Auto-Immunity Dendritic Cell Algorithm

Olubadeji Bukola
Department of computer science
Federal University of Technology, Akure

Adetunmbi A.O.
Department of computer science
Federal University of Technology, Akure

ABSTRACT

Security of information is of utmost importance to any organization or individual, which depend on computer system or internet for business transaction or source of information or research. Many viruses are able to recognize certain anti-virus software, and respond differently to such software than to programs designed for other purposes. Some viruses go after the databases stored by anti-virus products. Some viruses simply go after anti-virus products, trying to erase them. Immune systems also face this daunting control challenge. On the one hand, they need to minimize damage from pathogens, without wasting energy and resources, but on the other must avoid initiating or perpetuating autoimmune responses.

Several preventive measures including identification and authentication, logic access control, audit trails, digital signature and firewalls have been developed for the purpose of information security on system. As a result of inadequacies of these measures intrusion detection was introduced to complement these techniques and hence guarantee full protection of computing resources. Detection system is the process of identifying and detecting unauthorized access or abnormal incursions, actions and events in the system, which provides information for timely counter measures.

This paper presents a systematic approach to intrusion detection using artificial immune system (Dendritic Cell) to purging in order to avoid attack subversion and autoimmunity on network. In nature, dendritic cells function as natural anomaly detection agents, instructing the immune system to respond if stress or damage is detected. Dendritic cells are a crucial cell in the detection and combination of ‘signals’ which provide the immune system with a sense of context. The Dendritic Cell Algorithm which is based on an abstract model of dendritic cell behaviour, with the abstraction process performed in close collaboration with immunologists will be used. This algorithm consists of components based on the key properties of dendritic cell behaviour, which involves data fusion and correlation components. In this algorithm, four categories of input signal are used.

The DCA algorithm will be validated with a standard machine learning dataset. The validation of the Dendritic Cell Algorithm is performed. This is assessed through the algorithm’s application to the detection of intrusion and classification problems.

Keywords

Dendritic Cell Algorithm (DCA), Security, Auto – immune, K – means, Algorithms

1. INTRODUCTION

Purging is one of the approaches reviewed by [11] that can facilitate strategic robustness. It can be define as a systematically and permanently removal of infected computers from the network, purging is stronger than delete. It is often possible to regain deleted objects by undeleting them, but purged items are gone forever, Purging is only effective when individual replication rates are sufficiently large to tolerate the effects of removal of defective components. Thus, in immune system apoptosis (programmed cell death) is a common strategy for eliminating cells upon damage to their genomes or upon infection, provided these cell types are capable of regeneration. Nerve cells and germ cells produce factors which strongly inhibit apoptosis and removal, in these cases has deleterious consequences. In severe infection, it can make sense to purge nerve cells [12]. The vertebrate adaptive immune system uses an anti-robust strategy to deal with pathogens. They respond to signs of trouble by purging the damaged or infected cells, rather than by trying to stabilize these cells to help them live with the problem. This can be an effective way of dealing with a threat that otherwise can propagate, and anti-robustness at the cellular level confers robustness at the (multicellular) organismal level Krakauer, (2005).

Physiologists consider the purpose, function or goal of a biological structure when trying to understand how that structure works. Immunologists do the same thing. The goal of any immune system is to protect against pathogens and these systems have therefore evolved to increase the fitness of the organism by reducing the damage caused by such organisms [14] ideally without wasting energy and resources [16]. To use this functional approach successfully, one must account for the tradeoffs and constraints that organisms face. Here, we focus on purging which is an act of responding to signs of trouble and damaged or infected systems on the network, rather than by trying to stabilize the computer on the network to help them live with the problem. This can be an effective way of dealing with a threat that otherwise can propagate.

2. RELATED WORK

Since the early 1990s, biologically-inspired computing has become one of the most active interdisciplinary research topics. It generally involves biology, computer science, engineering and mathematics. Inspiration were been drawn from nature, to create computer systems encapsulating the appealing properties of biological systems.

Biologically-inspired computing is intimately related to the field of artificial intelligence, due to its tremendous influence on many machine learning techniques. examples include: [17] work on genetic algorithms that are inspired by the evolutionary process of organisms; cellular automata which mimics cells evolving based on the states of their neighbouring cells through a number of discrete time steps by [3]. A new research area, Artificial Immune Systems (AIS), has emerged, which focuses on drawing inspiration from theoretical immunology and observed immune functions, principles and models, [5] develop computer systems for real-world problem solving.

The natural immune system is evolved to protect the body from a wealth of invading micro-organisms, and AIS are designed and developed to provide same defensive properties within a computing context. It is believed that AIS could be advantageous over other conventional techniques in certain cases, due to a rich set of properties mapped from the natural system.

