Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/140516

An Efficient Snort NIDSaaS based on Danger Theory and Machine Learning

Hussein M. Elshafie[*], *Tarek M. Mahmoud and Abdelmgeid A. Ali*

Department of Computer Science, Faculty of Science, Minia University, Egypt

Received: 2 Mar. 2020, Revised: 2 Jul. 2020, Accepted: 4 Jul. 2020
Published online: 1 Sep. 2020

Abstract: Network Intrusion Detection System (NIDS) is a hardware or software application that allows computer networks to detect, recognize and avoid the harmful activities, which attempt to compromise the integrity, privacy or accessibility of computer network. Two detection techniques are used by the NIDSs, namely, the signature-based and anomaly-based. Signature-based intrusion detection depends on the detection of the signature of the known attacks. On the other hand, the anomaly-based intrusion detection depends on the detection of anomalous behaviours in the networks. Snort is an open source signature-based NIDS and can be used effectively to detect and prevent the known network attacks. It uses a set of predefined signatures (rules) to trigger an alert if any network packet matches one of its rules. However, it fails to detect new attacks that do not have signatures in its predefined rules. Thus, it requires constant update of its rules to detect new attacks. To overcome this deficiency, the present paper recommends using Danger Theory concepts inspired from biological immune system with a machine learning algorithm to automatically create new Snort rules, which can detect new attacks. Snort NIDS as a software as a Service (NIDSaaS) in cloud computing has been suggested. Experimental results showed that the proposed modifications of the Snort improved its ability to detect the new attacks.

Keywords: Cloud Computing, Danger Theory, Machine Learning, Network Intrusion Detection System NIDS, Snort

1 Introduction

Currently, the daily life dramatically relies on the complicated computer networks, which have been established for businesses, social media and governments. The computer networks, related to technology such as cloud computing, are the backbone of several applications and services that are needed in all our modern life activities. As regard cyber-security, these computer networks are more vulnerable than before and the modern attackers are more organised and have time, capability and funds to create the attacks that could not be detected even by secure networks [1–3]. Traditional methods, including data encryption techniques, user authentication mechanisms, and firewalls, cannot provide effective protection. Hence, the NIDS should be used to guard computer networks security [4]. NIDSs are based on two detection techniques; signature-based detection and anomaly-based detection [5]. The signature-based NIDS has signatures database of all known attacks. It observes the network packets. If there are any packets that resemble the existing signatures of the database, the system will release an attack alarm [6]. Signature-based NIDS has high accuracy and low rate of missing detection for known attacks. On the other hand, for unknown attacks not included in the signatures database, the signature-based NIDS has low accuracy and high rate of missing detection. The security administrator should always keep the signature database up-to-date [7].

The anomaly-based NIDS generates patterns of the behaviour of unpenetrated networks, i.e. normal profiles. These profiles are used to detect patterns that significantly deviate from them. These deviations may be actual intrusions or new behaviour that need to be added to the normal profiles [8]. One of the advantages of anomaly-based NIDS is its ability to detect new attacks. On the other hand, systems based on this technique has a high rate of false alarms [9].

Snort is one of the most prevalent and actively evolving open-source signature-based NIDS that uses a set of signatures known as Snort's rules, which are frequently updated on the Snort website [10]. Despite the Snort's efficiency in detection of known attacks, it suffers in detecting new attacks.

[*] Corresponding author e-mail: shafie2004@yahoo.com

Negative Selection Algorithm (NSA) is one of the main algorithms of Artificial Immune System (AIS), which is inspired by the biological immune system (BIS). The NSA is a substantial detector generation algorithm, it mimics the immune tolerance in the T-cell maturation operation of the BIS, and achieves effective discrimination of nonself antigens by deleting self-reactive candidate detectors. Because it effectively distinguishes between self and non-self patterns, the NSA is widely used in the field of anomaly detection and computer security [11].

The Clonal Selection Algorithm (CSA) is an immune system inspired algorithm that simulates the basic response of the adaptive immune system. The CSA inspires the principles of the adaptive immune system, such as mutation and clones' reselection. The basic idea of the CSA is that it reproduces only the detectors capable of detecting an antigen. The main objective of the CSA is to evolve the memory set of detectors that represents a solution of a problem [12].

Elshafie et al. [6, 13] used both NSA and CSA to improve the efficiency of Snort through increasing its ability to detect new attacks.

This paper expands our previous work in many significant ways. In addition to using the NSA and the CSA in the improved Snort intrusion detection technique, the concepts of danger theory and some machine learning algorithms are used to enhance the performance of detectors set. Then, they are used to automatically generate new Snort's rules. Finally cloud computing is used to improve the performance of the Snort NIDS.

The rest of this paper is organized as follows: Section 2 addresses the previous pieces of literature. Section 3 presents the theoretical background. In section 4, the proposed model is introduced with details. Section 5 shows experimental results to validate the effectiveness of the proposed improvement. Conclusion and future work are presented in Section 6.

2 Related Works

Several studies addressed enhancing the performance of Snort NIDS. Aickelin et al. [10] developed the Snort by generalising rules to recognise new attacks. The classic rule learning operators, such as generalisation and specialisation, were used to release and vary the conditions and parameters of current Snort rules. The experimental results using KDD cup99 dataset showed the effectiveness of this proposed approach to detect variants of various attacks depending on the processing time and the false rate alarm.

