CUP 99 dataset. The small size of the dataset facilitated the process of the GA–SVM classifier.

Another hybrid IDS was developed in [5]. It used kernel principal component analysis (KPCA), an SVM, and a GA. The kernel function of the SVM was RBF modified to include additional two parameters to obtain better feature values. These parameters were the mean value and the mean square difference values of feature attributes. The chromosome of the GA had three representations. The KDD CUP 99 dataset was used. The DR produced was 95.26%.

Al-Yaseen et al. [6] developed a hybrid IDS which uses K-means, SVM, and ELM. The focus in this work lay on building a training set with high quality that required little time for classification by using K-means. The KDD CUP 99 dataset was used. The DR produced was 95.17%.

Feng et al. [7] developed a hybrid IDS that used an SVM and an ant colony network. The hybrid system benefited from the high classification rate and runtime efficiency produced from the combination of these two algorithms. They used the KDD CUP 99 dataset and managed to achieve a DR of 94.86%.

Tao et al. [34] proposed an improved intrusion detection algorithm based on the GA and SVM methods. This paper proposed an alarm intrusion detection algorithm. Feature selection and weight and parameter optimization of the support vector machine were based on the genetic algorithm. The simulations and experimental results showed that the intrusion detection technology based on the "GA and SVM" proposed increased the intrusion DR, the Accuracy Rate, and the TPR while reducing the FPR. SVM training time was also reduced.

Agarwal and Mittal [42] proposed a hybrid approach for the detection of anomaly network traffic using data mining techniques. In this hybrid model, they applied entropy of network and SVM techniques. This hybrid method worked well and gave high accuracy for the detection of attack traffic with few false alarms. This method was not dynamic and did not detect or decide whether there was an attack.

Serpen and Aghaei [43] proposed a host-based misuse IDS using PCA feature extraction and a KNN classification algorithm. This system used basic analysis Eigen traces to extract data of the OS for a "trace data and k-nearest neighbor" algorithm for classification. This design exhibited very high performance and the ability to detect intrusion and attack plus the type of intrusion. This system worked on the Linux OS only.

Praveen et al. [44] proposed a hybrid IDS algorithm for private cloud services. They applied anomaly intrusion and misuse intrusion, and they used the NET framework as a front end and a SQL server as a back end to implement the algorithm. They conducted an overall study to build a hybrid IDS that would help to detect all types of intrusion in the cloud environment. According to the authors, the major characteristics that a hybrid IDS must have are a dynamic nature, self-adaptiveness, and scalability by the OS in the network and host.

3. The Proposed Hybrid IDS

The proposed hybrid IDS is a combination of two machine learning techniques, the GA and SVM, in which the GA is used for feature selection, which is considered as an optimization problem. Since the used dataset consisted of 79 features, and some of these featured had little to no effect on the identification of intrusion, a GA is applied to select the optimum number of features to maintain the high performance of IDS while reducing the overhead of the classification process. On the other hand, an SVM is applied to perform the actual classification of network data into normal and abnormal behaviors. However, before applying the algorithm to the dataset, it needs to be preprocessed so that the algorithm works smoothly on clean and consistent data. The preprocessing is explained in the following subsection.

3.1. Data Preparation

As mentioned in section one, one of the objectives of this work was to examine the proposed IDS on a recent dataset rather than the commonly used old dataset, KDD CUP 99. The dataset used in this work was the Intrusion Detection Evaluation Dataset [45]. Data covering only 5 days between 3 July 2017 and 7 July 2017 were used in this study. This data contained benign traffic and a limited number of several types of attacks such as brute-force FTP, brute-force SSH, SQL injection attack, and cross-site scripting. The dataset was known as CICIDS2017 and consisted of 170 K records and 79 features.

The first stage of preparation was to clean the data. Cleaning was done in terms of unifying the data type to integers, since some numerical values were entered as "infinity". Second, the missing values were replaced with zeros. Third, two columns were eliminated, as they were dramatically corrupted, causing the algorithm to fail. These columns were Fwd_Packets_s and Bwd_Packets_s. Fourth, the data was scaled, as some columns had one-figure values, while others had up to six-figure values. Scaling was performed using the sklearn function StandardScaler, which normalizes data according to the formula:

$$\frac{z = x - mean}{standard\ deviation}$$

The remaining features are listed in Table 2.

Table 2. Features of the CICIDS2017 dataset [45].

No.	Feature Name	No.	Feature Name	No.	Feature Name
1	Destination_Port	27	Bwd_IAT_Max	53	Avg_Bv/d_Segment_Size
2	Flow_Duration	28	Bwd_IAT_Min	54	Fwd_Header_Length
3	Total_Fwd_Packets	29	Fwd_PSH_Flags	55	Fwd_Avg_Bytes_Bulk
4	Total_Backward_Packets	30	Bwd_PSH_Flags	56	Fwd_Avg_Packets_Bulk
5	Total_Length_of_Fwd_Packets	31	Fv/d_URG_Flags	57	Fwd_Avg_Bulk_Rate
6	Total_Length_of_Bwd_Packets	32	Bwd_URG_Flags	58	Bwd_Avg_Bytes_Bulk
7	Fwd_Packet_Length_Max	33	Fwd_Header_Length	59	Bwd_Avg_Packets_Bulk
8	Fwd_Packet_Length_Min	34	Bwd_Header_Length	60	Bv/d_Avg_Bulk_Rate
9	Fwd_Packet_Length_Mean	35	Fv/d_Packets_s	61	Subflow_Fwd_Packets
10	Fwd_Packet_Length_Std	36	Bwd_Packets_s	62	Subflow_Fv/d_Bytes
11	Bwd_Packet_Length_Max	37	Min_Packet_Length	63	Subflow_Bv/d_Packets
12	Bwd_Packet_Length_Min	38	Max_Packet_Length	64	Subflow_Bv/d_Bytes
13	Bwd_Packet_Length_Mean	39	Packet_Length_Mean	65	Init_Win_bytes_forward
14	Bwd_Packet_Length_Std	40	Packet_Length_Std	66	Init_Win_bytes_backward
15	Flow_IAT_Mean	41	Packet_Length_Variance	67	act_data_pkt_fwd
16	Flow_IAT_Std	42	FIN_Flag_Count	68	min_seg_size_forward

Table 2. *Cont.*

No.	Feature Name	No.	Feature Name	No.	Feature Name
17	Flow_IAT_Max	43	SYN_Flag_Count	69	Active_Mean
18	Flow_IAT_Min	44	RST_Flag_Count	70	Active_Std
19	Fwd_IAT_Total	45	PSH_Flag_Count	71	Active_Max
20	Fwd_IAT_Mean	46	ACK_Flag_Count	72	Active_Min
21	Fwd_lATStd	47	URG_Flag_Count	73	Idle_Mean
22	Fwd_IAT_Max	48	CWE_Flag_Count	74	Idle_Std
23	Fwd_IAT_Min	49	ECE_Flag_Count	75	Idle_Max
24	Bwd_1_AT_Total	50	Down_Up_Ratio	76	Idle_Min
25	Bwd_IAT_Mean	51	Average_Packet_Size	77	Label
26	Bwd_lATStd	52	Avg_Fv/d_Segment_Size		

The second dataset used in this work was KDD CUP 99, which was used to benchmark the performance of the proposed system. The dataset consisted of only 42 features, which are listed in Table 3.

Table 3. Features of the KDD CUP 99 dataset [46].