Conventional Solutions to Intrusion Detection

Anomaly detection can be considered a binary classification problem, and techniques that are capable of performing binary classification are often applied to this problem. Machine learning techniques are popular in this domain, as they are able to learn from experience to separate insput data instances into the correct classes [18]. The application of machine learning techniques to particular problems refers to data mining. [13] propose a way of developing general and systematic methods for intrusion detection, using data mining approaches. A Multi-Layered Feed Forward (MLEF) neural networks approach was applied to discriminate between normal and abnormal behaviours on a user behaviour level in [20]. A number of popular supervised learning techniques were applied to a standard intrusion detection dataset, known as the KDD 99 dataset [2], which was derived from the Lincoln Lab DARPA 98 dataset.

Firstly, these conventional techniques usually sacrifice computational complexity for improving their learning and generalization ability, especially for the training phase of supervised techniques. This makes them relatively disadvantageous to techniques with low runtime complexity (e.g. linear), in terms of the scalability to data size and dimension-ality. Moreover, in order to deal with noise contained in the input data, they normally require the help of filtering techniques at either pre-processing or post-processing, which could add extra complexity to the system. Correctly classifying each data instance into the right classes, rather than correlating the potential causes to the identified anomalies through for example temporal correlation. They are often lack of the capability of dealing with the anomaly attribution aspect of anomaly-based intrusion detection. As a result, conventional techniques might not be suitable for the particular problem of interest, while solutions from other emerging areas, e.g. AIS, could be potentially advantageous.

AIS in Intrusion Detection

AIS are designed through encapsulating the perceived potentially useful properties of the natural immune system, for the protection and maintenance of a host. These properties include distributed, self-organised, lightweight, multi-layered, diverse and disposable [2]. Due to such properties, immune-inspired algorithms are often applied to intrusion detection problems that require robustness, configurability, extendibility, scalability, adaptability, global analysis and efficiency [9]. Some applications have shown that AIS approaches could be advantageous on detection performance over conventional techniques [7] In this section, a number of successful cases are reviewed to show their strengths and weaknesses. This is based on a review of immune system approaches to intrusion detection published by [10], with updates of more recent work.

Negative Selection Based Approaches

The body of research forming the development and application of the negative selection algorithm has formed a major contribution to the field of AISs. A naïve abstraction of the negative selection process is used as a metaphor of what immunologists describe as central tolerance. This naive approach has advantages: the resultant algorithm is easy to understand in a machine learning context, as it consists of training and a test process; the filtering process has relatively few steps. Negative selection has enjoyed some success when applied to real world data, as in the case of developing fault detection systems for cash machines [1] and aeroplanes [4]. However, when applied to anomaly detection, it suffers from scaling issues, detector coverage problems and the generation of excessive undesirable false positives. The criteria of the application of AISs to computer security is to develop systems which are robust, lightweight (in terms of computational overhead) and have the ability to detect novel attacks.

Aickelin proposes that through close collaboration with immunologists, computer scientists will be able to develop more biologically realistic However, existing immune-inspired techniques have not performed as well as expected when applied to the detection of intruders in computer systems. A novel immune-inspired algorithm based on the function of the dendritic cells of the human immune system was proposed to overcome the shortfall of the existing techniques. In nature, dendritic cells function as natural anomaly detection agents, instructing the immune system to respond if stress or damage is detected. Dendritic cells are a crucial cell in the detection and combination of ‘signals’ which provide the immune system with a sense of context. In this paper, the detection of anomalous on the network forms the application area used to demonstrate the use of a DC based algorithm applied for computer security. K – Means will be used to classify objects with similar features to the same cluster.

3. OVERCOMING SUBVERSION BY PATHOGENS AND AUTOIMMUNITY

The paper focus on the two main issues that have been instrumental in immune evolution: (i) Autoimmunity: immune systems need to minimize the risk of autoimmunity. A single autoimmune mistake is potentially lethal, if directed against essential components of the body; and (ii) attack subversion on computer network: immune systems must be strategically robust. The network need to work in ways that rapidly evolving pathogens cannot exploit, subvert, or sabotage.

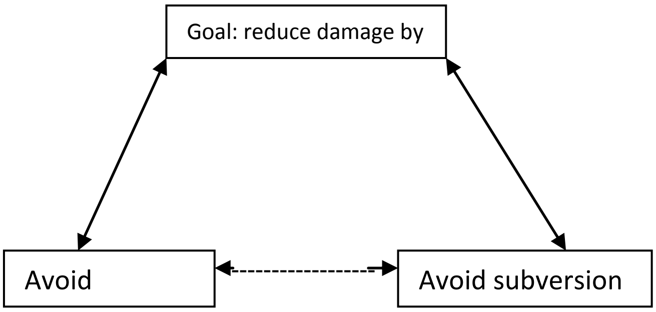


Figure1. Artificial Immune System

An effective immune system must prevent damage from infection while avoiding subversion by pathogens and autoimmune mistakes. As indicated by the dashed arrow, efforts to avoid autoimmunity might compromise efforts to avoid subversion, and vice versa.

