Prince Hussein Bin Abdullah College for Information Technology

Computer Science Department

Features Filtration for Intrusion Detection Incorporating Hopfield Artificial Neural Networks

ميزات الترشيح لالشبكات العصبية الاصطناعية لكشف التسلل دمج متعددة الارتباطات

By:

Bashair Fayez Al-Shdaifat

Supervisor:

Dr. Wafa' Slaibi Alsharfat

Co-Supervisor:

Dr. Mohammad El-Bashir

This Thesis was Submitted in Partial Fulfillment of the Requirements for the Master’s Degree in Computer Science

Al Al–Bayt University, Jordan

May, 2016

تفويض

انا بشائر فايز حامد الشديفات، أفوض جامعة آل البيت بتزويد نسخ من رسالتي للمكتبات او المؤسسات او الهيئات او الاشخاص عند طلبهم حسب التعليمات النافذة في جامعة.

التوقيع:

التاريخ:

اقرار والتزام بقوانين جامعة آل البيت وأنظمتها وتعليماتها

أنا الطالب: بشائر فايز حامد الشديفات	الرقم الجامعي :١٣٢٠٩٠١٠٠٣
التخصص: علم حاسوب	الكلية: تكنولوجيا المعلومات

اعلن بانني قد النزمت بقوانين جامعة آل البيت وأنظمتها وتعليماتها السارية المفعول المتعلق باعداد رسائل الماجستير والدكتوراه عندما قمت شخصيا بأعداد رسالتي بعنوان:

Features Filtration for Intrusion Detection Incorporating Hopfield Artificial Neural Networks

وذلك بما ينسجم مع الأمانة العلمية المتعارف عليها في كتابة الرسائل والأطاريح العلمية. كما انني أعلن بان رسالتي هذة غير منقولة او مستلة من رسائل او أطاريح او كتب او ابحاث او أي منشورات علمية تم نشرها او تخزينها في اي وسيلة اعلامية، وتاسيسا على ما تقدم فأنيي اتحمل المسؤولية بأنواعها كافة فيما لو تبين غير ذلك بما فيه حق مجلس العمداء في جامعة آل البيت بالغاء قرار منحي الدرجة العلمية التي حصلت عليها وسحب شهادة التخرج مني يعد صدورها دون ان يكن لي أي حق في التظلم او الأعتراض او الطعن بأي صورة كانت في القرار الصادر عن مجلس العمداء بهذا الصدد.

التوقيع: التاريخ:

Features Filtration for Intrusion Detection Incorporating Hopfield Artificial Neural Networks

By:

Bashair Fayez Al-Shdaifat

Supervisor: Dr. Wafa' Slaibi Alsharafat

Co-Supervisor: Dr. Mohammad El-Bashir

This Thesis was Submitted in Partial Fulfillment of the Requirements for the Master's Degree in Computer Science

Deanship of Graduate Studies

Al al-Bayt University

2016

ACKNOWLEDGEMENT

I am much grateful to the Almighty Allah for helping me fulfill the Master's degree requirements. I owe a great debt of gratitude to, and I am thankful for, many institutions and persons who provided me with support throughout this appealing, yet difficult and challenging, journey. While being thankful for every one of them, I am particularly grateful to some. I like first to express deep respect of my main supervisor, Dr. Wafa' Al Sharfat, and co-supervisor, Dr. Mohammad El-Bashir, for their guidance and support during the Master's journey. Their advice and valuable comments have greatly helped me to remain focused and kept me on track till the end. In addition, I especially thank the chairman of my examination committee, Dr. Qasem Radaideh, the external examiner and the internal examiner, Dr. Omar Shatnawi and Dr. Khaled Batiha I also like also to thank the academic and administrative staff in Al Al–Bayt University. In addition, I express my never-ending gratitude to, and appreciation for, lovely people; first, my parents for their strong support during my Master's journey, and, second, my husband, who, with patience, accompanied me all the way. Thanks are also extended to my friends, brothers, and sisters, who have been very kind to encourage me during my study. To all of you, I have to say: I respect and love you. May Allah bless you. Thanks to you all.

Bashair Fayez shdaifat

May, 2016

Table of Contents

Contents

List of Tables

List of Figures

List of Abbreviations

Expression	Abbreviation
Anomaly-Based Intrusion Detection System	ABIDS
Artificial Intelligence	AI
Anomaly Intrusion Detection	AID
Adaptive Neuro Fuzzy Inference System	ANFIS
Artificial Neural Network	ANN
Cloud Computing	CC
Cloud Computing System	CCS
Cluster-Supporting Object	CSO
Denial of Service	DoS
Detection Rate	DR
False Alarm Rate	FAR
Fuzzy Clustering	FC
Fuzzy Clustering by Local Approximation Memberships	FLAME
False Negative Rate	FNR
False Positive Rate	FPR
Genetic Algorithm	GA
Hopfield Artificial Neural Network	HANN
Host-Based IDS	HBIDS
Intrusion Detection System	IDS
Information Technology	IT
k-Nearest Neighbors	KNN
Network-Based IDS	NBIDS

Remote to Local	R2L
Radial Basis Function	RBF
Recurrent Neural Network	RNN
Simulated Annealing	SA
Simulated-Annealing Fuzzy *c*-Means	SA-FCM
Signature-Based Intrusion Detection Systems	SBIDS
Support Vector Machine	SVM
True Negative Rate	TNR
True Positive Rate	TPR
User Root	U2R
Extended Classifier System	XCS

Abstract

Cloud Computing offers framework to support end users easily owing to its ability to provide an unlimited amount of resources. Cloud Computing providers should protect the system, both from the outsiders and the insiders. We have suggested a system that allows the Cloud Computing Systems (CCSs) to realize the effectiveness of detecting the intrusions and strength of system security.

The Intrusion Detection System (IDS) is the most common system for protection of the CCSs from numerous sorts of attacks. The hybrid algorithm proposed here in lowers the number of features from 41 to 19 features. Based on testing in the current study, the normal distribution 10% of the KDD CUP 99 (KDD '99) benchmark was used for testing and training the suggested, the proposed hybrid algorithm results substantially influences the performance of the system, enhances the efficiency of the detection rates also the accuracy for different attack types and reduces the False Alarm Rate (FAR) in network's IDS.

Chapter I

Introduction

1.1 Introduction

Recently, the concept of green IT has appeared. Various researchers started to find ways to reduce the IT costs and Cloud Computing (CC) emerged. The CC has specially been developed with movement of the IT services. The CC is a new computing model, where the users have to pay for the use of services, without a need for carrying the cost of buying physical hardware.

1.2 Cloud Computing

More studies has recently been spotlighted on CC than on any other computing service due to its ability to provide unlimited resources which the customers, or end users, can use whenever there is satisfactory access to the Internet. The Cloud Computing Systems (CCSs) possess a multitude of private information and resources. Accordingly, these resources and information can be easily threatened by attackers and the CC service providers must protect these systems from outsiders and insiders, alike. Nowadays, the Intrusion detection Systems (IDSs) are the most common tools for protection of the CCSs from different kinds of attacks (Mathew and Jose, 2012).

The CCS can be described as distributed environment that comprises shared resources and services. The cloud environment satisfies scalability and manageability of the features of the computing services which respond to the user's requests through network (Kholidy, et al., 2012). The CC environment is a cost-effective and efficient environment as the users do not need to purchase any software or the technical infrastructure. Consequently, the cloud environment is potentially prone to attacks and security threats like Denial of Service (DoS), unauthorized access, and privacy violations (Raghav, et al., 2013). One of the major defense mechanisms for this kind of environment is the IDSs. These systems protect the cloud services and resources from intrusion and attempts of other types of attacks (Gyanchandani, et al., 2012).

1.3 IDSs

Teodoro, et al. (2009) defined the IDS as security tool, similar to other tools like firewalls and antivirus programs, which is suggested for the security of the information and communication systems to grow more secure and powerful. Thus, the IDS should be implemented to detect and control the events taking place in the network by using intelligent algorithms.

The first study of IDSs was published in 1980 by James Anderson in his attempt to improve the level of security for the computer system by detecting any unauthorized access. To achieve this goal, there was need for making use of auditing files that will detect the unauthorized access (SANS, 2001, a, b & c).

Poston, et al. (2012) indicated that there are different classifications of the IDSs, based on the data sources and the associated detection methods. A briefing on these classifications follows:

1) Classifications based on the Data Source:

Based on the sources of data, the IDSs may be divided into Network-Based IDSs (NBIDSs) and Host-Based IDSs (HBIDSs).

- ❖ The HBIDSs handle data and information collected from an entire system, or host, and are ideally assembled on a machine (SANS, 2001, a, b & c).
- ❖ The NBIDSs handle data and information collected from the network itself (SANS, 2001, a, b & c).

2) Classifications based on the Detection Method:

The IDSs may be divided into Signature-Based Intrusion Detection Systems (SBIDSs) and Anomaly-Based Intrusion Detection Systems (ABIDSs).

- ❖ The SBIDSs, also called misuse detection systems, are based on the fact that each type of attacks has a distinctive behavior (also called signature or pattern). Therefore, the SBIDSs detect only the attacks of known patterns, which is a process known as 'matched detection'. The known attacks are stored in records. If the detected behavior matches with stored record, i.e., if a match is found, then the SBIDS signals an intrusion. On the other hand, if the detected behavior does not match any stored record, then this behavior is classified as a normal pattern. One of the drawbacks of this sort of detection methods is that it can not detect new attacks; it only detects recorded behaviors.

- ❖ Anomaly-Based Intrusion Detection describes the process of detection 'abnormal activities' in networks that extend beyond the network's acquired knowledge of normal activities. Adaobi and Ghassemian (2009) employed predefined classes, attributes, and rules that were determined from various resources like a set of classification rules, a training dataset, and inferred procedures and parameters (Kaur and Gill, 2013).

The ABIDSs have several advantages over the SBIDSs. The first advantage is the ability of ABIDS to detect accounts with stolen passwords or insider attacks. If the normal user or somebody employing stolen account(s) begins conducting activities outside of the profile of the normal user, then the ABIDS produces alarm. The intrusion alarm will be initiated according to customized profiles. Hence, it will be highly difficult for the attacker to certainly know what particular activity he/she can perform without raising an alarm. However, the main advantage of the intrusive activity does not rely on a distinct traffic which corresponds to a known intrusive activity like the case in the SBIDS. The second advantage of ABIDS it may probably detect an attack in its first time.

Elhamahmy, et al. (2010) and Chen, et al. (2009), amongst other researchers, who defined the principal network attacks as Probe, Remote to Local (R2L), DoS, and User to Root (U2R) attacks. These attack types are briefly explained next.

1. **Denial of Service (DoS) Attack**:

   Denial of Service (DoS) describes attacks whereby the attacker makes a memory or computing resource too busy or highly loaded to process legitimate requests, or prevents user's access to a machine, e.g., Smurf attack.

2. **User to Root (U2R) Attack**:

   This sort of attack is defined as a kind of exploiting whereby the attacker starts with accessing normal user's account that is realized by a dictionary attack, social engineering, or sniffing passwords, and, subsequently, utilizes vulnerability to gain root access to the attacked system, e.g., buffer overflow.

3. **Remote to Local (R2L) Attack**:

   The Remote to Local (R2L) sort of attack is defined as the attack which takes place when attacker having the capability to send packets to a machine through a network but not having account on this machine utilizes vulnerability to acquire local access as user of that machine, e.g., Spy.

4. **Probing Attack**:

   This kind of attack is defined as attempt to collect information about network of computers in order to circumvent its own security controls, e.g., Ports wee.

   Table (1.1) provides examples on each type of network attacks (Paliwal and Gupta, 2012).

Table (1.1). Attack Types and Examples

Attack Type	**Examples**
DoS	Land, Pod, Teardrop, Back, Neptune, and smurf.
Probe	Nmap, Ports wee, Ipsweep, and Satan.

R2L	Spy, Phf, Multihop, ftp_write, Imap, and Warezmas.
U2R	Buffer over flow, Rootkit, Loadmodul, and Perl.

1.4 Motivation

Cloud environment is an open environment that is exposed to high security threats to availability and to keeping its data and resources away from unauthorized access and modification. These security requirements are referred to as confidentiality, integrity, and availability. In view of this, some new clustering techniques have been developed in the biomedical field for clustering DNA using Fuzzy Clustering by Local Approximation Memberships (FLAME), which gives good results. These results constitute a motivation for implementing FLAME to gain good results in the IDSs, which will be integrated with a combination of other Artificial Intelligence (AI) techniques.

On the other hand, there is a concern about reducing the number of features to take into consideration by maintaining the (most) important features and neglecting insignificant features. According to the researcher's own results, this feature reduction leads to better results.

1.5 Problem Statement

The IDS is implemented to detect the attacks on, and intrusions of, the network. Then, it saves this as an intrusion in a file or trains the system to detect similar future incidences of this type of attack based on the abnormal activities on a network. Several previous studies reported the overall detection rates (DRs), negligence accuracies, and False Alarm Rates (FARs). Some researches highlighted that the FARs were high and many others reported the DRs for the DoS and PROBE attacks without mentioning the DRs of the U2R or R2L attacks also intrusion detection accuracy reported in some previous studies was low.

1.6 Contributions

The main contributions of this study are:

1- Performing feature filtration by using clustering method called FLAME.

2- Implementing the Hopfield Artificial Neural Network (HANN). The HANN was assigned to each group of data according to the type of attack(s) to which they are prone.

3- Performing aggregation by applying the Simulated Annealing (SA) technique so as to determine whether the incoming packet is an attack or a normal activity.

4- Improving the intrusion DR and ACC, and reducing the FAR.

1.7 Thesis Structure

This thesis focused on applying the FLAME combined with other AI methods, namely, HANN and SA, to optimize and enhance the intrusion detection results. The thesis is organized in five chapters as follows:

Chapter 2 is the ‘Research Background’. It explains the main concepts of IDS, ANN, FLAME, and SA.

Chapter 3 is the ‘Literature Review’. In this chapter, the researcher discusses the results of recent, related studies.

Chapter 4 is the ‘Methodology’. It explains the proposed method.

Chapter 5 is the ‘Experimental Results and Evaluation’. It presents, evaluates, and discusses the research findings. A comparison of the results of this study with results of related, published works is also provided in this chapter.

Chapter II

Research Background

2.1 Introduction

The spread of attack in unfriendly environments is alarming and requires use of intrusion detection techniques so as to detect attacks on the networks. The IDSs collect information about the systems being observed (Toosi and Kahani, 2007). An IDS developed by Zhang, et al. (2001) detects network-based attacks in the form of anomalies by means of statistical preprocessing and a neural network classification. In the present study, the researcher employed a hybrid intrusion detection technique to compliment the IDSs by using three algorithms: FLAME for feature filtration, HANN for training the system, and SA for aggregation. In the current study, experiments with the suggested algorithms were conducted on the KDD Cup 99 intrusion detection dataset.

2.2 Fuzzy clustering

The IDSs usually determine the attack sort or classify the anomalous activities in certain groups. The goal is to integrate various soft-computing approaches into a classifying system that can differentiate normal behavior from an intrusion on the basis of the kind of attack in computer network. Of all soft-computing models, the genetic algorithms (GAs), fuzzy inference, and the neuro-fuzzy networks are discussed in many previous studies.

Parallel neuro-fuzzy classifiers are employed to produce initial classification. Fuzzy inference systems are dependent on neuro-fuzzy classifier’s outputs to decide on whether the suspect activity is intrusive or normal. Lastly, for obtaining the finest result, the GAs optimize the structures of their fuzzy decision algorithms.

By using fuzzy clustering technique, the whole training set can be divided into subsets which have less size and lower complexity. The aim of fuzzy cluster module is to partition a given set of data into clusters, and it should have the following properties: homogeneity within the clusters, concerning data in the same cluster, and heterogeneity between clusters, where data belonging to different clusters should be as different as possible. Through fuzzy clustering module, the training set is clustered into several subsets. (wang, et.al 2010).

2.3 Fuzzy Clustering by Local Approximation Membership (FLAME)

Fu and Medico (2007) used FLAME as a clustering technique in Biology. They proposed a clustering algorithm that combines simplicity with good performance and strength of system. Moreover, they have proposed a new clustering algorithm from the starting point of the process that serves to appropriate a specific structure of the data from the start of the detection process.