Muthuregunathan et al. [14] proposed a parallel clustering technique followed by usage of evolutionary computing comprising of Genetic Algorithm (GA) and Hill Climbing algorithm to optimize the clusters formed. The proposed technique was specifically developed to generate rule set of Snort NIDS efficiently. To accomplish the parallel computing task, the authors used Grid. The obtained results using KDD cup99 dataset showed that careful selection of fitness function could improve the efficiency of rule set generation depending on the true positive and false positive rate.

Fallahi et al. [15] suggested an approach to generate the rules automatically using the logs of performed attacks. The suggested approach has been implemented using two data mining algorithms called Ripper and C5.0. According to the obtained results using the ISCX 2012 dataset showed that the performance of Snort NIDS improved using automatic rule generation depending on four metrics: the false positive rate, recall, precision and F-Measure.

Guruprasad and D'Souza [16] suggested using the evolutionary approach to automate the Snort's rules generation. DARPA 1999, ISCX 2012 and ICMP network packets were used to test the efficiency of the generated rules in detecting attacks. The conducted experimental results showed that the proposed approach detected attacks with a high detection rate based on the average and the best fitness values.

Silalahi et al. [17] presented a signature-generating technique using the honeypot and signature generator. They used Polygraph to generate signatures because it can detect polymorphic worm. The attack data from honeypot was transformed into a rule that could be used by the Snort NIDS. The proposed approach could generate Snort rule for Apache Knacker and similar worms, which were generated from the worm generator made by the author.

Mahfouz et al. [18] presented a comprehensive analysis of some existing machine learning classifiers concerning determination intrusions in network traffic. They analysed the used classifiers along various dimensions, such as feature selection, sensitivity to hyper parameter selection, and class imbalance problems that are inherent to intrusion detection. The authors conducted experiments to evaluate the effectiveness of these classifiers using NSL-KDD dataset. Accuracy, true positive rate (TPR), true negative rate (TNR), recall, Precision, F-measure and Receiver Operating Characteristic (ROC) curves were used to evaluate the proposed model.

Sengaphay et al. [19] proposed a procedure to improve the Snort's rules for behaviour detection in private cloud using multi-sensors. They showed that each sensor would be installed in the private cloud and work in accordance with snort-IDS rules installed in their own selves. If any sensor detects intrusion behaviour, the alert data will be sent to the alert event database. The MIT-DAPRA 1999 and Nmap datasets are used to evaluate the detection performance of the proposed procedure. The obtained experimental results based on the number of detections showed that the proposed multi-sensors cooperated with the proposed snort-IDS rules could detect 51 cases of intrusion behaviours.

Hassan et al. [20] used Snort rules to detect Distributed Denial of Service (DDoS) attacks in cloud computing and remote control systems. They tested how Snort rules can detect DoS attacks and discussed some other existing techniques used for detecting DoS and DDoS. The Graphical Network Simulator-3 (GNS3) was used to develop virtual background for network topology to simulate a distributed nature of the DDoS attack. The performance measurements of the proposed work were accomplished by measuring the system resource consumption at different time interval.

3 Theoretical Background

3.1 Snort

Snort is the popular open-source signature-based NIDS. It represents signatures network attacks data as string patterns. Snort stores these signatures in a database in the form of rules, one for every known attack. This ruleset is updated when new attacks are discovered [21]. The lightweight description language is used to write the Snort rules. Each rule consists of two sections: a header section and an options section. The header section specifies the protocol, the source, and the destination of a network packet for which the rule is written and the type of action is to be taken if the rule is matched. Enclosed in parentheses, the options section of a rule is written in the next of the header section and consists of one or more keywords, and some of them may accept a value. The semicolon is used to separate the rule options, whereas the colon is used to separate the rule option keywords from its argument.

The four main categories of rule options are classified, as follows: General: supply the rule's information and do not cause any impact during detection.

Payload: look for data inside the packet's payload.

Non-payload: look for non-payload data.

Post-detection: specify the rule's specific triggers that will be released when the rule match any packet. Figure 1 describes the structure of the Snort's rules [22, 23].

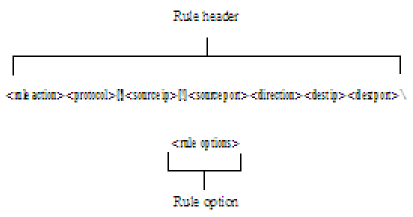
Rule header
<rule action> <protocol> [!] <source ip> [!] <source port> <direction> <dest ip> <dest port> \
<rule options>
Rule option

Fig. 1: Snort rule structure

3.2 Danger Theory

Because of the similarity between the NIDS and the BIS, the immune theory is a good inspiration of network intrusion detection system. They seek to identify normal things and eliminate abnormal things. In the immune system, the antibodies detect abnormal cells. Hence, the NSA is the key to build the NIDS based on the AIS [24]. The NSA creates a set of self-strings that define the normal state of the monitored network. Then, it generates a set of detectors that only distinguish nonself strings. This detector set is used to monitor the suspicious changes of the traffics in the network to classify them as self or non-self [25].

The Danger Theory (DT) was introduced to overcome the limitations of the NSA. It is used to improve the performance of the NSA to achieve better detection rates by integrating the basic danger concepts. The DT principle is the immune system response. It proposes the conditions for triggering the immune response when there is a danger not a suspicious element. In other words, the immune system does not react against the self, unless it is harmful, as it should react against non-self, unless it is harmless [26]. Neither all self nor all non-self is danger, as illustrated in Figure 2.