No.	Feature Name	No.	Feature Name
1	duration	22	is_guest_login
2	protocol_type	23	count
3	service	24	srv_count
4	flag	25	serror_rate
5	src_bytes	26	srv_serror_rate
6	dst_bytes	27	rerror_rate
7	land	28	srv_rerror_rate
8	wrong_fragment	29	same_srv_rate
9	urgent	30	diff_srv_rate
10	hot	31	srv_diff_host_rate
11	num_failed_logins	32	dst_host_count
12	logged_in	33	dst_host_srv_count
13	Inum_compromised	34	dst_host_same_srv_rate
14	Iroot_shell	35	dst_host_diff_srv_rate
15	Isu_attempted	36	dst_host_same_src_port_rate
16	Inum_root	37	dst_host_srv_diff_host_rate
17	Inum_file_creations	38	dst_host_serror_rate
18	Inum_shells	39	dst_host_srv_serror_rate
19	Inum_access_files	40	dst_host_rerror_rate
20	Inum_outbound_cmds	41	dst_host_srv_rerror_rate
21	is_host_login	42	label

3.2. Genetic Algorithm (GA)

As aforementioned, GAs constitute a family of mathematical models that operate on the principles of selection and natural evolution. GAs have multiple parameters, each of which can be implemented in several methods. GAs represent one of the best techniques for

optimization problems and feature selection. The following is a description of GA operators: initial population creation, crossover, mutation, and the proposed fitness function, which was used in the feature selection stage. These operators are summarized in Algorithm 1.

Algorithm 1: GA process

1: Create randomly first generation.
2: Evaluate fitness of each chromosome in the first generation.
3: **Loop**
4: Create next generation following these steps:
5: selection;
6: crossover;
7: mutation.
8: **Until** stop criteria is met.
9: The result is the best chromosome's content of the last generation.

3.2.1. Initial Population

The dataset utilized in this work consisted of 78 features, which was a relatively high number and could affect the quality of the output. Therefore, among these 78, the GA chose 25 features randomly to start the initial population. Knowing that the GA was tested with different numbers of features, from 10 to 25, the results were close, and the training and testing times of the data were adequate. After experimenting with 30 to 40 of the selected features, a decrease in the performance of the system was observed, as the accuracy percentage decreased and the times for training and testing data increased.

3.2.2. Crossover

The second operator of the GA is crossover, which was applied with a probability of 90%. In the crossover, we used single-point crossover, since it has frequently been used by other researchers. The single point was selected randomly such that it was within the first half of the chromosome length. Then the content of the chromosome to the left of the crosspoint was copied, as was the similar position in the offspring, while the content to the right of the crosspoint was switched between parent 1 and parent 2 to produce offspring 1 and offspring 2, respectively, as shown in Figure 5.

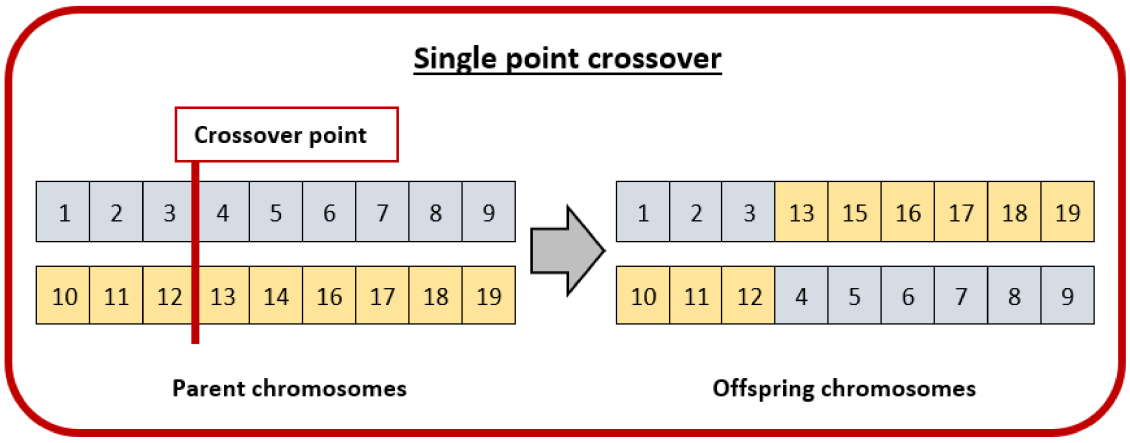

Figure 5. Crossover operation and its effects in generation of offspring.

3.2.3. Mutation

To avoid local convergence, mutation is performed with a very small probability, which was 1% in the present work. In mutation, one gene is selected randomly from the newly generated offspring and replaced with one from the search space, which in the present study was one of the 77 features. During replacement, a check is made to make sure

that the newly added gene is not duplicated in the offspring. This is to avoid redundancy of features. An example of the mutation process is illustrated in Figure 6.

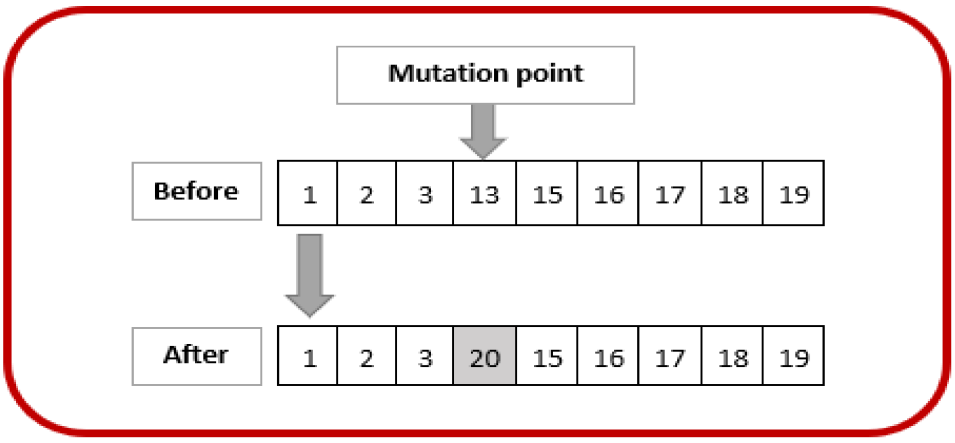

Figure 6. Mutation operation and its effects in generation of offspring.

3.2.4. Improved Fitness Function

The accuracy of classifying the test dataset was used to assess chromosome quality. The fitness function was represented in terms of accuracy. Accuracy is the percentage of the sum of true positive (*TP*) and true negative (*TN*) results divided by the sum of true positive (*TP*), true negative (*TN*), false negative (*FN*), and false positive (*FP*) results. It has been adopted by several researchers as a fitness function (as mentioned in the previous section) to evaluate GA performance. Accuracy was calculated per the following formula:

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

Accuracy cannot be considered a sole metric with which to evaluate a system, since it is not an accurate measure if the dataset is not balanced (both negative and positive classes have a different number of data instances). Hence, the proposed fitness function to be considered in this work was developed using several measures. These measures were F1-score, accuracy, and false positive rate (FPR). F1-score is a metric that considers both precision and recall and is defined as follows:

$$F1\ score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (2)$$

where precision is the fraction of true positive results among all retrieved results and calculated as follows:

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

On the other hand, recall, or true positive rate (TPR), is the fraction of true positive results among all true positive samples in the dataset and calculated as follows:

$$Recall(true\ positive\ rate) = \frac{TP}{TP + FN} \quad (4)$$

The proposed fitness function combined these three metrics in the following formula with different weights for each metric:

$$fitness = \alpha\ F1_{score} + \beta \times \text{accuracy} + \gamma \times \text{TPR} \quad (5)$$

where α, β, and γ are weighting factors that sum to 1. Since F1-score provides high indication to the results, it was assigned the highest value, 0.6, while both accuracy and TPR were considered as secondary measures and assigned 0.2 for each. The optimal value

of the fitness function can be obtained through examining all possible combinations of weighting factors, which sum up to 66 possible combinations.