The main process of FLAME is a clustering algorithm. Its particular processes are: (i) definition of every object's (sample or gene) neighborhood and determination of the objects having 'archetypal' characteristics, commonly referred to as Cluster-Supporting Objects (CSOs), so as to construct the clusters around which; and (ii) assigning a fuzzy membership vector for every single object. This vector is approximated by memberships of all neighboring objects through iterative-converging process wherein membership extends from the CSOs to their neighbors. In this regard, the FLAME function can handle very large datasets (Fu and Medico, 2007).

The FLAME clusters features into three main clusters; Inner, Outer, and Rest clusters.

1. Inner Cluster: The object with the highest density amongst all its neighbors.
2. Outer Cluster: The interception of an object's density with densities that have less than all of its neighbors' and less than predetermined threshold.
3. Rest Cluster: The object with a density that falls between the densities of the inner object and the outer object, close to the threshold.

The FLAME function differs from other clustering algorithms like the *c*-means and the *k*-means algorithms. As shown in Table (2.1), the *c*-means algorithm determines the Inner, Outer, and Rest clusters of the FLAME function by calculating the center for each cluster and dividing these clusters into ‘*c*’ clusters. The *k*-means algorithm divides the clusters into ‘*k*’ clusters.

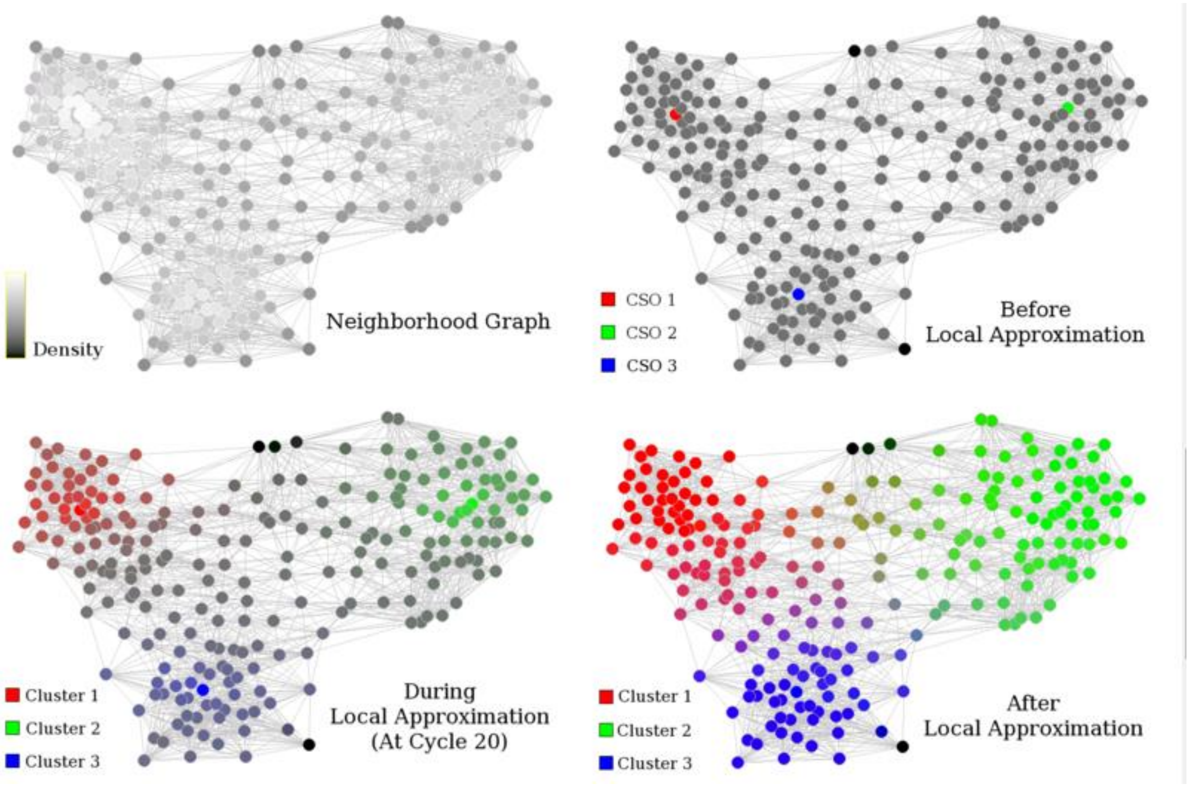

Figure (2.1). FLAME clustering algorithm.

2.4 Artificial Neural Networks (ANNs)

The Artificial Neural Networks (ANNs) are computational models that are inspired by the biological neural networks. They are employed to approximate functions and are suggested in view of the neurons’ behaviors and the electrical signals transmitted between the inputs (e.g., from the nerve endings in the hand or from the eyes), processing elements, and outputs of the brain (e.g., reacting to heat, light, or touch). Certain ANNs are in fact adaptive in nature and are employed, for example, to model environments and populations, which are ever changing. The neural networks can be software based (e.g., computer models) or hardware based like the neurons which are performed by physical components, and can employ a set of learning algorithms and topologies.

Figure (2.1) shows architecture of an ANN. The input units form the input layer. Each neuron’s output is the input for the following layer in the network, which comprises the interconnections between neurons and describes the network’s connectivity pattern. Each entry has a vector. When the entry is introduced to the network, the value of each vector in the corresponding neuron is copied. Afterwards, how each input unit receives the full vector information, processes it, and generates a single output is disseminated by connections between neurons. When the input has spread through the entire network, an output vector is produced whose components are generated by each of the input values of the output neurons.

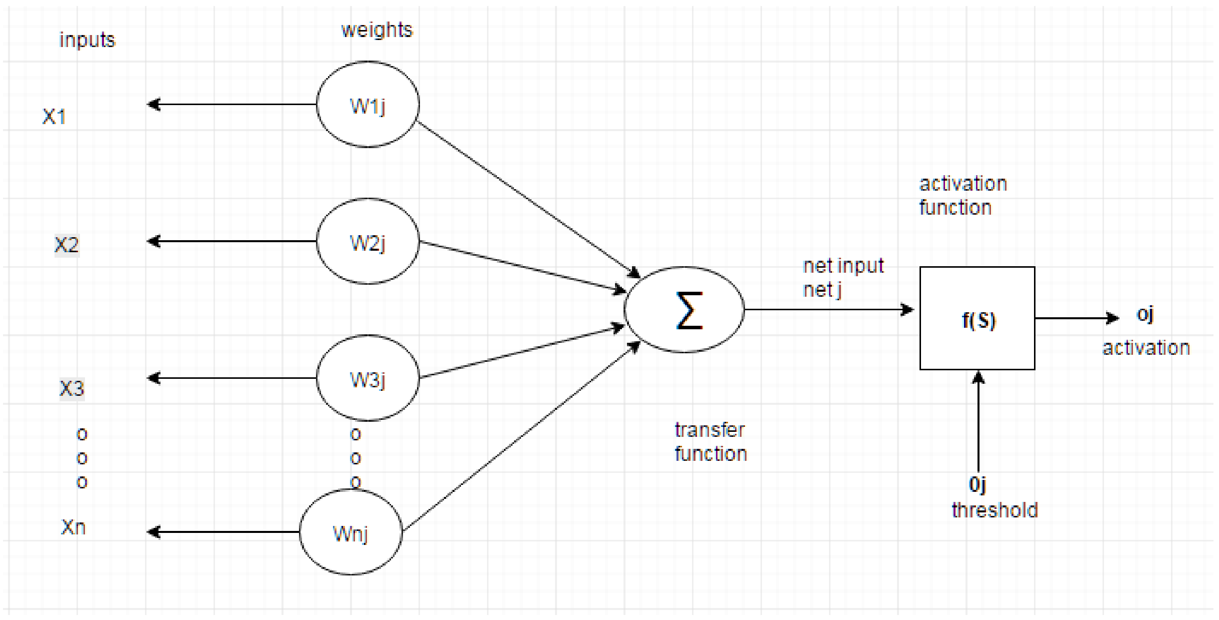

Figure (2.2). Architecture of ANN.

The ANNs are divided into five main types:

1. **Feed-forward neural networks**

   These are the simplest ANN type. In this kind of network, information travels in a single direction only. Data of the input nodes are passed to the hidden nodes first then to the output nodes. In other words, in this neural network type, there are no loops or cycles.

2. **Radial basis function (RBF) network**

The Radial basis function (RBF) is a function incorporating a distance measure with reference to specific center. It is a type of the three-layer, feed-forward neural networks that is made up of input layer with *n* nodes, a hidden layer of neurons, and an output layer comprising a single node. The RBF is applied as a hidden neuron in this sort of ANNs (Tiwari, et al., 2013).

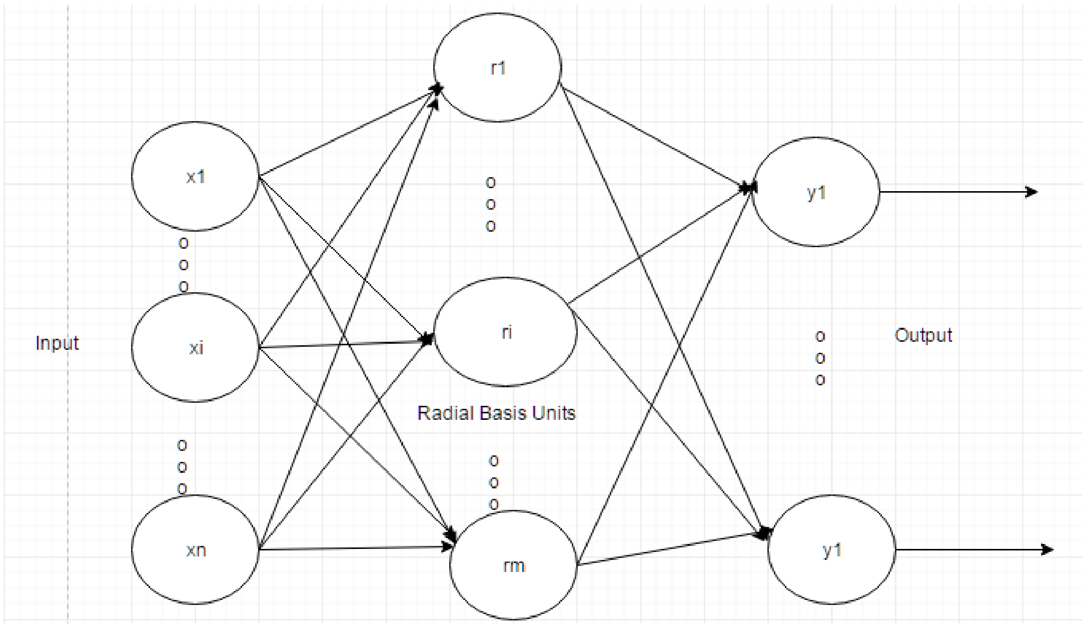

Figure (2.3): Architecture of the RBF Neural Network

3. **Recurrent neural network (RNN)**

These networks are models with a bi-directional data stream. Whereas the feed-forward network does linearly propagate the data from the inputs to the output, the RNNs propagate the data from the latest to the early processing stages. They may be utilized as general-sequence processors. One of the examples of this type of neural networks is the HANN.

4. **Modular neural networks**

   In this network type, many small networks compete or cooperate in order to resolve the problem of concern.

5. **Physical neural networks**

   The physical neural network encircles electrically-adjustable resistant material so as to simulate the artificial synapses.

2.5 Hopfield Artificial Neural Network (HANN)

The ANNs perform better, mainly with noisy data. They demand huge data for training and it is usually difficult to choose the optimum architecture for a neural network. Therefore, this study employed the SA technique to choose the best data from the entire cluster, followed by the HANN.

The HANN is a type of RNN developed in 1982 by John Hopfield . The Hopfield nets stand as content-addressable memory systems with binary threshold nodes. They are warranted to converge to local minimum. However, convergence to some false pattern (faulty local minimum) instead of the stored pattern (the anticipated local minimum) may take place. Moreover, the Hopfield networks present a model for understanding the human memory, even though Hopfield based this network on the RNNs, which are actually bi-directional data flow models, whereas the feed-forward ANN linearly transmits the data to the output layer from the input layer.

The units in the Hopfield nets are binary threshold units, that is, they assume two values only for their states and the particular value is specified based on whether the unit's input does, or does not, surpass its threshold. Normally, the Hopfield nets possess units which assume the values of -1 or 1, even though some studies in the literature used units that take the values of 0 and 1. In a Hopfield network, every pair of units *j* and *i* has connection, which is described by connectivity weight, w_{ij}. Thus, the Hopfield network may be described officially as a perfect, undirected graph G=<V,f>, where V is some set of Mc Culloch-Pitts neurons and f: $V^2 \rightarrow R_f$ is a function which connects the node pairs to a true value, namely, the connectivity weight. Typically, the Hopfield net connections have the following two constraints:

$w_{ii} = 0, \forall_i$ (no unit has a connection with itself), and

$W_{ij} = W_{ij}, \forall_i, j$ (connections are symmetric)

The requirement that weights be symmetric is typically used as it will guarantee that (i) the energy function declines in a monotonic manner while abiding by the activation rules, and (ii) the network may demonstrate chaotic or periodic behavior if non-symmetric weights are employed. Nonetheless, Hopfield discovered that the chaotic behavior is restricted to comparatively small sections of the phase space and that it does not deteriorate network's capability to function as a system of content-addressable associative memory.

Updating a unit (i.e., a node in the plot that simulates the artificial neuron) in the Hopfield network is realized according to the rule:

$$S_i \leftarrow \begin{cases} \text{'1'} & \text{if } \sum j \, Wij \, Sj \;>=\; \theta i, \\ '-1' & \text{otherwise.} \end{cases}$$

where W_{ij} is strength of the connection weight from j to i; S_j is state of unit j, and θ_i is threshold of unit i.

Weight updating in Hopfield networks may be implemented in either of a couple of ways:

1- **Asynchronous updating.** One unit only is updated at a time. The unit to update my be randomly picked, or taken in a sequence that may be set at the beginning.

2- **Synchronous updating.** Units are updated simultaneously. This demands central clock for the system so as to sustain synchronization. This updating technique is less realistic because the physical or biological systems do not have global clock keeping the track of time.

The weight between two units has high influence on the values of neurons. If the connection weight, w_{ij}, between two neurons i and j is greater than 0, then the updating rule entails that when $S_j = 1$, neuron j has positive contribution to the weighted sum. Consequently, S_i is drawn by j in the direction of its value, namely, $Sj = 1$. However, when $Sj = -1$, then j has negative contribution to the weighted sum. Then, Sj is again drawn by j in the direction of its value $S_j = -1$. In consequence, the values of j and i converge when the interconnecting weight is positive and diverge when the interconnecting weight is negative.

Numerous learning rules may be employed to save the information in the Hopfield Network's memory. However, it is desired for the learning rule to be local and incremental. The learning rule is 'Local' when every weight is updated by using the information available for the neurons on any connection side concomitant to that specific weight. On the other hand, the learning rule is 'Incremental' when the new patterns can be learnt without use of information from old patterns, which too were used in training. In other words, when new pattern is utilized in training, values of the new weight are dependent on the new pattern(s) and the old values. These two properties are preferable owing to that a learning rule that satisfies both properties is biologically more possible. For instance, as the human brain is ever learning new ideas, it may be reasoned that human learning is actually cumulative. In general, whichever learning system which is not accunmulative in nature can be trained once only, using large batches of training data. Two of the most common learning rules for the HANN are the Hebbian and the Storkey learning rules. A description of the two follows.

1- The Hebbian Learning Rule for the Hopfield Networks

The Hebbian Theory was presented in 1949 by Donald Hebb for explaining the associative learning wherein concurrent activation of the neuron cells brings forth profound increments in the synaptic strength between those cells. In effect, the Hebbian rule is both incremental and local. When learning N binary patterns in the Hopfield Networks, the Hebbian rule proceeds as follows:

$W_i^j = 1/n \sum^n_{M=1} €_i^M €_j^M$, where $€_i^M$ refers to bit i in pattern M.

If the bits that correspond to the neurons j and i are identical in the pattern M, then the associated product $€_i^M \, €_j^M$ is indeed positive. This will, in consequence, have positive influence on the weight, W_i^j, and the values of j and i tend to be identical. The contrary takes place when the bits of neurons j and i are different.

2- The Storkey Learning Rule for the Hopfield Networks

The Storkey learning rule, which is both incremental and local, was introduced in 1997 by Amos Storkey who demonstrated that a Hopfield network that is trained following this rule will have higher capacity than a respective network that is trained following the Hebbian rule. An attractor neural network's weight matrix is judged to be following the learning rule of Storkey when it adheres to:

$$W^v_{ij} = W^{v-1}_{ij} + 1/n\,[(€_i^v €_j^v) - (€_i^v h_{ji}^v) - (€_i^M h_{ij}^v)]$$

where $h_{ij}^v = \sum_{k=1, k=!i,j}^{n} W^{v-1}_{ik} €_k^v$ is a local field form at the neuron i.