The NIDS which uses the DT technique aims to detect the signals created in the early stages of an intrusion and to use them as triggers for the activation of defensive response and healing algorithms [27].

The danger model is different from the negative selection model which detects non-self-antigens or pathogenic molecules. The danger model detects the existence of danger signals, released as a result of necrotic cell death within the organism. The DT consists of active suppression with the natural death of the cells (apoptosis), combined with rapid activation when the cells have undergone a pathological death (necrosis) [28]. The Dendritic Cell (DC) is the crime-scene inspector of the natural immune system, i.e. it walks around the tissue for evidence of damage (signals), and for probable suspicious cause of this damage (antigen) [29]. Each DC is able to collect the relative proportions of input signals to produce its set of output signals. There are three types of input signals:

1. Pathogenic Associated Molecular Patterns (PAMP): The PAMP usually indicates an anomalous situation.

2. Danger signal: corresponding to the necrosis process when cells undergo a pathological death. The presence of danger signal indicates that the probability of an anomaly is higher than that in normal situations.

3. Safe signals: corresponding to the apoptosis process, when natural death of the cells, that means there is no bad behaviour. The presence of safe signal indicates that no anomalies are present [28].

In the context of a NIDS, the DT is used in two manners: The first is to generate and manage the three types of signals which define the state of the network, and the second manner is to improve the detector generation process using the Dendritic Cell Algorithm (DCA) [30].

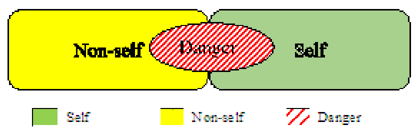

Fig. 2: The relationship between self, non-self and danger

The DCA is a detectors generation process. For each generation, dendritic cells present a set of fixed size elements, randomly chosen from the whole antigen with the corresponding context. According to the context of the presented element, a number of operations will be established to allow the memory detectors population to detect intrusive elements. If the element's context is dangerous, the algorithm checks whether this element is detectable by the memory detectors population. Thus, only this protein will be deleted from the set of introduced elements. If dangerous element is not detectable by the memory detectors population, the algorithm checks in the population of mature detectors if there exists a detector that can detect this element. If such detector exists, it will be added to the memory detectors population and the corresponding protein will eventually be removed from all the elements presented. If the presented element is harmless, the algorithm checks if this element is detectable by the memory detectors population to remove the corresponding detector [26].

3.3 Machine Learning (ML)

ML is a form of artificial intelligence (AI) that provides systems with the ability to automatically improve data without being explicitly programmed. There are three different learning techniques of the ML: supervised, unsupervised and semi-supervised. Picking the appropriate technique depends on the nature of the problem being handled, the type and the volume of the data used for learning [31]. Throughout this paper, the supervised learning technique will be used in the proposed NIDS.

Supervised ML assumes a function (F) given input data (x) to predict an output value (y), y = F(x) by training on a labeled dataset, which is a set of examples with paired input records and their desired outputs. The algorithm continue takes predictions on the training data and stops only when a reasonable level of performance is accomplished [32].

ML techniques can automatically build the model for intrusion detection based on the training data set, which contains data instances that can be represented using a set of features and corresponding labels [33].

3.4 Cloud Computing

In the present world of information technology (IT), the cloud computing is rapidly growing computational model. It solves various IT industrial problems such as computing overloads and potentially expensive investments in hardware for data processing and backups. It can transform the IT industry, making both software and infrastructure more effective, by reshaping the designing and purchasing way of the hardware [34]. Cloud computing introduces a framework that allows end users to easily get benefits from the powerful services and applications through Internet. It provides suitable and available network access to a shared aggregation of configurable computing resources (e.g. networks, servers, storage, applications, etc.); as a service on the Internet for achieving computing requests of the end users [35]. Figure 3 shows three different cloud models depending on the type of the service [36]:

Infrastructure as a Service (IaaS)

Platform as a Service (PaaS)

Software as a Service (SaaS)

In the IaaS,the service provider provides basic computing hardware to the consumers. The consumer gains control of the storage, networks, and other computing capabilities by renting the services from the provider [37]. **In the PaaS,** a development platform is offered as a service. The platform enables consumers to build their applications that run on the service provider's infrastructure. The developer gets applications and platforms, which include tools; libraries and programming languages, and is supported by the service provider that helps create custom applications. **In the SaaS model,** the consumers can access the provider's application through a user interface like a Web browser installed on different consumer devices [38]. In this model, a single instance on the service provider's end supports multiple access instances on the consumer's side. The main advantage of this model is that the licensing costs significantly reduce because only a single instance of an application is required to support multiple consumers' access [37].

Previously, most institutions set their strategy to deploy their NIDS on dedicated hardware to protect computing infrastructures, which contain an important component, from cyber-attacks. However, because of the high cost of security issue and the financial cost saving of the cloud computing such strategy is no longer effective for small and medium institutions that conveniently turn into cloud computing, which provides them with infrastructure, platform and software as services on a pay-per-use basis [39, 40].