Many systems, including the immune system, must be robust: they need to operate in a range of background conditions, function in the presence of noise and despite variation in internal structure, and keep working even if multiple internal components fail. Systems that have to deal with internal subversion must go one step further and be strategically robust: that is, they need to function properly despite efforts to sabotage their workings.

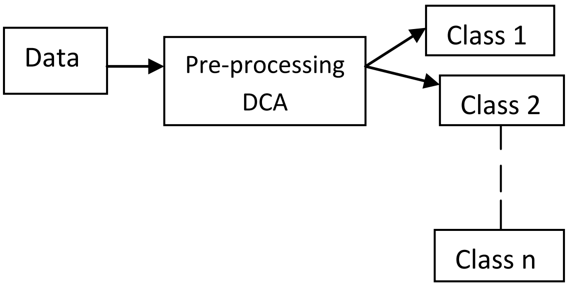
The distinction between robustness and strategic robustness becomes clear through analogy. A robust computer circuit would function effectively even if a few resistors burned out at random. A strategically robust computer circuit would function even if a disgruntled technician tried to sabotage the machine by removing precisely those resistors that were most crucial. As the system goes from robustness to strategic robustness, its like going from a simple optimization problem to a game-theoretic one, in which antagonists each try to maximize their own payoffs at the possible expense of the others. Thus, the task of implementing strategic robustness is as much in the spirit of the mechanism design problem from economics [19] in which the designer aims to set up the rules of the game so as to make multiple self-interested players behave as the designer wishes, as it is in the spirit of control theoretic approaches from engineering.

The task of an immune system in (Figure 1) is difficult because efforts that meet one challenge often compromise efforts to meet another. To avoid autoimmunity, immune systems must have ways of terminating accidental self-directed responses; however, these 'shutdown' pathways can be strategic vulnerabilities. Attack can and do evolve to exploit the mechanisms that immune systems use for self-regulation [6]. This paper explores the ways that immune systems deal with the challenges of strategic robustness and autoimmune avoidance.

Design principles for overcoming attack subversion on the computer network

Robust biological systems use multiple strategies to function effectively across a range of conditions, despite internal and external noise and variation, and component failure. Purging is one of the design principle that facilitate strategic robustness which respond to signs of trouble by purging the damaged or infected cells, rather than by trying to stabilize these cells to help them live with the problem.

3.1 Application of the DCA to facilitate robustness of computer network



Focus is on the algorithm's roles in performing the anomaly detection from a conceptual perspective, the DCA will be used as an intrusion detection and classification system. Suppose there is a monitored system where data is collected at a regular basis, e.g. at every second, and the task is to detect the system's misbehaviour, i.e. anomalies. The collected data consists of two types. One is related to quantitative measures of the system's behaviour, represented as signals. The other is some identifiers that could be linked to the potential causes of anomalies, represented as antigens. Three phases are usually involved, namely pre-processing, detection and analysis, is described as follows.

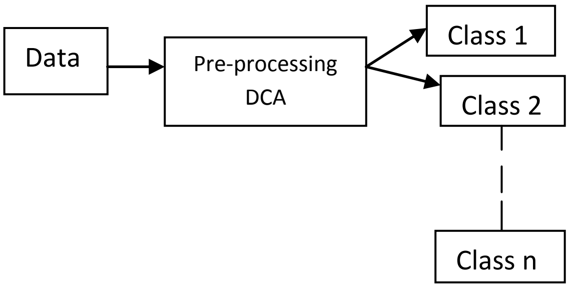


Figure 2: Architecture of DCA classification system

3.2 Pre-Processing Phase

Pre-Processing Phase

The application of the DCA often requires a data pre-processing phase to appropriately map a given problem domain to the input space of the algorithm. The pre-processing phase of the DCA is of interest. The pre-processing phase of the DCA usually involves signal selection and categorization, to generate the input signal stream of the algorithm. Signal selection is required to select the most interesting features from the original feature set. This is equivalent to the task of feature abstraction or selection in the area of machine learning, which is often accomplished by applying dimensionality reduction techniques.

The DCA has the ability of building a good classifier even on small training some dataset but work only on discretized data. Since, real life data is made of both or either continuous and discrete attributes valves then the need for discretization before training commence. Discretization can be defined as set of cuts over domains of attributes, representing an important pre-processing task for numeric data analysis.

