The ISC Int'l Journal of Information Security

July 2013, Volume 5, Number 2 (pp. 209–225)

http://www.isecure-journal.org

# STLR: A Novel Danger Theory Based Structural TLR Algorithm

Reza Azmi [1,*] and Boshra Pishgoo [1]

[1] *Alzahra University, Computer Engineering Department, Operating System Security Laboratory, Tehran, Iran*

ARTICLE INFO.

*Article history:*
**Received:** 13 January 2013
**Revised:** 21 November 2013
**Accepted:** 27 November 2013
**Published Online:** 20 March 2014

*Keywords:*
Anomaly Detection, Artificial Immune Systems, Danger Theory, TLR algorithm, STLR algorithm.

ABSTRACT

Artificial Immune Systems (AIS) have long been used in the field of computer security and especially in Intrusion Detection systems. Intrusion detection based on AISs falls into two main categories. The first generation of AIS is inspired from adaptive immune reactions but, the second one which is called danger theory focuses on both adaptive and innate reactions to build a more biologically-realistic model of Human Immune System. Two algorithms named TLR and DCA are proposed in danger theory field that both of them are trying to identify the antigens based on a simple identifier. Both of them suffer from low accuracy and detection rate due to the fact that they are not taking the structure of antigens into account. In this paper, we propose an algorithm called STLR (structural TLR), which is an extended form of TLR algorithm. STLR tries to model the interaction of adaptive and innate biological immune systems and at the same time considers the structure of the antigens. The experimental results show that using the structural aspects of an antigen, STLR can lead to a great increase in the detection rate and accuracy.



## 1 Introduction

Intrusion detection is the process of detecting an unauthorized use, or attack upon a computer or a telecommunication network. Intrusion Detection Systems (IDS) constantly monitor the system behavior to aid in deterring and mitigating the damage that can be caused by attackers or other security violations [1]. According to their location, IDSs can be categorized into Network-based Intrusion Detection System (NIDS) and Host-based Intrusion Detection Systems (HIDS). NIDSs try to detect intrusions by monitoring the network traffic. In contrast HIDSs reside inside of a computer system and try to monitor and analyze the internals of a system as well as the network packets on its interfaces. HIDSs mostly focus on the dynamic behavior of local machines to protect the computer system against misuse or attack, thus making it much harder for the intruder to break into the system [1].

Generally there are two broad approaches to intrusion detection according to the used method: misuse detection and anomaly detection. The misuse detection approach tries to examine the system for abnormal behavior or misuse and mostly utilizes pattern-matching techniques [2]. Naturally it should keep a database of all known misuse signatures and needs to be upgraded periodically. The anomaly detection approach usually tries to make a profile of normal user's behavior and any deviation from this, is considered abnormal. A mixture of these two approaches, called hybrid, is also used. The misuse approach is simpler, but it is not capable of detecting novel attacks. On the other hand the anomaly approach can detect new attacks, but it suffers from high false positive rate.

* Corresponding author.
Email addresses: azmi@alzahra.ac.ir (R. Azmi), Boshra.pishgoo@student.alzahra.ac.ir (B. Pishgoo)
ISSN: 2008-2045 

Another disadvantage of systems using anomaly detection is that, the attacker can mimic the normal user's behavior and evade the detection by IDS, resulting in false negative alarm [2]. One of the main goals of our research is to minimize the false positive and false negative rate.

There are different methods in computational intelligence field which have been frequently used for anomaly detection up now [3]. In this paper, we employ artificial immune systems for anomaly detection. This approach is inspired from Human Immune Systems (HIS), and is a suitable option for intrusion detection because of its distributed, self-organized and lightweight nature. HIS has a multi-layered protection architecture, including physical barriers, physiological barriers, an innate immune system, and an adaptive immune system [3]. Each method of AIS is inspired from some parts of this architecture. From this viewpoint, all AIS methods can be divided into two main categories. The first one is the largest branch of AIS methods and is inspired from adaptive immune system. The adaptive immune system is a complex of great variety of cells. Among its cells, two lymphocyte types, T cells (TC) and B cells (BC), cooperate to distinguish self from non-self antigens. Negative selection [5], Clonal selection [4] and immune network [6] are three important branches of this category. The second category is new generation of AIS, and it is inspired from both innate and adaptive immune systems. Danger theory methods fall into this category.

The fundamental idea of the first generation of AIS, Immune responses are triggered when the body encounters non-self antigens; but Matzinger proposed the Danger Model [7, 8, 63], and claimed that immune responses are triggered by the unusual death of normal tissues, not by non-self antigens. Danger Theory suggests that cells do not release alarm signals when they die by normally planned processes (known as apoptosis), whereas cells do release alarm signals when they are stressed, injured, or die abnormally (known as necrosis). These signals are raised via innate immune systems. A type of cells known as Dendritic Cells (DC) act as important medium between innate and adaptive immune system and passing these alarm signals to the adaptive immune system [3].

Aickelin and his research group applied Danger Theory to intrusion detection systems as a project called "Danger Project" [9] in 2003. The results of their works can be summarized as one innate immunity architecture, and two danger theory based algorithms namely the Dendritic Cell Algorithm (DCA) [10–16, 59–62, 64–68] and TLR algorithm [17–19]. Both DCA and TLR employ the model of DCs but, their implementation focuses on different aspects of the DC model. The DCA relies on the signal processing aspect by using multiple input and output signals, while the TLR emphasizes the interaction between DCs and T cells, and only uses danger signals [3]. Neither of these algorithms takes the structure of the antigens as an input to the system. In this paper we propose an algorithm named STLR (Structural TLR) which is an extended form of TLR algorithm [56]. Like TLR, we try to model the interaction between innate and adaptive immune system through using DCs and TCs. But unlike TLR, which matches antigens with DCs only through an identifier, our algorithm also takes the structure of the antigen into account and that's why we named it "structural TLR".