The NIDS is introduced as a Service (NIDSaaS) model. The cost of setting up a server, hiring and training system administrators, and installing all the needed applications is mostly eliminated [41]. For NIDS which used rules set for detecting the attacks e.g. Snort, rule

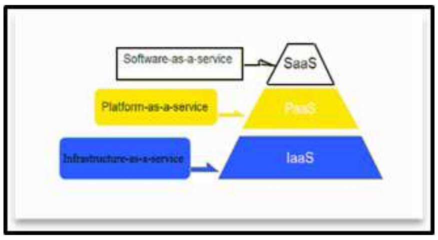

Fig. 3: Cloud computing models

customization is another advantage of NIDSaaS. The cloud provider customizes the rule set by creating rules for new attacks and eliminating deserted rules without adding any additional burden on the consumer. This approach can decrease the occurrence of false positive alerts [42].

4 The Proposed Model

The main steps of the proposed model are defined, as follows:

Step 1: Randomly generate the first generation of detectors set using the NSA described in [13]. This set is denoted by NSA detectors set.

Step 2: Use CSA described in [6] to improve the NSA detectors set obtained in step 1. The obtained improved detectors set is called the CSA detectors set.

Step 3: Pass the CSA detectors set obtained in step 2 to the DCA to produce the improved detectors set and denote it as DCA detectors set.

Step 4: Pass the DCA detectors set to the ML algorithm to produce the more improved detectors set and denote it as ML detectors set.

Step 5: Use the final improved detectors set (ML Detectors Set), generate the new Snort's rules set.

Step 6: The Snort with the new generated rules is implemented in cloud as SaaS.

It is noticeable that the detectors set plays a significant role in defining the self (normal) and non-self (abnormal) patterns for the proposed automatic Snort's rules generation process. In this implementation, the r-chunk matching scheme is used to match between detectors and abnormal patterns. The r-chunk matching rule is defined as follows:

given a string $x = x_1x_2...x_n$ **and a detector**

$d = d_1d_2...d_l$ **with** $l \leqslant n$ **and** $i \leqslant n-l+1$

d matches x $\equiv x_j = d_j$ for $j = i,...,i+l-1$

Where i represents the position where the r-chunk starts [13].

The NSL-KDD dataset is used in training and testing the detectors. Cloning and mutation processes are used in the CSA (Step 2) to generate CSA detectors set. In the CSA detectors set generation step, all detectors in NSA detectors set undergo cloning and mutation. The original detectors remain unchanged, only the clones of the detectors are mutated. If the clone detector produces better affinity than its original detector, the clone replaces its original detector in the new improved CSA detectors set. Otherwise, the original detector remains in the CSA detectors set [6]. The affinity is the matching degree between detector and intruder.

DCA is normally used to generate the detector set from the scratch. This implementation suggests using DCA to select the detectors, which are produced in Step2 and can distinguish between danger and non-danger patterns (Step 3). The DCA is programmed by the Python 3.7.6 release programming language. Python is an interpreted, high-level, general-purpose programming language. The programmed algorithm runs under Kali Linux 2019.3 release operating system. The Kali Linux is an open source Debian-derived Linux distribution used for security and penetration testing designed and funded by Offensive Security, a developer of global information security training and penetration testing services. It can be freely downloaded from its web site [43]. The experimental results indicate that the DCA increased the detectors set true positive detection rate and reduced its false positive detection rate. The machine learning classifiers (Step 4) is used to build a detector classification model. Four ML classifiers J48, IBK, Logistic Regression, and MLP [18] programmed by Python with their default settings will be trained on the training dataset provided by NSL-KDD dataset using 10-folds cross-validation without any pre-processing for the dataset. Performance of the four classifiers will be tested by NSL-KDD test set. Then, the most relevant features of the NSL-KDD dataset are selected using the InfoGainAttributeEval algorithm provided by Weka, which is an open-source machine learning workbench [44]. Performance of the four ML classifier will be compared. The classifier with the best performance will be applied on the improved DCA detectors set to produce the final improved ML detectors set. The automatic Snort's rules generation system will use the ML detectors set to produce the new rules set (Step 5). The automatically generated rules set will be added to the Snort release 2.9.15.0 with the Talos rules which can be freely download from its website [45]. Snort's performance will be compared after applying the new rules set to its performance before adding it. The experiment will be executed in computer with processor Intel Core i5 7200U 2.4 GHZ with 8GB RAM. The generated rules will be inserted into a local rules file (file containing user-generated Snort's rules) as shown in Figure 4.

A NIDS framework that uses the cloud technology SaaS was proposed (Step 6). The proposed framework got benefit from affordable computational cost of the Cloud. The proposed framework employed the danger theory signal mechanism to reduce the false positive rate. The NIDSaaS consists of three main components: NIDS, Rule-set Manager, and the Danger Signal Sensors, as shown in Figure 5. The Snort is the core component of NIDSaaS. It also analyzes and compares network traffic to its predefined rule-set. In addition, it is responsible for collecting the danger signals from its Danger Signal Sensors. The Rule-set Manger is responsible for the automatic update of the Snort rules set by applying the technique proposed in the previous steps. The Danger Signal Sensor collects danger signals within the network. The danger signal may be one of the following cases:

- The presence of the indications of a breach, e.g. port scans and other types of network sniffing.

- A normal user trying to perform administrative tasks.

- A device using multiple accounts and credentials to access network resources.

- A user trying to find valuable data in file servers.

- The existence of command and control activity or persistent access mechanisms [46].

