Distributed Network Intrusion Detection System: An Artificial Immune System Approach

Obinna Igbe Ihab Darwish Tarek Saadawi
Electrical Engineering Department
City University of New York, City College

Abstract*—Intrusion detection is the identification of unauthorized use, misuse, and abuse of computer system infrastructures by both system insiders and external intruders. Detecting intrusion in distributed network from outside network segment as well as from inside is a difficult problem. Network based Intrusion Detection System (NIDS) must analyze a large volume of data while not placing a significant added load on the monitoring systems and networks. This paper presents a framework for a distributed network intrusion detection system (dNIDS) based on the artificial immune system concept. In this framework, an adaptive immune mechanism through unsupervised machine learning methods is proposed to classify network traffic into either normal ("self") and suspicious profiles ("non-self") respectively. Experimentally, our approach distributes the NIDS among all connected network segments, allowing NIDS in each segment to identify potential threats individually and enabling the sharing of identified threat vectors between the communicating distributed NIDSs. Analysis of the technique for distribution of this information about threat vectors is presented.

***Keywords*—Artificial Immune System; Negative Selection; Genetic Algorithm; IDS; dNIDS**

I. INTRODUCTION

Many techniques have been deployed to combat security issues inherent in computer systems and its interconnecting networks. More especially networks comprising of computers that hold sensitive information like medical records of patients that visit a hospital. One of these techniques involves the deployment of an intrusion detection system (IDS). An IDS serves as an additional wall for protecting critical systems due to the ability of hackers to subvert other protection systems like firewalls [1].

The importance of protecting networks interconnecting medical devices cannot be overemphasized. Examples of attacks against such networks include malware attacks which a hacker can use to create backdoors for siphoning data out of the network and ransomware attack which is a form of attack where the hacker encrypts the files in devices connected to the health care network and demands money before such files will be decrypted.

Currently, health care networks are protected from malicious traffic via a multilayered security infrastructure that includes network intrusion detection systems (NIDS). Though NIDS are useful for identifying malicious activity in a network, they only have a single vantage point, which limits their ability to detect distributed or coordinated attacks [2]. Also, a *zero-day* attack [3] (an attack for which no signature exists) experienced by an organization's NIDS located, say in Texas, USA might not be a zero-day attack for another of its NIDS located, say in Delhi, India or another company's NIDS located in the same state. Therefore, if all the NIDS exchanged their detected threat information, more network threats can be stopped by a coordinated effort of all participating NIDS from the different or same organization.

In this paper, we propose a fully distributed NIDS (dNIDS) model to protect an organization's health care network from cyber-attacks. All participating NIDS will use the negative selection algorithm (NSA) inspired by the human immune system (HIS) to monitor the network interconnecting medical devices and equipment. The dNIDS components include IDSs that are located at the entry point of various interconnected network segments. These segments might be owned by the same organization or different organizations who have agreed to share their knowledge of threat vectors.

The remaining part of the paper is organized as follows; Section II discusses intrusion detection systems in more depth and the related work in the field of distributed and non-distributed IDS. Section III deals with artificial immune system and the algorithms that exist in this field. In section IV, we will explain the components of our proposed distributed IDS, its objectives, and the implementation details. The experimental setup and results are discussed in section V followed by our conclusion and future work in sections VI.

II. INTRUSION DETECTION SYSTEM (IDS)

Intrusion detection is defined as the process of intelligently monitoring the events occurring in a computer system or network, analyzing them for signs of violations of the security policy. The primary aim of Intrusion Detection Systems (IDS) is to protect the availability, confidentiality and integrity of critical networked information systems [4].

A. IDS Classification

IDS can be classified based on the detection method used, the source of information or its topological structure [5].

IDSs can be classified according to the detection approach they utilize into anomaly-based IDS and the signature based IDS. Anomaly-based intrusion detection monitors network traffic and user activities for abnormal behavior. On the other hand, a signature-based IDS uses a database with signatures to identify possible attacks and malicious activities. IDSs are also classified, according to how the data being used for intrusion detection is obtained, into host-based IDS and network-based

IDS. Host-based IDS (HIDS) monitors specific host machines while a network-based IDS (NIDS) identifies intrusions on key network points. Another classification of IDS is according to its topology or structure which can be centralized or distributed. In the distributed configuration, the distributed IDS (DIDS) consists of multiple IDSs over a large network communicating with each other or with a central server. These communicating IDSs could be all NIDS -forming a distributed NIDS (dNIDS), all HIDS or a combination of both. Distributed monitoring allows early detection of planned and coordinated attacks and thereby allowing the network administrators to take preventive measures. DIDS also helps to control the spreading of worms, improve network monitoring and incident analysis, attack tracing and helps detect new threats [5].

