Adaptive Gene Intrusion Detection System

Pim Wijn

Leiden Institute of Advanced Computer Science
Leiden University, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
p.wijn@umail.leidenunv.nl

Abstract. Artificial immune systems provide a good model for intrusion detection systems for networks, which has similar requirements as a natural immune system. An immune system has to categorize substances into self and non-self to find intrusions. Such a system makes use of several algorithms and concepts inspired by natural immune systems such as Negative Selection, mutation and distributed selection. We will look at a new negative selection algorithm. This algorithm builds a non-self detector set that can be used in such an intrusion detection architecture. An adaptation based on repulsion often seen in flocks is then proposed. Both algorithms will then be evaluated in the context of a distributed intrusion detection algorithm.

1 Introduction

The natural immune system is a highly complex system that protects the human body from harm. The goal of the immune system is to categorize all cells and molecules in self or non-self substances, or non-dangerous and dangerous substances. It does this using a highly distributed task force of cells. These cells take local action in the form of binding to antibodies. However they also trigger global actions, using a network of chemical messengers.

The immune system has two branches. The innate immune system and the adaptive immune system. The innate system is unchanging and detects and destroys invading substances. The adaptive branch responds to unknown foreign substances and builds a response that will stay in the body for a long time. The behavior of the system is decided by matching, diversity and distributed control. The antibodies have to match antigens (but not self). Antibodies should be diverse to respond to the widest range of threats possible. Finally all behavior has to happen through interactions between cells without any central controller. These aspects make the immune system highly distributed, adaptive and self organizing. It also has memory of past encounters and the ability to continually learn from new encounters. All of these properties make the immune system model interesting for computer scientists. This has lead to the field of artificial immune systems.

Artificial immune systems are suited for problems where multiple solutions are important, either directly when having multiple objects or indirectly when a group of solutions generate the desired outcome together. They are also very

suitable for problems that change over time and need to be solved multiple times. They are less proficient at one-off optimizations and optimizing a single function.

First in section 2 we will take a look at artificial immune systems in general and what the requirements are for intrusion detection systems.

In this paper we will describe an architecture based on the architecture developed in [5] and [4]. It is a recently developed architecture for intrusion detection based on artificial immune systems. They describe a high level architecture that can be implement in a network. However the focus is more specifically on the two aspects of the architecture that are most closely related to artificial immune systems. Detector generation and the detection of intrusions using these detectors. The authors present two algorithms that increase accuracy with respect to older approaches.

We will take a deeper look at these two algorithms in section 4. First of all we will try to formalize and clarify important concepts and do this in a more consistent manner. These formal definitions will be presented in section 3. Later the algorithms will be described in more detail. Once the algorithms have been presented, we will propose a change to the detector generation algorithm that could improve performance.

Finally we will evaluate the results of the original algorithm and the updated algorithm to see the change in performance. For this the KDDCUP'99 dataset will be used which is considered a standard in evaluation intrusion detection algorithms.

2 Preliminaries

2.1 Artificial Immune Systems

Artificial immune systems are not an algorithm that we can directly use. It is a model with certain properties that we can use to build an immune system architecture. Since the model contains multiple concepts, several algorithms have been developed. Examples are NSA (Negative Selection Algorithm), DCA (Dendric Cell Algorithm) and danger theory based algorithms. [1]

Most artificial immune systems use a combination of these algorithms, which we will in the proposed architecture as well.

In general an immune system architecture has four important aspects, namely: encoding, the similarity measure, selection and mutation.

Encoding The genes, antibodies and antigens, have to be represented in the system in a format that can be easily manipulated. Originally this was a string of digits, often binary, that represented a gene. This is closely related to the representation of antibodies in the natural immune system. For artificial systems such a representation often has no real advantages and genes can be represented as vectors containing any type of value. For the approach described in this paper we will use a vector of real values between 0 and 1.