This learning rule is local because synapses consider the neurons at their sides only. This rule utilizes further information from weights and patterns than the generalized Hebbian rule because of the the local field's impact.

The neural networks are considered as effective tools for classifying patterns. They are also used to detect anomalous users' activities. However, the long training cycles have prevented their processes.

2.6 Simulated Annealing

Simulated annealing (SA) is a general probabilistic metaheuristic for the global optimization problem of finding reasonable approximation of the global optimum of provided outcomes in broad pace of search results. It is frequently employed with discrete search spacees (e.g., the tours visiting given number of cities). For specific problems, the SA can be more efficient than the exhaustive enumeration on the condition that the objective is only finding some good, acceptable solution in fixed period of time rather than the finest probable solution.

The SA annealing algorithm is a type of a heuristic searching method. The natural technique and relevant random factors of the physical SA process were introduced to this algorithm. The SA algorithm can produce fast convergence and high ACC. This algorithm proved to be effective global optimum algorithm. However, its computation speed is high as it entails lots of iterations. Owing to that SA is a sort of heuristic random searching technique, it jumps out of the local minimum and gets a lower minimum that is reputed as a better superiority point (Gao and Tian, 2009). The ability of the SA algorithm to jump out of the local optimum solution, integrated with fuzzy *c*-means clustering, is used first so as to achieve a global optimum clustering. Normal and anomalous data are hence identified by a normal cluster ratio. The SA algorithm improves the capability for searching for the global optimum solution and it can, thus, be utilized in anomaly intrusion detection (AID). Experiments using the KDD CUP 99 data set confirmed that this method can generate a low false positive rate (FPR) and high DRs (Jian and Rui, 2009).

The SA algorithm is an iterative heuristic algorithm that realizes the global optimum solution of the optimization problem or approximates it by means of a simulation of the annealing process of high-temperature objects. When optimizing a problem using the SA algorithm, the starting temperature is first chosen to be 0. Then, an initial solution, X_o, is picked up randomly to serve as the present solution for calculating the objective function value, J_o, of the present solution. Afterwards, a new solution, x_n, in the domain of the present solution is produced and the objective function value, J_n, is to be computed. It will then be judged whether the new solution is the present solution or not following the Metropolis principle. Specifically, when $J_n \leq J_o$, then x_n is accepted as the present solution. However, when $Jn > Jo$, then, some random figure, R, in [0, 1] is provided to compute the probability of acceptance, $P = \exp((J_o-J_n)/t_k)$, where t_k is the current annealing temperature. If $P \geq R$, then x_n is the present solution, or x_o is taken as the present solution in order to iterate repeatedly. When the iterations number approaches length of the Markov chain, the annealing temperature is decreased and the acceptance probability becomes lower. The iterative process of 'generating a new solution- >judging >accepting/abandoning' is repeatedly implemented in order to satisfy the termination requirement. The solution obtained from the SA algorithm is non-related to state of the starting solution. Additionally, it may iterate according to the direction of decrease of the objective function and may accept an objective function growing under specific probability. This is the fundamental difference between local-searching algorithms and the SA algorithms that makes SA skip the local minimal point (Jian and Rui, 2009).

Chapter III

Related Works

3.1 Introduction

The different researchers employed various AI techniques for intrusion detection. These include ANNs, fuzzy logic, and GAs for the detection of network attacks, particularly in the cloud environment which is regarded as intruders' preferred target for exploitation of the weak points in this environment.

3.2 IDS with ANN

Zhang, et al. (2001) employed a neural network classifier and analysis of the stimulus vector from the statistical model so as to ultimnately decide if network traffic is, or is not, normal. They noticed that the lately-arriving events were at first saved in the event buffer. Afterwards, the saved events were compared with the reference model of the related layer and the comparison outcomes were supplied to the neural network classifier in order to suggest the status of the network during that time window.

Al Sharafat (2010) developed a model for an IDS consisting of two phases. In the first phase, a feature filtration process is implemented using the ANN so as to choose the best set of features for every potential kind of network attacks. In the second phase, an IDS is designed using an Extended Classifier System (XCS) with interior modification for the classifier generator to have a better DR. The proposed IDS can efficiently and effectively detect attacks. The R2L and U2R types of attacks were both detected at low rates. This may be due to the small number of records that need to be allocated for these attacks in the KDD '99 dataset and in the test set as well. The ANN-XCS model's DR was 98.01% and the FPR was 0.9%.

Selman (2011) proposed an IDS model on the basis of Fuzzy Logic and which comprises three elements: an Input Reduction System (IRS) that employs Principal Component Analysis and decreases the inputs' number from 41 to 13; a Classification System that employs the Fuzzy *c*-Means algorithm to produce data clusters from training data; and a

Pattern Recognition System founded on the Nearest Neighbor technique, which classifies the new data records according to their expert clusters. Depending on the various sorts of attacks, the classification performance of the system varies and the optimum performance (a success rate of 99.3%) was associated with the PROBE attack. Meantime, the best performance in pattern recognition (a success rate of 58.8%) was concomitant with the U2R.

Gaikwad, et al. (2012) developed an intrusion detection approach, referred to as FC-ANN, on the basis of ANN and fuzzy clustering. The fuzzy clustering method uses the *c*-means algorithm and the heterogeneous training set is partitioned into many homogenous subsets. In consequence, incorporation of the training subsets is lowered and detection performance is improved. By so doing, the system grows highly stable and efficient, and it overcomes, in an efficient matter, the impediments of low detection accuracy and weak detection stability.

Tiwari, et al. (2013) developed a hybrid model for an IDS using a combination of the Firefly Algorithm for feature selection, RBF Neural Network, and the Rough Set Theory as a tool for intelligent data mining and analysis. The KDD '99 dataset was used with 32 features and four types of attacks. Eventually, the DRs of DoS, Prob, R2L, and U2R attacks were 0.99, 0.98, 0.97, and 0.95, respectively.

Khazaee and Faez (2014) suggested a new technique of traffic classification and intrusion detection using hybrid fuzzy clustering and ANNs. This approach has remarkable performance in the sense of recall, precision, DR, the *f*-value, ROC curve analysis, ACC, and FAR than other proposed approaches. After preprocessing and the fuzzy splitting, the training dataset is split into a couple of subsets: (i) a subset which contains the suitable samples for training and (ii) a subset that contains the outliers (this particular subset is called 'Removed-Dataset'). The outlier data are not deleted but rather used in a different manner during the training phase. Accuracy of this new approach to traffic classification and intrusion detection was 99.5% and the FAR was 0.45%.

Dominges and Martin (2015) developed a robust technique that allows for some effective and intelligent intrusion detection which is suitable for the constantly-changing personal behaviors. They developed an analysis technique and analyzed it using the ANN, which can be used for pattern recognition, clustering, and classification. The ANNs were employed to generate a dynamic profile for the user that behaves as a differentiated individual for access to information systems so as to eventually get better intrusion detection.

3.3 IDS with HANN

Jabez and Muthukumar (2007) developed and applied a Knowledge-Based AID framework that was founded on the Hyperbolic Hopfield Neural Network with anonalies trained by data tested on the KDD '99 network dataset. The Hyprolic Hofeild Neural Network proved to have greater efficiency than the GAs and the Fuzzy Neural Network and had a DR of 95.0%.

3.4 IDS with Fuzzy Clustering

Usually, the IDSs determine the kind of attack or classify the inspected activities in certain, distinct categories. The main aim of the work of Toosi and Kahani (2007) was to integrate various soft-computing methods into a classifying system in order to ultimately distinguish the normal behaviors from intrusions on the basis of sort of attack in computer networks. Of all the investigated soft-computing paradigms, the GAs, fuzzy inference, and neuro-fuzzy network approaches were investigated. Several parallel neuro-fuzzy classifiers were employed to perform preliminary classification. Then, the fuzzy inference system then utilizes the outcomes of the neuro-fuzzy classifiers to decide if the activity under investigation is intrusive or normal. Lastly, so as to obtain the finest possible results, the GA optimizes the

the fuzzy decision engine's structure. Performance of this method was tested on the KDD CUP 99 data.

Wang, et al. (2010) suggested novel method, known as FC-ANN, on the basis of ANN and fuzzy clustering, to produce low FPR and high DR. This approach employs the *c*-means fuzzy clustering method first. Each data subset is allotted to independent ANN model in order to produce differing training subsets. The various generated training subsets, according to varied ANN models, are then trained to create various base models. Afterwards, these researchers used fuzzy aggregation so as to aggregate the outcomes. The optimum reported mean precision of this approach was 96.7% when the number of clusters (*k*) was 6.

Shanmugavadivu and Nagarajan (2013) designed anomaly-based IDS for detection of intrusion behavior in networks. They developed fuzzy decision-making module to create a highly-accurate system for the purpose of attack detection on the basis of the fuzzy inference method. Firstly, the specific rules were produced through mining single-length frequent items from the attack and the normal data.The relevant fuzzy rules were then identified via fuzzifying the distinct rules, which were then introduced to the fuzzy system that classifies the test data. An analysis of the detection results indicated that the overall performance of this proposed system was improved greatly; the system achieved greater than 90% detection accuracy for all sorts of attacks.

3.5 IDS with Supported Vector Machine (SVM)

Shrivastava and Jain (2011) proposed a model for improving AID. The model was intended to obtain a high DR and a low FPR based on using Rough Set, which reduces the data features, and Support Vector Machine (SVM) to train and test the data. The proposed model only uses 6 features out of 41 features. The model reduced the Central Processing Unit (CPU) and memory utilization of the system and enhanced trust in intrusion detection. The ACC of the proposed system was 95.98 % and it had FPR of 7.52%.

3.6 IDS with GA

Panja, et al. (2014) presented a highly-accurate classification of traffic network using GAs and Adaptive Neuro Fuzzy Inference System (ANFIS) to enhance data classification and obtain good classification. The GA applies genetic operators like selection, mutation, and crossover on the present population in order to spawn identical patterns to be utilized over and over again till a certain stopping measure is satisfied. Initially, the proposed method had a DR of 92.74%. This rate increased in the second run to 94.87%.

3.7 IDS with Simulated Annealing

Jian and Rui (2009) illustrated the capability of the SA algorithm to jump out of the local optimum solution, in combination with fuzzy *c*-means clustering, which is first used to achieve global optimal clustering. Anomalous and normal data are both specified by a normal cluster ratio. Afterwards, these clusters may be employed to detect any intruding activities. Experimentation with the KDD CUP 99 dataset pointed out that this method has better intrusion detection than the traditional fuzzy *c*-means algorithm. This SA-FCM method had a DR of 95.3% and an FPR of 10.5% on the testing set.

Lin, et al. (2012) provided evidence on improvement of AID. Their proposed method consists of feature selection for AID as a first step. The second step is detection of new attacks by the decision rules obtained from the dataset. The AID methodology proposed by these researchers was tested on the KDD '99 dataset. Both the SVM and SA methods were applied to determine the finest subsets of features. The SA and Decision Trees proved their efficiency in producing decision rules for detection of new attacks. In this method, the variables of the Decision Trees and SVM are automatically obtained and the classification ACC for the testing set was 99.96%.

3.8 FLAME

Fu and Medico (2007) used FLAME as a clustering technique in Biology. They proposed a clustering algorithm that combines simplicity with good performance and strength of system. As well, they discovered a new clustering algorithm from the starting point of the process that helps in finding a specific appropriate structure of the data set from the beginning of the process.

The FLAME function is a clustering algorithm. Its characteristic steps are: (i) identifying neighborhood of every object (sample or gene) and specifying the objects with archetypal features (that is, the CSOs), in order to construct the clusters around which; and (ii) assigning fuzzy membership vector to every object. The fuzzy membership vector is approximated from the memberships of neighboring objects by a process of iterative convergence whereby the membership extends from the CSOs through their neighbors. The FLAME can handle very large datasets (Fu and Medico, 2007).

Chapter IV

Methodology

4.1 Introduction

This chapter presents an adjustment for IDS by adapting several AI algorithms to obtain an improved DR and a low FAR. For this purpose, a network-based IDS was implemented according to the source of data in the cloud environment, taking advantage of the knowledge of the normal behaviour of network environment users. The proposed technique consists of a set of stages that start from grouping the data until an aggregation stage which is concerned with providing a final decision on whether the incoming data is an attack or not. The following sections provide a description for each stage.

4.2 System Stages

To sum up, Figure 4.1 displays the proposed system's architecture.

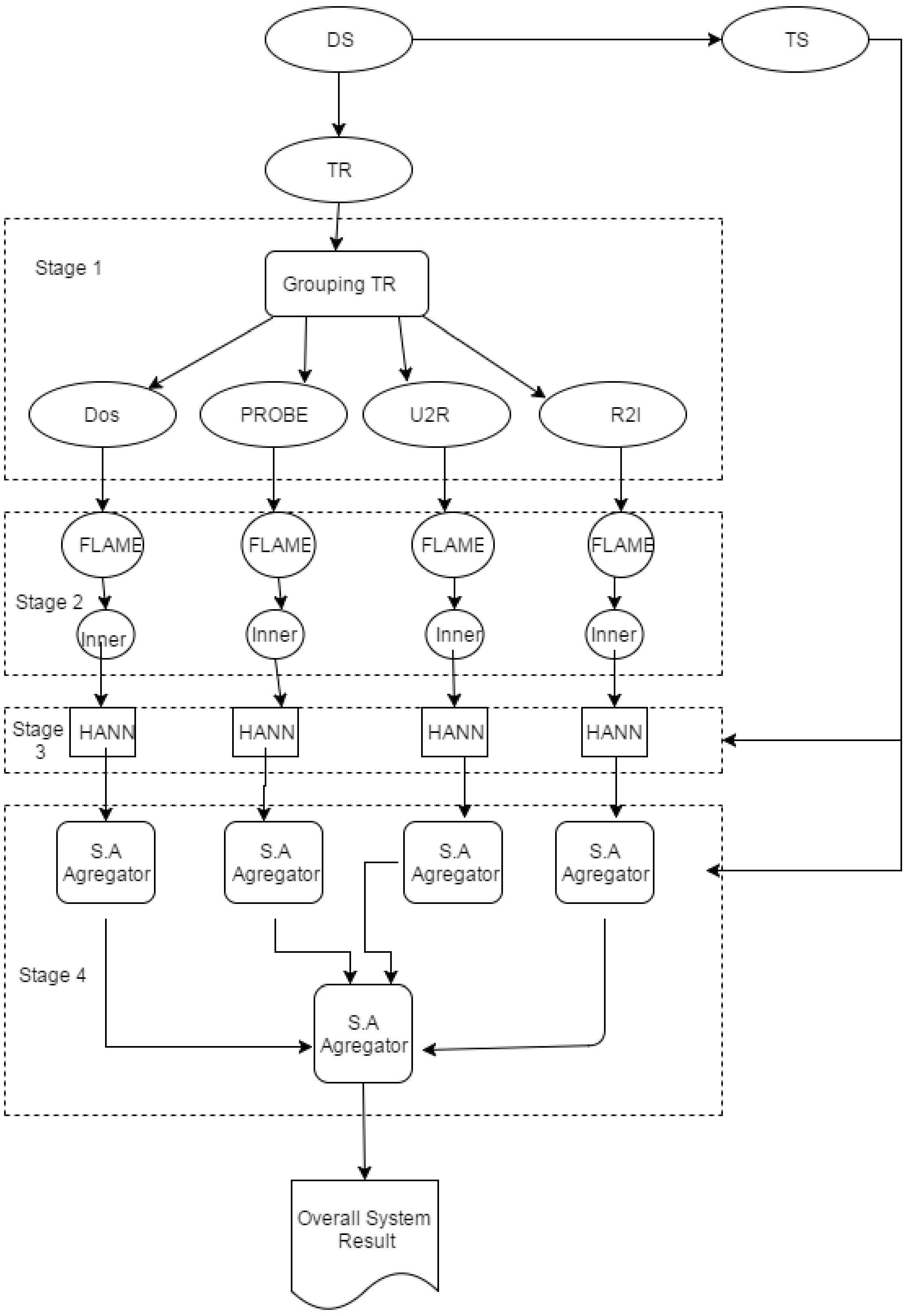

Figure (4.1). Architecture of the Suggested IDS.