B. *Related Work*

There is a great deal of work that is currently being done in the area of intrusion detection. Much of the work centers on improvement in the ability of systems to detect attacks and the speed of network traffic that can be handled.

Snapp et al. [6] proposed a centralized DIDS model. This method makes the DIDS director a central point of failure for this architecture. Crosbie and Spafford [7], proposed distributed IDS where communication IDSs would broadcast activities that have been flagged as malicious activities between themselves to assist in intrusion detection. On distributed IDSs that utilized artificial immune system (AIS), Hosseinpour et al., [8] proposed a DIDS based on the AIS which uses a central engine. All the participating IDSs are synced with this central engine which also serves as a middle-man between two IDSs intending to share a detector record with each other. Afzali and Azmi [9] proposed a multi-agent AIS (MAIS-IDS) approach. MAIS-IDS has higher recognition accuracy by collaboration between virtual machines than individual works.

Our approach to solving this problem differs in the sense that it uses a fully distributed NIDS approach where no central controller is needed, and participating NIDS (each located at the entry point of the network to which they are a member of) communicate with each other directly. And instead of sending all “suspicions” of intrusions to all agents, it only informs the agents of actual intrusions that it was successful in detecting.

III. Artificial Immune System

The algorithms that exist in the field of AIS exploit the immune system's characteristics of learning and memory to solve diverse problems. These algorithms are based on human immune system (HIS) models taken from the field of immunology. Immunology uses models for understanding the structure and function of the immune system. The self-nonself (SNS) model is an immunology model that have been successfully utilized in AIS in the design of IDS systems to detect network attacks; both insider and outsider. [10] [3] [11].

A. *Self-nonself (SNS) Model*

This model focuses on the adaptive nature of the immune system, i.e., it uses the adaptive immune system and its memory or self-learning capability. In this model, the B cells (which are called detectors in AIS) would have antigen specific receptors that can recognize non-self or foreign bodies and in turn initiate an immune system response that is specific to the system where this AIS model is applied. In this technique, the first step according to Forrest et al. [11] involves randomly generating detectors (which is the AIS’s equivalent of B cell in HIS). These detectors that are still immature are then exposed to a set of self-structures. Any detector that reacts or matches any member of the self set is eliminated. The remaining members of the detector set that were unreactive with any member of the self set become mature detectors. This detector selection technique is called negative selection and the algorithm used to perform this computation is called a negative selection algorithm (NSA) [11].

When a mature detector encounters a pathogen or non-self entity that it has been exposed to previously during the negative selection phase, it mounts a very rapid and efficient response that is called the *secondary response*. When a new pathogen that does not bind with any mature detector is encountered, the immune system mounts a *primary response* during which the immune system tries to learn the pattern of this unseen pathogen so that it can mount a secondary response next time the same pathogen shows up.

From the discussions so far in this section, it is seen that AIS would be a good approach to solving the security problems inherent in computer networks. This is because computer networks are similar to the human body; thus, a digital immune system can be created for networks to fight intrusion just like the way the immune cells fight pathogens. In dNIDS, especially in a fully dNIDS, it is necessary that information about attacks be exchanged between constituent NIDS. This exchange involves moving attack information across the network from one NIDS to another. This migration technique is similar to the way the immune cells recirculate through the blood, homing at various secondary lymphoid organs where they interact with foreign antigens.

IV. Proposition of a new Approach to dNIDS Design Based on Negative Selection

A. *The Proposed Approach Objectives*

Attack strategies employed by computer hackers have become complex and well-coordinated in recent years. Worms that propagate and reproduce in a network have become prevalent. Also, given that most organizations have multiple sites spanning across an entire country or state, if a NIDS agent is installed on these sites (at the entry point of each) and made to share information about intrusions, this could lead to an effective and fast way to detect attacks across the organization’s network. The goal of this proposed approach is to build a distributed NIDS to detect attacks across an organization’s network. Also, this dNIDS would able to detect zero-day attacks through interactions with other NIDS in another geolocation.