Similarity Measure The similarity or distance measure is closely related to the encoding of the genes and the properties of the data. For standard binary strings the hamming distance can be used. Another measure often used for this representation is the longest contiguous match. This measure is particularly relevant for data where the ordering is important. Since we are using vectors of real values the Euclidean distance and similarity will be used.

Selection In an artificial immune system selection can be both positive and negative. For positive selection the system is empty at the beginning, target data items will be encoded as antigens. Antibodies are created and each starts with a specific concentration value. This concentration decreases over time (death rate) and antibodies below a certain concentration will be removed from the system. An antibody's concentration is increased when it matches antigens. The increase in concentration is related to the strength of the match (a process called *stimulation*). To prevent overlapping antibodies, they are *suppressed* by similar antibodies, this encourages diversity. The change in concentration for an antibody can be calculated using equation 1.

$$\begin{aligned}\frac{dx_i}{dt} &= c[(\text{antibodies recognized}) - (\text{suppression}) + (\text{antigens recognized})] - (\text{death rate}) \\ &= c\left[\sum_{j=1}^{N} m_{ji}x_i x_j - k_1 \sum_{j=1}^{N} m_{ij}x_i x_j + \sum_{j=1}^{n} m_{ji}x_i y_j\right] - k_2 x_i\end{aligned} \tag{1}$$

Positive selection is useful for adapting to, and learning from new threats. The second type of selection, namely negative selection has another purpose. Artificial immune systems need to detect threats, however at the start of the life-cycle it has not seen any threats yet. Therefore it needs to model threats using only the information it already has, which is the self set. With negative selection the goal is to create antibodies that cover the non-self space as much as possible. Antibodies are randomly generated (or using a heuristic)[7]. For each of the antibodies it is then checked whether they match any gene in the self set (an auto-immune response). If the antibody does not match self then it is kept as an immature antibody. From that point on in needs to achieve a certain activation threshold within a certain timespan to be remembered, otherwise it is discarded.

Mutation The type of mutation most commonly used in artificial immune systems is similar to that used in genetic algorithms. In binary strings bits are flipped. In vectors of real numbers or characters values are changed at random. The amount of mutation is dependent on the closeness of the match, which can increase or decrease the strength of mutation depending on the type used.

2.2 Intrusion Detection Systems

The main goal of intrusion detection systems is to detect unauthorized use of systems by both insiders and outsiders. They are often connected to the network and analyze all incoming traffic. Most intrusion detection systems rely on suspicious signatures based on known intrusions. This means that most intrusion detection systems cannot react to previously unseen attacks. We want and IDS to be scalable, distributed, adaptive and self organizing. Preferably the system can be scaled as much as necessary. The detection of intrusions should be done distributed over many nodes in the network. Each node should be able to adapt to new threats by itself and communicate knowledge to the rest of the system [2]. This makes the system also self-organizing, since there is no central components that evolves the system.
Monolithic systems and strictly hierarchical systems are often not well scalable, distributed and are more vulnerable since they have a single point of failure. The parallels between the requirements for intrusion detection systems and artificial immune systems have lead to much research in this application area. The other advantages of an artificial immune system based intrusion detection system are that it can react to new attacks and it does not need any non-self information to start with.
There are several types of attacks that occur, as described by Harmer et al. in [3].

- Misuse/Abuse: Authorized users performing unauthorized actions.
- Reconnaissance: finding exploitable systems and/or services.
- Penetration: Accessing resources someone is not authorized for.
- Trojanization: Adding unauthorized processes.
- Denial of Service: Preventing legitimate access to computation/services.

3 Definitions and Notations

In this section we will give definitions and formalize the different concepts used in the artificial immune system architecture.