4.2.1 Stage One: Grouping the Data

The first stage in the method proposed in this study is grouping the data, which is a step that aims for grouping the data into four clusters using the Weka tool according to the type of attack; Probe, DoS, R2L, and U2R. Figure 4.2 and Figure 4.3 present the grouping dataset classifier pseudocode. The groping stage helps in determining the significant features of each type of attack out of 41 network features. Achievement of this target will be reflected by an improved DR and a reduced FAR.

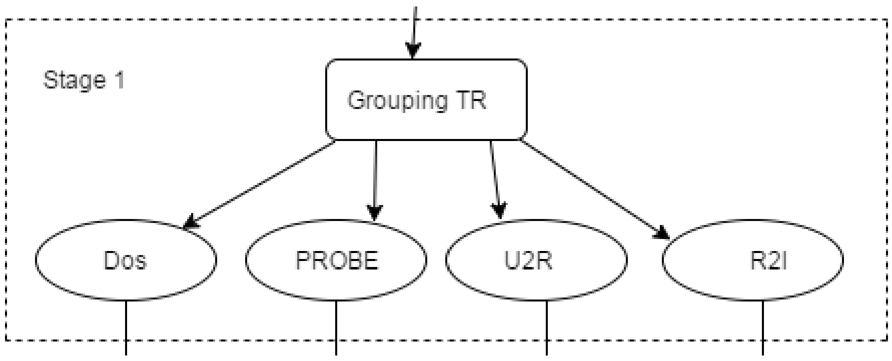

Figure (4.2). Stage one of the Proposed IDS: Grouping the Data.

```
Classifier :

  Classify Data set based on attack type (using Weka tool)

    [BACK, LAND , NEPTUNE , POD , SMURF] --> DoS

          [BUFFER_OVERFLOW , LOADMODULE , PERL , ROOTKIT] --> U2R

          [FTP_WRITE , GUESS_PASSWD , IMAP , MULTHOP , PHF , SPY , WAREZ_CLIENT ,
WAREZ_MASTER] --> L2R

    [SATAN , IPSWEEP , NMAP , PORT_SWEEP] --> PROBE
```

Figure (4.3). Classifier Pseudocode for Data Grouping.

4.2.2 Stage Two: Feature Filtration

The FLAME methodology is applied for feature filtration by reducing the number of features to less than 41, and, in consequence, get a highly-accurate system with a high DR.

Each type of attack (DoS, Probe, U2R, and R2L) has its own features, which may intersect and some of them may be shared in common by the different types of attacks.

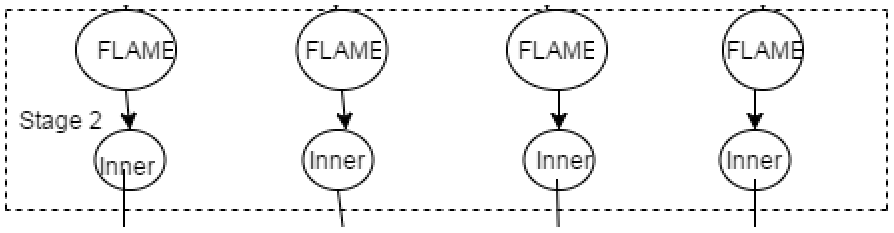

Figure (4.4). Stage II of the Proposed IDS: Feature Filtration.

The FLAME is algorithm for data-clustering. It defines the clusters and implements cluster assignment according to the neighborhood relations amongst the various objects. The main characteristic of the FLAME is that every cluster has sub-clusters (Inner, Outer, and Rest clusters). The distinct data of a cluster are saved in the Inner sub-cluster; either the data do, or do not, belong to the cluster saved in the Outer sub-cluster. The remaining data are then saved in the Rest sub-cluster. Fu and Medico (2007) clarified that the purpose of data clustering is to extract information from the expression of data. The FLAME clustering algorithm defines the elements of the data introduced to it following eight steps:

Step 1: Determine the data element which needs to be clustered.

Step 2: Fill out the data elements in a list of vectors.

Step 3: Compute the point weights for the data using the density concept (high values indicate low densities while small values point out to high densities).

Step 4: Define the outlier(s) and the CSO point (clustering) on the basis of the density values. If a density score is less than those of all neighbors, then the point is CSO, otherwise, it is an outlier.

Step 5: Remove the CSOs that are close to the outliers.

Step 6: Transform the cluster counts so as to indicate the new cluster ID.

Step 7: Go through the points and remove any clusters with a count of 1.

Step 8: Get the clustering results using HANN to determine the extent to which this network learnt the examples (current network, test network)

After grouping the training data into types of attack, the FLAME function starts extracting information from the data and divides it into the four attack kinds (Probe, DoS, R2L, and U2R). Afterwards, by using KNN for estimating the density for each cluster, each type of attack has three types of sub-clusters (Inner, Outer, and Rest sub-clusters). The Inner cluster has the highest density and contains the most relevant features that give indications of a specific type of attack. The Outer cluster contains features that do not give indications of the same type of attack, and their features may belong to Inner clusters in a different attack type that has the highest membership. Meanwhile, the features may intersect with different clusters. By programming the system.

4.2.3 Stage Three: Network Attack Detection Using HANN

After the feature-filtering stage, the detection stage starts by applying HANN, taking into account internal tuning of the internal weights of the HANN for neurons. By assigning HANN for each group of data, e.g., DoS to detect DoS attacks and predict intrusion, each HANN gives a result.

The Hopfield networks normally have nodes that assume the values of 0 or 1 to give the result as a binary variable. The weights which the HANN usually take range from 0 to 1. When the program is running; it will store the connections in an array (content addressable memory). The aim of this action is to allow each neuron to be connected to every other neuron. The connection weights added to this array, also referred to as the weight matrix, allow the neural network to call back specific patterns when they are presented.

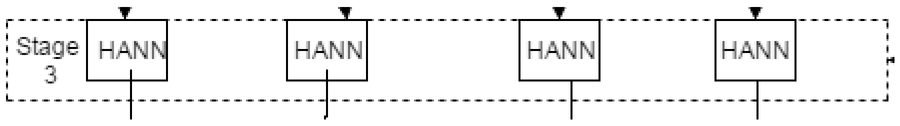

Figure (4.5). Stage III of the Suggested IDS: Training.

By the end of this stage, there will be intermediate detection results. As previously mentioned, four HANN's HANNs are assigned to every type of attack and the four outputs will pass through an aggregation stage to find out if the incoming network record is, or is not, an attack. Figure 4.6 shows the HANN pseudocode.

```
//Calculate weight for each DataSetRow
for (double key : values.keySet())
  weight += (neoWeight * key * values.get(key));

//Calculate ACC equation regarding attack type (PROBES , DoS , U2R , L2R)
ACC = ((double) weight / testSet.getRows().size()) * 100;

//Calculate Falarm Rate equation regarding attack type (PROBES , DoS , U2R
, L2R)
FalarmRate = 100 * ((double) (testSet.getRows().size() - sum) /
testSet.getRows().size());

//Calculate Detection Rate equation regarding attack type (PROBES , DoS ,
U2R , L2R)
DetectionRate = 100 * ((double) sum / testSet.size());
```

Figure (4.6). Pseudocode of the HANN.

4.2.4 Stage Four: Aggregation

The aggregation stage starts with taking inputs from the HANN, which are allotted to each attack sort, to provide final decision on whether the incoming packet is, or is not, an attack, and to reduce the detection errors. In this aggregation stage, the SA algorithm was applied (Figure (4.7)).

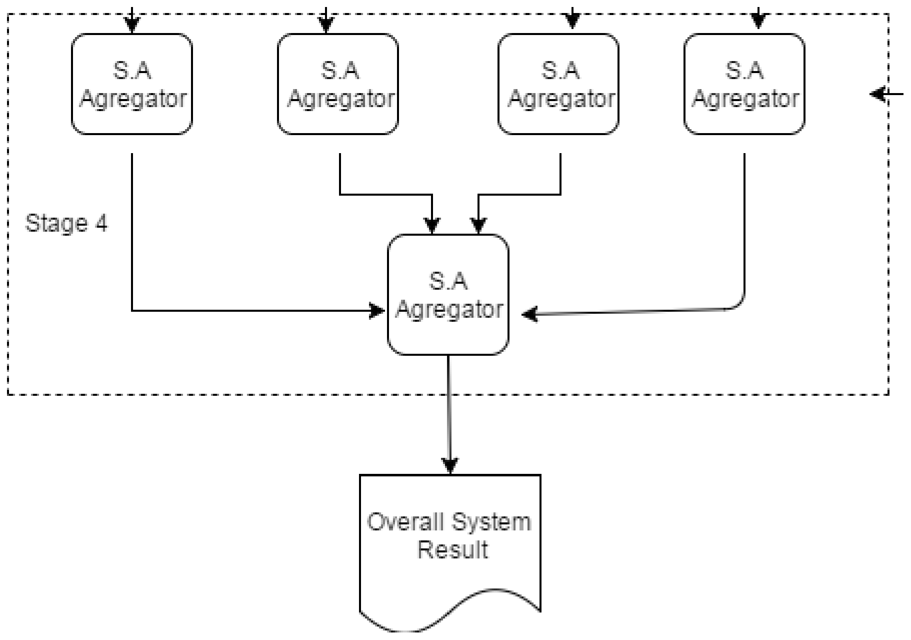

Figure (4.7). Stage IV of the Proposed IDS: Aggregation.

The global optimum solution for a combinatorial optimization problem can be obtained. As well, the best results from the HANN clusters can be obtained. When optimizing by the algorithm of SA, the preliminary value is chosen as 0. Then, the initial solution is selected such that it is the present solution. This solution is specified as follows: if the current > the last, then the value = the current (Figure (4.8)).

```
                    if (finalProbesAcc < probesAcc && finalProbesDetectionRate <
probesDetectionRate) { finalProbesAcc = probesAcc;
                          finalProbesDetectionRate = probesDetectionRate;
                          finalProbesFalarmRate = probesFalarmRate;}}
if (finalDosAcc < dosAcc && finalDosDetectionRate < dosDetectionRate) {
                          finalDosAcc = dosAcc;
                          finalDosDetectionRate = dosDetectionRate;
                          finalDosFalarmRate = dosFalarmRate;}
             }      if (finalU2RAcc < u2RAcc && finalU2RDetectionRate < u2RDetectionRate) {
                          finalU2RAcc = u2RAcc;
                          finalU2RDetectionRate = u2RDetectionRate;
```

```
            finalU2RFalarmRate = u2RFalarmRate;}}
        if (finalL2RAcc < l2RAcc && finalU2RDetectionRate < l2RDetectionRate) {
            finalL2RAcc = l2RAcc;
            finalL2RDetectionRate = l2RDetectionRate;
            finalL2RFalarmRate = l2RFalarmRate;}
```

Figure (4.8). Pseudocode of the S.A.

4.3 Implementations and Tools

Some tools were used with JAVA code to enhance the work such as Weka tool for grouping dataset and learn tool for learning the system in HANN.

4.3.1 Weka Tool

The Waikato Environment for Knowledge Analysis (Weka) is a comprehensive suite of Java class libraries that implement many state-of-the-art machine learning and data mining algorithms. Weka is freely available on the World-Wide Web.

The primary learning methods in Weka are "classifiers", and they induce a rule set or decision tree that models the data. Weka also includes algorithms for learning association rules and clustering data. All implementations have a uniform command-line interface. A common evaluation module measures the relative performance of several learning algorithms over a given data set (witten,et.al, 1999).

4.3.2 Learn Tool

Neuroph is an object-oriented neural network framework written in Java. It can be used to create and train neural networks in Java programs. Neuroph provides Java class library as well as GUI tool easyNeurons for creating and training neural networks.

Neuroph's core classes correspond to basic neural network concepts like artificial neuron, neuron layer, neuron connections, weight, transfer function, input function, learning rule etc. Neuroph supports common neural network architectures such as Multilayer perceptron with Backpropagation, Kohonen and Hopfield networks. All these classes can be extended and customized to create custom neural networks and learning rules. Neuroph has built-in support for image recognition (El Farissi, et.al, 2012).

To sum up, Figure 4.9 provides the system's pseudocode.

```
Step A (Classifier):

START

Input → (training dataset(500000 record))

Classify the data based on attack type using Weka tool.
    [BACK, LAND , NEPTUNE , POD , SMURF] --> DoS
            [BUFFER_OVERFLOW , LOADMODULE , PERL , ROOTKIT] --> U2R
            [FTP_WRITE , GUESS_PASSWD , IMAP , MULTHOP , PHF , SPY , WAREZ_CLIENT ,
WAREZ_MASTER] --> L2R
    [SATAN , IPSWEEP , NMAP , PORT_SWEEP] --> PROBE

Output →(Four clusters ( DoS, PROBE, U2R, R2L)).

Step B (Feature Filtration Using FLAME):

Input → (Four clusters ( DoS, PROBE, U2R, R2L)).
Step 1: Determine the data element which needs to be clustered
Step 2: Fill out the data elements in a list of vectors
Step 3: Compute the point weights for the data by using the density concept (high values indicate low densities while small values point out to high densities)
 Step 4: Define the outlier(s) and the CSO point (clustering) on the basis of the density values. If a density score is less than those of all neighbors, then the point is CSO, otherwise, it is an outlier.
Step 5: Remove the CSOs that are close to the outliers
Step 6: Transform the cluster counts so as to indicate the new cluster ID
Step 7: Go through the points and remove any clusters with a count of 1
Step 8: Get the clustering results using ANN to determine how the extent to which this network learnt the examples (current network, test network)
Output →(Four Inner clusters ( DoS (inner cluster), PROBE (inner cluster), U2R (inner cluster), R2L (inner cluster))).

Step C: (HANN):

Input →(Four Inner clusters ( DoS (inner cluster), PROBE (inner cluster), U2R (inner cluster), R2L (inner cluster))).
Step 1: Calculate weight for each data row
for (double key : values.keySet())
  weight += (neoWeight * key * values.get(key));

Step 2: Calculate ACC equation for each attack type (PROBES , DoS , U2R , L2R)
ACC = ((double) weight / testSet.getRows().size()) * 100;

Step 3: Calculate Falarm Rate equation for each attack type (PROBES , DoS , U2R , L2R)
FalarmRate = 100 * ((double) (testSet.getRows().size() - sum) / testSet.getRows().size());
```

Step 4: Calculate the DR equation for each attack type (PROBES , DoS , U2R , L2R)
DetectionRate = 100 * ((double) sum / testSet.size());
Output →(Multiple result for each type of attack cluster).

<u>Step D (S.A):</u>
Input→(Multiple result for each type of attack cluster).

```
If (finalProbesAcc < probesAcc && finalProbesDetectionRate < probesDetectionRate) {
finalProbesAcc = probesAcc;
                                        finalProbesDetectionRate = probesDetectionRate;
                                        finalProbesFalarmRate = probesFalarmRate;}}
If (finalDosAcc < dosAcc && finalDosDetectionRate < dosDetectionRate) {
                                        finalDosAcc = dosAcc;
                                        finalDosDetectionRate = dosDetectionRate;
                                        finalDosFalarmRate = dosFalarmRate;}
                    }          If (finalU2RAcc < u2RAcc && finalU2RDetectionRate < u2RDetectionRate)
{
                                        finalU2RAcc = u2RAcc;
                                        finalU2RDetectionRate = u2RDetectionRate;
                                        finalU2RFalarmRate = u2RFalarmRate;}}
                              If (finalL2RAcc < l2RAcc && finalU2RDetectionRate < l2RDetectionRate) {
                                        finalL2RAcc = l2RAcc;
                                        finalL2RDetectionRate = l2RDetectionRate;
                                        finalL2RFalarmRate = l2RFalarmRate;}
```

Final Output →(Single DoS result → DR, ACC, FAR)

(Single Probe result → DR, ACC, FAR)

(Single U2R result → DR, ACC, FAR)

(Single R2L result → DR, ACC, FAR)

(Single Overall system result → DR, ACC, FAR)

Main→

```
{
training_set.Grouping();
probeFlame.start();
dosFlame.start();
u2rFlame.start();
r2lFlame.start();
applyANN(Probesclusters,probes);
applyANN(Dosclusters,Dos);
applyANN(u2rclusters,u2r);

applyANN(r2lclusters,l2r);

Probe.SA();

Dos.SA();

U2r.SA();

R2l.SA();

}

End;
```

Figure (4.9). Pseudocode of the Proposed IDS.