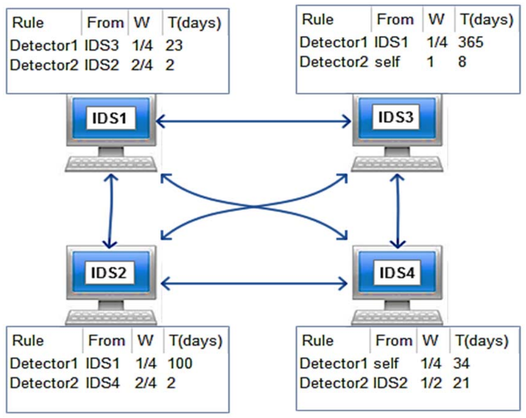
B. *Description of the Proposed Approach*

We propose an approach to dNIDS design that uses the negative selection algorithm derived from AIS explained in the preceding section. This dNIDS would consist of autonomous agents that communicate with each other. See Fig. 1. These agents: IDS1, IDS2, IDS3 and IDS4 (all of which are network-based, i.e., NIDSs) independently run the NSA to generate rules for detecting and classifying network intrusions into self and nonself. Each NIDSs has a table of rules or detectors and due to their inter-communication with one another, they also have information about detection rules from other NIDS.

The detector that matches an attack after detector generation and training would be promoted to a memory detector. It is these memory detectors that IDSs exchange with each other. The mature detectors (i.e., detectors that have been generated, trained, but have not matched any attack over the course of its live span) is not part of this detector exchange.

In Fig. 1, T (in days) represents how long a detector has been in existence from the time it became a memory detector. T can be used with a chosen detector lifecycle to eliminate expired rules from the table.

Since the NIDS agents are running the NSA independently, a duplicate of a particular rule (detector) could be received. Each IDS agent assigns a weight called W to each detector rule. In Fig. 1, IDS1 agent received detector 2 from two out of the 4 communicating IDS agents. W can be used as a means to determine if an entry in the table comes from a false positive. This is because, if two geologically disperse NIDS that was trained using different data reports that same memory detector, then it is highly likely that this detector is a true positive. It can be seen that IDS1 agent only keeps a single copy, and fills up the "from" field of the table with the name of the first IDS agent (i.e. IDS2) that informed it of this new rule.



Rule	From	W	T(days)
Detector1	IDS3	1/4	23
Detector2	IDS2	2/4	2

Rule	From	W	T(days)
Detector1	IDS1	1/4	365
Detector2	self	1	8

Rule	From	W	T(days)
Detector1	IDS1	1/4	100
Detector2	IDS4	2/4	2

Rule	From	W	T(days)
Detector1	self	1/4	34
Detector2	IDS2	1/2	21

Fig. 1. Rule table as seen in each network intrusion detection system.

C. *Implementation*

In this section, implementation details are presented. Implementation of the distributed NIDS framework involves the following steps:

1. Data capture phase.
2. Feature selection phase.
3. Data preprocessing phase.
4. Detector generation phase.
5. Monitoring phase.
6. Memory detector distribution phase.

1) Data capture:

In this phase, traffic flowing into the network is captured by the NIDS agents positioned at the entry point of each subnetwork. This data capturing can be achieved with network packet sniffer tool like Tcpdump [15].

2) Feature selection phase:

Feature selection is the process of selecting a subset of relevant features for use in model construction. The central premise when using a feature selection technique is that the data contains many features that are either redundant or irrelevant, and can thus be removed without incurring much loss of information [16].

The information gain (IG) feature selection technique which falls under the filter feature selection method was used in this phase. For details about this technique, see [17].

3) Data preprocessing:

In data preprocessing, the data related to the features selected in the preceding subsection is analyzed. All the nominal data like the protocol type is converted into a suitable representation which could be binary or an integer value depending on the data classification algorithm requirements. This is because most data classification techniques are affected by noisy data and extreme values. Hence, selected features with unormalized continuous values, x_i are normalized to a value between 0 and 1 using (1):

$$X_{i,[0,1]} = \frac{x_i - x_{min}}{x_{max} - x_{min}} \quad (1)$$

4) Detector generation:

The first NSA algorithm [11] which was proposed by forest et. al. is an exhaustive approach. The limitation of this approach is the computational difficulty of generating valid detectors, which grows exponentially with the size of the self [18]. So, to solve the problems of the exhaustive approach, we need a technique for implementing the NSA which locates a detector instead of selecting them at random as in the case of the exhaustive approach. For this, we employ an evolutionary approach using a genetic algorithm (GA).