The remainder of this paper is organized as follows. In Section 2 we review related works and algorithms proposed in this area. The biological inspiration from HIS, which is the base of TLR and STLR is briefly introduced in Section 3. In Section 4, we explain TLR algorithm and then, in Section 5 our proposed algorithm is described. Section 6 is devoted the evaluation of our proposed method and the results of comparing STLR algorithm and some classic methods in AIS field (like Negative Selection (NS), DCA and TLR algorithms) are presented. Finally, the conclusion is given in Section 7.

## 2 Related Work

Host based anomaly detection is a very active research area and different methods have been frequently used to improve it. Most of these studies employ web information or user-level information such as system calls for model construction and intrusion detection. Here we concentrate on different methods that are applied on system calls and describe them in this section.

All methods in the field of anomalous system call detection fall into two main categories namely i) specification based methods and ii) learning based methods [20]. Specification based techniques rely on application-specific models. These models can be written manually [21–23] or derived using program analysis techniques [24] or created interactively with the user's help [25, 26]. These application-specific models describe normal behaviors of system and an anomaly is detected when a non-conforming system call invocation is found. A major problem of specification-based systems is the fact that they exhibit only a very limited capability for generalizing from written or derived specifications [20].

Learning-based techniques do not use any written or derived specifications but, they employ some profiles that are built by analyzing system call invocations during normal execution [20]. An example of this

approach is presented by Forrest [27]. In His method there are two phases. During the training phase, the system collects all distinct system call sequences of a certain specified length and during the detection phase, all actual system call sequences are compared to the set of legitimate ones. If no match is found, an alarm is raised. This approach has been further refined in some researches.

Early researches in this field only uses sequences of system calls and constructed behavioral models using simple algorithms like look ahead pairs algorithm [28–30]. A few later, Wagner and Dean proposed that it was possible to craft sequences of system calls that exploited an attack, but appeared normal [24]. This strategy, which is called Mimicry attack, have become increasingly sophisticated using automated attack methods including model checking [31], genetic algorithm [32] and symbolic code execution [33].

In order to fix this problem, in one side, some researches combined sequences of system calls with system call arguments to increase the system view and Kruegel went one step further, looking only at the arguments and disregarding sequence of system calls altogether [34]. On the other side, different learning-based methods have been employed for detecting anomalous system calls like finite state automata [35–37], hidden Markov models [27, 38], neural networks [39–42], Bayesian networks [20, 43], graph-based methods [44, 45] and some common machine learning classifiers such as k nearest neighbor [46] and support vector machines [47].

Generally, learning-based methods can lead to relatively acceptable results in terms of false alarm and detection rates but, most of them cannot perform well enough in distributed and variable environments.Therefore, some studies focused on artificial immune systems for anomaly detection. As mentioned before, this approach is a main branch of computational intelligence and is inspired from Human Immune Systems. Due to distributed, adaptive, self-organized and lightweight nature of HIS, AISs can be a suitable option for anomaly detection in distributed and variable environments. So, some researches tried to detect anomalous system calls using negative selection algorithms [48] and some Danger theory methods like DCA [10–16, 59–62, 64–68] and TLR [17–19].

DCA algorithm is an earliest method in danger theory field and is implemented by Greensmith *et al.* [10–16, 59–61]. The DCA is an intelligent way of fusing and correlating information from different signal sources like PAMP signal, Safe signal, danger signal and inflammation. This algorithm simulates power of dendritic cells, which are able to activate or suppress immune responses by the correlation of signals representing their environment, combined with the locality markers in the form of antigens. The DCA is not in the scope of this paper, refer readers to its related references for further information.

TLR algorithm which is another method in danger theory field is proposed by Twycross *et al.* [17–19]. TLR employs both DCs and TCs, and simulates the function of adaptive and innate immune systems and interaction between them simultaneously. Here the DCs collect antigens and at the same time process signals. Unlike DCA, this algorithm does not utilize different groups of input signals and only utilizes from danger signals.

Finally, STLR which is our detection method falls into danger theory field, too. This algorithm is an extended form of TLR and is applied on system calls and their arguments for anomaly detection. This algorithm show considerable improvement in detection rate and accuracy respect to the other methods in artificial intelligence field.

## 3 Biological Inspiration

In this section we describe the biological principles of human immune system that are the base of the danger theory algorithms. As mentioned in previous sections, two important layers of HIS architecture are innate and adaptive immune systems that danger theory methods are inspired from both of them. Two critical agents of these layers are dendritic cells (DC) and T cells (TC). In human body, DCs have a dual role, as garbage collectors for tissue debris, and as commanders of the adaptive immune system. DCs belong to the innate immune system, and do not have the adaptive capability of the lymphocytes of the adaptive immune system [12]. These cells have three distinct states: immature, semi-mature and mature. Figure 1 shows these different types of DCs. As shown in this figure, immature DCs exist in tissue and the others are in lymph node.

Immature dendritic cells (iDC) capture antigens and alarm signals using their receptors. Each iDC has two types of receptors: antigen receptors (AgR), that is responsible for antigen reception, and toll-like receptors (TLR) that capture different alarm signals [3]. The alarm signals are derived from numerous sources, including pathogens, healthy dying cells, damaged cells and inflammation. From this viewpoint, all signals can be divided into four categories namely PAMP signals (PS), danger signals (DS), safe signals (SS) and inflammation. Each DC has the capability to combine input signals to produce a set of output signals [14].