Chapter V

Experimental Results and Evaluation

5.1 Introduction

This chapter offers the experimental results and evaluates the proposed technique. The researcher holds comparisons in this chapter with relevant works that focused on IDS in the CC environment. All the experiments were conducted on Acer desktop computers with a core i7 CPU, 6 GB of RAM, and Hard Disk size of 500 GB, under the Microsoft Windows 7 platform and eclipse LUNA for implementation of JAVA.

5.2 Performance Measurements

In general, evaluation of an IDS is a complicated task that may be based on the extent to which the IDS can classify intrusions correctly and precludes false detection. The difficulty of evaluating an IDS stems from that the IDS should function appropriately under unknown circumstances and handle new kinds of attacks in the network environment. The main evaluation measures are the DR, FAR, and ACC. Table 5.1 lists the estimation parameters used in evaluating the efficiency of the proposed IDS algorithm (Gaidhane, et al., 2014; Chen, et al., 2009; Wang, et al., 2010).

Table (5.1) Measurement Parameters

Parameter	**Definition**
True Positive Rate (TP)	Attack Occurs and an Alarm is Raised
False Positive Rate (FPR)	No Attack Occurs but Alarm Raised
True Negative Rate (TN)	No Attack Occurs and No Alarm is Raised
False Negative Rate (FN)	Attack Occurs but No Alarm is Raised

Following is a briefing on the main evaluation measures employed in the present study.

1- Accuracy (ACC):

To determine the overall number of correct predictions, the percentages of true positive detections plus true negative detections are calculated and divided by the sum of the TP, TN, FP, and FN detections (Elhamahmy, et al., 2010) as illustrated by Equation (5.1):

$$ACC = \frac{\text{TP+TN}}{\text{TP+TN+FP +FN}} \qquad \text{…………………. (5.1)}$$

2- Detection Rate (DR):

The DR can be defined as ratio of number of attacks correctly-detected to total number of attacks (Kumar, 2014). It is calculated as percentage of TP detections divided by the TP plus the FN detections. The value is reported as given in Equation (5.2):

$$DR = \frac{\text{TP}}{\text{TP+FP}} \qquad \text{…………………. (5.2)}$$

3- False Alarm Rate (FAR):

The FAR is defined as number of normal patterns categorized as attacks (FP) divided by total number of the normal patterns (Elhamahmy, et al., 2010), as given by Equation (5.3):

$$\text{FAR} = \frac{\text{FP}}{\text{FP +TN}} \qquad \text{...………………..…. (5.3)}$$

5.3 The Dataset

The KDD '99 dataset is the most common and the benchmark dataset for evaluation of the IDSs since 1999. It has been compiled in 1998 from the DARPA Intrusion Detection and Evaluation, which was derived from 4.0 gigabytes of raw, compressed binary TCP/Dump data for seven-week network traffic that may be processed into nearly 5.0 million connection records, each with almost 100 bytes (Tavallae, et.al, 2009). This two-week data correspond about 2.0 million connection records, each of which is a vector that includes 41 features categorized as either attack (DoS, Probe, U2R, and R2L,) or normal activity (KDD Cup 1999, http://kdd.ics.uci.edu/databases/kddcup 99/kddcup99.html). All studies of IDSs in cloud environments employ the KDD '99 dataset, or its normal distribution, as a benchmark.

In the current study, the normal distribution one tenth of the KDD '99 dataset has been used. This represents a normal distribution of KDD '99. Almost (34.99%) of the data were used as a testing set. The remainder data 65.0% were used as a training set. One tenth of the KDD '99 data was used to avoid the 'Java heap' error, which results from the huge size of the KDD '99 dataset. Table (5.2) shows the distribution of network attacks sizes in KDD '99.

Table (5.2). Distribution of network attacks in 10% of the KDD '99 dataset

Attack Type	**Percentage of Data**
DoS	51%
PROBE	28%
R2L	11%
U2R	8%
Normal	2%

The records in the KDD '99 dataset, each, contain information about 41 network packets (Farid, et al., 2010) features (Table 5.3).

Table (5.3). Features of the KDD '99 Dataset

eature Number	**Feature Name**	**Type** (1)	**Description**
	Duration	C	length (number of seconds) of the connection
2	Protocol type	D	type of the protocol, e.g. tcp, udp, etc.
3	Service	D	network service on the destination, e.g., http, telnet, etc.
4	Flag	D	normal or error status of the connection
5	Src_bytes	C	number of data bytes from source to destination
6	Dst_bytes	C	number of data bytes from destination to source
7	Land	D	1 if connection is from/to the same host/port; 0 otherwise
8	Wrong fragment	C	number of ``wrong'' fragments
9	Urgent	C	number of urgent packets
0	Hot	C	number of ``hot'' indicators
1	Num_failed_logins	C	number of failed login attempts
2	Logged in	D	1 if successfully logged in; 0 otherwise
3	Num_compromised	C	number of ``compromised'' conditions
4	Root shell	C	1 if root shell is obtained; 0 otherwise
5	Su_attempted	C	1 if ``su root'' command attempted; 0 otherwise
6	Num_root	C	number of ``root'' accesses
7	Num_file_creations	C	number of file creation operations
8	Num_shell	C	number of shell prompts
9	Num_access_files	C	number of operations on access control files
0	Num_outbound_cmds	C	number of outbound commands in an ftp session
1	Is_host_login	D	1 if the login belongs to the ``hot'' list; 0 otherwise
2	Is_guest_login	D	1 if the login is a ``guest'' login; 0 otherwise
3	Count	C	number of connections to the same host as the current connection in the past two seconds
4	Srv_count	C	number of connections to the same service
5	Serror_rate	C	% of connections that have ``SYN'' errors
6	Srv_serror_rate	C	% of connections that have ``SYN'' errors
7	Rerror_rate	C	% of connections that have ``REJ'' errors
8	Srv_rerror_rate	C	% of connections that have ``REJ'' errors
9	Same_srv_rate	C	% of connections to the same service
0	Diff_srv_rate	C	% of connections to different services
1	Srv_diff_host_rate	C	% of connections to different hosts
2	Dst_host_count	C	count for destination host
3	Dst_host_srv_count	C	srv_count for destination host
4	Dst_host_same_srv_rate	C	same_srv_rate for destination host
5	Dst_host_diff_srv_rate	C	diff_srv_rate for destination host
6	Dst_host_same_src_port_rate	C	same_src_port_rate for destination host
7	Dst_host_srv_diff_host_rate	C	diff_host_rate for destination host
8	Dst_host_serror_rate	C	serror_rate for destination host
9	Dst_host_srv_serror_rate	C	srv_serror_rate for destination host
0	Dst_host_rerror_rate	C	rerror_rate for destination host
1	Dst_host_srv_rerror_rate	C	srv_serror_rate for destination host
2	Attack name	-	-
1) C: Continuous; D: Discrete.			

Figure (5.1) is example of network record that has been taken from the training dataset while Figure (5.2) introduces example of network record that has been taken from a testing dataset.

```
0,tcp,http,SF,158,20300,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,
210,255,1.00,0.00,0.00,0.01,0.00,0.00,0.00,0.00,normal
```

Figure (5.1). Example of a Training Data Record.

```
0,tcp,http,SF,158,12382,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,6,0.00,0.00,0.00,0.00,1.00,0.00,0.67,
182,182,1.00,0.00,0.01,0.00,0.00,0.00,0.00,0.00
```

Figure (5.2). Example of a Testing Data Record.

5.4 Feature Filtration Using FLAME

One of the main stages in the proposed work is feature filtration. This step aims at selecting the features that have a significant role in detection of each type of attack. Thus, the FLAME function was implemented to achieve that purpose. This function clusters the features into three main clusters; Inner, Outer, and Rest, as follows:

1. Inner Cluster: The object with the highest density amongst all its neighbors.
2. Outer Cluster: The interception with less than its neighbors' densities and less than predetermined threshold.
3. Rest Cluster: The object with a density that falls between the densities of the inner object and the outer object, close to the threshold.

Table (5.4) provides the results of FLAME clustering. It shows that the Inner cluster achieved the highest DR and ACC; which is 99.66 for DR and 95.20 for ACC, respectively. As regards the FAR, the Inner cluster achieved the lowest FAR value;

0.3636. The number of features which leads to better results will be 19 features rather than 41 features.

Table (5.4). FLAME clusters and their results for 19 features.

Cluster name	**DR**	**ACC**	**FAR**	**# of feature**
Inner	***99.6363***	***95.2040***	***0.3636***	***19***
Outer	49.3640	3.9334	50.6359	19
Rest	81.2036	53.516	18.7963	19

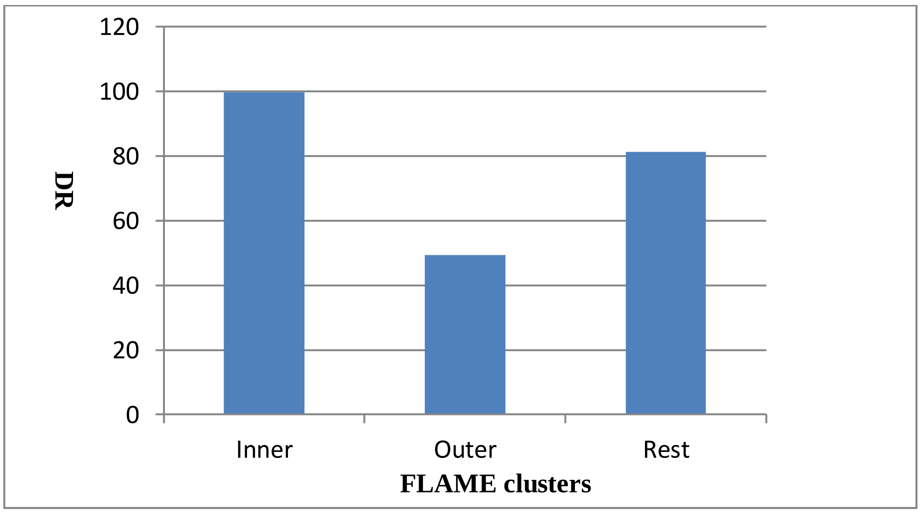

Figure (5.3). The DR Associated with FLAME Clustering.

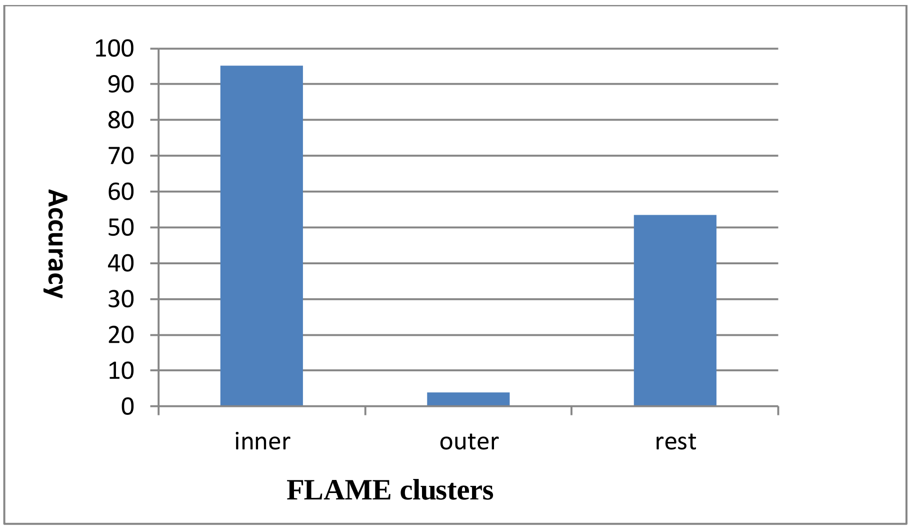

Figure (5.4). The ACC Associated with FLAME Clustering.

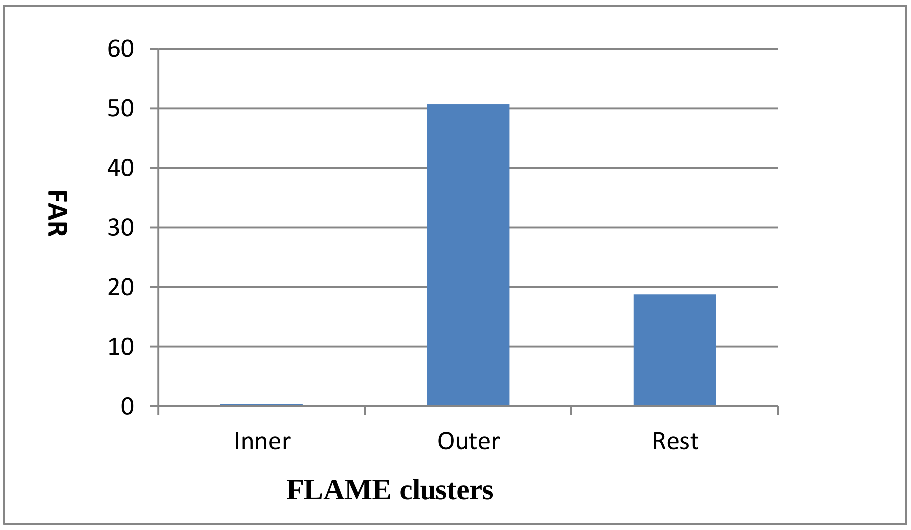

Figure (5.5). The FAR Associated with FLAME Clustering.

Figures (5.3), (5.4), and (5.5) show the results that were introduced by Table (5.4). These figures show that the Inner cluster has the highest DR and ACC and the lowest FAR, most probably because the Inner cluster had the more related features than the other clusters. In more detail, FLAME clustering was performed for detecting the system's overall DR, ACC, and FAR. The FLAME function was also adopted for the purpose of specifying the sort of attack. The Inner cluster has the most related features in that it produced the highest DR and ACC, and the lowest FAR.

5.4.1 Inner Cluster:

This part of the study comprised the following 19 features selected: 37, 38, 39, 34, 35, 36, 31, 32, 33, 29, 30, 40, 41, 5, 6, 7, 2, 3 and 4.

Table (5.5) summarizes the DR, ACC and FAR results for the Inner cluster for each type of attack.

Attack type	**DR%**	**ACC %**	**FAR%**
DoS	98.8585	94.6336	1.14150
Probe	99.7385	86.2340	0.26153
U2R	99.9486	99.9486	0.05137
R2L	99.9486	99.9486	0.05137

Figures (5.6), (5.7), and (5.8) display the DR, ACC and FAR results for the Inner cluster shown in Table (5.5).

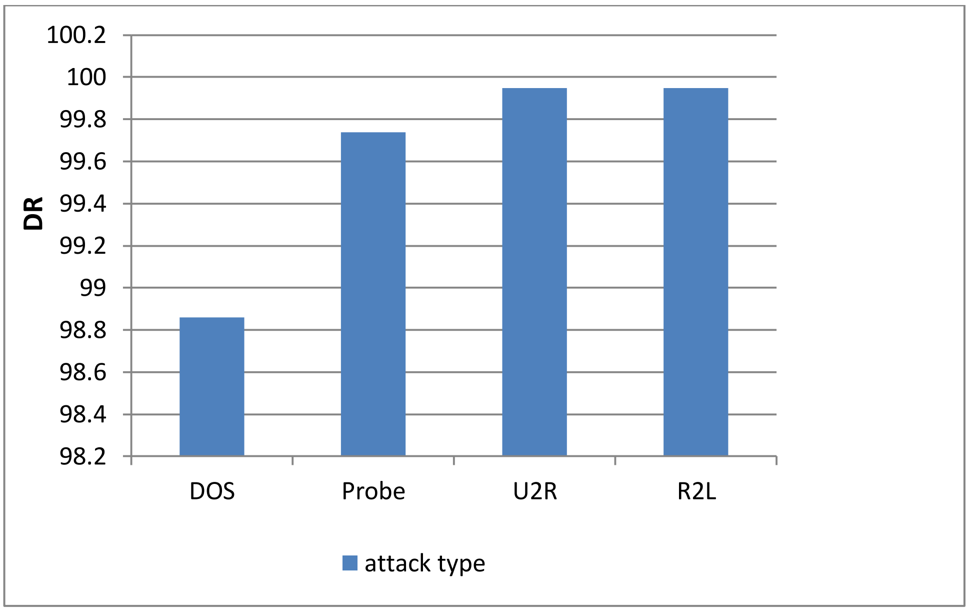

Figure (5.6). The DR for the Inner Cluster for each Sort of Attack.