The presence of a danger signal concurrent with the observation of abnormal behaviour increases the possibility of an intrusion. The Snort as SaaS eliminates the need for training new technical, setting up new release, updating new rules or upgrading the hardware of cloud customer, who can get opportunities and responsibility to administrate the NIDSaaS which brings more reliable feeling to them.

Three bare-metal servers, each with two 2.6 GHz Intel Xeon CPUs (16 cores) and 128 GB RAM, will be used to implement the proposed NIDSaaS. The proposed NIDSaaS will implemented on the OpenStack test environment. OpenStack is a free open source cloud computing software deployed to manage IaaS in both public and private clouds. The OpenStack controls diverse, multi-vendor hardware pools of processing, storage, and networking resources throughout a data center where virtual servers and other resources are accessible to the users, who can manage it either through a dashboard or the OpenStack API [47].

The OpenStack includes one controller node, one network node and one compute nodes. Four instances run Kali Linux and Snort software each of them contains rules for only one type of the four attack types. Eight instances running Kali Linux and Snort software running in packet sniffer mode will play the role of danger signal sensors. Kali Linux 2019.3 release operating system will be installed at the top of an instance. The Snort release 2.9.15.0 with the full Talos rules and the new created rules set will be installed to play the role of the central NIDS of the proposed NIDSaaS. This central NIDS is responsible for the coordination between the other NIDS and the danger sensors , as well as managing and dealing with danger signals in the system.

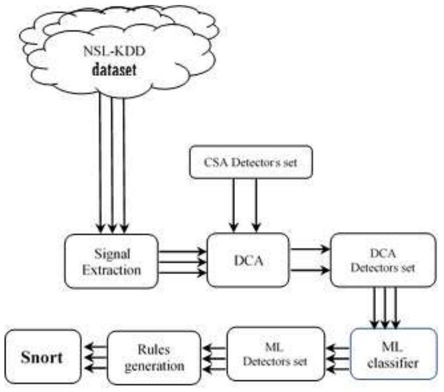

Fig. 4: Automatic Snort rule generation process

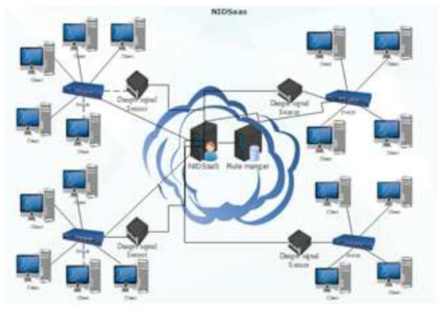

Fig. 5: The NIDSaaS framework

4.1 NIDS Evaluation

The evaluation of NIDS is important to judge its ability to carry out an accurate and effective detection. Performance of NIDS is evaluated by calculating the number of the expected records correctly classified and the number of records incorrectly classified.

Four evaluation metrics are used to evaluate NIDS: Recall (R), Precision (P), False positive rate (FPR) and F-scores. The specific definitions of the four metrics are shown in equations (1)-(4). These metrics can be calculated based on four elements, i.e. **true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN)**. TP and TN, respectively, denote the instances which are correctly classified as attacks and normal instances. FP and FN, respectively, denote the instances which are incorrectly classified as attacks and normal instances. The Recall (R) or (True Positive Rate - TPR) is the ratio of correctly predicted attacks to the size of the actual attacks, as shown in equation 1.

$$R = \frac{TP}{(TP+FN)} \tag{1}$$

Table 1: Description of five class labels of the NSL-KDD data set

Label NO.	Record Type	Description
1	Probe	Probe attack: Scanning and detecting network's security breaches
2	Dos	Denial of Service attack: Denying the network's services
3	U2R	User to Root attack: Illegally access to local root users
4	R2L	Remote to Local attack: unauthorised remote access
5	Normal	normal record

The Precision (P) is the ratio of correctly predicted attacks to the predicted size of the attacks, as shown in equation 2.

$$P = \frac{TP}{(TP+FP)} \tag{2}$$

The False positive rate (FPR) is the ratio of incorrectly predicted attacks to the actual size of the normal instances, as shown in equation 3.

$$FPR = \frac{FP}{(FP+TN)} \tag{3}$$

The F-Score is the weighted harmonic mean of R and P, as shown in equation 4.

$$F-score = \frac{2*P*R}{(P+R)} \tag{4}$$

The popular dataset NSL-KDD is used to train, test, and evaluate many NIDSs. It is a data set suggested to solve some inherent problems of the KDD'99 data set [48]. All the attributes of NSL-KDD data set are similar to those of KDD99 data set. The NSL-KDD dataset was used to evaluate many detection techniques. Each labelled record in the NSL-KDD dataset consists of 41 attributes and the class label. There are five class labels, four attack types (i.e., Probe, DoS, R2L, and U2R) and one normal class, as shown in Table 1 [30]. The NSL-KDD data set can be freely downloaded from the Canadian Institute for Cybersecurity (CIC) website [48].

5 Experimental Results

The Snort's preprocessor has been used to merge the abnormal detection algorithm with Snort. The abnormal detection algorithm used the DCA detectors set, which contains 49 detectors. Then, the Snort was evaluated by applying it on the NSL-KDD test data. Its detection performance, according to evaluation metrics for the binary classes (normal and attack), was compared with the results of the other approaches reported in the previous work [6, 13] after applying them on the NSL-KDD test data. The comparison is listed in Table 2. For best comparison, the best performance is highlighted in boldface.