3.2.5. Stopping Criteria

The process of the GA is repetitive. For each iteration, there is a generation, which consists of a set of chromosomes that represent a solution. The maximum number of iterations was set to 100. Therefore, the stopping criteria were either the GA running for 100 iterations or a lack of improvement in the solution obtained. Improvement was measured in terms of the fitness function defined above, so the stopping criterion was a lack of improvement in the obtained accuracy. To define the improvement, we used a threshold. If the difference of accuracy between two consecutive generations was less than the threshold, then the process stopped. The threshold for this system was set to be 0.05.

3.3. Support Vector Machine (SVM)

The support vector machine (SVM) is one of the best machine learning algorithms used for binary classification. IDS can be viewed as binary classification, since transactions are classified as either normal or intrusions regardless of the type of attack.

This work adopted an SVM because of the following advantages. SVMs are highly effective in high-dimensional spaces such as the case we studied here. Even if the number of dimensions is greater than the number of samples, an SVM still produces effective results. SVMs are memory efficient, because in the decision function, they use a subset of training points. SVMs are also versatile, as the decision function can be specified using different kernel functions [27].

3.4. Integration between GA and SVM

The process starts by applying a genetic algorithm to select the needed number of features. At this stage, the preprocessed data (explained later in Section 4.1) is fed into the proposed GA to extract the proper set of features, since the original data consist of a high-dimensional feature set with 77 features, out of which the best features are extracted. In the next stage, the dataset of these features is divided into training and testing sets. After that, a support vector machine is applied to the training set, and then the fitness is calculated. If the fitness is not accepted, the SVM parameters are changed, and SVM is applied again on the training set to calculate the new accuracy. What is meant by not accepted is that the accuracy is less than the threshold. For this work, the threshold was set to the minimum accuracy achieved by other researchers, which was 94.86% [7]. If the results are accepted, they are saved for future comparison with results obtained from other iterations of different feature selections. The next step is to check the total number of iterations; if it is greater than the threshold, then these results, along with the parameters that led to these results, are stored, and the process stops. If the iterations are less than the threshold, the process loops again to the first step. Figure 7 shows the flowchart of the proposed hybrid IDS. In the present work, the GA at the first stage was run for 100 generations. At each generation, a collection of features was examined using the fitness function on that sample. The proposed Hybrid IDS algorithm is presented in Algorithm 2:

Algorithm 2: Hybrid IDS using GA and SVM.

```
1: Apply Algorithm 1 to select features.
2: loop
3:    Create initial generation.
4:    for i=1 to n /*n is the number of folds*/
5:       Select 1/n of the population as a test set and (n−1)/n for training.
6:       Apply SVM to the current training set with specific hyperparameters.
7:       Evaluate the performance of the results using the fitness function.
8:    end for
```

Algorithm 2: *Cont.*

9: performance of the current generation= average score of the results of n folds
10: Save the results of step 6.
11: **until** the stop criteria are met.
12: Display the best result, along with the hyperparameters for the SVM that produced it.

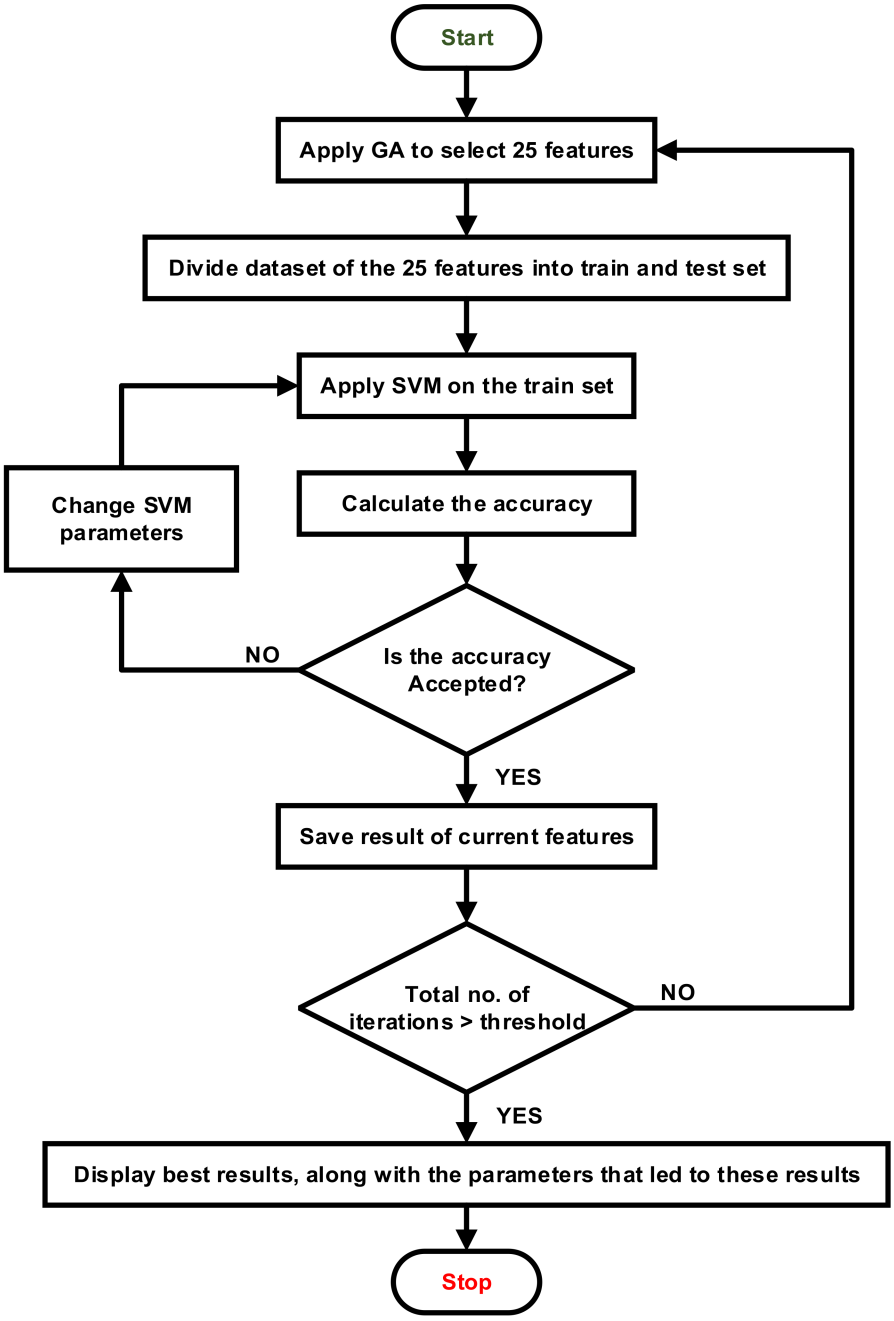

Figure 7. Flowchart of the proposed hybrid IDS.

4. Implementation and Results

This section explains the implementation process. It starts with data preparation, then explains the measures used to evaluate the system performance. Finally, it discusses the obtained results and how they were evaluated using different metrics so they could be compared with existing techniques.