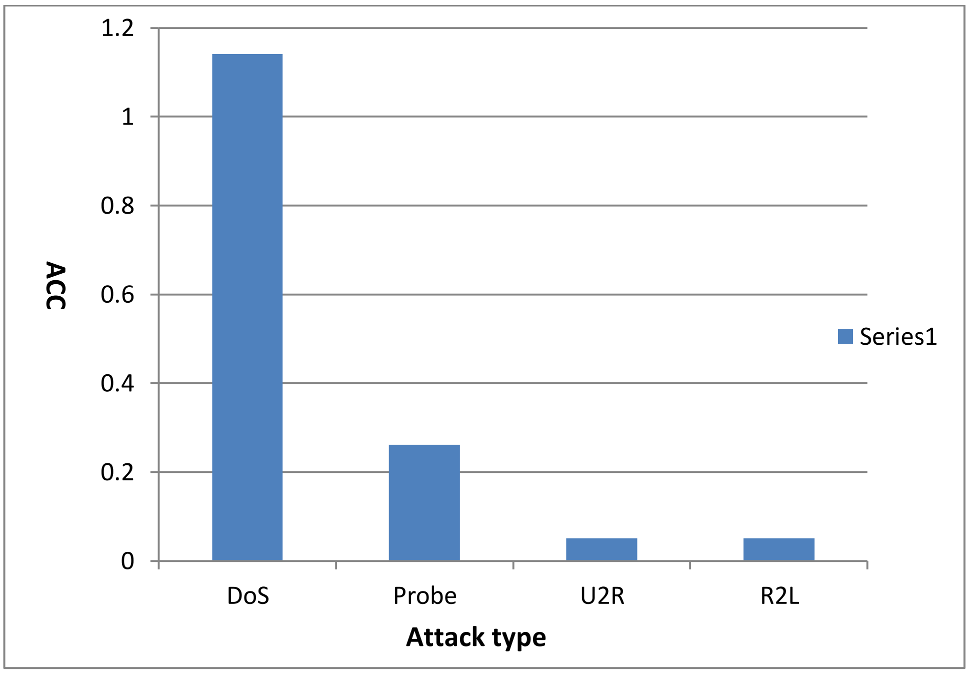

Figure (5.7). Accuracy for the Inner Cluster for each Sort of Attack.

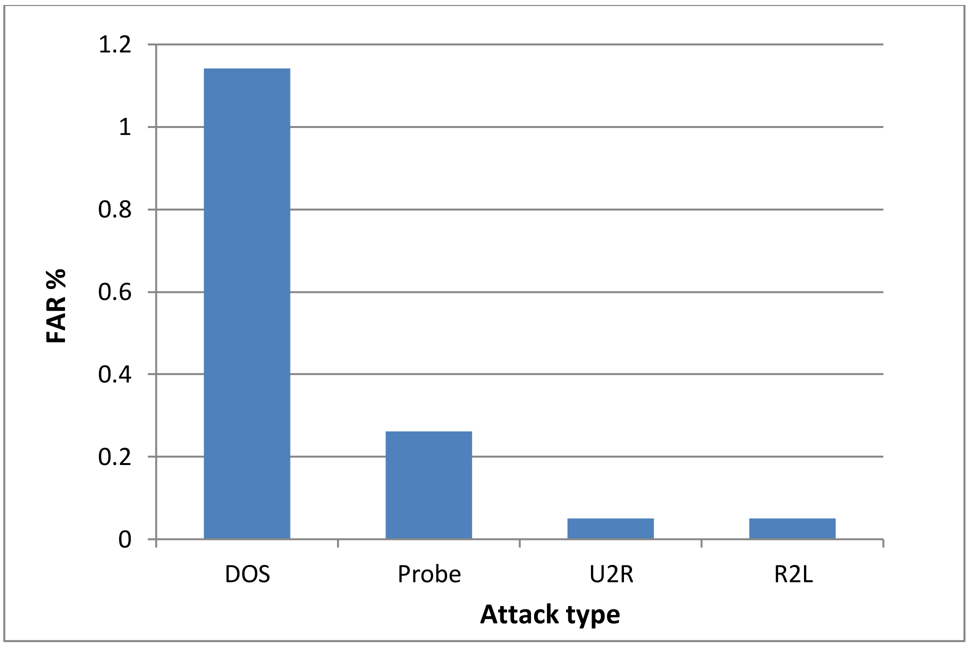

Figure (5.8). The FAR for the Inner Cluster for each Sort of Attack.

For more enhancement there is multiple values for weight used in HANN, the best weight that gives the best result for DR, ACC and FAR is **1.0**, as will be mentioned in the next section.

Table (5.6) shows the result for inner cluster where weight =1.0, which is the best weight.

Table (5.6). Results for the Inner cluster when weight = 1.0.

Attack type	DR	ACC	FAR	# of features
DoS	98.8585 %	94.6336%	1.1414%	19
Probe	99.7384 %	86.2339%	0.2615%	
U2R	99.9486 %	99.9486%	0.0513%	
R2L	99.9486 %	99.9486%	0.0513%	
Overall System Result	99.6363%	95.2040%	0.3636%	

Table (5.7) points out that the best number of features which positively affect the DR, ACC, and FAR is associated with the Inner cluster.

Table (5.7). Results for the Inner cluster

Overall System Result	DR	ACC	FAR	Weight	# of features
	99.6363%	95.2040 %	0.3636%	1	19

5.4.2 Outer Cluster

This part of the study comprised the following 10 features selected: 4, 9, 14, 17, 18, 19, 20, 21, 24, and 15.

The Outer cluster contains insignificant features, thus leading to lower DR and ACC, and higher FAR than the other clusters. The IDS using the Outer cluster could not achieve its purpose. Table (5.8) shows the DR, ACC, and FAR results for each type of attack for the Outer cluster.

Table (5.8). Results for the Outer cluster when weight = 1.0

Attack type	DR	ACC	FAR	Weight
DoS	45.2059%	3.6551%	54.7940%	1.0
Probe	52.6571%	4.3946%	47.3428%	
U2R	45.0561%	4.4204%	54.9438%	
R2L	54.5368%	3.2634%	45.4631%	
Overall System results	49.3640%	3.9334%	50.6359%	

5.4.3 The Rest Cluster

This part of the study comprised the following 13 features: 4, 9, 14, 15, 17, 18, 19, 20, 21, 24, 26, 27, and 31. This cluster contains features whose densities fall between the densities of the features of the Inner and the Outer clusters. However, this cluster did not give much good results for the proposed IDS. For each type of attack, it produced low DR and ACC values and high FAR for U2R and R2L attacks, as evidenced by the results in Table (5.9).

Table (5.9). Results for the 'Rest' cluster for the weights of 1.0.

Attack type	**DR**	**ACC**	**FAR**	**Weight**
DoS	99.8370%	99.8370%	0.1629%	1
Probe	99.9999%	99.9999%	1.56745%	
U2R	43.2657%	1.0220%	56.7342%	
R2L	43.2657%	1.0220%	56.7342%	
Overall System results	81.2036%	53.5169%	18.7963%	

For higher system accuracy and so as to explore any contradictory evidence, the researcher examined all the clusters and potential combinations of clusters of the FLAME. Elaborations on the results obtained with the various possible cluster combinations follow.

5.4.4 Inner and Rest Clusters

This part of the study comprised the following 19 features: 1, 2, 3, 8, 9, 10, 11, 12, 19, 13, 14, 15, 4, 5, 6, 7, 16, 23, and 33. Using this combination of features from the Inner and the Rest clusters, Table (5.10) indicates that the DR and ACC values are not high where as the FAR is not low for each type of network attack.

Table (5.10). Results for the combination of 'Inner' and 'Rest' clusters when weight = 1.0.

Attack type	**DR**	**ACC**	**FAR**	**Weight**
DoS	78.3714%	71.2648%	21.62853863057326	1.0
Probe	99.9998%	99.9426%	1.66443724130448344	
U2R	72.9339%	71.4254%	27.066007681242844	
R2L	72.93399%	71.42543%	27.066007681242844	
Overall System results	80.1304%	61.3916%	19.869579355416057	

5.4.5 Inner and Outer Clusters

By using a combination of features from the Inner and the Outer clusters, Table (5.11) brings to notice that the ACC is not high while the FAR is not high for the four network attack types.

Table (5.11). Results for the combination of 'Inner' and 'Outer' clusters when weight = 1.0.

Attack type	**DR**	**Accuracy**	**FAR**	**Weight**	**# of features**
DoS	99.91812 %	9.9918121%	0.0818784%	1.0	19
Probe	99.99998 %	9.9999982%	1.7481870%		
U2R	100.0 %	10%	0.0%		
R2L	100.0 %	10%	0.0%		
Overall system result	92.653114%	9.2653114%	7.346885%		

5.4.6 Outer and Rest Clusters

The combination of features from the Inner and Rest clusters generated the worst results. In specific, Table (5.12) demonstrates that the DR and ACC values are low while the FAR is high for each kind of network attack type.

Table (5.12). Results for the combination of 'Outer' and 'Rest' clusters when weight = 1.0

Attack Type	**DR**	**Accuracy**	**FAR**	**Weight**	**# of features**
DoS	99.9999 %	9.9999%	3.374312%	1.0	19
Probe	99.9999 %	9.9999%	3.374312%		
U2R	39.65062 %	2.70393%	60.34937%		
R2L	39.6506 %	2.70393%	60.34937%		

Over all system result	70.9722 %	6.38015 %	29.027723%		

5.5 Hopfield Artificial Neural Network (HANN)

The HANN has an optimization method. Different researchers have applied ANNs on their networks. For choosing the best weight in the range of 0 and 1, the weight of the neural network is increased at a constant rate until the program reaches to the best result for the data as shown in Table (5.13), which presents the DRs for each type of attack and for multiple weights within the weight range of [0.1 -1].

Table (5.13). The DRs based on type of attack using 19 features with multiple weights

Weight	DoS %	Probe %	U2R %	R2L %
0.1	99.7453	99.8212	99.9999	99.9999
0.2	89.5983	99.9999	99.9999	99.9999
0.3	98.4672	44.4359	36.9715	36.9715
0.4	99.6364	99.9566	99.9749	99.9749
0.5	59.5786	99.9999	99.9999	99.9999
0.6	83.9917	99.9706	100	100
0.7	84.2444	99.8940	43.2778	43.2778
0.8	95.5747	99.9802	42.6386	42.6386
0.9	69.9271	99.1507	99.9999	99.9999
1.0	98.8585	99.7384	99.9486	99.9486

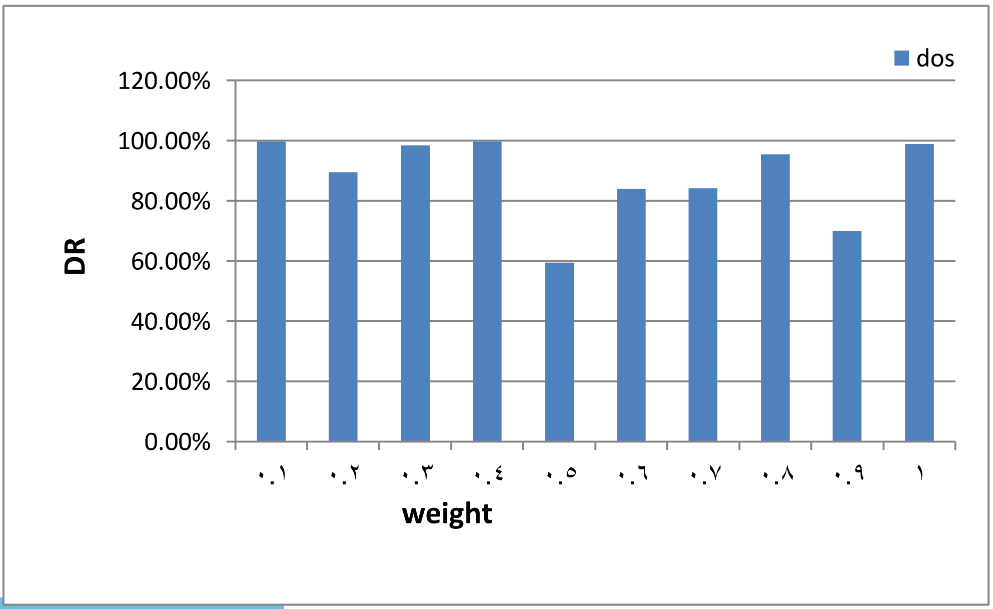

Figure (5.9). The DRs for the DoS Attack with Multiple Weights.

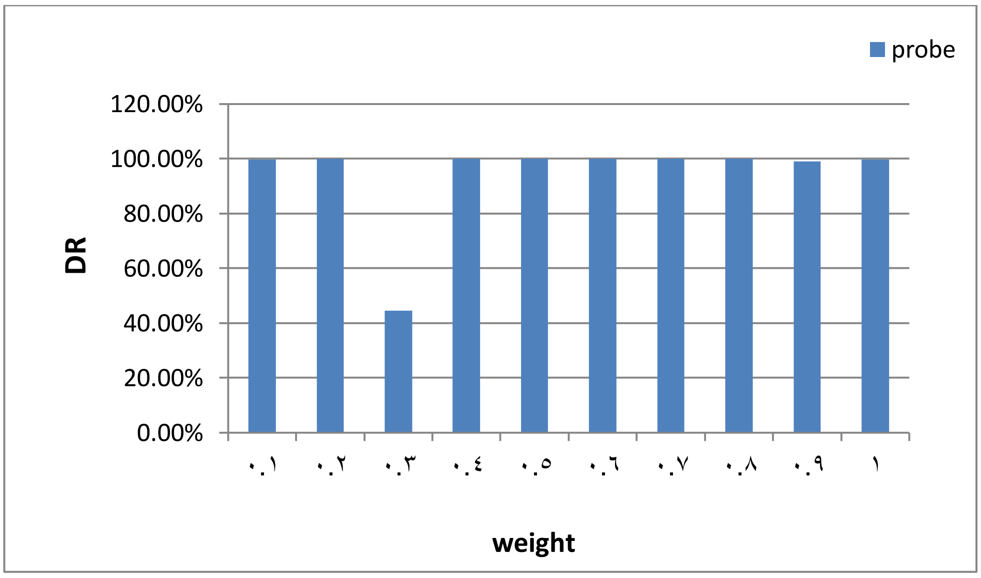

Figure (5.10). The DRs for the Probe Attack with Multiple Weights.

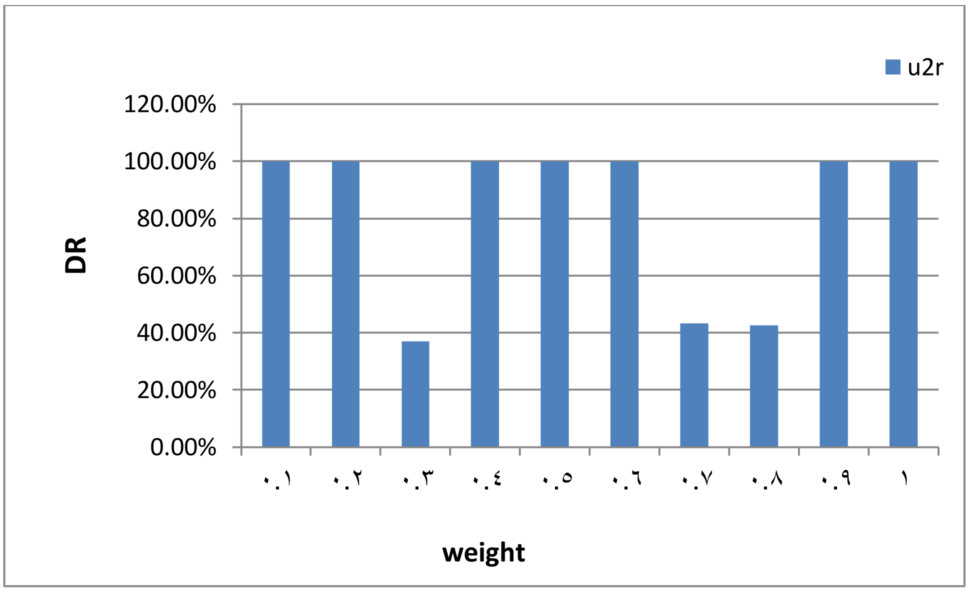

Figure (5.11). The DRs for the U2R Attack with Multiple Weights.