PAMP or Pathogenic associated molecular patterns

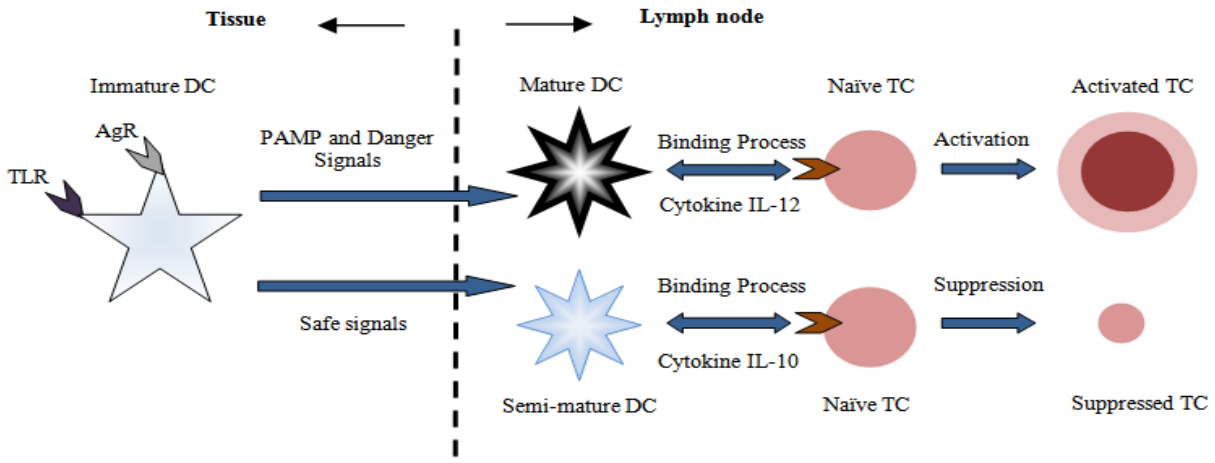


**Figure 1**. Illustration of different types of dendritic cells and T cells.

are proteins expressed exclusively by bacteria, which can be detected by DCs and their presence usually indicates an anomalous situation. Danger signals are produced as a result of unplanned necrotic cell death. DCs are sensitive to changes in danger signal concentration. The presence of danger signals may or may not indicate an anomalous situation, however the probability of an anomaly is higher than under normal circumstances [12].

Safe signals are produced via the process of normal cell death, namely apoptosis. Cells must apoptose for regulatory reasons into the tissue. The presence of the safe signal almost certainly indicates that no anomalies are present.On the other hand various immune-stimulating molecules can be released as a result of injury. It is not possible to say whether an anomaly is more or less likely if inflammatory signals are present. However, their presence amplifies the effects of three other signals [14]. Once iDCs collect antigens and signals, they act as natural data fusion agents and translate the signal information received in the tissue into a context for antigen presentation, i.e. the antigen presented in an overall "normal" or "anomalous" context. iDCs that are exposed to predominantly PAMP and danger signals turn into mature dendritic cells (mDC) and those that are exposed to predominantly safe signals turn into semi-mature dendritic cells (smDC).

After this process, smDCs and mDCs migrate from the tissue compartment to a lymph node. In the lymph node, DCs attempt to bind expressed antigen with receptors of naive T cells (nTC). nTCs with a high enough affinity for the presented antigen are influenced by the output signals of the DC. When an nTC in the lymph node binds to those antigens collected by iDCs, it will be activated only if the antigens are presented by an mDC. This is because mDCs secrete a type of cytokines called IL-12 which activates nTCs, while smDCs secrete a type of cytokines called IL-10 which suppresses nTCs [3].

Finally, it should be noted that each of the danger theory algorithms are inspired only from some parts of HIS mechanisms. For example The DCA algorithm has strong signal processing based on PAMP, safe and danger signals but, TLR and STLR algorithms do not use different groups of input signals and only utilize from danger signals. DCA algorithm ignores the function of adaptive immune system and only simulate power of dendritic cells but, TLR and STLR simulate the function of adaptive and innate immune systems and interaction between them simultaneously. In the following sections, we describe these algorithms in detail.

## 4 TLR Algorithm

As mentioned in previous sections, our proposed method called STLR is an extended version of TLR algorithm and falls in the danger theory field. So, in this section we describe TLR algorithm and explain how it works. TLR algorithm emphasizes on the interaction between adaptive and innate immune systems. It employs dendritic cells (DC) and T cells (TC) as two critical agents for anomaly detection. According to biological inspiration, both DCs and TCs appear in different states.

Three various types of DCs are immature DC (iDC), semi-mature DC (smDC) and mature DC (mDC). iDCs are responsible for gathering antigens and signals in tissue compartment and they have two receptors namely AgR, and DSR for capturing antigens and danger signals respectively. In contrast, mDCs and smDCs are responsible for presenting antigens to TCs and secreting different cytokines (IL-10 and IL-12)

in lymph node. On the other hand, three various types of TCs are nave TC (nTC), activated TC (aTC) and suppressed TC (sTC). nTCs receive antigens and secreted cytokines with their receptors in lymph node. If an nTC binds to an antigen and receives cytokine IL-12 with its receptor, it will be activated and turned into aTC. Otherwise if it receives cytokine IL-10 with its receptor, it will be suppressed and turned into sTC. TLR algorithm is completed in has two phases: training phase and detection phase.In the following subsections we investigate these two phases, separately.

### 4.1 Training Phase

The training phase of TLR algorithm has two main stages: initializing signal receptors of iDCs in tissue compartment and initializing antigen receptors of nTCs in lymph node. Like other danger theory methods, TLR requires a number of external sources for generating danger signals. So, for initializing signal receptors of iDCs, first the sources of danger signal must be indicated (here we consider CPU and memory usage as danger signals) and then, all possible values that are generated with these sources should be collected in a specific list which is named "danger signal values". After that, some signal values that are created using normal training samples must be removed from "danger signal values" list according to negative selection mechanism to finally it contains only some signal values that are not generated in normal situations. This new list is named "non-self danger signal values" and in training phase, all signal receptors of iDCs are randomly initialized with one of its members.