GAs are adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics. As such they represent an intelligent exploitation of a random search used to solve optimization problems. Each generation consists of a population of character strings that are analogous to the chromosome that we see in our DNA. Each individual represents a point in a search space and a possible solution. The individuals in the population are then made to go through a process of evolution [19]. Algorithm 1, shows the steps taken by our approach to detector generation using GA.

A detector is defined as $d = (c, r_d)$, where $c = (c_1, c_2, \ldots, c_m)$ is an m-dimensional point that corresponds to the center of a unit hypersphere with r_d as its radius. In this detector generation phase, the main task is to generate a set of detectors, with the center of each detector being at least $(r_d + r_s)$ distance away from the center of its nearest self element (which has radius = r_s).

The GA is initialized with random individuals. Two parents; P_1 and P_2 are drawn from the initialized samples at random with equal probability. Crossover is performed with these two parents to create two offspring (C_1 and C_2) with a probability C_p called the crossover probability. The kind of crossover used here is the one-point crossover.

Algorithm 1: NSA using Genetic Algorithm

```
Input: S∈U ("self-set"); $r_S$ =self radius
Output: a set of detector D∈U ("detector set")
1:  population ← random individuals
2:  for the specified number of generations do
3:      for the size of the population do
4:          Select two individuals, (parent1 and parent2),
            with uniform probability.
5:          Apply crossover with probability $C_p$ to
            generate two offspring (child1 and child2).
6:          Mutate child1 and child2 with probability $M_p$
7:          If distance (child1,parent1) > $r_S$ and fitness
            (child1) > fitness (parent1) then
8:              parent1 ← child1
9:          end if
10:         If distance (child2,parent2) > $r_S$ and fitness
            (child2) > fitness(parent2) then
11:             parent2 ← child2
12:         end if
13:     end for
14: end for
```

To perform mutation on C_1 and C_2, a position in the chromosome of each is chosen at random from [1,2…m] and flipped with a mutation probability M_p. This flipping is done by replacing the value of the attribute in the randomly selected location with a randomly generated value that lies between [0, 1].

After mutation, the fitness of C_1 and C_2 is evaluated using the function in (2):

$$fitness(individual) = e^{-r_s/D} \qquad (2)$$

Where r_s is a threshold value (allowable variation) of a self point; in other words, a point at a distance greater than r_s from the self sample is considered to be abnormal. D is the distance between the individual and the nearest self. This distance measure is calculated using the Euclidean distance measure given by (3):

$$D(X,Y) = \left(\sum_{i=1}^{n}(x_i - y_i)^2\right)^{1/2} \qquad (3)$$

The fitness of both parents used for the crossover together with the resulting offspring is calculated. Also, the distance between (C_1 and P_1) and (C_2 and P_2) are both calculated. P_1 in the population is replaced by C_1 if C_1 has a better fitness and the distance between its center and that of P_1 is greater than r_s. Similar action is taken for replacing P_2. The fittest individuals are selected as the detectors.

5) Monitoring/Testing phase:

After the detectors have been generated using NSA, it is tested to see how many of these detectors would match to a nonself entity. If the distance between the center of a test sample and the center of the nearest detector is less than or equal to r_d (which is set be equal to r_s to create hyperspheres of the same size i.e. fixed size detectors), this test sample is said to be abnormal (an attack). Otherwise, it is classified as normal (self).

6) Memory detector distribution:

A dNIDS table distribution application is written using the python programming language to facilitate the sharing and updating of the detector table explained in section 4. Every participating NIDS runs this application.

Once a detector is promoted to a memory detector in a NIDS running this application, a thread in the application would update the table with this new rule (i.e., detector). After an interval called the update interval (usually set to a small value), this new rule is communicated to all the NIDS directly connected to originating NIDS just like in distance vector routing algorithms.

The content of the update includes the rule(s) that haven't been shared since the last update interval and the time since each of these detectors or rules became a memory detector. The receiving NIDS(s), updates its table by first searching to see if an entry for this detector exists. If an entry exists, it updates W using (4):

$$W = \frac{no\ of\ times\ rule\ was\ received}{no\ of\ participating\ NIDS} \qquad (4)$$

If this rule entry does not exist, it is created, and the numerator of (7) becomes one. Also, the "from" field is filled up with the ID of the sending NIDS, which in this case is the name of the NIDS. Fig. 2, shows the distributed network of NIDS. The red circles indicate the NIDSs with an unsent update. These NIDSs will wait until the next update interval.