In the proposed architecture we use real valued genes, this means a gene is a vector of numbers between 0 and 1.

$$g = \langle g_0, \ldots, g_n \rangle \wedge \forall_{g_i} [0 < g_i < 1] \tag{2}$$

A detector contains a gene g and has an binding threshold bt.

$$d = \langle bt, g \rangle \wedge 0 < bt < 1 \tag{3}$$

We use Euclidean distance, so the matching between a detector d and antigen g is defined as

$$m_{dg} = \frac{1}{1 + \sqrt{\sum_i (d_i - g_i)^2}} \tag{4}$$

A detector d matches a gen g if the matching value m_{dg} is lower than the binding threshold d_{bt} of the detector.

$$match(d, g) = \begin{cases} \text{true} & \text{if } m_{dg} > d_{bt} \\ \text{false} & \text{if } m_{dg} \leq d_{bt} \end{cases} \tag{5}$$

Multiple detector sets will be generated containing a specified amount of detectors. Each of the detector sets will be assigned to a detector agent.

$$DS = \{d_0, \ldots, d_n\} \tag{6}$$

A detector agent DA is a component that will be assigned to a node in the network. Each detector agent has an assigned detector set DS and a risk factor rf. Once the risk factor is too high, the system might decide to remove the detector agent and create a new one.

$$DA = \langle rf, DS \rangle \tag{7}$$

The self set S is a set of genes that characterizes self, the antibodies should not match any gene in S.

$$S = \{g_0, \ldots, g_N\} \tag{8}$$

To train the artificial immune system with self in the first step a test set TS is needed, this set should be a subset of self, so that we can use negative selection to find antibodies that do not match any $g \in S$.

$$TS \subseteq S \tag{9}$$

4 Adaptive Gene Intrusion Detection System

In this section we will give a high level description of the proposed architecture and its components. We will then look into more detail at the algorithm for creation of detectors and the detection of intrusions.

4.1 Architecture

The architecture consists of three high level components: the TM (Training Module), VAM (Vulnerability Assessment Module) and RM (Response Module). The training module is responsible for the generation and selection of detectors. During the initialization of the system it creates the detector sets that will be assigned to the detector agents. During the life-time of the system new detector sets can be generated to replace outdated detector agents.

The vulnerability assessment module consists of multiple detector agents. Each of these agents evaluates network traffic to find intrusions. Once an intrusion is found it will be communicated to the response module.

The response module receives alerts from the VAM, each of these alerts is then evaluated and a response is generated. The information that is learned for the intrusion is communicated with a component called the intelligence component, it can use this information to improve generation of detectors. Each detector agent keeps track of the false positives and false negatives it finds. Using this the risk factor of that agent is calculated. Once the risk factor is too high the detector agent is removed and a new one is generated with the updated information on intrusions.

In the rest of this section we will first describe an algorithm for the initial creation and selection of detectors. Afterwards we will show how the detector agents work together to find intrusions and minimize the false positive and false negative rates.

Finally we propose a change to the algorithm for detector creation and selection that could improve the accuracy of detectors.

The rest of the architecture is outside of the scope of this paper.

4.2 Detector Creation and Negative Selection

The detector creation and selection algorithm is based on the NSA algorithm. It generates random real-valued detectors which are then compared with the test set. However in contrast to standard NSA the detectors that match self are not removed from the generated detectors. These *weak* detectors are then tuned until they do not match any self gene. This is done by increasing the binding threshold with the tuning value, this reduces the area of the space that is matched by that detector.

Algorithm 1 works as follows: Until the required detector set size is reached, new detectors are created. Each detector is checked for matches with the self set and matching detectors are tuned as described previously. If the detector is not strong enough it will be rejected.

4.3 Intrusion Detection

Detector agents $DA_0 \dots DA_n$ are distributed over all nodes in the system. Each with an assigned detector set $DS_0 \dots DS_n$. All of these detector agents monitor the network traffic for intrusions. This is modeled using a test set of antigens. Each detector agent tries to match the antigen g to each of its detectors. Once detector is found that matches the antigen, the detector agent increases the vote count for that antigen and continues with the new antigen. Each of the other detector agents also tries to match its detector set with the antigen and contributes to the votes. Once the vote count becomes higher than the alert threshold the antigen is flagged as an anomaly. The algorithm for detection in a DA is 2.