Antigen receptors of TCs can be initialized similar to signal receptors of DCs. TLR does not utilize antigen structures and assumes that each antigen only has an identifier. This identifier is not unique for each, but it is a member of a fixed and finite list which is named "antigen identifier values". Here we consider Process ID (PID) of each system call as identifier. So, for initializing the antigen receptors of TCs, first all the values of antigen identifier of normal training samples must be indicated and then, they should be removed from "antigen identifier values" list according to negative selection mechanism to finally new set contains only some antigen values that are not generated in normal situations. This set is named "non-self antigen identifier values" and in the training phase, all antigen receptors of TCs are randomly initialized with one of its members.

### 4.2 Detection Phase

The detection phase of TLR algorithm starts from tissue compartment and ends in lymph node. Once an antigen (a test sample that its type is unknown) enters to tissue compartment, it is captured by one or more iDCs. On the other side, danger signals are generated from their source in different times and each signal is captured only with iDC receptors that has the same value. Each iDC remains in the tissue for a certain time to capture antigens and signals using its receptors. After this time, it translates the received information in the tissue into a context for antigen presentation. If iDC is exposed to predominantly danger signals, it turns into an mDC, otherwise if it does not receive a danger signal, it turns into a smDC.

After this process, deformed iDC (mDC or smDC) migrates from tissue compartment to a lymph node to present its antigen to nTC storage. At this step, each detector of nTC storage receives antigen and signals presented by mDC or smDC using its receptors and evaluates this antigen for binding. Then, if the value of an nTC receptor is exactly equal with the value of a present antigen, the receptor binds to the antigen, but it will be activated only if the antigen is presented by an mDC. An mDC secrets a type of cytokines called IL-12 which activates nTCs. This activated nTC (aTC) can kill the antigen. So, in this case, TLR algorithm assigns abnormal label to the antigen.

In the case that an nTC receptor binds to the antigen which is presented by an smDC, the nTC will be suppressed, because an smDC secrets a type of cytokines called IL-10 which it suppresses nTCs. This suppressed nTC (sTC) cannot kill the antigen. So, TLR assigns normal label to the antigen.

As the final state, it is evident that if any nTC receptors do not bind to the antigen, TLR assigns normal label to that antigen, because the killing process of antigen must be done using an nTC which is binds to it. Figure 2 shows the pseudo code of TLR algorithm.

## 5 STLR: The Proposed Algorithm

In this section we describe our proposed algorithm and its implementation in detail. As mentioned in previous sections, the danger theory algorithms lead to more accurate and scalable results since they are modeling both adaptive and innate immune system of the body and their interaction in the system. Our proposed method falls in danger theory field.

As discussed in previous section, TLR algorithm does not utilize antigen structures and assumes that each antigen only has an identifier like PID or system call number. So, this algorithm has to do binding process using exact matching. This process is even sensitive to very small changes so, it is not suitable in many applications. In order to solve this problem, here

TLR Algorithm

**1. Training Phase**
- 1.1. Initializing Signal receptors of iDCs.
- 1.2. Initializing antigen receptors of TCs.

**2. Detection Phase**
- 2.1. Processes of tissue compartment
  - 2.1.1. Capturing antigens by antigen receptors of iDCs.
  - 2.1.2. Capturing danger signals by signal receptors of iDCs.
  - 2.1.3. Processing captured signals by iDCs using exact matching.
  - 2.1.4. Turning iDCs to mDCs or smDCs according to signals.
  - 2.1.5. Migrating smDCs and mDCs.
- 2.2. Processes of lymph node
  - 2.2.1. presenting antigens of smDCs or mDCs to nTCs.
  - 2.2.2. exact matching between antigens and nTCs for binding.
  - 2.2.3. assigning labels to entered antigens as follows
    - 2.2.3.1. antigen is abnormal if it binds to at least one TC and it had been presented by an mDC.
    - 2.2.3.2. otherwise it is normal.

**Figure 2**. Pseduo Code of TLR Algorithm.

we propose a novel algorithm which is called STLR which is an extended form of TLR algorithm.

This method models each antigen as a vector of features. This feature vector forms the structure of the antigen and that's why we call our algorithm "structural TLR" or STLR. The structure of each antigen can be consisted of different features in various datasets. Figure 3 represents the structure of an antigen in a general state, where $F$ is the symbol of a feature and its subscript indicates the number of that feature. So, $N$ is the total number of all features.

| $F_1$ | $F_2$ | $F_3$ | $F_4$ | $F_5$ | ... | $F_{N-1}$ | $F_N$ |
|---|---|---|---|---|---|---|---|

**Figure 3**. An overal view of the Structure of each antigen.

STLR and TLR algorithms are basically the same but, there are some differences between these two. To indicate these differences, in Figure 4, we describe the pseudo code of STLR algorithm in the same manner with Figure 2. According to this figure, STLR is completed in training and detection phases, too.

## 5.1 Training Phase

The training phase of STLR algorithm has two main stages: initializing signal receptors of iDCs in tissue compartment and, Defining non-self region as anomaly detector in lymph node. In the following sections we will describe these two processes.

STRL Algorithm

**1. Training Phase**
- 1.1. Initializing Signal receptors of iDCs.
- 1.2. Defining non-self region as anomaly detector.

**2. Detection Phase**
- 2.1. Processes of tissue compartment
  - 2.1.1. Capturing antigens by antigen receptors of iDCs.
  - 2.1.2. Capturing danger signals by signal receptors of iDCs.
  - 2.1.3. Processing captured signals by iDCs using threshold.
  - 2.1.4. Turning iDCs to mDCs or smDCs according to signals.
  - 2.1.5. Migrating smDCs and mDCs.
- 2.2. Processes of lymph node
  - 2.2.1. presenting antigens of smDCs or mDCs to nTCs.
  - 2.2.2. binding process between antigens and non-self region.
  - 2.2.3. assigning labels to entered antigens as follows
    - 2.2.3.1. antigen is abnormal if it binds with non-self region and it had been presented by an mDC.
    - 2.2.3.2. otherwise it is normal.