After an update, the nodes become blue colored. From Fig. 2, it is seen that the NIDSs are positioned at the entry point of each site's network. This allows all the traffic passing through the network in each site to be inspected by its NIDS. Given that all these sites are owned by the same organization, the table distribution application has a global site view of the status of all NIDS.

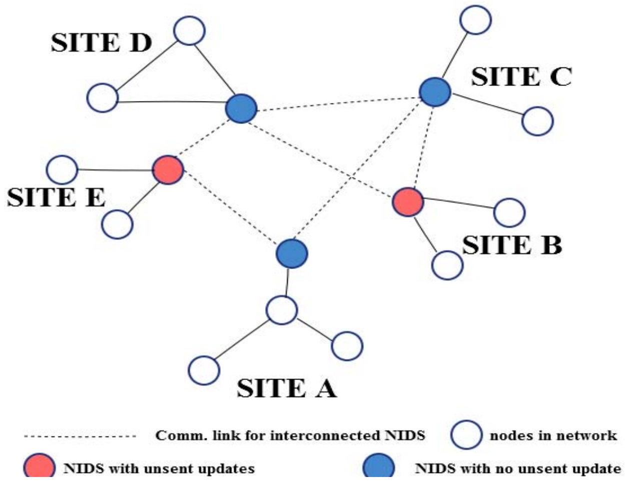


Fig. 2. Topology showing interconnected distributed network intrusion detection systems.

V. Experimental Results and Discussion

The first 5 stages that constitute this dNIDS implementation technique (data capture, feature selection, data preprocessing, detector generation and monitoring phase), are initially carried out in a test network where the yet-to-be-deployed NIDSs are trained with the company's "self" traffic to recognize non-self elements. On completion of the training, these NIDSs are moved to the distribution network. After these NIDSs become live, they execute the last phase which involves memory detector distribution. Going forward, packets passing through the networks protected by these NIDSs are copied by the NIDS in the destination network and compared to the detectors stored in detector table. If it matches a Detector (i.e. falls within the detection radius), this packet is flagged as an intrusion. Given that these NIDSs are trained to differentiate between self and non-self, with good detector coverage, new attacks that have not been seen before (zero day attacks) or even variants of the one the company is already aware of, can also be detected.

A. Data Capture and Feature Selection

For the data capture phase explained in section IV, NSL-KDD [20] dataset was used. This dataset consists of 41 features and 4 different attack classes. The KDDTrain+20% was used to train the detectors. The KDDTrain+20% consists of 25192 instances out of which 13449 is normal data and 11743 is the attack data. For detailed information about this dataset, see [21].

The 41 features were reduced to 8 using IG to ensure a small detector table size. These 8 features were selected based on their information gain shown in Fig. 3. Only the features with information gain above 0.42 (line 1 of Fig. 3) were selected i.e. 5,3,6,4,30,29,33 and 34. Table 1 below shows the mapping of the IG selected features to its name and a brief description of each.

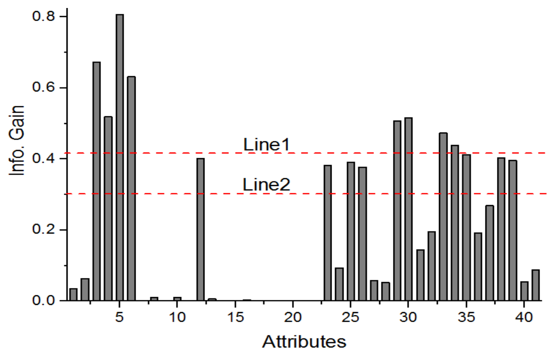


Fig. 3. Information gain of all forty-one features of the dataset.

TABLE I. Description of Selected Features

Feature	Description
5	**(src_bytes):** Number of data bytes transferred from source to destination in single connection
3	**(service):** Destination network service used
6	**(dst_bytes):** Number of data bytes transferred from destination to source in single connection
4	**(flag):** Status of the connection – Normal or Error
30	**(diff_srv_rate):** The % of connections that were to different services, among the connections aggregated in count
29	**(same_srv_rate):** The % of connections that were to the same service, among the connections aggregated in count
33	**(dst_host_srv_count):** The % of connections that were to the same service, among the connections aggregated in dst_host_count
34	**(dst_host_same_srv_rate):** The % of connections that were to different services, among the connections aggregated in dst_host_count

B. Negative selection and detector distribution

For the Genetic algorithm used for the negative selection, the following parameter values were chosen: number of generations: 100; mutation prob. (M_p): 2/(no of features); crossover prob. (C_p): 1.0 and a crossover point of 3.