Algorithm 1 Detector Set Generation

Require: TS the test set such that $TS \subseteq S$
Require: PT power threshold such that $0 < PT < 1$
Require: t_v tuning value such that $0 < t_v < 1$

$DS \leftarrow \emptyset$
$WDS \leftarrow \emptyset$
while $|DS| < D_n$ **do**
  Generate detector d with random values for $d_0, \ldots, d_n$
  $d_p = 0$
  for $g \in TS$ **do**
    while $match(d, g)$ **do**
      $d_p \leftarrow d_p - 1$
      $d_{bt} \leftarrow d_{bt} + t_v$
    end while
    $d_p \leftarrow d_p + 1$
  end for
  if $\frac{d_p}{|TS|} > PT$ **then**
    $DS \leftarrow DS \cup \{d\}$
  else
    discard d
  end if
end while
return DS

$$anomaly(TeS_i, V) = \begin{cases} \text{true} & \text{if } V > A_t \\ \text{false} & \text{if } V \leq A_t \end{cases} \tag{10}$$

Algorithm 2 Intrusion Detection

Require: DS
Require: TeS test set
Require: A_t alert threshold, such that $0 < A_t < 1$

for $g \in TeS$ **do**
  $votes \leftarrow 0$
  for $d \in DS$ **do**
    if $match(d, g)$ **then**
      $vote \leftarrow vote + 1$
      return $vote$
    end if
  end for
  if $votes \geq A_t$ **then**
    Flag Anomaly
  end if
end for

4.4 Adaptive Genes

In this section we will describe the proposed change to algorithm 1. The original algorithm tunes the strength of the detectors with the tuning value. This decreases the area in which a detector detects genes. This tuning is done until the detector does not match any self gene. The change proposed will not only tune the strength of the detector but also the values of the gene in the detector using a similar repulsion principle as in flocking behavior. In a way this adds mutation to the detectors that were not directly accepted. This should prevent much of the shrinking of the size of the area a detector covers.

The gene of a detector will be moved along the direction of $v = d_g - g$ where d_g is the gene in the detector and g is the gene in the training set. The strength of the movement will be dependent on the closeness of the match and repulsion value r_v.

$$d'_g = d_g + m_{dg} r_v \frac{d_g - g}{|d_g - g|} \tag{11}$$

Algorithm 3 Adaptive Detector Set Generation

Require: TS the test set such that $TS \subseteq S$
Require: PT power threshold such that $0 < PT < 1$
Require: t_v tuning value such that $0 < t_v < 1$
Require: r_v tuning value such that $0 < r_v < 1$
$DS \leftarrow \emptyset$
$WDS \leftarrow \emptyset$
while $|DS| < D_n$ **do**
  Generate detector d with random values for $d_0, \ldots, d_n$
  $d_p = 0$
  for $g \in TS$ **do**
    while $match(d, g)$ **do**
      $d_p \leftarrow d_p - 1$
      $d_{bt} \leftarrow d_{bt} + t_v$
      $d_g \leftarrow d_g + m_{dg} r_v \frac{d_g - g}{|d_g - g|}$
    end while
    $d_p \leftarrow d_p + 1$
  end for
  if $\frac{d_p}{|TS|} > PT$ **then**
    $DS \leftarrow DS \cup \{d\}$
  else
    discard d
  end if
end while
return DS

5 Experiments

In this section we will compare the original detector generation algorithm 1 to the changed algorithm to see the differences in performance. The algorithms will be evaluated on the KDDCUP'99 dataset for intrusion detection.

The data was preprocessed using a OneShot encoder for categorical values. The values were normalized using a Robust normalization algorithm. Finally Principal Component Analysis (PCA) was used to extract the five most relevant features (like the original authors). These steps are necessary to make the genes real valued and keep computational complexity of the algorithm in reasonable bounds.

It is however not possible to compare the implemented algorithms to the algorithms described in the paper [5]. The authors have many inconsistencies in the description of their implementation and leave gaps. Furthermore it seems they used a specific type of PCA was used called KernelPCA. This algorithm is heavily dependent on the implementation and I was unable to use a standard implementation for this algorithm. Since the data is very skewed applying KernelPCA might be a requirement to get a reasonable performance.