**Figure 4**. Pseduo Code of STLR Algorithm.

### 5.1.1 Initializing Process

Like TLR algorithm, our proposed algorithm requires a number of external sources for generating danger signals. Therefore, to initializ signal receptors of iDCs, STLR algorithm first indicates the sources of danger signals and then collects all possible values that are generated with these sources in *"danger signal values"* list. After that, by using negative selection mechanism we remove some signal values from *"danger signal values"* list, these values are are created by normal training samples then, STLR algorithm creates *"non-self danger signal values"* list which contains only some signal values that are not generated in normal situations and in training phase, all signal receptors of iDCs are randomly initialized with one member of this list.

### 5.1.2 Defining the Self and Non-self Regions

The first difference between TLR and STLR algorithms is in this step of training phase. Unlike TLR which initializes antigen receptors of nTCs by ignoring the structure of antigens, STLR defines non-self region by using the structure of antigens. This region belongs to adaptive immune system and can be used as anomaly detector in detection phase.

Before defining non-self region, we need to define self region using normal training data. It is noteworthy that each of the normal training data's structure is

similar to Figure 3. Self region consists of putting some spheres with constant radius side by side such that the centers of them are normal training samples. This region can be defined by (1).

$$Self - Region = \bigcup_{1<i<Y} SS_i \;\; ; SS_i = (N_i, r_s) \quad (1)$$

Where $r_s$ is the constant radius of all self spheres and $n_i$ is $i^{th}$ sample in normal training dataset and the $i^{th}$ center of Self Sphere $SS_i$. $Y$ is the number of samples in the normal training dataset.

The second difference between TLR and STLR algorithms appears in this signal processing step. In TLR algorithm, iDCs utilize exact matching and capture an entered signal as their danger signal only if it matches with pre-initialized values of their receptors.

After defining the self-region, we can define non-self region completely out of the self-region. The use of this region will be described in detection phase in more detail.

### 5.2 Detection Phase

Like TLR, the detection phase of STLR algorithm starts from tissue compartment and ends in lymph node. The Detection phase of STLR algorithm has three main stages: maturation process, binding process, and label assignment process. The first stage is done in tissue compartment and two other stages are performed in lymph node. Figure 5 shows the sequence of these stages in the form of a flowchart. In the following section we will describe these three processes separately.

#### 5.2.1 Maturation Process

This step is the first part of detection process which starts from tissue compartment. In this tissue, there are many iDCs for signal and antigen capturing. Once an antigen enters tissue compartment, it is captured by one or more iDCs. On the other side, danger signals are generated from their sources in different times. After signal and antigen capturing by iDCs, the captured danger signals must process using these immature dendritic cells for maturing process. But in STLR algorithm, iDCs uses a threshold and capture an entered signal as their danger signal only if it is higher than a predefined threshold. When iDCs capture danger signals, their danger signal receptors (DSRs) will be set to one, otherwise it will be set to zero. Equation (2) shows this process.

$$DSR = \begin{cases} 1 \text{ if } & captured\ signal\ by\ iDC \geq T \\ 0 \text{ if } & otherwise \end{cases} \quad (2)$$

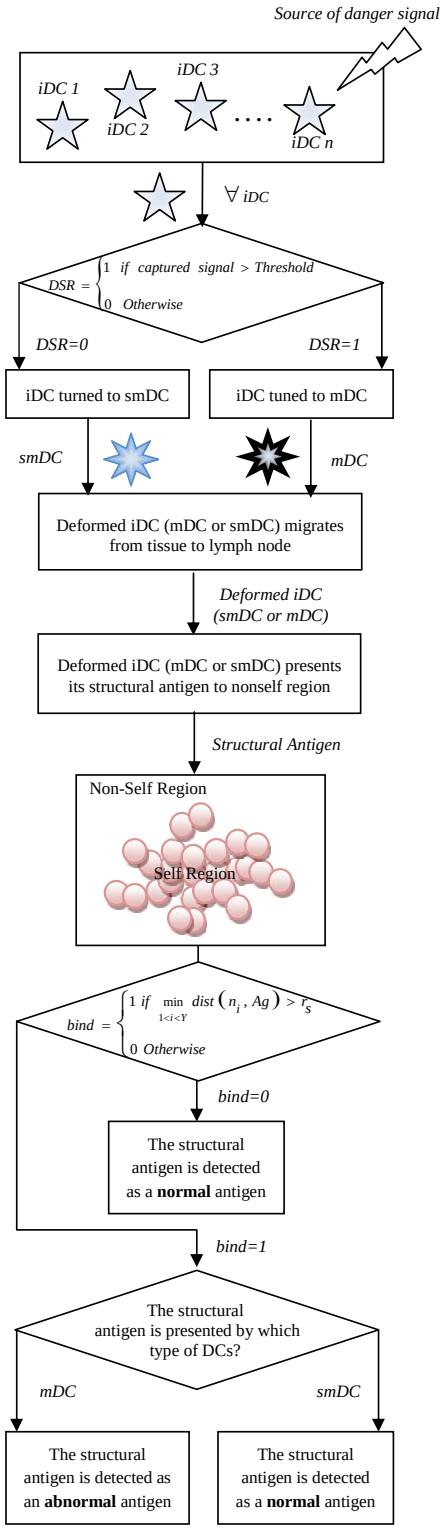


**Figure 5**. Flowchart of STLR algorithm.

Where $T$ is a threshold which can be defined in training phase according to normal training data. The mentioned signal processing in STLR algorithm can decrease the amount of sensitivity of TLR with respect to exact value of danger signals.