The numerical (continuous) attributes in dataset are discretized based on Entropy, a supervised splitting technique exploring class distribution information in its calculation and determination of split-point. Entropy discretization technique leads to reduction of data size and makes use of class information, which may assist in improving classification accuracy. In discretizing a numerical attribute A, the value of A with the minimum entropy value is selected as split-point, and the resulting intervals are recursively partitions to arrive at a hierarchical discretization computer as follows [8].

Given D consisting of data tuples defined by a set attributes and a class label attribute

A split-point for A can partition the tuples in D into two subsets satisfying the conditions

A ≤ split point and A > split point respectively, thereby creating a binary discretization.

The expected information requirement for classifying a tuple in D based on partitioning by A is given by

$$Info_A(D) = \frac{|D_1|}{|D|}Entropy(D_1) + \frac{|D_2|}{|D|}Entropy(D_2)$$

Where D_1 and D_2 correspond to the data tuples in D satisfying the conditions A $\leq$ split point and A > split point respectively. |D| is the number of tuples in D.

The entropy function for a given set is computed based on the class distribution of the tuples in the set. For example, given n classes, $C_1, C_2, \ldots C_n,$ the entropy of D_1 is

$$Entropy(D_1) = \sum_{l=l}^{n} P_1 \log_2 (p_1) \tag{4.4}$$

Where P_1 is the probability of $C_1\ D_1$, determined by dividing the number of tuples of C_1 in D_1 by $|D_1|$, the total number of tuples in D_1.

Hence, in selecting a split-point for attribute A, the chosen attribute is the one with attribute value that gives the minimum expected information required (i.e. min (*Info*^{A} *(D))*. The process of determining a split-point is recursively applied to each partition obtained until the information requirement is less than a small threshold $\varepsilon(0)$.

3.2.1 Data

The NSL-KDD data set is used to validate the DCA and is a well understood two-class data set. The data consists 125972 items, classified by their corresponding real valued attributes, with membership of the data items to either class one or class two. These data can be used to give preliminary figures of DCA performance in terms of accuracy and precision. The data ID is used to form antigen, with a pre-processed subset of attributes used to form the signals.

The raw data of NSL KDD before pre-processing

```
0,tcp,private,REJ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,121,19,0.0
0,0.00,1.00,1.00,0.16,0.06,0.00,255,19,0.07,0.07,0.00,0.00,0.0
0,0.00,1.00,1.00,normal,21
```

```
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,166,9,1.00,1
.00,0.00,0.00,0.05,0.06,0.00,255,9,0.04,0.05,0.00,0.00,1.00,1.
00,0.00,0.00,neptune,21
```

Generated cut_point during pre-processing

```
cut_point0[60]={0.5,21,26,113.5,127,148.5,210.5,213,215,22
5.5,241,260,261.5,11305,12437,13517,13553,14873,15068,15
214,15361,15397,15508,15705,18062,21010,21320,21786,23
586,23910,24014,24024,25018,25044,25084,25220,25390,25
672,25678,28006,28024,29964,112624,112824,116344,11866
4,120944,128912,176368,210080,210096,210160,211920,212
528,215168,216816,217232,218688,220864,232848,};
```

this is the pre-processed NSL-KDD Dataset

```
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1.
```

```
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1.
```

```
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,2.
```

```
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,2.
```

Step to be taking in executing DCA

1. The signal transformation function O : $Time \rightarrow \mathbb{R} \times \mathbb{R}$ is:

$$O(t) = \begin{cases} W^T S_{(t),} & if\ S(t) \epsilon\ Signal; \\ 0, & otherwise \end{cases} \tag{2}$$

This operation is executed whenever $S(t)\epsilon\ Signal\ holds$, and it performs the multiplication between a transposed $2 \times m$ matrix and a m-dimensional vector to produce a two-dimensional vector of output signals, namely `CSM' and `K'.

2. The lifespan update function $F : Time \times Population \rightarrow \mathbb{R}\ is\ defined\ as$

$$F\ (t, i) = \begin{cases} I(I), & if\ t = 1; \\ I(i) - \pi_i\big(O(t)\big), & if\ F(t-1, i) \leq 0; \\ F(t-1, i) - \pi_1\big(O(t)\big), & otherwise \end{cases} \tag{3}$$

When $t = 1,$ the initial value of F is $I(i)$, which is the initial lifespan of the DC with an index i. It is repeatedly subtracted by CSM signal until the termination condition,$F(t-1, i) \leq 0$, is reached.

3. The signal profile updates function $G : Time\ \ Population \rightarrow \mathbb{R}\ is$:

$$G(t, i) = \begin{cases} 0, & if\ t = 1 \\ 0 + \pi_2\big(O(t)\big), & if\ F(t-1, i) \leq 0; \\ G(t-1, i) + \pi_2(O)(t)), & otherwise \end{cases} \tag{4}$$

When $t = 1$, the value of G is zero, which is the initial signal profile of the DC with an index i. It is repeatedly added by K signal until the termination condition is reached.