4.1. Experiments Procedure and Results

The conducted experiments had two folds. The first was to find the optimal set of features, while the second was to identify the set of parameters values for the SVM that produce the maximum performance in terms of the proposed fitness function. For the first fold, the GA started with a chromosome length equal to 10, which represents the number of features. For the second fold, the SVA was applied to these 10 features with different parameter values, which are listed in Table 4. The SVA has three basic parameters: kernel, gamma, and degree. The kernel functions used in this work were linear, polynomial, RBF and sigmoid. For linear, the degree is useless, but degree affects the results when the kernel function is polynomial. The final parameter of the SVM that was applied in the present experiment is gamma. It can be either Scale or Auto. The experiment was conducted using 12 different combinations of these parameters, as illustrated in Table 5. For each combination, the GA was executed with a different set of features for a maximum predefined number of generations. The best result of the last generation was considered in this study. The results of this experiment are shown in Table 5. The same type of experiment was conducted for chromosome lengths of 15, 20, 25, 30, 35, and 40, producing the results listed in Tables 6–11. Beyond 40 features, the system experienced unacceptable slowness; hence, these results were not considered in the analysis.

Table 4. List of SVM parameters.

Kernel	Gamma	Degree
Linear	Scale AUTO	3
Poly	Scale AUTO	1, 2, 3
RBF	Scale AUTO	3
Sigmoid	Scale AUTO	3

Table 5. Results of 12 experiments on the proposed hybrid IDS using 10 features and different SVM parameters.

SVM Parameters	Fitness Value
svclassifier = SVC (kemel = 'linear', gamma = 'scale', degree = 3)	0.994302
svclassifier = SVC (kernel = 'linear', gamma = 'auto', degree = 3)	0.994302
svclassifier = SVC (kernel = 'poly', gamma = 'scale', degree = l)	0.994302
svclassifier = SVC (kemel = 'poly', gamma = 'scale', degree = 2)	0.994542
svclassifier = SVC (kernel = 'poly', gamma = 'scale', degree = 3)	0.994542
svclassifier = SVC (kernel = 'poly', gamma = 'auto', degree = l)	0.994302
svclassifier = SVC (kemel = 'poly', gamma = 'auto', degree = 2)	0.994583
svclassifier = SVC (kernel = 'poly', gamma = 'auto', degree = 3)	0.994542
svclassifier = SVC (kernel = 'rbf', gamma = 'scale', degree = 3)	0.994624
svclassifier = SVC (kernel = 'rbf', gamma = 'auto', degree = 3)	0.994624
svclassifier = SVC (kemel = 'sigmoid', gamma = 'scale', degree = 3)	0.991761
svclassifier = SVC (kemel = 'sigmoid', gamma = 'auto', degree = 3)	0.991947
Average	0.994031

Names of the selected features: Total Backward Packets, Bwd Packet Length Std, Fwd IAT Total, Bwd IAT Mean, Fwd Header Length, URG Flag Count, CWE Flag, Count, Fwd Avg Packets Bulk, Subflow Fwd Bytes, Active Max.

Table 6. Results of 12 experiments on the proposed hybrid IDS using 15 features and different SVM parameters.

15 Features	
SVM Parameters	**Fitness Value**
svclassifier = SVC (kernel = 'linear', gamma = 'scale', degree = 3)	0.994562
svclassifier = SVQ (kernel = 'linear', gamma = 'auto', degree = 3)	0.994562
svdassifler = SVC (kernel = 'poly', gamma = 'scale', degree = l)	0.994281
svdasslfter = SVC (kernel = 'poly', gamma = 'scale', degree = 2)	0.994583
svclassifier = SVC (kernel = 'poly', gamma = 'scale', degree = 3)	0.994583
svclassifier = SVC (kernel = 'poly', gamma = 'auto', degree = l)	0.994302
svclassifier = SVC (kernel = 'poly', gamma = 'auto', degree = 2)	0.994583
svdassifier = SVC (kernel = 'poly', gamma = 'auto', degree = 3)	0.994583
svdassifier = SVC (kernel = 'rbf', gamma = 'scale', degree = 3)	0.994604
svdassifier = SVC (kernek = 'rbf', gamma = 'auto', degree = 3)	0.994604
svdassifier = SVC (kernel = 'sigmoid', gamma = 'scale', degree = 3)	0.992954
svdassifier = SVC (kernel = 'sigmoid', gamma = 'auto', degree = 3)	0.992954
Average	0.994263

Names of the selected features: Fwd Packet Length Std, Bwd IAT Max, Bwd I AT Min, Fwd PSH Flags, Fwd Header Length, Fwd Packets_s, Bwd Packets_s, ECE Flag Count, Avg Fwd Segment Size, Avg Bwd Segment Size, Fwd Avg Packets_Bulk, Bwd Avg Packets_Bulk, Subflow Fwd Bytes, Subflow Bwd Bytes, Min Seg_Size_Forward.

Table 7. Results of 12 experiments of the proposed Hybrid IDS using 20 features and different SVM parameters.

20 Features		**Confusion Matrix**
SVM Parameters	**Fitness Value**	
svclassifier = SVC (kernel = 'linear', gamma = 'scale', degree = 3)	1	[[18758 551] [411 79084]]
svdassifier = SVC (kernel = 'linear', gamma = 'auto', degree = 3)	1	[[18758 551] [411 79084]]
svclassifier = SVC (kernel = 'poly', gamma = 'scale', degree = l)	1	[[18762 547] [850 78645]]
svclassifier = SVC (kernel = 'poly', gamma = 'scale', degree = 2)	1	[[19215 94] [106 79389]]
svclassifier = SVC (kernel = 'poly', gamma = 'scale', degree = 3)	0.997437	[[33494 193] [47 340]]
svclassifier = SVC (kernel = 'poly', gamma = 'auto', degree = l)	1	[[33429 258] [48 339]]
svclassifier = SVC (kernel = 'poly', gamma = 'auto', degree = 2)	1	[[33487 200] [47 340]]
svclassifier = SVC (kernel-'poly', gamma = 'auto', degree = 3)	0.999979	[[33686 1] [368 19]]
svclassifier = SVC (kernel = 'rbf', gamma = 'scale', degree = 3)	0.994901	[[19228 81] [61 79434]]
svclassifier = SVC (kernel = 'rbf', gamma = 'auto', degree = 3)	0.998482	[[33521 166] [49 338]]

Table 7. *Cont.*

20 Features		Confusion Matrix
SVM Parameters	**Fitness Value**	
svclassifier = SVC (kernel = 'sigmoid', gamma = 'scale', degree = 3)	0.998649	[[33399 288]
svclassifier = SVC ('kernel-sigmoid', gamma = 'auto', degree = 3)	0.99344	[233 154]]
Average	0.998574	[[33436 251]

Names of the selected features: Total Length of Bwd Packets, Bwd Packet Length Mean, Flow IAT Mean, Flow IAT Max, Fwd IAT Std, Bwd IAT Mean, Bwd IAT Max, Bwd Header Length, Fwd Packets_s, PSH Flag Count, URG Flag Count, ECE Flag Count, Avg Bwd Segment Size, Fwd Avg Bytes Bulk, Bwd Avg Packets Bulk, Subflow Bwd Bytes, Init Win Bytes Backward, Active Min, Idle Mean, Idle Max.

Table 8. Results of 12 experiments on the proposed hybrid IDS using 25 features and different SVM parameters.