Generally, each iDC remains in the tissue for a certain time to capture antigens and signals using its receptors. After this time; if iDC is exposed to predominantly danger signals and the value of its DSR is equal to 1, it matures completely and turns into a mature dendritic cell (mDC), otherwise if it did not received a danger signal and the value of its DSR is equal to 0, it matures incompletely and turns into a semi-mature dendritic cell (smDC).

#### 5.2.2 Binding Process

After maturation process, deformed iDC (mDC or smDC) migrates from tissue compartment to a lymph node to present its antigen to non-self region for binding process. Non-self region, which is defined in training phase, plays the role of anomaly detector in STLR algorithm.

Binding process in lymph node is the last difference between TLR and STLR algorithms. As mentioned previously, this process is based on exact matching between antigens and nTC receptors in TLR algorithm, but STLR checks the ability of binding between antigens and non-self region using (3).

$$DSR = \begin{cases} 1 \text{ if } & min_{1<i<Y}\ dis(n_i, Ag) > r_s \\ 0 \text{ if } & otherwise \end{cases} \quad (3)$$

Where $Ag$ is a structural antigen which is presented using mDCs or smDCs, $r_s$ is the constant radius of all self spheres, $Y$ is the total size of normal training dataset, and $n_i$ is $i^{th}$ sample in normal training dataset and the $i^{th}$ center of Self Sphere $SS_i$. $dist(.)$ is a function that calculates Euclidean distance between its arguments. In this equation, *bind* parameter is equal to 1 only if the entered antigen binds with non-self region and becomes out of self region. Otherwise this parameter is equal to 0 if the entered antigen does not bind with non-self region and becomes in self region.

#### 5.2.3 Label Assignment Process

After binding process, STLR algorithm will be capable to assign "normal" or "abnormal label to each entered antigen. This process is performed based on the value of parameter which is obtained using (3).

According to biological inspiration of HIS, if the antigen binds with non-self region and the *bind* parameter is equal to 1, STLR algorithm assigns "abnormal" label to that antigen if it is presented by an mDC, otherwise if the antigen is presented by an smDC, this algorithm assigns "normal" label to it.

If the antigen does not bind with non-self region and the *bind* parameter is equal to 0, STLR algorithm assigns normal label to that antigen since, this situation shows that the entered antigen will not be detected by non-self region as anomaly detector.

## 6 Evaluation

In this section we evaluate the performance of some classic methods in AIS field (like NS, DCA and TLR algorithms) and our proposed algorithm to show that STLR algorithm performs better and produces higher detection rate and lower false alarm rate compared to previous algorithms. For this purpose, first we describe a dataset which is constructed for this paper and another dataset that is used for testing. Then we describe the evaluation metrics used for comparing the mentioned algorithms and finally, we present experimental results.

### 6.1 Dataset

In order to evaluate and compare the performance of TLR and STLR algorithms, we need a dataset with two types of record: i) structural antigen records and ii) danger signal records.

Twycross has evaluated TLR algorithm using two datasets called rpc.statd and wuftpd in his Ph.D. thesis [17]. The rpc.statd dataset is used to provide an initial analysis of system calls and signal levels for a server under normal and attack conditions and the wuftpd dataset expands the range of normal and attack usage and signal sources for a second FTP server.

Both these datasets satisfy the second condition of our needed dataset but none of them support the first condition. These datasets contain normal and abnormal antigens but, they do not take the antigens' structure into account since, the TLR algorithm does not need this information. So, they are not suitable for evaluating STLR algorithm. Therefore, we constructed a new dataset called "Structural Syscall" which satisfies both conditions. Section 6.1.1 describes the construction process of this dataset. For more accurate evaluation, we used another dataset called NSL-KDD. Unlike rpc.statd and wuftpd datasets, this new dataset satisfies only the first condition and does not contain danger signal records. To overcome this problem, we used a technique presented in [13] to generate danger signals for each antigen. Section 6.1.2

describes this process in details.

### 6.1.1 Structural Syscall dataset

In this section we describe the construction process of *Structural Syscall* dataset. In order to obtain a suitable dataset, we had two hosts running Ubuntu 10.04 with kernel 2.6.32 with one of them having audit daemon installed on it. Audit daemon is a user space component for Linux Auditing System [49] and is responsible for logging system calls initiated by different programs on the host system. With the aid of audit daemon, we were able to define which system calls to audit [50]. Additionally on the target host we have some users who are doing their regular jobs indicating the normal behavior of the system.

We also, have an intruder who's going to access the target machine remotely through an open port provided by the K-beast Rootkit (which is the latest public rootkit available for kernel 2.6.32) [51]. Since our algorithm uses the contextual information like CPU usage and memory usage of the system as danger signals, we also had to issue a top command periodically to obtain the run time information.

To provide the required dataset, we parsed the log file created by audit daemon and extracted each system call as an antigen whose structure is defined by its parameters and some additional arguments. These additional fields are pid, gid, uid, a1 and a2 The structure of this antigen is presented in Figure 6.

| $F_1$ | $F_2$ | $F_3$ | $F_4$ | $F_5$ | $F_6$ |
|---|---|---|---|---|---|
| SysCall-No | Mode | Flag | Uid | Gid | Pid |

**Figure 6**. Antigen structure in "Structural Syscall" dataset.

Here PID is the id of the process initiating the system call, UID and GID are the user id and group id of the user and a1 and a2 are especial arguments which are totally system call dependent such as mode and flag which for simplicity are converted into binary format. We also tried to group the system calls or antigens according to their contextual signals which are the run time information such as CPU and memory usage of the system at that time.

Our final dataset was gathered in two phases. One is the training phase data which consist of 94000 records. All these records are collected during normal usage of the system before any attack has happened upon the system. The other is the test phase data. This dataset contains 3093 records with 2345 record labeled as normal and 748 records labeled as abnormal. These records are calculated regardless of the records related to system run time information.