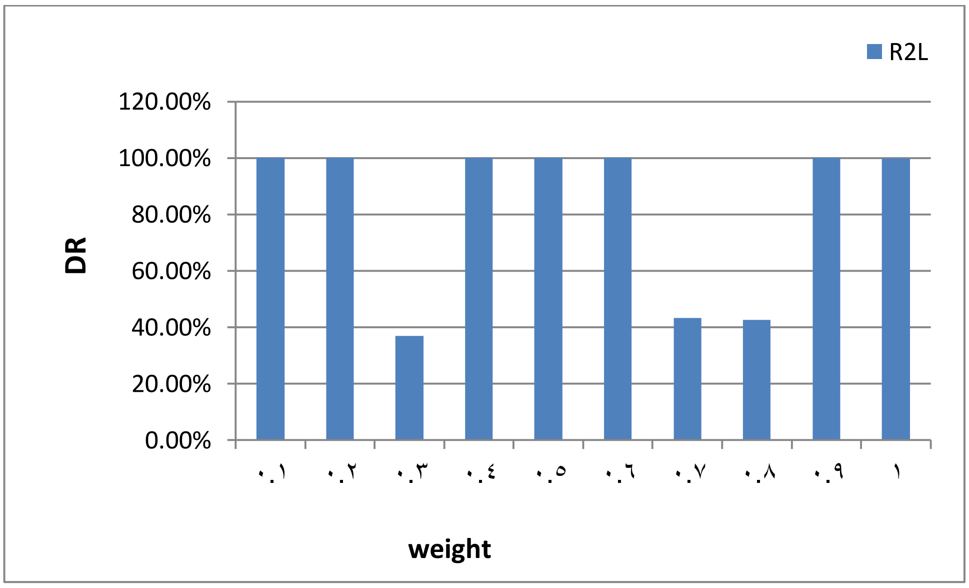

Figure (5.12). The DRs for the R2L Attack with Multiple Weights.

These figures reveal the differences between the obtained results in view of differences in attack type and weights. The researcher finds that the weights of 1.0 and 0.4 result in high DR, with higher DRs associated with the weight of 0.40 than 1.0. Table 5.14 shows the ACC values for each type of attack and for varied weights in the range [0.1 - 1]. The results in this table point out that the weight giving the best ACC is 0.1.

Table (5.14). The ACCs associated with the types of attacks using 19 features and multiple weights

Weight	**DoS %**	**Probe %**	**U2R %**	**R2L %**
0.1	99.96943	99.9821	95.9999	95.9999
0.2	16.9460	19.9827	19.9999	19.9999
0.3	17.5968	11.7389	6.23795	6.2379
0.4	37.3859	39.7972	39.9893	39.9893
0.5	10.4035	49.9999	49.9999	49.9999
0.6	39.7133	59.9823	60.0	60.0
0.7	44.4864	65.4165	30.1911	30.1911
0.8	62.4408	79.9003	19.9272	19.92724
0.9	52.0826	89.0192	89.9999	89.9999
1.0	94.6336	86.2339	99.9486	99.9486

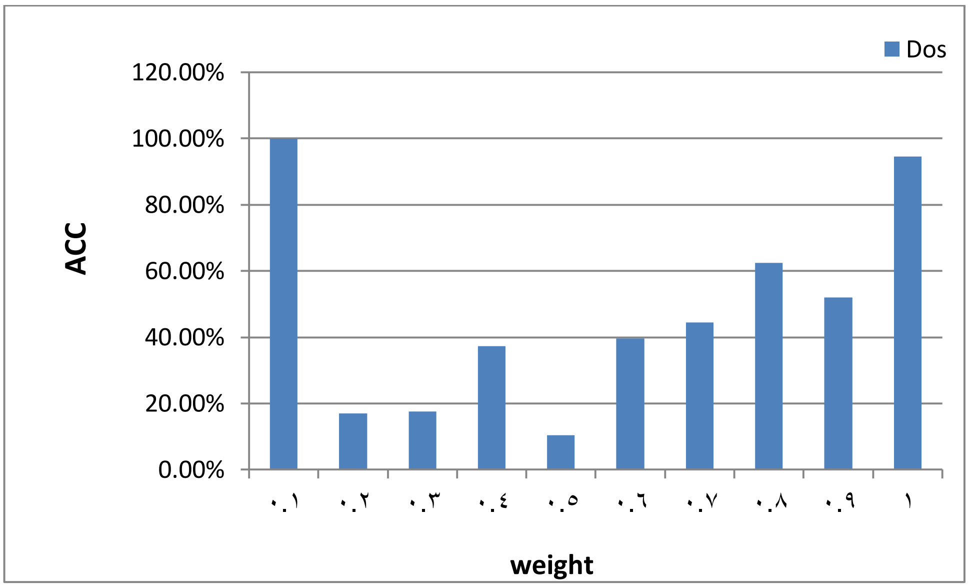

Figure (5.13), The ACCs for the DoS Attack with Multiple Weights.

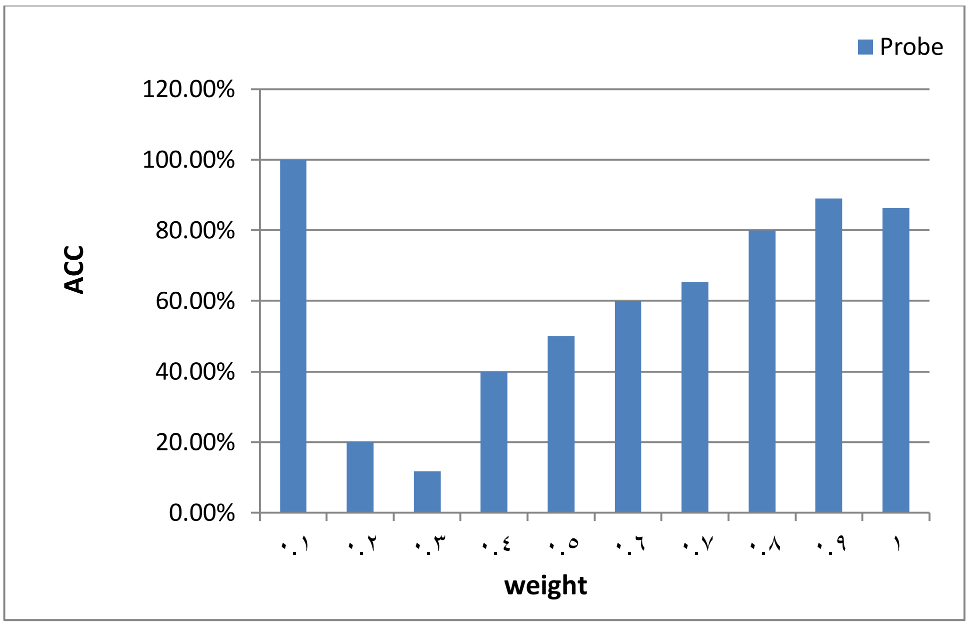

Figure (5.14). The ACCs for the Probe Attack with Multiple Weights.

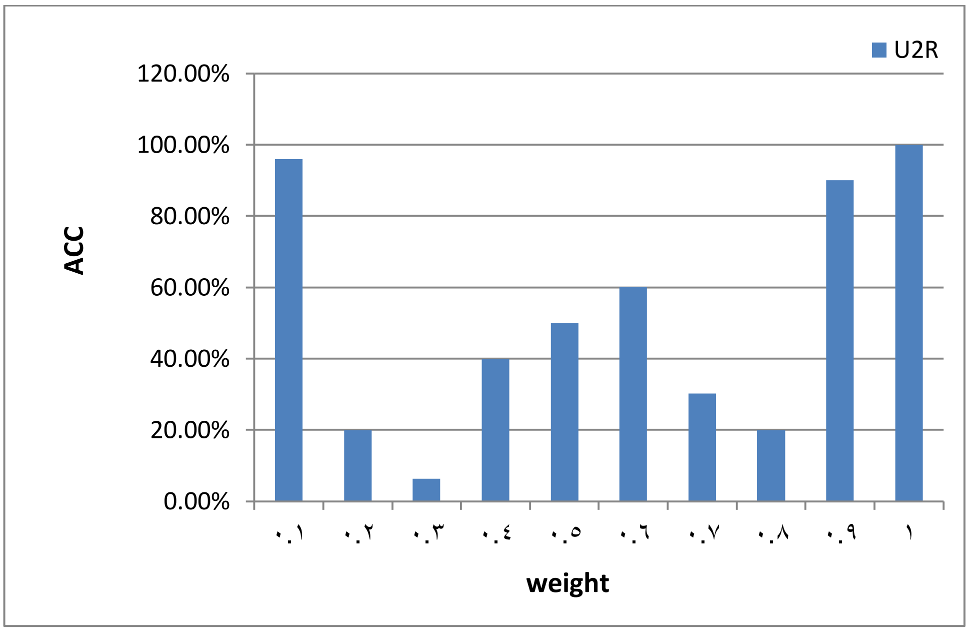

Figure (5.15). The ACCs for the U2R Attack with Multiple Weights.

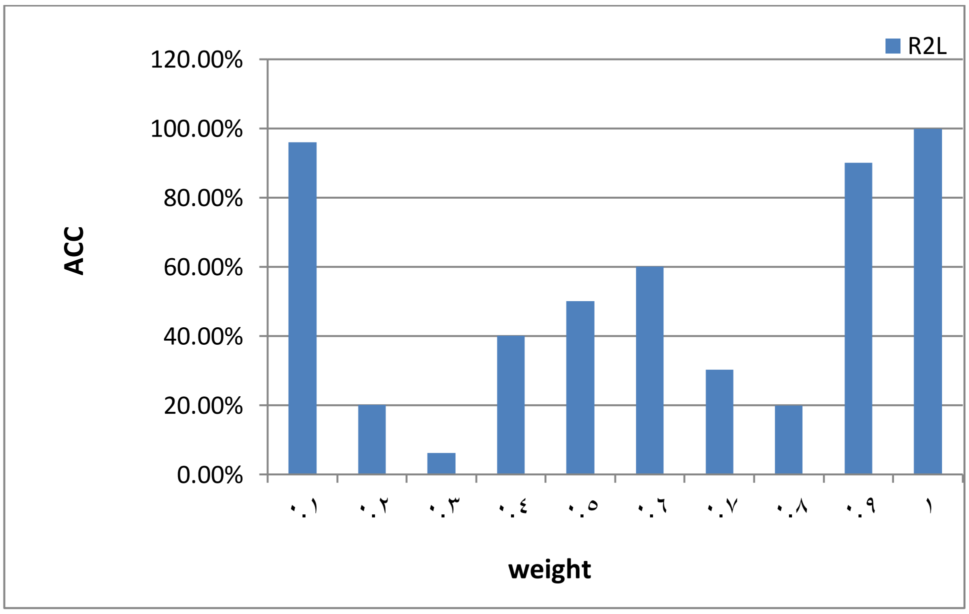

Figure (5.16). The ACCs for the R2L Attack with Multiple Weights.

As Figures (5.13), (5.14), (5.15), and (5.15) demonstrate, the weight which results in the highest system accuracy is 1.0. This weight produces higher ACC than any other weight, including 0.4, which actually has a high DR. Table (5.15) shows the FAR results for each type of network and a multitude of weights in the range of [0.1 - 1]. The results underline that the weight that gives the lowest FAR is 1.

Table (5.15). The FARs associated with the different types of attacks using 19 features and multiple weights

Weight	**DoS**	**Probe**	**U2R**	**R2L**
0.1	0.2546	0.1787	5.0414	5.0414
0.2	10.4016	2.3322	4.8835	4.8835
0.3	1.5327	55.5640	63.0284	63.0284
0.4	0.3635	0.0433	0.0250	0.0250
0.5	40.4213	1.8805	3.6533	3.65332
0.6	16.0082	0.0293	0.0	0.0
0.7	15.7555	0.1059	56.7221	56.7221
0.8	4.4252	0.0197	57.3613	57.3613
0.9	30.0728	0.8492	1.9618	1.9618
1.0	1.1414	0.2615	0.0513	0.0513

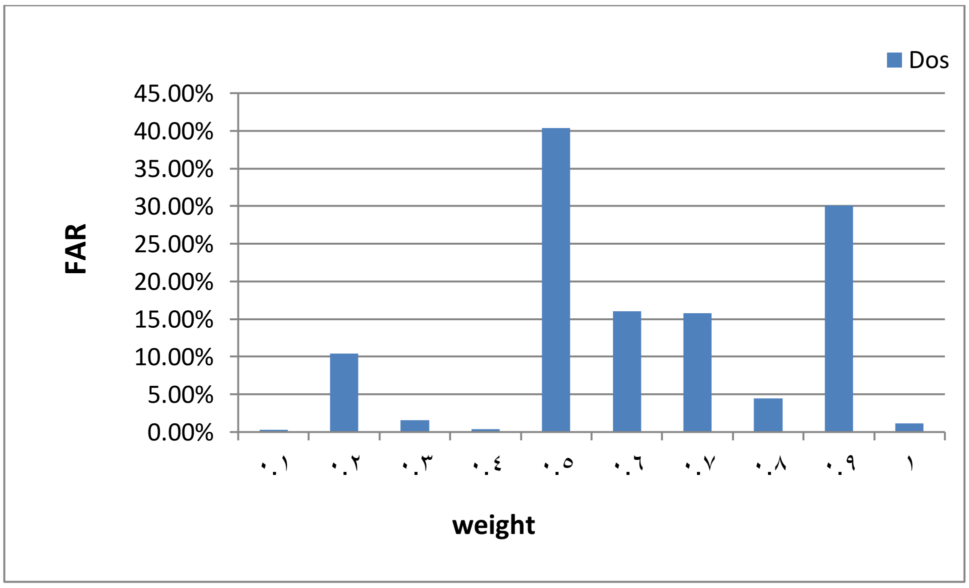

Figure (5.17). The FARs for the DoS Attack with Multiple Weights.

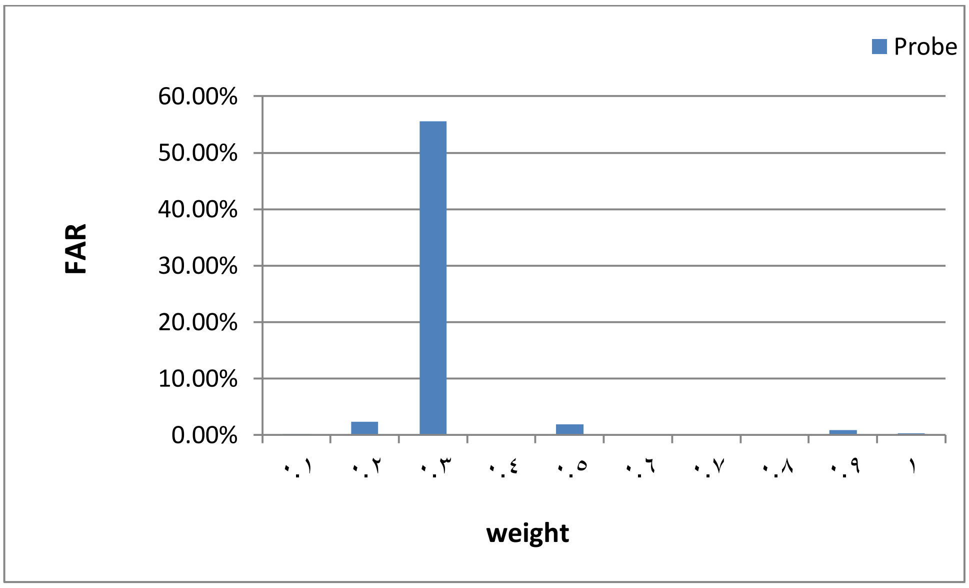

Figure (5.18). The FARs for the Probe Attack with Multiple Weights.

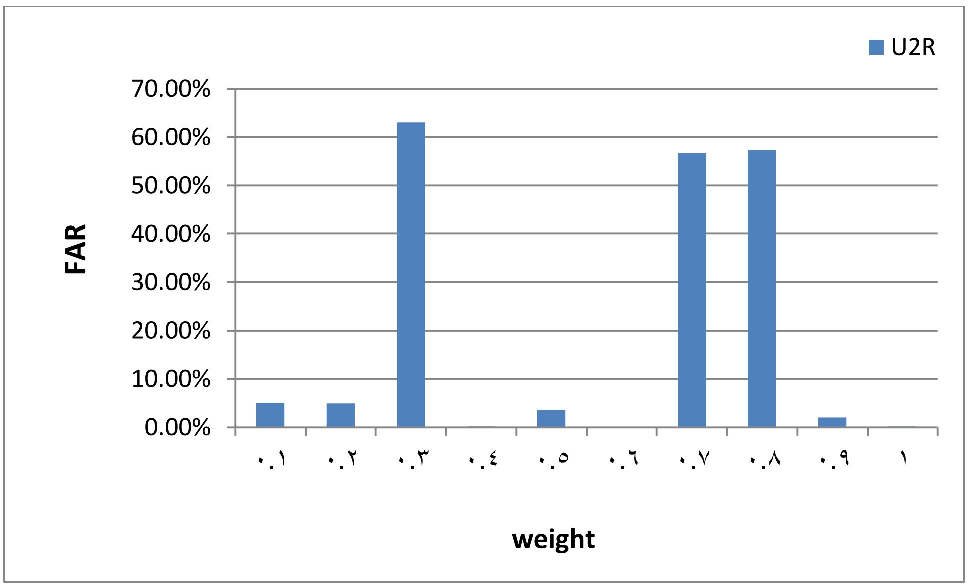

Figure (5.19). The FARs for the U2R Attack with Multiple Weights.