Table 2: Comparisons of detection results obtained by applying the DCA and previous approaches on the NSL-KDD test data.

Approach	R	P	FPR	F-score
Snort	0.242	0.722	0.098	0.358
Snort + NSA [13]	0.596	0.707	0.298	0.642
Snort + CSA [6]	0.762	0.870	0.141	0.808
Snort + DCA	**0.766**	**0.904**	**0.096**	**0.825**

Table 3: The performance metrics for the trained classifiers on the train data set.

Classifier	R	P	FPR	F-score
J48	0.922	0.904	0.108	0.912
IBK	0.972	0.907	0.109	0.931
Logistic Regression	0.845	0.903	0.102	0.871
MLP	0.868	0.913	0.094	0.889

The performance of the snort with the DCA detectors set was better than the previous approaches. It achieved higher values on Recall, Precision, and F-score than the others. However, it kept the FPR value lower than the other approaches.

Performance of the four ML classifiers was tested by NSL-KDD test set after training them on the training dataset provided by NSL-KDD dataset using 10-folds cross-validation without any pre-processing for the dataset. The results of this phase are summarized in Tables 3 and 4 where Table 3 summarizes performance metrics of the trained four classifiers and the same performance metrics of the trained classifiers on the test dataset are summarized in Table 4.

The four ML classifiers were trained on the training dataset provided by NSL-KDD dataset using 10-folds cross-validation after selecting the most 27 relevant features of 41 features, which were selected by InfoGainAttributeEval algorithm. The performance of the four classifiers was tested by NSL-KDD test set. The results of this phase are summarized in Tables 5 and 6 where Table 5 summarizes performance metrics for the trained four classifiers after selecting the most relevant features and the same performance metrics for the trained classifiers on the test dataset are summarized in Table 6. The results indicate that the J48 classifier outperforms other classifiers with the best performance metrics when applying the feature selection.

The J48 classifier was applied on the DCA detectors set. Then, the selected detectors (23 detectors) were passed to automatic rule generation algorithm to generate 23 new rules. The snort with the new generated rules was evaluated by applying it on the NSL-KDD test data. Its detection performance, according to evaluation metrics, was compared with its detection performance with the DCA detectors set. This comparison is listed in Table 7.

Table 4: The performance metrics for the classifiers on the test dataset.

Classifier	R	P	FPR	F-score
J48	0.854	0.898	0.109	0.873
IBK	0.846	0.896	0.109	0.862
Logistic Regression	0.802	0.898	0.102	0.844
MLP	0.830	0.911	0.088	0.867

Table 5: The performance metrics for the trained classifiers on the train data set after selecting the most 27 relevant features.

Classifier	R	P	FPR	F-score
J48	0.916	0.912	0.104	0.908
IBK	0.907	0.899	0.109	0.894
Logistic Regression	0.797	0.887	0.120	0.838
MLP	0.845	0.899	0.113	0.863

Table 6: The performance metrics for the classifiers on the test dataset after selecting the most 27 relevant features.

Classifier	R	P	FPR	F-score
J48	**0.847**	**0.906**	**0.104**	**0.877**
IBK	0.795	0.889	0.109	0.838
Logistic Regression	0.769	0.877	0.132	0.816
MLP	0.777	0.897	0.106	0.827

Table 7: Comparisons of detection results obtained by the Snort with the new rules set and the Snort with the DCA detectors set on the NSL-KDD test data.

Approach	R	P	FPR	F-score
Snort + DCA	0.766	0.904	0.096	0.825
Snort with the new rule set	**0.937**	**0.921**	**0.088**	**0.928**

The performance of the snort with new rule set is preferred to that with the DCA detectors set. It achieves higher values on Recall, Precision, and F-score than the previous steps. However, it kept the lower value of the FPR.

Implementation of NIDSaaS after adding danger signals simulator as well as applying the clonal selection and danger theory principles was evaluated by NSL-KDD test set. The comparison between evaluation metrics of the implemented NIDSaaS and the standalone NIDS in the previous stage is listed in Table 8.

According to Table 8, although an improvement occurs in Recall, Precision and F-score, the significant improvement is in FPR. Another improvement of the NIDSaaS is the increment of the response time by 43% as a result of distributing the rules and using clonal selection and danger theory principles. There exist cost reduction in implementing NIDSaaS and complexity reducing of the defensive mechanism. This is useful not only for cost reduction, but also for open research communities.

Table 8: Comparisons of evaluation metrics of the NIDSaaS and the standalone NIDS.