### 6.1.2 NSL-KDD dataset

NSL-KDD dataset [52] is an improved version of "KDD cup 1999" or KDD99 dataset [53, 54] which is an standard and famous dataset of UCI repository. The KDD99 dataset was derived in 1999 from the DARPA98 network traffic dataset by assembling individual TCP packets into TCP connections. It was the benchmark dataset used in the International Knowledge Discovery and Data Mining Tools Competition. Each TCP connection has 41 features with a label which specifies the status of a connection as either being normal, or attack.

NSL-KDD dataset solves some of the inherent problems of the KDD99 dataset by removing the redundant records and applying some other processing on it according to [52]. So, in this paper we utilized this dataset for evaluating our proposed algorithm. Here we used only 19 of 41 features based on feature selection method proposed in [55]. Thus, antigen structure in this dataset can be shown as a feature vector with 19 features which is illustrated in Figure 7. More details about these 19 features can be found in references [52–54].

As mentioned before, this dataset satisfies only the first condition of our needed dataset and do not contain danger signal records. For solving this problem, we used a technique presented in [13] to generate danger signals for each antigen. In this method, four data items of each antigen record, that have the largest standard deviation, create the danger signals. For each of these attributes the mean was calculated over all normal records. Subsequently, the absolute difference from the mean was calculated for each entered antigen, within four selected attributes, separately. The average of the four attribute mean differences comprises the derived danger signal concentration. NSL-KDD dataset is prepared for evaluation, after applying this process on its records.

## 6.2 Evaluation Criteria

If we consider normal label as "Negative" and abnormal label as "Positive", then we can define four basic concepts; True Negative (TN) which is the number of samples that their predicted labels and actual labels are the same as Negative, True Positive (TP) which is the number of samples that their predicted labels and actual labels are the same as positive, False Negative (FN) which is the number of samples that their predicted labels are Negative while their actual labels are Positive and False Positive (FP) which is the number of samples that their predicted labels are Positive while their actual labels are Negative.

For the evaluation, we use seven criteria that could

| $F_1$ | $F_2$ | $F_3$ | $F_4$ | $F_5$ | $F_6$ | $F_7$ |
|---|---|---|---|---|---|---|
| Duration | Service | Flag | src_bytes | dst_bytes | wrong_fragment | Urgent |

| $F_8$ | $F_9$ | $F_{10}$ | $F_{11}$ | $F_{12}$ | $F_{13}$ | $F_{14}$ |
|---|---|---|---|---|---|---|
| hot | num_failed _logins | num _compromised | su_attempted | num_root | num_file _creations | num _shells |

| $F_{15}$ | $F_{16}$ | $F_{17}$ | $F_{18}$ | $F_{19}$ |
|---|---|---|---|---|
| num_access _files | Count | srv_count | dst_host _count | dst_host _srv_count |

**Figure 7**. Antigen structure in "Structural Syscall" dataset.

be defined based on these concepts. The first criterion is False Negative Rate (FNR) which indicates the rate of False Alarms (FA) for an HIDS. The ideal value of this criterion is 0 and it can be calculated using (4).

$$FPR = FA = FP/(TN + FP) \quad (4)$$

The other criterion is True Positive Rate (TPR) which indicates the Detection Rate (DR) of our system and also known as sensitivity and recall. The ideal value of this criterion is 1 and it is shown in (5).

$$TPR = DR = sensitivity = TP/(TP + FN) \quad (5)$$

The simultaneous evaluation of two above criteria can be a suitable option to illustrate the power of our system. ROC [14] is a space that its horizontal and vertical axes shows FPR and TPR, respectively. So, the power of each classifier can be indicated as a unique point in this space. A ROC graph is made from join of point (0,0) to the unique point of classifier and unique point to point (1,1). The area under this graph (SROC) which is calculated from (6) is our 3rd criterion. According to this equation, the unique point of a perfect classifier is (0,1) and SROC is 1 for it.

$$S_{ROC} = \frac{2 \times (1 - FPR) \times TPR}{2} + \frac{(FPR \times TPR)}{2} + \frac{(1 - FPR) \times (1 - TPR)}{2} \quad (6)$$

Two other criteria are TNR which is the complement of FPR and PPV or precision that indicates which rate of predicted abnormal records is actually "abnormal". The ideal values of these two criteria are 1 and they are shown in (7) and (8) respectively.

$$TNR = specify = 1 - FPR \quad (7)$$

$$PPV = precision = TP/(TP + FP) \quad (8)$$

F-score which computes a harmonic mean for TPR and PPV and Accuracy that indicate which rate of normal or abnormal records is predicted (labeled) correctly are two final criteria. The ideal value for these two criteria is 1 and they are shown in (9) and (10) respectively.

$$F1 = F\_score = (2 \times PPV \times TPR)/(PPV + TPR) \quad (9)$$

$$Acc = (TP + TN)/(TP + TN + FP + FN) \quad (10)$$

### 6.3 Experimental Results

In order to evaluate the detection power of our proposed method, we applied TLR, STLR and two other algorithms named NS and DCA on two datasets which are explained in Section 6.1, separately and then, we compared their performances according to the criteria described in Section 6.2. For this experiment, we implemented Negative Selection (NS) algorithm with variable detector size according to [57, 58]. DCA algorithm was implemented according to technique which is discussed in [13] and finally, TLR and STLR algorithms were implemented according to Section 4 and Section 5, respectively. But before discussing about the results of this experiment, we present some details about DCA and NS algorithms.

DCA algorithm is another algorithm in danger theory field. DCA ignores the function of adaptive immune system and only simulate power of dendritic cells which are able to activate or suppress immune responses by the correlation of signals representing their environment, combined with the locality markers in the form of antigens. Unlike previous algorithms, DCA has no training phase.