4. The antigen profile update function $H{:}\ Time \times Population \rightarrow (a_{i1}, a_{i2}, \ldots, a_{ik})$ is defined as

$$H(t, 1) = \begin{cases} \big(a_{i1}, a_{i2}, \ldots, a_{ik}, a_{i(k+1)}\big) s.t.\ a_{i(k+1)} = S(t), & if\ S(t)\epsilon\ Antigen; \\ (a_{i1}, a_{i2}, \ldots, a_{ik}), & otherwise \end{cases} \tag{5}$$

Where H is initially empty. As a new antigen instance arrives, it is sampled by the DC with an index i and its antigen profile is updated until the termination condition is reached.

Let $r_i = G(t, i) s.t\ F(t-1, i) \leq 0$ be the signal profile of a DC, and $L : \mathbb{N} \rightarrow Antigen \times \mathbb{R}$ denote the function that maps an index $j\ \epsilon\ \mathbb{N}$ to an element of the output list.

The output record function is:

$$L(j) = (a_{ik}, r_i)\ \ \forall k \tag{6}$$

where $L(j)$ is the jth element of the list. This function is responsible for recording the decision of a DC into the output list when the termination condition is reached.

5. The antigen counter function $C : \mathbb{N} \times Antigen \rightarrow \{0,1\}$ is:

$$C(j, \alpha) = \begin{cases} 1, & if\ \pi_1\big(L(j)\big) = \alpha; \\ 0, & otherwise \end{cases} \tag{7}$$

6. The signal profile abstraction function $R : \mathbb{N} \times Antigen \rightarrow \mathbb{R}$ is defined as

$$R(j,\alpha) = \begin{cases} \pi_2\big(L(j)\big), & if\ \pi_1\big(L(j)\big) = \alpha \\ 0, & otherwise \end{cases} \qquad 8$$

In the two functions above, $\alpha \epsilon\ Antigen$ is an antigen type. The function C counts the number of instances of antigen type α, and the function R calculates the sum of all K values associated with antigen type α.

7. The anomaly metric calculation function is defined as.

$$K(\alpha) = \frac{\gamma}{\beta}\ with\ \beta = \sum_{j=1}^{n} C(j,\alpha) and\ \gamma = \sum_{j=1}^{n} R(j,\alpha) \qquad 9$$

As Antigen $\neq \emptyset$ *and* $\alpha \epsilon\ Antigen$, the number of this antigen type $\beta \geq 1$. A threshold ε can be applied for further classification. The value of the threshold depends on the underlying characteristics of the dataset used. An antigen type α is classified as anomalous if $K(\alpha) > \varepsilon$, and normal otherwise.

In this experiment the distribution between the two classes is used to bias this value, the calculation is displayed in Equation 10 to demonstrate the process of deriving anomaly threshold at, an is the number of anomalous data items, tn is the total number of data items and at is the derived anomaly threshold (receptor).

$$at = \frac{an}{tn} \qquad 10$$

Once the MCAV is derived for each antigen, a threshold is applied. Antigen with a MCAV greater than anomaly threshold generated, are classified as class two or *anomalous*.

4. EXPERIMENTAL SET UP

Using the intrusion detection NSK-KDD data designed to demonstrate the capability of the DCA to differentiate signals in the same distinct contexts. By making use of one step data order i.e. the class label one before class label two in order to generate the signal correctly. The classification threshold is set to a MCAV of 5.0 as previously stated. Items below the threshold are classified as class one and above as class two. The resulting classified antigens are compared to the labels given in the original data set. The rate of errors is recorded for further analysis. Performance is assessed in terms of true positives and false positives.

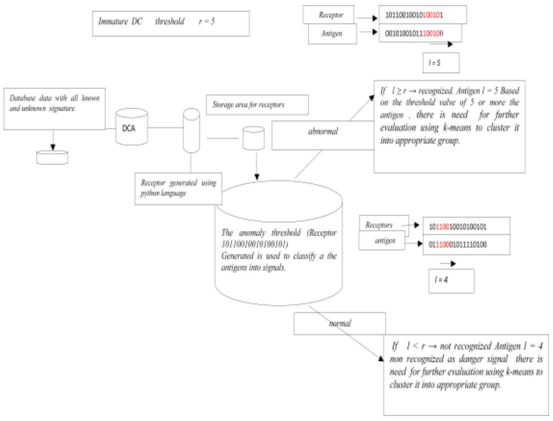

Immature DC threshold r = 5
Receptor
Antigen
10110010010100101
00101001011100100
l = 5
Database data with all known and unknown signature.
DCA
Storage area for receptors
abnormal
If l ≥ r → recognized. Antigen l = 5 Based on the threshold valve of 5 or more the antigen , there is need for further evaluation using k-means to cluster it into appropriate group.
Receptor generated using python language
The anomaly threshold (Receptor 10110010010100101) Generated is used to classify a the antigens into signals.
Receptors
antigen
10110010010100101
01110001011110100
l = 4
normal
If l < r → not recognized Antigen l = 4 non recognized as danger signal there is need for further evaluation using k-means to cluster it into appropriate group.