25 Features	
SVM Parameters	**Fitness Value**
svclassifier = SVC (kernel = 'linear', gamma = 'scale', degree = 3)	0.994427
svclassifier = SVC (kemel = 'linear', gamma = 'auto', degree = 3)	0.994427
svclassifier = SVC (kernel = 'poly', gamma = 'scale', degree = l)	0.99426
svclassifier = SVC (kemel = 'poly', gamma = 'scale', degree = 2)	0.995953
svclassifier = SVC (kernel = 'poly', gamma = 'scale', degree = 3)	0.996404
svclassifier = SVC (kernel = 'poly', gamma = 'auto', degree = l)	0.99426
svclassifier = SVC (kernel = 'poly', gamma = 'auto', degree = 2)	0.995953
svclassifier = SVC (kernel = 'poly', gamma = 'auto', degree = 3)	0.996341
svclassifier = SVC (kemel = 'rbf', gamma = 'scale', degree = 3)	0.996762
svclassifier = SVC (kernel = 'rbf', gamma = 'auto', degree = 3)	0.996762
svclassifier = SVC (kernel = 'sigmoid', gamma = 'scale', degree = 3)	0.986537
svclassifier = SVC (kemel = 'sigmoid', gamma = 'auto', degree = 3)	0.986537
Average	0.994052

Names of the selected features: Flow Duration, Total Fwd Packets, Fwd Packet Length Std, Flow IAT Min, Fwd IAT Max, Fwd IAT Min, Bwd IAT Total, Fwd PSH, Flags, Max Packet Length, Packet Length Variance, SYN Flag Count, RST Flag Count, URG Flag Count, CWE Flag Count, ECE Flag, Count, Fwd Header Length, Fwd Avg Bulk Rate, Subflow Fwd Bytes, Init_Win_Bytes_Forward, Init_Win_Bytes_Backward, Act Data Pkt Fwd, Active Min, Idle Mean, Idle Max, Idle Min.

Table 9. Results of 12 experiments on the proposed hybrid IDS using 30 features and different SVM parameters.

30 Features	
SVM Parameters	**Fitness Value**
svclassifier = SVC (kemel = 'linear', gamma = 'scale', degree = 3)	0.994583
svclassifier = SVC (kernel = 'linear', gamma = 'auto', degree = 3)	0.994583
svclassifier = SVC (kemel = 'poly', gamma = 'scale', degree = l)	0.994583
svclassifier = SVC (kernel = 'poly', gamma = 'scale', degree = 2)	0.994624
svclassifier = SVC (kernel = 'poly', gamma = 'scale', degree = 3)	0.994539
svclassifier = SVC (kernel = 'poly', gamma = 'auto', degree = l)	0.994583
svclassifier = SVC (kernel = 'poly', gamma = 'auto', degree = 2)	0.994624
svclassifier = SVC (kernel = 'poly', gamma = 'auto', degree = 3)	0.996804
svclassifier = SVC (kernel = 'rbf', gamma = 'scale', degree = 3)	0.99458

Table 9. *Cont.*

30 Features	
SVM Parameters	**Fitness Value**
svclassifier = SVC (kernel = 'rbf', gamma = 'auto', degree = 3)	0.99458
svclassifier = SVC (kernel = 'sigmoid', gamma = 'scale', degree = 3)	0.987238
svclassifier = SVC (kernel = 'sigmoid', gamma = 'auto', degree = 3)	0.987363
Average	0.993557

Names of the selected features: Total Fwd Packets, Total Length of Fwd Packets, Total Length of Bwd Packets, Bwd Packet Length Max, Bwd Packet Length Min, Flow IAT Mean, Fwd IAT Total, Fwd IAT Std, Fwd IAT Max, Bwd IAT Mean, Bwd IAT Std, Fwd PSH Flags, Bwd PSH Flags, Bwd Header Length, Fwd Packets_s, Packet Length Variance, SYN Flag Count, PSH Flag Count, CWE Flag Count, ECE Flag Count, DownJJp Ratio, Average Packet Size, Avg Bwd Segment Size, Fwd Avg Packets_Bulk, Bwd Avg Bytes_Bulk, Bwd Avg Bulk Rate, Subflow Bwd Bytes, Active Max, Active Min, Idle Mean.

Table 10. Results of 12 experiments on the proposed hybrid IDS using 35 features and different SVM parameters.

35 Features	
SVM Parameters	**Fitness Value**
svclassifier = SVC (kernel = 'linear', gamma = 'scale', degree = 3)	0.992532
svclassifier = SVC (kemel = 'linear', gamma = 'auto', degree = 3)	0.992532
svclassifier = SVC (kemeI = 'poly', gamma = 'scale', degree = l)	0.996033
svclassifier = SVC (kernel = 'poly', gamma = 'scale', degree = 2)	0.996951
svclassifier = SVC (kernel = 'poly', gamma = 'scale', degree = 3)	0.996951
svclassifier = SVC (kemel = 'poly', gamma = 'auto', degree = l)	0.995778
svclassifier = SVC (kernel = 'poly', gamma = 'auto', degree = 2)	0.992633
svclassifier = SVC (kernel = 'poly', gamma = 'auto', degree = 3)	0.992574
svclassifier = SVC (kernel = 'rbf', gamma = 'scale', degree = 3)	0.993822
svclassifier = SVC (kernel = 'rbf', gamma = 'auto', degree = 3)	0.993808
svclassifier = SVC (kernel = 'sigmoid', gamma = 'scale', degree = 3)	0.986881
svclassifier = SVC (kernel = 'sigmoid', gamma = 'auto', degree = 3)	0.992652
Average	0.993808

Names of the selected features: Total Length of Fwd Packets, Fwd Packet Length Max, Fwd Packet Length Min, Fwd Packet Length Mean, Bwd Packet Length Min, Bwd Packet Length Std, Flow IAT Std, Flow IAT Max, Flow IAT Min, Fwd IAT Std, Bwd IAT Std, Fwd URG Flags, Fwd Header Length, Bwd Header Length, Fwd Packets_s, Bwd Packets_s, Min Packet Length, Packet Length Variance, SYN Flag Count, Average Packet Size, Avg Bwd Segment Size, Fwd Avg Bytes_Bulk, Fwd Avg Packets_Bulk, Fwd Avg Bulk Rate, Bwd Avg Bulk Rate, Subflow Fwd Packets, Subflow Bwd Bytes, Init_Win_Bytes_Forward, Init_Win_Bytes_Backward, Min_Seg_Size_Forward, Active Mean, Active Std, Active Min, Idle Mean, Idle Std.

These experiments produced the following observations: from 10 to 15 features the results were very close, and the training and testing times of the data were adequate. From 25 to 40 features, a decrease in the performance of the system was observed, as the accuracy percentage decreased, and the time of training and testing increased.

To benchmark the performance of the proposed system with other systems using the KDD CUP 99 dataset, another experiment was conducted using the same 12 combinations of SVM parameters. The results of this experiment are shown in Table 12. Analysis of these experiments is presented in the next subsection.

Table 11. Results of 12 experiments on the proposed hybrid IDS using 40 features and different SVM parameters.