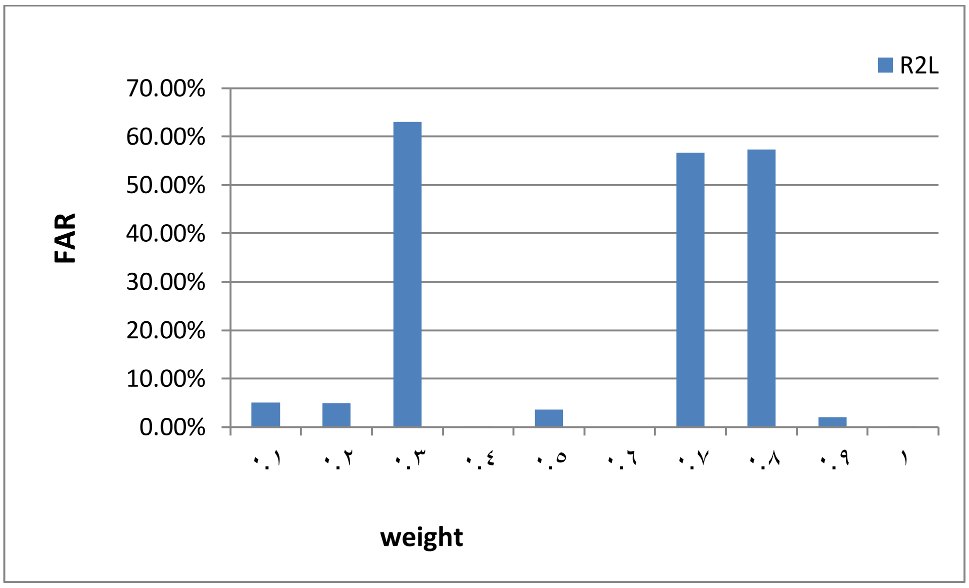

Figure (5.20). The FARs for the R2L Attack with Multiple Weights.

Figures (5.17), (5.18), (5.19), and (5.20) uncover that the weight 0.4 generated the lowest FAR. However, the best weight should have the highest DR, highest system's ACC, and lowest FAR. In light of this, the researcher underlines that the study results suggest that all these requirements have been fulfilled by the weight of **1**.

5.6 Optimal Solution of the System

Table (5.16) summarizes the results of search for the optimum solution for the system comprising 19 features, with a weight of 1.0 and a time of 3.0 seconds.

Table (5.16). Results of search for the optimal solution for the system comprising 19 features with a weight of 1.0 and a time of 3.0 seconds

Attack type	**DR %**	**ACC %**	**FAR%**	**Weight %**	**# of features**
DoS	98.85850	94.63361	1.1414953	1	19
Probe	99.73847	86.23397	0.2615266		
U2R	99.9486	99.94862	0.0513715		
R2L	99.9486	99.94862	0.0513715		
Overall System Result	99.63639	95.20404	0.363604		

The 19 features give good results because the combination of these features is strong. Table 5.17 presents results of comparison held between outcomes of the proposed system and those of a number of related studies.

Table (5.17). Comparison between outcomes of the suggested method and those of some related studies.

Method	**# of features**	**DR%**	**ACC%**	**FA%**
Proposed alg.	19	99.63639	95.204049	0.3636040
selman	13	99.3% for probe and 58.8% for U2R	NA	NA
FC-ANN	41	96.71%	NA	NA
Hybrolic hopfeild	41	95 %	NA	NA
Khazaee and Faez	41	NA	NA	45%

In Table (5.18) and the subsequent paragraphs, the researcher introduces results of a comparison between the SVM and the proposed method.

Table (5.18). Comparison between the SVM and the proposed method

Attack type	**SVM**	**Proposed method**
	DR%	
DoS	97.5	99.998673 %
Probe	86.6	99.997969 %
U2R	81.3	99.953693 %
R2L	92.8	99.953693 %

In the sequent paragraphs, the researcher presents results of a comparison between FC-ANN and the proposed method. Some researchers used the detection method known as Recall, which was developed by Wang (2010). In the FC-ANN method, two performance criteria are used: Precision and the *F*-value. Therefore, the researcher embedded these two respective equations in this work. The following tables summarize the results produced by these equations.

Table (5.19). Comparison between Wang's (2010) method and the proposed method for the DoS attack

	Wang (FC-ANN)	**Proposed method**
Precision %	99.91	99.96
Recall %	96.70	99.99867
F-value %	98.28	97.05

Table (5.20). Comparison between Wang's (2010) method and the suggested method for the Probe attack

	Wang (FC-ANN)	**Proposed method**
Precision %	48.12	50.05
Recall %	80.00	99.99796
F-value %	60.09	79.89

Table (5.21). Comparison between Wang's (2010) method and the suggested method for the U2R attack

	Wang (FC-ANN)	**Proposed method**
Precision %	83.33	93.5
Recall %	76.92	99.9536
F-value %	80.00	89.30

Table (5.22). Comparison between Wang's (2010) method and the suggested method for the R2L attack

	Wang (FC-ANN)	**Proposed method**
Precision %	93.18	96.44
Recall %	58.57	99.9536
F-value %	71.93	83.77

5.7 Timing

In terms of CPU computational time, the algorithms can not be compared because they were executed on different machines, and there was no possible access to the original code.

5.8 Feature Filtration

In this study, the researcher used feature filtration as was mentioned in Chapter IV. While trying to improve the intrusion detection results, it was found that filtration enhances the results by using 19 features and a weight of 1.0 as can be seen in Table (5.23). The 41 features produced lower DR than the 19 features did.

Table (5.23). The best number of features to detect so as to achieve a high DR

Attack type	**DR %**	**ACC %**	**FAR%**	**Weight**	**# of features**
DoS	98.85850	94.63361	1.1414953	1	19
Probe	99.73847	86.23397	0.2615266		
U2R	99.9486	99.94862	0.0513715		
R2L	99.9486	99.94862	0.0513715		
DR ACC FAR of system	99.63639, 95.20404, 0.363604 Time span = 3 seconds				

VI

Conclusion and Future Works

6.1 Conclusion and Future Work

This thesis presents an adjustment for IDS by adapting several AI algorithms to obtain an improved DR, ACC and a low FAR. For this purpose, a network-based IDS was implemented according to the source of data in the cloud environment, taking advantage of the knowledge of the normal behavior of network environment users. The proposed technique consists of a set of stages that start from grouping the data until an aggregation stage, which is concerned with providing a final decision on whether the incoming data is an attack or not.

The main contributions of this study are:

A. grouping the data into four clusters using the Weka classifier according to the type of attack; Probe, DoS, R2L, and U2R. . The groping stage helps in determining the significant features of each type of attack out of 41 network features. Achievement of this target will be reflected by an improved DR and a reduced FAR.

B. The FLAME methodology is applied for feature filtration by reducing the number of features to less than 41, and, in consequence, get a highly-accurate system with a high DR. Each type of attack (DoS, Probe, U2R, and R2L) has its own features, which may intersect and some of them may be shared in common by the different types of attacks, by performing feature filtration using FLAME algorithm features where reduced to 19 features in the inner cluster, that gives the best result for DR, ACC and FAR, which produces an efficient IDS, when using outer and rest clusters of FLAME, the result was bad for DR, ACC and FAR, of the system and each type of attack; because they uses insignificant features.

C. Applying HANN, taking into account internal tuning of the internal weights of the HANN for neurons. By assigning HANN for each group of data, e.g., DoS to detect DoS attacks and predict intrusion, each HANN gives a result, the weight that gives the best result equals 1.0, this result come from testing the system.

D. For aggregation stage, the SA algorithm was applied to get the global optimal solution for a combinatorial optimization problem and obtain the best results from the HANN clusters.

The best result of IDS is using 19 features from the inner cluster for FLAME algorithm and weight equals 1.0 and gives best and high result for DR and ACC also lower FAR.

In view of the study findings, the researcher maintains that it will be quite interesting for future works to develop an IDS that modifies weights adaptively, rather than selecting weights manually, so as to speed up the performance of this type of systems.

References

References:

Adaobi, O., & Ghassemian, M. (2010, December). Analysis of an Anomaly-based Intrusion Detection System for Wireless Sensor Networks. In 1st international conference communications engineering (pp. 59-63).

Chen, R. C., Cheng, K. F., Chen, Y. H., & Hsieh, C. F. (2009, April). Using rough set and support vector machine for network intrusion detection system. In Intelligent Information and Database Systems, 2009. ACIIDS 2009. First Asian Conference on (pp. 465-470). IEEE.

Das, A., & Sathya, S. S. (2012, July). A fuzzy approach to feature reduction in KDD intrusion detection dataset. In Computing Communication & Networking Technologies (ICCCNT), 2012 Third International Conference on (pp. 1-5). IEEE.

Domínguez, J. E. L., Martín, A. S., & de Chalco, C. U. U. V. Intrusion detection pattern recognition using an Artificial Neural Network., International Journal of Engineering Science and Innovative Technology (IJESIT).

El Farissi, A. I., Azizi, M., & Moussaoui, M. (2012, May). Detection of smart card attacks using neural networks. In Multimedia Computing and Systems (ICMCS), 2012 International Conference on (pp. 949-954). IEEE.

Elhamahmy, M. E., Elmahdy, H. N., & Saroit, I. A. (2010). A New Approach for Evaluating Intrusion Detection System. CiiT International Journal of Artificial Intelligent Systems and Machine Learning, 2(11).

Fu, L., & Medico, E. (2007). FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data. BMC bioinformatics, 8(1), 3.

Gaidhane Roshani, Prof Vaidya. C, and Dr. Raghuwanshi M, International Journal of Advance Foundation and Research in Computer (IJAFRC), (2014), Survey: Learning Techniques for Intrusion Detection System (IDS).

Gaikwad, D. P., Jagtap, S., Thakare, K., & Budhawant, V. (2012, November). Anomaly Based Intrusion Detection System Using Artificial Neural Network and Fuzzy Clustering. In International Journal of Engineering Research and Technology (Vol. 1, No. 9 (November-2012)). ESRSA Publications.

Gao, M., & Tian, J. (2009, April). Network intrusion detection method based on improved simulated annealing neural network. In Measuring Technology and Mechatronics Automation, 2009. ICMTMA'09. International Conference on (Vol. 3, pp. 261-264). IEEE.

Gyanchandani, M., Rana, J. L., & Yadav, R. N. (2012). Taxonomy of anomaly based intrusion detection system: a review. International Journal of Scientific and Research Publications, 2(12), 1.

http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (20/12/2015).

Jabez, J., & Muthukumar, B. (2014). INTRUSION DETECTION SYSTEM: TIME PROBABILITY METHOD AND HYPERBOLIC HOPFIELD NEURAL NETWORK. Journal of Theoretical & Applied Information Technology, 67(1).

Jian, W., & Rui, F. G. (2009, November). Intrusion detection based on simulated annealing and fuzzy c-means clustering. In Multimedia Information Networking and Security, 2009. MINES'09. International Conference on (Vol. 2, pp. 382-385). IEEE.

Kaur Harjinder, Gill Nivit, (2013), Performance Comparison of Host based and Network based Anomaly Detection using Fuzzy Genetic Approach (FGA), International Journal of Computer Trends and Technology (IJCTT).

Kholidy, H. A., Baiardi, F., Hariri, S., Elhariri, E. M., Yousof, A. M., & Shehata, S. A. (2012). A Hierarchical Intrusion Detection System For Clouds: Design And Evaluation. International Journal on Cloud Computing Services and Architecture (IJCSA).

Khazaee, S., & Faez, K. (2014). A Novel Classification Method Using Hybridization of Fuzzy Clustering and Neural Networks for Intrusion Detection.International Journal of Modern Education and Computer Science (IJMECS),6(11), 11.

Kumar, G. (2014). Evaluation Metrics for Intrusion Detection Systems-A Study.International Journal of Computer Science and Mobile Applications, II, 11.

Lin, S. W., Ying, K. C., Lee, C. Y., & Lee, Z. J. (2012). An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection.Applied Soft Computing, 12(10), 3285-3290.

Paliwal, S., & Gupta, R. (2012). Denial-of-Service, Probing & Remote to User (R2L) Attack Detection using Genetic Algorithm. International Journal of Computer Applications, 60(19).

Panja, B., Ogunyanwo, O., & Meharia, P. (2014, June). Training of intelligent intrusion detection system using neuro fuzzy. In Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2014 15th IEEE/ACIS International Conference on (pp. 1-6). IEEE.

Poston, H. E. (2012, July). A brief taxonomy of intrusion detection strategies. In 2012 IEEE National Aerospace and Electronics Conference (NAECON).

Raghav, I., Chhikara, S., & Hasteer, N. (2013). Intrusion Detection and Prevention in Cloud Environment: A Systematic Review. International Journal of Computer Applications, 68(24).

Sahu, S. K., & Jena, S. K. (2014). A study of K-Means and C-Means clustering algorithms for intrusion detection product development. International Journal of Innovation, Management and Technology, 5(3), 207.

SANS, (2001), The History and Evolution of Intrusion Detection, Institute Reading Room site.(Rights, R. F. (2001). SANS Institute InfoSec Reading Room.)

Selman, A. H. (2013). Intrusion Detection System using Fuzzy Logic.SouthEast Europe Journal of Soft Computing, 2(1).

Shanmugavadivu R., Nagarajan N., (2013), Network Intrusion Detection System using Fuzzy Logic, Indian Journal of Computer Science and Engineering (IJCSE).

Shrivastava, S. K., & Jain, P. (2011). Effective anomaly based intrusion detection using rough set theory and support vector machine. International Journal of Computer Applications, 18(3), 35-41.

Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. computers & security, 28(1), 18-28.

Tiwari, S., Roy, S. S., Charaborty, S., & Kumar, A. (2013, December). A novel hybrid model for network intrusion detection. In Green Computing, Communication and Conservation of Energy (ICGCE), 2013 International Conference on (pp. 685-688). IEEE.

Toosi, A. N., & Kahani, M. (2007). A new approach to intrusion detection based on an evolutionary soft computing model using neuro-fuzzy classifiers.Computer communications, 30(10), 2201-2212.

Wang, G., Hao, J., Ma, J., & Huang, L. (2010). A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering. Expert Systems with Applications, 37(9), 6225-6232.

Witten, I. H., Frank, E., Trigg, L., Hall, M., Holmes, G., & Cunningham, S. J. (1999). Weka: Practical machine learning tools and techniques with Java implementations.

Zhang, Z., Li, J., Manikopoulos, C. N., Jorgenson, J., & Ucles, J. (2001, June). HIDE: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification. In Proc. IEEE Workshop on Information Assurance and Security (pp. 85-90).

ملخص

تقدم الحوسبة السحابية إطار لدعم المستخدمين النهائيين بسهولة نظرا لقدرتها على توفير كمية غير محدودة من الموارد. يجب على مقدمي الحوسبة السحابية حماية النظام، سواء من الغرباء والمطلعين. لقد اقترح نظاما يسمح للأنظمة السحب الآلي (CCSS) لتحقيق فعالية الكشف عن الاختراقات وقوة أمن النظام.

هو النظام الأكثر شيوعا لحماية صندوق (IDS) نظام كشف التسلل الضمان الاجتماعي من العديد من أنواع الهجمات. الخوارزمية المختلطة المقترحة هنا في يقلل من عدد من المزايا ٤١-١٩ الميزات. وبناء على اختبار في الدراسة الحالية، تم استخدام العادي للاختبار KDD CUP 99 معيارا (KDD توزيع ١٠٪ من ٩٩' وتدريب واقترح، الهجين النتائج الخوارزمية المقترحة يؤثر بشكل كبير على أداء النظام، ويعزز كفاءة معدلات الكشف أيضا دقة في (FAR) لأنواع مختلفة هجوم ويقلل من معدل انذار كاذب الشبكة IDS.