Sixty percent (60%) of the normal samples drawn from the NSL-KDD training data (specifically the KDDTrain+20%) is used as the self samples for the NSA while the remaining forty percent (40%) and all the anomaly samples are used for testing the generated detector after training.

In order to compare performance with other classification techniques that could be employed in the dNIDs, we also apply support vector machine (SVM), Naïve Bayes and J48 (see [22] for more information on these classifiers) to the same dataset with the same selected features. These techniques are compared in terms of accuracy. The idea is to calculate the number of true positives (TP, anomalous elements identified as anomalous), true negatives (TN, normal elements identified as normal), false positives (FP, normal elements identified as anomalous) and false negatives (FN, anomalous elements identified as normal). These values are, however, used to calculate two measures of effectiveness):

$$Detection\ Rate(DR) = \frac{TP}{TP + FN} \quad (5)$$

$$False\ alarm\ rate\ (FA) = \frac{FP}{TN + FP} \quad (6)$$

In general, we want a very high detection rate with a very low false alarm rate. However, there is a tradeoff between these two measures. Fig. 4, shows the different FA and DR values gotten for the different intrusion detection approaches.

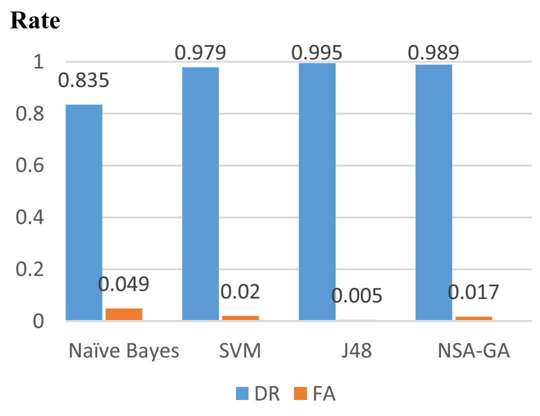


Fig. 4. Summary of detection result on the KDDTrain+20% dataset.

The NSA-GA was selected as the classifier to be used in this dNIDS framework instead of the J48 because to train the NIDSs (which will later become part of the dNIDSs) using NSA-GA, only the self samples were used, unlike the J48 which requires both self and nonself samples to build detection rules. This requirement by the J48 makes the J48 susceptible to strains of attacks; even strains of attacks it encountered during the training phase. By training with only self samples, the NIDSs using NSA-GA need not know the nature of abnormal activities or how their traffic statistics look like before being able to detect such (similar to how the human body need not know the nature of a pathogen before detecting its presence). To increase the DR, a high number of GA generation run is recommended. And to reduce the FA, the detection radius can be reduced from 0.4 used in this experiment to 0.1, but the only challenge in doing this is that this will lead to an increase in the number of detectors needed to fill up the search space.

To test the detector distribution, the developed table distribution python application was installed on 4 out of 16 Ubuntu 14.04 [23] virtual machines which are connected to form the topology shown in Fig. 2. Here, 4 subnetworks are created with each having a NIDS (that runs the NSA) at its network input. 300 random attacks are selected from the NSL-KDD test data which was not used for the training or monitoring phase, and this was considered as a zero-day attack. These attacks were presented to a NIDS in one of the machines. This NIDS was successful in detecting 207 of these attacks after a processing time of 4 secs. After identification of these attacks by the NIDS, the detectors that matched these new attacks were sent to the neighboring NIDSs after a duration of 3 secs. The same attacks were launched against these receiving NIDSs and they were able to utilize the detectors they received in classifying the attacks within a processing time of less than 1 sec. Hence, these attacks which could have been a zero-day attack to the receiving NIDS was successfully detected and classified. The size of the table in each NIDS is proportional to the number of NIDSs that make up the dNIDS (which in turn is proportional to the number of subnetworks).