Approach	R	P	FPR	F-score
Snort NIDS	0.937	0.921	0.088	0.928
Snort as NIDSaaS	**0.941**	**0.951**	**0.058**	**0.946**

6 Conclusion and Future Work

In this paper, the dendritic cell algorithm and machine learning classifier (J48) were suggested to generate Snort's rules automatically. The conducted experimental results showed that the generated Snort's rules using DCA and ML classifier improved the performance of the Snort NIDS. The suggested Snort framework was implemented as a software as service (SaaS) in the cloud computing, whose major advantage is saving cost. The performance of the suggested Snort framework as NIDSaaS improved, as well as the false positive rate and response time reduced. In the future work, the integration between some deep learning algorithms and the biological immune system concepts should be addressed to enhance the performance of the Snort NIDS.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this article

References

[1] S. A. R. Shah and B. Issac, Performance comparison of intrusion detection systems and application of machine learning to Snort system, Future Generation Computer Systems, **80**, 157-170 (2018).
[2] C.-R Wang, R. -F Xu, S. -J Lee, C. -H Lee, Network intrusion detection using equality constrained-optimization-based extreme learning machines, Knowledge-Based Systems, **147**, 68-80 (2018).
[3] S. Parampottupadam and A. -N Moldovann, Cloud-based Real-time Network Intrusion Detection Using Deep Learning, 2018 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), 1-8 (2018).
[4] V. Hajisalem and S. Babaie, A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection, Computer Networks, **136**, 37-50 (2018).
[5] D. Papamartzivanos, F. Marmol, G. Kambourakis, Introducing Deep Learning Self-Adaptive Misuse Network Intrusion Detection Systems, IEEE Access, **7**, 13546-13560 (2019).
[6] H. M. Elshafie, T. M. Mahmoud, A. A. Ali, Improving the Performance of the Snort Intrusion Detection Using Clonal Selection, 2019 International Conference on Innovative Trends in Computer Engineering (ITCE), 104-110 (2019).

[7] X. Sun, D. Zhang, M. Liu, Z. He, H. Li, J. Li, Detecting and Resolving Inconsistencies in Snort, 2018 IEEE 17th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC), 552-560 (2018).
[8] T. Mohammed, A. Mohamed, S. T. Bella, Intrusion Detection Model Using Naive Bayes and Deep Learning Technique, The International Arab Journal of Information Technology, **17**, (2019).
[9] S. M. Hussein, Performance Evaluation of Intrusion Detection System Using Anomaly and Signature Based Algorithms to Reduction False Alarm Rate and Detect Unknown Attacks, 2016 International Conference on Computational Science and Computational Intelligence (CSCI), 1064-1069 (2016).
[10] U. Aickelin, J. Twycross, T. Hesketh-Roberts, Rule generalisation in intrusion detection systems using SNORT, arXiv preprint arXiv:0803.2973 (2008).
[11] Z. Liu, T. Li, J. Yang, T. Yang, An improved negative selection algorithm based on subspace density seeking, IEEE Access, **5**, 12189-12198 (2017).
[12] H. Jantan, S. Sa?dan, A. M. F. Baskaran, Artificial Immune Clonal Selection Based Algorithm in Academic Talent Selection, Journal of Informatics and Mathematical Sciences, **8**, 225-234 (2016).
[13] T. M. Mahmoud, A. A. Ali, H. M. Elshafie, A Hybrid Snort-Negative Selection Network Intrusion Detection Technique, International Journal of Computer Applications, **975**, (2016).
[14] R. Muthuregunathan, S. Siddharth, R. Srivathsan, S. Rajesh, Efficient snort rule generation using evolutionary computing for network intrusion detection, 2009 First International Conference on Computational Intelligence, Communication Systems and Networks, 336-341 (2009).
[15] N. Fallahi, A. Sami, M. Tajbakhsh, Automated flow-based rule generation for network intrusion detection systems, 2016 24th Iranian Conference on Electrical Engineering (ICEE), 1948-1953 (2016).
[16] S. Guruprasad and R. D'Souza, Development of an Evolutionary Framework for Autonomous Rule Creation for Intrusion Detection, 2016 IEEE 6th International Conference on Advanced Computing (IACC), 534-538 (2016).
[17] D. Silalahi, Y. Asnar, R. S. Perdana, Rule generator for IPS by using honeypot to fight polymorphic worm, 2017 International Conference on Data and Software Engineering (ICoDSE), 1-5 (2017).
[18] A. M. Mahfouz, D. Venugopal, S. G. Shiva, Comparative Analysis of ML Classifiers for Network Intrusion Detection, Fourth International Congress on Information and Communication Technology, 193-207 (2020).
[19] K. Sengaphay, S. Saiyod, N. Benjamas, Creating Snort-ids rules for detection behavior using multi-sensors in private cloud, Information Science and Applications (ICISA) 2016, Springer, 589-601, (2016).
[20] Z. Hassan, R. Odarchenko, S. Gnatyuk, A. Zaman, M. Shah, Detection of Distributed Denial of Service Attacks Using Snort Rules in Cloud Computing & Remote Control Systems, 2018 IEEE 5th International Conference on Methods and Systems of Navigation and Motion Control (MSNMC), 283-288 (2018).
[21] S. S. Gill and R. Buyya, SECURE: Self-protection approach in cloud resource management, IEEE Cloud Computing, **5**, 60-72, (2018).
[22] I. Roy, A. Srivastava, M. Grimm, M. Nourian, M. Becchi, S. Aluru, Evaluating High Performance Pattern Matching on the Automata Processor, IEEE Transactions on Computers (2019).
[23] Cisco, Snort Users Manual Snort Release: 2.9.15.1, Retrieved November (2019).
[24] L. Ma and Y. Chen, An improved Algorithm of Generating Network Intrusion Detector, 2nd International Conference on Electronic & Mechanical Engineering and Information Technology (2012).
[25] N. Elisa, L. Yang, Y. Qu, F. Chao, A revised dendritic cell algorithm using k-means clustering, 2018 IEEE 20th International Conference on High Performance Computing and Communications, 1547-1554 (2012).
[26] A. Khannous, A. Rghioui, F. Elouaai, M. Bouhorma, MANET security: An intrusion detection system based on the combination of Negative Selection and danger theory concepts, 2014 International Conference on Next Generation Networks and Services (NGNS), 88-91 (2014).
[27] M. E. Pamukov, MANET security: Application of artificial immune systems for the creation of IoT intrusion detection systems, 2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), **1**, 564-568 (2017).
[28] O. Igbe, I. Darwish, T. Saadawi, Deterministic dendritic cell algorithm application to smart grid cyber-attack detection, 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), 199-204 (2017).
[29] J. Greensmith and U. Aickelin, The deterministic dendritic cell algorithm, International Conference on Artificial Immune Systems, 291-302 (2008).
[30] O. Igbe, O. Ajayi, T. Saadawi, Denial of service attack detection using dendritic cell algorithm, 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), 294-299 (2017).
[31] J. P. Mueller and L. Massaron, Machine learning for dummies, John Wiley & Sons, (2016).
[32] D. Gibert, C. Mateu, J. Planes, The rise of machine learning for detection and classification of malware: Research developments, trends and challenges, Journal of Network and Computer Applications (2020).
[33] S. Omar, A. Ngadi, H. H. Jebur, Machine learning techniques for anomaly detection: an overview, International Journal of Computer Applications, **79**, (2013).
[34] J. R. Vacca, Cloud computing security: foundations and challenges, CRC Press, (2016).
[35] Z. Chiba, N. Abghour, K. Moussaid, M. Rida, A cooperative and hybrid network intrusion detection framework in cloud computing based on snort and optimized back propagation neural network, Procedia Computer Science, **83**, 1200-1206 (2016).
[36] L. F. Bittencourt, A. Goldman, E. R. Madeira, N. L. da Fonseca, R. Sakellariou, Scheduling in distributed systems: A cloud computing perspective, Computer Science Review, **30**, 31-54 (2018).
[37] Q. Alajmi, A. S. Sadiq, A. Kamaludin, M. A. Al-Sharafi, Cloud Computing delivery and Delivery Models: Opportunity and Challenges, Advanced Science Letters, **24**, 4040-4044 (2018).