Figure 3: DCA as Classification System

A simple experiment was performed using NSL KDD dataset which contained 125973 dataset is used to demonstrate the capability of the DCA to classify unordered data between two distinct contexts. The data is partitioned into three sections, resulting in a two-step data order. Performance metrics is used to measure the accuracy, detection and false alarm rate. The datasets yield similar result with the algorithm.

Table 1: Result of Normal Dendritic Cell Algorithm

Experiment	TP	TN	FP	FN	Class 1	Class 2
2 step	32025	30542	26605	36801	57197	68825

Two-step data order

120 class1 460 class 120 class 1

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\frac{32025 + 30542}{32025 + 30542 + 26605 + 36801} = \frac{62567}{125973} = 0.50\%$$

$$Detection\ rate = \frac{TP}{TP+FN}$$

$$\frac{32025}{32025 + 36801} = \frac{32025}{68826} = 0.47\%$$

$$False\ alarm = \frac{FP}{FP+TN}$$

$$\frac{26605}{26605+30542} = \frac{26605}{57197} = 0.47\%$$

For the two-step data 26605 errors out of 125973 are recorded, the error is on the high side, with a classification rate of 0.47%. These shows that the DC is unable to locate a T cell with a high affinity to antigen that it has classed as dangerous, therefore, the DCA has a propensity to have a high false positive rate. A possible improvement is the introduction of a similar barrier in order to improve the algorithm's false positive rate. This could simply be in the form of identifying signatures of "normality" at the antigen level and simply preventing those antigens from being allowed to report as anomalous.

4.1 Auto-immunity Dendritic Cell Algorithm

The word automatic is shortened as auto which is a process or mechanism functioning by itself without human intervention. Immunity is the body's ability to resist a disease; it may exist naturally or as a result of inoculation or previous infection. Auto – immunity means the body's ability to resist infection without external intervention. Since the DCA has a propensity to have a high false positive rate which limits its functionality, there is need to for it to be vaccinated so as to improve its classification capabilities.

Steps taking in producing vaccine for an auto-immunity dendritic cell algorithm:

a. After running the DCA depending on the threshold the value of receptors corresponds to the threshold value that determines the specificity when an antigen is matched against the receptor. Depending on this threshold value the antigens may be classified as normal or attack.

b. Calculate all the mean of the correctly classified attack and normal and divide it by attack which will give a new dataset and,

c. Also, the new data will be added to the old dataset one after the other; this will change most of the wrongly classified attack to normal. Since a signature of normality has been imputed into it because vaccine is giving to weakening the actions of an attack.

d. Then another anomaly threshold value will be generated which serves as a discriminating value between normal and attack.

$$at = \frac{an}{tn} \qquad 11$$

The newly generated dataset is of the mean of the truly classified normal added to the mean of the truly classified attack and divide it by the attack mean, thereby generating a new dataset

$$New\ data = \ attack\ mean + normal\ mean \qquad 12$$

Table 2: Result for Auto-immunity Dendritic Cell Algorithm

TP	TN	FP	FN	Class 1	Class 2
67826	56,897	250	1000	57147	68,826

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\frac{67{,}826 + 56{,}897}{67{,}826 + 56{,}897 + 250 + 1000} = \frac{124{,}723}{125{,}973} = 0.99\%$$

$$Detection\ rate = \frac{TP}{TP+FN}$$

$$\frac{67{,}826}{67{,}826+1000} = \frac{67{,}826}{68{,}826} = 0.98\%$$

$$False\ alarm = \frac{FP}{FP+TN}$$

$$\frac{250}{250+56{,}897} = \frac{250}{57{,}147} = 0.00\%$$

The effect of vaccine administration can be seen in table 3.9, the false positive generated in the normal DCA reduced appreciably in the Auto-immune DCA in which it gives a low rate of error. Also, there is still need for further clarification by using data clustering technique to classify the NSL KDD dataset and the result will be compared.

Auto – immunity K – Means Clustering

Cluster based epidemic control requires effective clustering approach. Clustering is a well known technique and there exist a large number of clustering algorithms. These methods can be categorized as: partitioning methods, hierarchical methods, density based methods and grid-based methods. Here concentration will be on algorithm that closely related to our investigation. K-means algorithm will be used to cluster the data set and to filter it.