VI. Conclusion

In this paper, we have managed to describe the framework for a fully distributed NIDS that utilizes the NSA in its constituent NIDS. Genetic algorithm was explored as a technique for generating the detectors which form the building block of the NSA. Also, the performance of other techniques including J48, SVM and Naïve Bayes which could be used as an alternative agent (running on the NIDS) were compared to our proposed algorithm. A technique for distribution of detectors between participating NIDS was also explained. Our future work will expand on this area further by implementing a fully distributed NIDS that that uses both the self-nonself model and another AIS algorithm called danger theory for anomaly detection, and other techniques to facilitate intercommunication between NIDS.

References

[1] J. Thomas and A. Abraham, "Distributed Intrusion detection systems: a computational intelligence approach, applications of information systems to Homeland Security and Defense," chapter 15, pp. 105-135, 2005.

[2] P. Barford, S. Jha, V. Yegneswaran, "Fusion and filtering in distributed intrusion detection systems," In Proc. Annual Allerton Conference on Communication, Control and Computing, September, 2004.

[3] M. M. Elhaj, H. Hamrawi, M. Suliman, "A multi-layer network defense system using artificial immune system," In Computing, Electrical and Electronics Engineering (ICCEEE), 2013 International Conference on pp. 232-236. IEEE.

[4] N. Bashah, I. B. Shanmugam, and A. M. Ahmed, "Hybrid intelligent intrusion detection system," Proceedings of World Academy of Science, Engineering and Technology, vol. 6, June 2005.

[5] P. Kazienko and P. Dorosz, "Intrusion detection systems (IDS) Part 2 - classification; methods; techniques," Online. Retrieved 1 January 2016, from http://windowssecurity.com

[6] S. Snapp, J. Brentano, and G. Dias et al., "DIDS (Distributed Intrusion Detection System) – motivation, architecture, and an early prototype," In Proceedings of the 14th National Computer Security Conference, October 1991.

[7] M. Crosbie and G. Spafford, "Defending a computer system using autonomous agents," Technical Report 95-022, COAST Laboratory - Purdue University, 1994.

[8] F. Hosseinpour, A. Meulenberg, S. Ramadass, P. Vahdani, and Z. Moghaddasi, "Distributed agent based model for intrusion detection system based on artificial immune system," Int. J. Digit. Content Technol. its Appl., vol. 7, pp. 206–214, 2013

[9] N. Afzali and R. Azmi, "MAIS-IDS: a distributed intrusion detection system using multi-agent AIS approach," Eng. Appl. Artif. Intell. vol. 35, pp. 286–298, 2014

[10] A. EshghiShargh, "Using artificial immune system on implementation of Intrusion detection systems", Third UKSim European Symposium on Computer Modeling and Simulation, pp. 164 168, 2009

[11] S.A. Hofmeyr and S. Forrest, "Architecture for an artificial immune system," Evolutionary Computation, vol. 8, no. 4, pp. 443–473, 2000

[12] P. Matzinger, "Tolerance, Danger and the Extended Family," Annual Review of Immunology, Vol. 12, pp. 991-1045, 1994.

[13] J Greensmith, U Aickelin, S Cayzer, "Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection," Proceedings ICARIS-2005, 4th International Conference on Artificial Immune Systems, LNCS 3627, pp. 153-167, Springer-Verlag, 2005.

[14] U. Aickelin and J. Greensmith, "Sensing danger: innate immunology for intrusion detection," In Information Security Technical Report, ISSN 1363-4127 (In Press). ELSEVIER, 2007

[15] Tcpdump. http://www.tcpdump.org

[16] Feature selection. In Wikipedia, The Free Encyclopedia. Retrieved 02:23, January 7, 2016, from https://en.wikipedia.org/w/index.php?title=Feature_selection&oldid=695639260

[17] A. Moore, "Information Gain," Lecture Notes. 2003. http://www.autonlab.org/tutorials/

[18] D. Dasgupta, F. Nino, "Immunological computation: theory and applications," Auerbach Publications, 2008.

[19] Introduction to Genetic Algorithms, http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.html (accessed January 2, 2016).

[20] "Nsl-kdd data set for network-based intrusion detection systems." Available on: http://nsl.cs.unb.ca/KDD/NSLKDD.html, March 2009.

[21] L. Dhanabal, S.P. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," International Journal of Advanced Research in Computer and Communication Engineering Vol. 4, Issue 6, June 2015.

[22] Classification methods,http://www.d.umn.edu/~padhy005/Chapter5.html

[23] Ubuntu 14.04.3, http://releases.ubuntu.com/14.04/