This could be especially the case since there has been criticism on the KDDCUP'99 dataset [6]. The dataset does not seem to be a good representation of the problem and after analyzing the dataset it has been found that some trivial features predict intrusions in almost all cases.

We will therefore only look at the difference in performance of the two algorithms with otherwise the same code and exactly the same dataset.

Experiment settings For the experimental settings we will mostly keep the values that the authors have proposed, since they have been experimentally evaluated to be effective. We take the power threshold $PT = 0.7$, the self-tuning factor $t_v = 0.01$, the binding threshold for detectors is initialized at $bt = 0.3$. The number of detector sets and detector agents was not defined in the original paper. This makes it difficult to compare. We will take a value that is as high as possible while still keeping a reasonable running time, which is $|DS| = 50$ and we use one detector set. For the test set we will take $|TS| = 10000$. The alert threshold will be $A_t = 1$. This would not be used in a realistic scenario but makes it easier to evaluate the likelihood of getting false positives. The test set TeS will contain 100000 values which are both normal and intrusions.

For the adaptive algorithm we will take multiple values for the repulsion value $r_v \in \{0.005, 0.01, 0.02, 0.05, 0.1\}$

Each configuration will be run 20 times and we will take the average. The performance will be evaluated with the detection rate DR and False Alarm Rate FAR.

$$DR = \frac{TP}{TP + FN}, FAR = \frac{FP}{TN + FP} \tag{12}$$

r_v	0	0.005	0.01	0.02	0.05	0.1
DR	0.018	0.017	0.017	0.019	0.454	0.864
FAR	0.023	0.024	0.025	0.031	0.154	0.176

Results As we can see in the result table, the repulsion aspect greatly increases the chances of detection. It also increases the FAR. The combination of both can be fine-tuned however using more detector agents, bigger detector sets and a different A_t.

6 Conclusions

In this paper we have given a more formal description of the algorithms used in the architecture proposed in [5]. Furthermore several important concepts have been made more consistent and formal in section 3.
Second we have proposed a change to the original detector generation algorithm 1. This was done using a repulsion factor often used in flocking behavior. The aim was to move detectors away from the self set instead of just shrinking the area in which they match genes. This should keep as much coverage as possible without adding more detectors.

We have compared implementations of both algorithms in the context of a intrusion detection architecture. The results indicate that adding a repulsion factor can greatly increase the detection rate. It is however difficult to say if this would work in a more practical setting, since the dataset is not considered to be a good benchmark. It was also not possible to replicate the original experimental setup presented in [5]. This makes it difficult to see whether this change would have the same effect in the original implementation.

Further research could be done by applying these approaches on a different dataset and using an experimental setup closer to the original.

References

1. U. Aickelin, D. Dasgupta, and F. Gu. Artificial immune systems (INTROS 2). *CoRR*, abs/1308.5138, 2013.
2. S. Cayzer, J. Smith, J. A. R. Marshall, and T. Kovacs. What have gene libraries done for ais? In *Proceedings of the 4th International Conference on Artificial Immune Systems*, ICARIS'05, pages 86–99, Berlin, Heidelberg, 2005. Springer-Verlag.
3. P. K. Harmer, P. D. Williams, G. H. Gunsch, and G. B. Lamont. An artificial immune system architecture for computer security applications. *Trans. Evol. Comp*, 6(3):252–280, June 2002.
4. J. Kim and P. Bentley. An artificial immune model for network intrusion detection. 10 2001.
5. P. Saurabh and B. Verma. An efficient proactive artificial immune system based anomaly detection and prevention system. 60, 03 2016.
6. S. Terry Brugger and J. Chow. An assessment of the darpa ids evaluation dataset using snort. 01 2007.
7. H. Yang, T. Li, X. Hu, F. Wang, and Y. Zou. A survey of artificial immune system based intrusion detection. 2014:156790, 03 2014.