40 Features	
SVM Parameters	**Fitness Value**
svclassifier = SVC (kernel = 'linear', gamma = 'scale', degree = 3)	0.995964
svclassifier = SVC (kemel = 'linear', gamma = 'auto', degree = 3)	0.995964
svclassifier = SVC (kernel = 'poly', gamma = 'scale', degree = l)	0.989691
svclassifier = SVC (kernel = 'poly', gamma = 'scale', degree = 2)	0.991477
svclassifier = SVC (kemel = 'poly', gamma = 'scale', degree = 3)	0.995303
svclassifier = SVC (kernel = 'poly', gamma = 'auto', degree = l)	0.99463
svclassifier = SVC (kernel = 'poly', gamma = 'auto', degree = 2)	0.995839
svclassifier = SVC (kernel = 'poly', gamma = 'auto', degree = 3)	0.999979
svclassifier = SVC (kernel = 'rbf', gamma = 'scale', degree = 3)	0.996609
svclassifier = SVC (kernel = 'rbf', gamma = 'auto', degree = 3)	0.996547
svclassifier = SVC (kemel = 'sigmoid', gamma = 'scale', degree = 3)	0.991734
svclassifier = SVC (kernel = 'sigmoid', gamma = 'auto', degree = 3)	0.994776
Average	0.994876

Names of the selected features: Flow Duration, Total Length of Fwd Packets, Total Length of Bwd Packets, Fwd Packet Length Max, Bwd Packet Length Max, Bwd Packet Length Std, Flow IAT Max, Flow lATMin, Fwd IAT Max, Bwd IATTotal, Bwd lATStd, Bwd lATMin, Fwd PSH Flags, Bwd PSH Flags, Bwd URG Flags, Fwd Header Length, Bwd Header Length, Fwd Packets_s, Bwd Packets_s, Packet Length Mean, Packet Length Variance, SYN Flag Count, RST Flag Count, PSH Flag Count, URG Flag Count, CWE Flag Count, ECE Flag Count, Average Packet Size, Fwd Avg Packets_Bulk, Fwd Avg Bulk Rate, Bwd Avg Bytes_Bulk, Bwd Avg Bulk Rate, Subflow Fwd Packets, Act Data Pkt Fwd, Min Seg Size Forward, Active Std, Active Min, Idle Mean, Idle Max, Idle Min.

Table 12. Results of 12 experiments of the proposed Hybrid IDS using 15 features from (KDD CUP99) dataset.

15 Features	
SVM Parameters	**Fitness Value**
svclassifier = SVC (kernel= 'linear', gamma = 'scale', degree = 3)	0.980213
svclassifier = SVC (kernel = 'linear', gamma = 'auto', degree = 3)	0.980213
svclassifier = SVC (kernel = 'poly', gamma = 'scale', degree = l)	0.972088
svclassifier = SVC (kernel = 'poly', gamma = 'scale', degree = 2)	0.985217
svclassifier = SVC (kernel = 'poly', gamma = 'scale', degree = 3)	0.992792
svclassifier = SVC (kemel = 'poly', gamma = 'auto', degree = l)	0.972032
svclassifier = SVC (kernel = 'poly', gamma = 'auto', degree = 2)	0.985074
svclassifier = SVC (kemel = 'poly', gamma = 'auto', degree = 3)	0.991780
svclassifier = SVC (kernel = 'rbf', gamma = scale', degree = 3)	0.992975
svclassifier = SVC (kemel = 'rbf', gamma = 'auto', degree = 3)	0.992650
svclassifier = SVC (kernel = 'sigmoid', gamma = 'scale', degree = 3)	0.893446
svclassifier = SVC (kernel = 'sigmoid', gamma = 'auto', degree = 3)	0.886727
Average	0.968767

Name of the selected features: service, flag, src_bytes, num_failed_logins, logged_in, lnum_file_creations, lnum_outbound_cmds, count, srv_count, rerror_rate, same srv rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst host rerror rate, dst host srv rerror rate.

4.2. Analysis of Results

Different types of SVM kernel functions were applied. It was observed that if the kernel was linear, then the accuracy did not depend on any other parameters (neither gamma nor degree). If the kernel was polynomial and gamma was scale, then the accuracy depended on the degree. When the degree was two or three, the SVM produced higher accuracy than if the degree was one in most cases. In terms of the number of features, it was noted that the highest ever scored fitness, which was 100%, was obtained when the number of features was 20. This high performance was achieved when SVM kernel function was linear, or when the kernel function was polynomial and the degree was either one or two. In addition, the highest average of all 12 combinations was achieved when the number of features was 20 as well. Graphically, this is illustrated in Figure 8. This showed

that the optimal number of features was 20, which was 25.6% of features in the dataset. These features are listed in Table 7. Achieving the optimal performance of 100% using an only quarter of features proved that the proposed method could perform well using a limited number of features. On the other hand, one could conclude that the dataset had many low-value features that could be eliminated from the dataset safely. However, this conclusion needs to be further examined using other ML techniques.

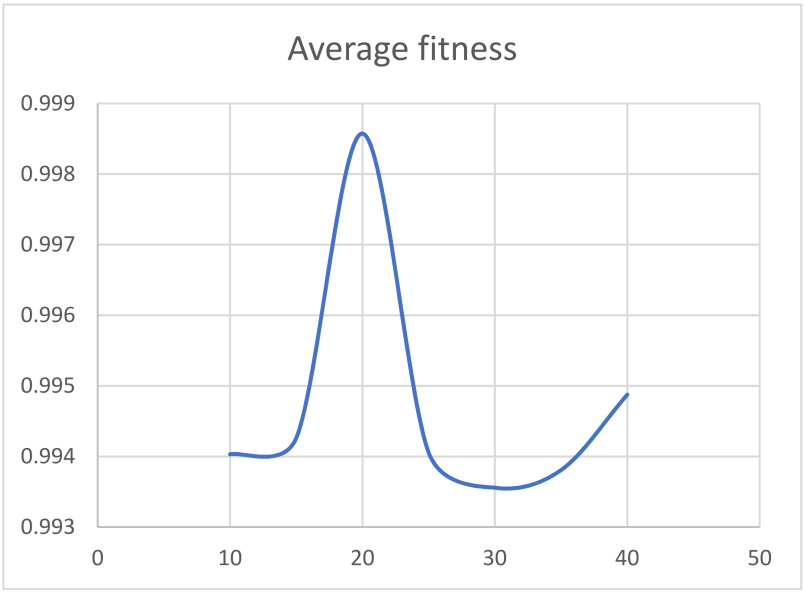

Figure 8. Average fitness per number of features.

Comparing the obtained results using the CICIDS2017 dataset with those of the other hybrid IDS mentioned in Section 2, which used the KDD CUP 99 dataset, it was noted that [4] achieved a 96.38% detection rate while our system achieved a maximum of 100%, which means that the proposed system achieved an improvement of 3.32%. Reference [5] achieved a 95.26% detection rate compared with the 99.50% achieved in the present study, which means that the proposed system achieved an improvement of 4.74%. Reference [6] achieved a 95.17% detection rate, which was lower than that obtained in the present study by 4.83%. Reference [7] achieved a 94.86% detection rate, which was 5.14% less than that obtained by the proposed hybrid IDS. These results are summarized in Table 13.

Table 13. Summary of results analysis using the CICIDS2017 dataset.

Authors	Detection Rate (DR)	Proposed Method (DR)	Achieved Improvement (%)
M. Moukhafi et al., 2018 [4]	96.38%	100%	3.32
F. Kuang et al., 2018 [5]	95.26%		4.74
Al-yaseen et al., 2018 [6]	95.17%		4.83
W. Feng et al., 2014 [7]	94.86%		5.14

The proposed system was also tested with the KDD CUP 99 dataset to compare the present results with previous works that used the same dataset in their works. It was noted that [4] achieved a 96.38% detection rate, while our system achieved a maximum detection rate of 99.3% using the same dataset, which means the proposed system achieved an improvement of 3.03%. Reference [5] achieved a 95.26% detection rate, while the proposed hybrid IDS system achieved a maximum detection rate of 99.65%, which means that the proposed system achieved an improvement of 4.24%. Reference [6] achieved a 95.17% detection rate, which is lower than that obtained by the proposed system by