K-means Clustering

K-means clustering is a clustering analysis algorithm that groups objects based on their feature values into K disjoint clusters. Objects that are classified into the same cluster have similar feature values. K is a positive integer number specifying the number of clusters, and has to be given in advance. Here are the four steps of the K-means clustering algorithm:

1) Define the number of clusters K.

2) Initialize the K cluster centroids. This can be done by arbitrarily dividing all objects into K clusters, computing their centroids, and verifying that all centroids are different from each other. Alternatively, the centroids can be initialized to K arbitrarily chosen, different objects.

3) Iterate over all objects and compute the distances to the centroids of all clusters. Assign each object to the cluster with the nearest centroid.

4) Recalculate the centroids of both modified clusters.

5) Repeat step 3 until the centroids do not change any more.

A distance function is required in order to compute the distance (i.e. similarity) between two objects. The most commonly used distance function is the Euclidean one which is defined as:

$$d(x,y) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2} \qquad 13$$

Where $x = (x_1, \ldots., x_m)$ and $y = (y_1, \ldots \ldots, y_m)$ are two input vectors with m quantitative features.

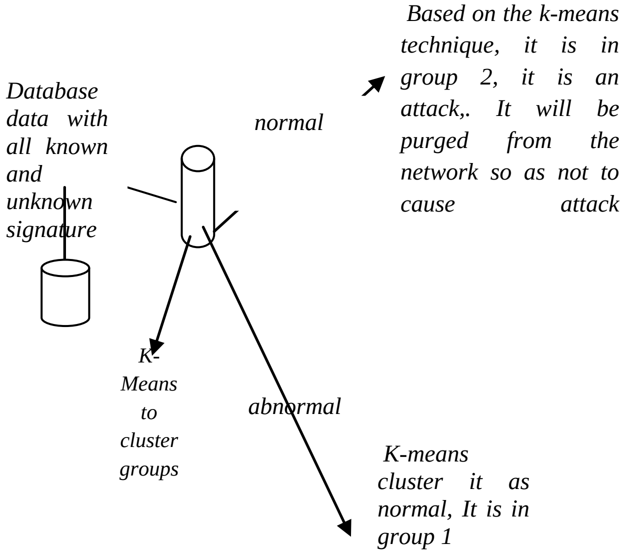

Database data with all known and unknown signature
normal
Based on the k-means technique, it is in group 2, it is an attack,. It will be purged from the network so as not to cause attack
K-Means to cluster groups
abnormal
K-means cluster it as normal, It is in group 1

Figure 4: K-means as a classification system

Experimental Setup and Classification result

The NSL KDD intrusion detection dataset is used for experimental purpose. The 125973 records left after removing the redundant records were used in carrying out the clustering.

K means technique which will be used in solving a classification problem, in order to solve the problem of false.

Table 3: Result of the Normal K – Means Data clustering

TP	TN	FP	FN	Class 1	Class 2
51648	66034	1309	6982	67343	58630

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\frac{51648 + 66034}{51648 + 66034 + 1309 + 6982} = \frac{117682}{125{,}973} = 0.94\%$$

$$Detection\ rate = \frac{TP}{TP+FN}$$

$$\frac{51648}{51648+6982} = \frac{51648}{58630} = 0.90\%$$

$$False\ alarm = \frac{FP}{FP+TN}$$

$$\frac{1309}{1309+66034} = \frac{1309}{67393} = 0.02\%$$

The result shows that k-means approach is better in solving classification problem in terms of accuracy, detection and in false alarm generation.

There is need to vaccinate the system again for clarity purpose using auto – immune K-Means clustering technique to classify the data into appropriate groups based on the minimum distance, to know if the network is free of virus. If there still remain a group that is farther from the minimum distance then the system needs to be purged out instead of trying to manage it. [15] propose that purging the system will help in overcoming subversion of attack and autoimmunity avoidance on the network.

Auto-immune in K – Means Clustering

The newly generated dataset has the signature of normal and attack, with introduction of vaccine to suppress the actions of the attack on the network. This network will be monitored using a clustering technique (k – means) to determine the centroid coordinate, determine the distance of each object to the centroid and group the object based on the minimum distance, until there is no changes in K cluster centres.

Table 4: Result of the Auto immune K – Means Data clustering

TP	TN	FP	FN	Total Class 1	Total Class 2
58580	67343	0	50	67343	58630

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\frac{58580+67343}{58580+67343+0+50} = \frac{124973}{125973} = 0.99\%$$

$$Detection\ rate = \frac{TP}{TP+FN}$$

$$\frac{58580}{58580+50} = \frac{58580}{58630} = 0.98\%$$

$$False\ alarm = \frac{FP}{FP+TN}$$

$$\frac{0}{0+67343} = \frac{0}{67343} = 0.00\%$$

The effectiveness and efficiency of vaccine was measured using the auto-immune k – means clustering the performance was excellent. It shows that an introduction of signature of normality will improve the algorithm's false positive rate, this will help in overcoming attacks on our networks.