[38] T. H. Noor, S. Zeadally, A. Alfazi, Q. Z. Sheng, Mobile cloud computing: Challenges and future research directions, Journal of Network and Computer Applications, **115**, 70-85 (2018).
[39] W. Yassin, N. I. Udzir, Z. Muda, A. Abdullah, M. T. Abdullah, A cloud-based intrusion detection service framework, Proceedings Title: 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), 213-218 (2012).
[40] T. Alharkan and P. Martin, Idsaas: Intrusion detection system as a service in public clouds, 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), 686-687 (2012).
[41] K. Nguyen and T. D. Thi, Develop a distributed Intrusion detection system based on Cloud Computing, International Journal of Computer Science and Information Security (IJCSIS), **15**, (2018).
[42] T. Alharkan, IDSaaS: Intrusion Detection system as a Service in public clouds, M.A. thesis, Queen's University, Canada (2013).
[43] Kali Linux Web Site, cited 2019, Available from: https://www.kali.org.
[44] S. Lang, F. Bravo-Marquez, C. Beckham, M. Hall, E. Frank, WekaDeeplearning4j: A deep learning package for Weka based on Deeplearning4j, Knowledge-Based Systems, **178**, 48-50 (2018).
[45] Snort Web Site, cited 2019, Available from: https://www.snort.org.
[46] Five signs an attacker is already in your network, cited 2020, Available from: https://www.networkworld.com/article/3085141/five-signs-an-attacker-is-already-in-your-network.html.
[47] OpenStack Web Page, cited 2020, Available from: https://www.openstack.org.
[48] The NSL-KDD dataset Web Page, cited 2019, Available from: https://www.unb.ca/cic/datasets/nsl.html.

Hussein M Elshafie received the Bachelor's degree in Mathematics and Computer Science from Minia University, Egypt in 1998 and the Master of Sciences degree from Minia University, Egypt in 2009. He is currently IT specialist at Minia University, Egypt. His research interests include computer security, biologically-inspired computing, bioinformatics, computer networks, data mining, pattern recognition, machine learning, neural networks and artificial intelligence.

Tarek M. Mahmoud received the Bachelor's degree in Mathematics from Minia University, Egypt in 1984 and the Master of Sciences degree from Assiut University, Egypt in 1991. Finally, he received the Philosophy Doctorate in Computer Science (Computer Networks) from Bremen University, Germany in 1997. He is currently professor of computer science at Computer Science Department, Faculty of Science, Minia University, Egypt. His research interests include computer networks, pattern recognition, Social networks, Web and text mining and artificial intelligence.

Abdelmgeid A. Ali received the PhD degree in Computer Science in 1996. Currently he works as a Professor in Computer Science Department, Minia University, El Minia , Egypt. He has published over 80 research papers in prestigious international journals, and conference proceedings. He has supervised over 60 Ph.D. and M. Sc. Students. Prof Ali is a member of the International Journal of Information Theories and Applications (ITA). Prof Ali interests are Information Retrieval, Software Engineering, Image Processing, Data security, metaheuristics, IOT, Digital Image Steganography, Data Warehousing .