4.33%. Referenced [7] achieved a 94.86% of detection rate, which is 4.68% lower than that obtained by the proposed hybrid IDS. Compared with other IDS systems that applied fuzzy vectorized GA as a core technique, it was noted that [39] used a vectorized fitness function on the NSL-KDD dataset with 42 features. The NSL-KDD dataset is the enhanced version of KDD CUP 99. In the experiment in [39], six different models were applied, four of which achieved accuracy ranging between 91.86 and 95.53%. On the other hand, two models achieved accuracies of 99.18% and 99.02%. The last two scores were much closer to the achieved results in this work; they were less by only 0.12% and 0.28%, respectively. Another work that applied GA is [47]. The authors developed a GPSVM, which combined both genetic programming and an SVM. The detection rate varied depending on the type of attack. The maximum detection rate achieved was 97.59%, for detecting U2R attacks. It was noted that this detection rate was 1.75% lower than that obtained by our system. Considering the results obtained by a recent study, [48], in which intrusion was detected using a real-time sequential deep extreme learning machine, said machine achieved a maximum accuracy rate of 93.58% when applied on a fused dataset (NSL-KDD and KDD CUP 99). One of the recent studies is [49]. The system therein used a deep extreme learning machine (DELM) to identify any malice or intrusion. Its accuracy using KDD CUP 99 was 94.6%, which was better than its accuracy using NSL-KDD by 0.69%. It is worth pointing that our proposed system outperformed the DELM by up to 5.47%. Table 14 summarizes these results.

Table 14. Summary of results analysis using the KDD CUP 99 and NSL-KDD datasets.

Authors	Accuracy Rate (%)	Dataset	Achieved Improvement (%)
Proposed Hybrid IDS (this work)	99.3	KDD CUP 99	—
M. Moukhafi et al., 2018 [4]	96.38		3.03
F. Kuang et al., 2018 [5]	95.26		4.24
Al-yaseen et al., 2018 [6]	95.17		4.34
W. Feng et al., 2014 [7]	94.86		4.68
Khan et al., 2021 [49]	94.6		4.97
Bhattacharjee et al., 2017 (f1) [39]	99.18	NSL-KDD	0.12
Bhattacharjee et al., 2017 (f2) [39]	99.02		0.28
Pozi et al., 2016 [47]	97.59		1.75
Khan et al., 2021 [49]	93.91		5.74
Khan et al., 2021 [48]	93.58	KDD CUP 99 and NSL-KDD	6.11

5. Conclusions

In this work, we propose a novel hybrid intrusion detection system to secure data in cloud computing based on an improved genetic algorithm (GA) and support vector machine (SVM) algorithm to create an intelligent IDS model. The improved GA is characterized by a newly developed fitness function that is used to evaluate the performance of the hybrid IDS. The fitness function combines three measures: F1-score, accuracy, and TPR, with different weights for each measure. This system was examined using two datasets: the newly developed dataset CICIDS2017, which consists of normal and most up-to-date common attacks, and the well-known dataset KDD CUP 99. The brilliance of the proposed system is that the GA and SVM are executed simultaneously to achieve two objectives at once, which are obtaining the best subset of 79 features with maximum accuracy. The SVM is used to classify data into benign and abnormal using different values for its hyperparameters, including the kernel function, gamma, and degree. Folding of the dataset is applied to change the training and testing dataset to examine the behavior of the system on these kinds of data. The results were analyzed and compared with similar works that applied GA

and SVM for IDS. The results showed that the proposed model outperformed these works in terms of accuracy by a maximum of 5.14% and a minimum of 3.32% using CICIDS2017, a maximum of 4.97% and a minimum of 3.03% using the KDD CUP 99 dataset, and a maximum of 5.74% and a minimum of 0.12% using the NSL-KDD dataset. This system proved its efficiency through the results obtained. To generalize the results, further experiments must be conducted. These must include using even larger datasets, using different datasets, trying different machine learning algorithms or combinations of ML algorithms, and trying more hyperparameters and parameter values for these ML algorithms.

Author Contributions: A.A. Conceptualization, Methodology, software, project administration, supervision, writing—review and editing. F.A. Formal analysis, investigation, Methodology, data curation, software, writing—original draft. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Medical Data in the Crosshairs: Why Is Healthcare an Ideal Target? 14 August 2021. Available online: https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/medical-data-in-the-crosshairs-why-is-healthcare-an-ideal-target (accessed on 15 May 2020).
2. Conaty-Buck, S. Cybersecurity and healthcare records. *Am. Nurse Today* **2017**, *12*, 62–64.
3. eGOVERNMENT, Cloud Computing Initiatives. 22 April 2021. Available online: https://www.bahrain.bh/ (accessed on 15 July 2021).
4. Moukhafi, M.; El Yassini, K.; Bri, S. A novel hybrid GA and SVM with PSO feature selection for intrusion detection system. *Int. J. Adv. Sci. Res. Eng.* **2018**, *4*, 129–134. [CrossRef]
5. Kuang, F.; Xu, W.; Zhang, S. A novel hybrid KPCA and SVM with GA model for intrusion detection. *Appl. Soft Comput. J.* **2014**, *18*, 178–184. [CrossRef]
6. Al-Yaseen, W.L.; Othman, Z.A.; Nazri, M.Z.A. Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system. *Expert Syst. Appl.* **2017**, *67*, 296–303. [CrossRef]
7. Feng, W.; Zhang, Q.; Hu, G.; Huang, J.X. Mining network data for intrusion detection through combining SVMs with ant colony networks. *Future Gener. Comput. Syst.* **2014**, *37*, 127–140. [CrossRef]
8. Ambusaidi, M.A.; He, X.; Nanda, P.; Tan, Z. Building an intrusion detection system using a filter-based feature selection algorithm. *IEEE Trans. Comput.* **2016**, *65*, 2986–2998. [CrossRef]
9. Mustapha, B.; Salah, E.H.; Mohamed, I. A two-stage classifier approach using RepTree algorithm for network intrusion detection. *Int. J. Adv. Comput. Sci. Appl. (IJACSA)* **2017**, *8*, 389–394.
10. Tuan, A.; McLernon, D.; Mhamdi, L.; Zaidi, S.A.R.; Ghogho, M. Intrusion Detection in SDN-Based Networks: Deep Recurrent Neural Network Approach. In *Advanced Sciences and Technologies for Security Applications*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 175–195.
11. Nguyen, K.K.; Hoang, D.T.; Niyato, D.; Wang, P.; Nguyen, D.; Dutkiewicz, E. Cyberattack detection in mobile cloud computing: A deep learning approach. In Proceedings of the IEEE Wireless Communications and Networking Conference, Barcelona, Spain, 15–18 April 2018.
12. He, D.; Qiao, Q.; Gao, Y.; Zheng, J.; Chan, S.; Li, J.; Guizani, N. Intrusion detection based on stacked autoencoder for connected healthcare systems. *IEEE Netw.* **2019**, *33*, 64–69. [CrossRef]
13. Mudzingwa, D.; Agrawal, R. A study of methodologies used in intrusion detection and prevention systems (IDPS). In Proceedings of the IEEE Southeastcon, Orlando, FL, USA, 15–18 March 2012.
14. Aljawarneh, S.; Aldwairi, M.; Yassein, M.B. Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *J. Comput. Sci.* **2018**, *25*, 152–160. [CrossRef]
15. Ludinard, R.; Totel, É.; Tronel, F.; Nicomette, V.; Kaâniche, M.; Alata, É.; Bachy, Y. Detecting attacks against data in web applications. In Proceedings of the 7th International Conference on Risks and Security of Internet and Systems (CRiSIS), Cork, Ireland, 10–12 October 2012.
16. Li, X.; Xue, Y.; Malin, B. Detecting anomalous user behaviors in workflow-driven web applications. In Proceedings of the IEEE Symposium on Reliable Distributed Systems, Irvine, CA, USA, 8–11 October 2012; pp. 1–10.
17. Le, M.; Stavrou, A.; Kang, B.B. DoubleGuard: Detecting intrusions in multitier web applications. *IEEE Trans. Dependable Secur. Comput.* **2012**, *9*, 512–525. [CrossRef]