Table 5: Results and Discussions

Detectors	Accuracy	Detection rate	False alarm
Normal DCA	0.50%	0.47%	0.47%
Auto-immune DCA	0.99%	0.98%	0.00%
K-Means Data Clustering	0.94%	0.90%	0.02%
Auto – immune K-Means Data Clustering	0.99%	0.98%	0.00%

Table 5 shows the comparison between Normal DCA, Auto-immune DCA, K – Means Data Clustering and Auto-immune K-Means Data Clustering techniques. This table shows that the proposed auto – immune DCA and Auto – immune K – Means data clustering is better in terms of accuracy, detection and false positive generation.

5. CONCLUSION

The DCA and K – Means clustering technique was applied to classification problems. The effectiveness and efficiency of vaccine was measured as an auto-immune DCA and auto-immune k – means clustering the performance was excellent. It shows that an introduction of signature of normality will improve the algorithm's false positive rate, this will help in overcoming attacks on our networks. An evaluation of the algorithm showed success when applied to the classification problem on the network. The performances of the DCA and K- means was carried out using kdd'99 intrusion detection implemented using python programming language.

6. RECOMMENDATION

While the vaccinated (Auto – immune DCA) and (Auto – immune K – means) has performed well on the problems presented, it means that a signature of normality is necessary in other to overcome classification error that is to overcome high false positive generation on data that are not ordered.

7. REFERENCES

[1] Ayara, .M., Timmis, J., de Lemo, .R and Forrest, .S (2005): Immunising automated teller machines. In *Proc. of the 4th International Conference on Artificial Immune Systems (ICARIS), LNCS 3627*, pages 404–417.

[2] Blake, .C .L., Hettich, .S and Merz C. J. (1998): UCI repository of machine learning databases.

[3] Codd .E. F. (1968): Cellular Automata, academic Press.

[4] Dasgupta (1999): Immunity-based intrusion detection system: a general framework, Proceeding of the 22nd National Information Systems Security Conference (NISSC)", Arlington, Virgina, pp.147-160.

[5] de Castro, .L and Timmis, .J (2002): *Artificial Immune Systems: A New Computational Approach*. Springer-Verlag, London. UK.

[6] Gooding, L.R. (1992): Virus proteins that counteract host immune defenses. Cell 71, 5–7

[7] Greensmith .J. and Aickelin .U (2008): The Deterministic Dendritic Cell Algorithm. In Proceedings of the 7th International Conference on Articial Immune Systems (ICARIS), pages 291-303.

[8] Jiawei and Micheline, K.(2006) Data Mining: Concepts and techniques, second edition, Elsevier inc.

[9] Kim J. Bentley P. Aickelin U. Greensmith J. Tedesco G. and Twycross J. (2007): Immune System Approaches to Intrusion Detection - A Review. Natural Computing,6(4):413-466.

[10] Kim J. W. (2002): Integrating Artificial Immune Algorithms for Intrusion Detection. PhD thesis, Department of Computer Science, University College London.

[11] Krakauer, D.C. and Plotkin, J.B. (2005) Principles and parameters of molecular robustness. In Robust Design: A Repertoire of Biological, Ecological, and Engineering Case Studies (Jen, E., ed.), pp. 71–103, Oxford University Press

[12] Krakauer, D.C. (2005) Robustness in biological systems: a provisional taxonomy. In Complex Systems Science in Biomedicine (Deisboeck, T.S. and Kresh, Y., eds), pp. 185–207, Plenum

[13] Lee W. and Stolfo S. J. (1998):Data mining approaches for intrusion detection. In Proceedings of the 7th conference on USENIX Security Symposium.

[14] Matzinger, P. (1998) An innate sense of danger. Semin. Immunol. 10, 399–415

[15] Olubadeji .B and Adetunmbi .A .O (2014): Design principle for overcoming subversion of attack on the network. ICACSET 2014. Conference.

[16] Schmidt-Hempel, P. (2005) :The evolutionary ecology of insect immune defense. Annu. Rev. Immunol. 50, 529–551

[17] Mitchell T. M. (1997): Machine Learning. McGraw-Hill.

[18] Mitchell T. M. (1998): An Introduction to Genetic Algorithms. The MIT Press.

[19] Myerson, R.B. (1987): Mechanism design. In The New Palgrave: Allocation, Information, and Markets (Eatwell, J. et al., eds), pp. 191–206, W.W. Norton & Co.

[20] Scholkopf, .B and Smola, .A .J (2002): Vector Machines, Regularization, Optimization, and Beyond. The MIT Press.