18. Nascimento, G.; Correia, M. Anomaly-based intrusion detection in software as a service. In Proceedings of the 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W), Hong Kong, China, 27–30 June 2011; pp. 19–24.
19. Ariu, D. Host and Network Based Anomaly Detectors for HTTP Attacks. Ph.D. Thesis, University of Cagliari, Cagliari, Italy, 2010.
20. Gimenez, C.; Villaegas, A.; Alvarez, G. An Anomaly-Based Approach for Intrusion Detection in Web Traffic. *J. Inf. Assur. Secur.* **2010**, *5*, 446–454.
21. Saraniya, G. Securing the Network Using Signature Based IDS in Network IDS. *Shodhshauryam Int. Sci. Refereed Res. J.* **2019**, *2*, 99–101.
22. Gao, W.; Morris, T. On Cyber Attacks and Signature Based Intrusion Detection for Modbus Based Industrial Control Systems. *J. Digit. Forensics Secur. Law* **2014**, *9*, 37–56. [CrossRef]
23. Uddin, M.; Rehman, A.A.; Uddin, N.; Memon, J.; Alsaqour, R.; Kazi, S. Signature-based multi-layer distributed intrusion detection system using mobile agents. *Int. J. Netw. Secur.* **2013**, *15*, 97–105.
24. Kumar, U.; Gohil, B.N. A Survey on Intrusion Detection Systems for Cloud Computing Environment. *Int. J. Comput. Appl.* **2015**, *109*, 6–15. [CrossRef]
25. Lewis, P.M. The characteristic selection problem in recognition system. *IEEE Trans. Inf. Theory* **1962**, *8*, 171–178. [CrossRef]
26. Shakya, V.; Makwana, R.R.S. Feature selection-based intrusion detection system using the combination of DBSCAN, K-Mean++ and SMO algorithms. In Proceedings of the 2017 International Conference on Trends in Electronics and Informatics (ICEI), Tirunelveli, India, 11–12 May 2017; pp. 928–932. [CrossRef]
27. Almomani, O. A Feature Selection Model for Network Intrusion Detection System Based on PSO, GWO, FFA and GA Algorithms. *Symmetry* **2020**, *12*, 1046. [CrossRef]
28. Gul, A.; Adali, E. A feature selection algorithm for IDS. In Proceedings of the 2017 International Conference on Computer Science and Engineering (UBMK), Antalya, Turkey, 5–8 October 2017; pp. 816–820. [CrossRef]
29. Gharaee, H.; Hosseinvand, H. A new feature selection IDS based on genetic algorithm and SVM. In Proceedings of the 2016 8th International Symposium on Telecommunications (IST), Tehran, Iran, 27–28 September 2016; pp. 139–144. [CrossRef]
30. Parimala, G.; Kayalvizhi, R. An Effective Intrusion Detection System for Securing IoT Using Feature Selection and Deep Learning. In Proceedings of the 2021 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 27–29 January 2021; pp. 1–4. [CrossRef]
31. Hakim, L.; Fatma, R. Influence Analysis of Feature Selection to Network Intrusion Detection System Performance Using NSL-KDD Dataset. In Proceedings of the 2019 International Conference on Computer Science, Information Technology, and Electrical Engineering (ICOMITEE), Jember, Indonesia, 16–17 October 2019; pp. 217–220. [CrossRef]
32. Ahmadi, S.S.; Rashad, S.; Elgazzar, H. Efficient Feature Selection for Intrusion Detection Systems. In Proceedings of the 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 10–12 October 2019; pp. 1029–1034. [CrossRef]
33. Wang, W.; Du, X.; Wang, N. Building a Cloud IDS Using an Efficient Feature Selection Method and SVM. *IEEE Access* **2019**, *7*, 1345–1354. [CrossRef]
34. Tao, P.; Sun, Z.; Sun, Z. An Improved Intrusion Detection Algorithm Based on GA and SVM. *IEEE Access* **2018**, *6*, 13624–13631. [CrossRef]
35. Hsu, C.-W.; Chang, C.-C.; Lin, C.-J. *A Practical Guide to Support Vector Classification*; Department of Computer Science, National Taiwan University: Taipei, Taiwan, 2003.
36. Rai, N.; Rai, K. Genetic algorithm-based intrusion detection system. *Int. J. Comput. Sci. Inf. Technol.* **2014**, *5*, 4952–4957.
37. Ojugo, A.A.; Eboka, A.O.; Okonta, O.E.; Yoro, R.E.; Aghware, F.O. Genetic Algorithm Rule-Based Intrusion Detection System (GAIDS). *J. Emerg. Trends Comput. Inf. Syst.* **2012**, *3*, 1182–1194.
38. Bhattacharjee, P.S. Intrusion Detection System for NSL-KDD Data Set using Vectorised Fitness Function in Genetic Algorithm. *Adv. Comput. Sci. Technol.* **2017**, *10*, 235–246.
39. Kalavadekar, M.P.N.; Sane, S.S. Building an Effective Intrusion Detection System using Genetic Algorithm based Feature Selection. *Int. J. Comput. Sci. Inf. Secur.* **2018**, *16*, 97–110.
40. Gong, R.H.; Zulkernine, M.; Abolmaesumi, P. A software implementation of a genetic algorithm-based approach to network intrusion detection. In Proceedings of the Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Network, Towson, MD, USA, 23–25 May 2005; Volume 2005, pp. 246–253.
41. Pawar, S.N.; Bichkar, R.S. Genetic algorithm with variable length chromosomes for network intrusion detection. *Int. J. Autom. Comput.* **2015**, *12*, 337–342. [CrossRef]
42. Agarwal, B.; Mittal, N. Hybrid Approach for Detection of Anomaly Network Traffic using Data Mining Techniques. *Procedia Technol.* **2012**, *6*, 996–1003. [CrossRef]
43. Serpen, G.; Aghaei, E. Host-based misuse intrusion detection using PCA feature extraction and kNN classification algorithms. *Intell. Data Anal.* **2018**, *22*, 1101–1114. [CrossRef]
44. Praveen, K.; Rajendran, M.; Rajesh, R. Healthcare Systems Hybrid Intrusion Detection Algorithm for Private Cloud. *Indian J. Sci. Technol.* **2015**, *48*, 325–329.

45. CICIDS2017, Intrusion Detection Evaluation Dataset, Last Update (2020). Available online: https://www.kaggle.com/cicdataset/cicids2017 (accessed on 25 February 2021).
46. KDD Cup 1999 Data. Available online: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (accessed on 23 February 2021).
47. Pozi, M.; Sulaiman, M.N.; Mustapha, N.; Perumal, T. Improving Anomalous Rare Attack Detection Rate. *Neural Process. Lett.* **2015**, *44*, 279–290. [CrossRef]
48. Khan, M.A.; Ghazal, T.M.; Lee, S.W.; Rehman, A. Data Fusion-Based Machine Learning Architecture for Intrusion Detection. *Comput. Mater. Contin.* **2021**, *70*, 3399–3413. [CrossRef]
49. Khan, A.; Abbas, S.; Rehman, A.; Saeed, Y.; Zeb, A.; Uddin, M.I.; Nasser, N.; Ali, A. A machine learning approach for blockchain-based smart home networks security. *IEEE Netw.* **2021**, *35*, 223–229. [CrossRef]