The DCA starts with creating a population of immature DCs. Each iDC collects many antigens and four different signals (safe, PAMP, danger and inflammation signals). Moreover each antigen can be collected by more than one iDC. Each iDC processes its input signals periodically and transforms them by an equation [13] to three output concentrations: co-

stimulatory molecules (csm), smDC cytokines (semi) and mDC cytokines (mat). Csm tracks the maturation of a DC. When this quantity is larger than a pre-defined threshold, the corresponding DC is said to be mature. The other two outputs, semi and mat, will determine if this DC develops to be an smDC or mDC. Each smDC assigns "normal" label to its antigens and each mDC assigns "abnormal" label to them. Whereas, each antigen can be collected by more than one iDC, so it can have more than one label. In such case, the final label of each antigen can be calculated by voting.

Negative Selection (NS) algorithm is an algorithm in the first generation of artificial immune system field which unlike previous algorithms is not in danger theory branch. Unlike DCA algorithm, NS algorithm ignores the function of innate immune system and only simulates the behavior of lymphocytes of adaptive immune system (T and B cells).

NS algorithm has two phases. In training phase, first it indicates self region which consists of putting some spheres with constant radius side by side such that the centers of them are normal training samples. After that it defines detector storage which contains a number of spheres that the centers of them are determined randomly and their radiuses are variable. The detail information for calculating variable radiuses can be found in [57, 58]. In detection phase, each antigen is presented to detector storage. Then each detector evaluates this antigen for binding. NS algorithm assigns "abnormal" label to each antigen which it does not bind to any detectors,otherwise they assign "normal" label to it.

Table 1 to Table 4 are confusion matrixes which show the results of applying NS, DCA, TLR and STLR algorithms on Structural Syscall dataset. Here Nact and Pact indicate the number of records with actual labels of Negative (normal) and positive (anomalous), respectively. Equally, Npred and Ppred indicate the number of records with predicted labels of normal and anomalous.

**Table 1**. Confusion matrix of NS algorithm in *Structural Syscall* dataset.

| $NS$ | $N_{pred}$ | $P_{pred}$ | $Total_{act}$ |
|---|---|---|---|
| $N_{act}$ | 2344 | 1 | 2345 |
| $P_{act}$ | 321 | 427 | 748 |
| $Total_{pred}$ | 2665 | 428 | 3093 |

According to these tables and as discussed in Section 6.1.1, the total number of records with actual abnormal label is 748 but, since both of these algorithms suffer from a little inaccuracy; the number of normal and abnormal records predicted by them is not exactly the same as actual number of normal and abnormal records. We are also aware of the fact that the error rate of these four algorithms is different from each other. Table 5 presents a quantitative comparison of applying these four algorithms on Structural Syscall dataset and Figure 8 visually compares the power of danger theory algorithms (DCA, TLR and STLR) in terms of criteria described in Section 6.2.

**Table 2**. Confusion matrix of DCA algorithm in *Structural Syscall* dataset.

| $DCA$ | $N_{pred}$ | $P_{pred}$ | $Total_{act}$ |
|---|---|---|---|
| $N_{act}$ | 1686 | 659 | 2345 |
| $P_{act}$ | 2 | 746 | 748 |
| $Total_{pred}$ | 1688 | 1405 | 3093 |

**Table 3**. Confusion matrix of TLR algorithm in *Structural Syscall* dataset.

| $TLR$ | $N_{pred}$ | $P_{pred}$ | $Total_{act}$ |
|---|---|---|---|
| $N_{act}$ | 2125 | 220 | 2345 |
| $P_{act}$ | 274 | 474 | 748 |
| $Total_{pred}$ | 2399 | 694 | 3093 |

**Table 4**. Confusion matrix of STLR algorithm in *Structural Syscall* dataset.

| $DCA$ | $N_{pred}$ | $P_{pred}$ | $Total_{act}$ |
|---|---|---|---|
| $N_{act}$ | 2164 | 181 | 2345 |
| $P_{act}$ | 0 | 748 | 748 |
| $Total_{pred}$ | 2164 | 929 | 3093 |

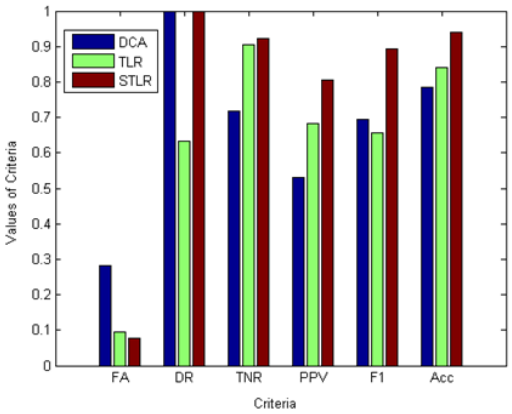


**Figure 8**. The comparison between DCA, TLR and STLR algorithms in *Structural Syscall* dataset.

The first four columns of Table 5 show the results of four mentioned algorithms in terms of different criteria and next column presents the average results of the first columns. Sixth column shows the amounts of

**Table 5**. Comparative results between NS, DCA, TLR and STLR algorithms in *Structural Syscall* dataset

| Criteria \ Algorithms | NS | DCA | TLR | STLR | Average | STLR improvement vs. Average (%) | STLR improvement vs. TLR (%) |
|---|---|---|---|---|---|---|---|
| FPR(FA) | 0.0004 | 0.2810 | 0.0938 | 0.0772 | 0.1131 | + 3.59 | + 1.66 |
| TPR(DR) | 0.5709 | 0.9973 | 0.6337 | 1.0000 | 0.8005 | + 19.95 | + 36.63 |
| TNR(Specificity) | 0.9996 | 0.7190 | 0.9062 | 0.9228 | 0.8869 | + 3.59 | + 1.66 |
| PPV(Precision) | 0.9977 | 0.5310 | 0.6830 | 0.8052 | 0.7542 | + 5.10 | + 12.22 |
| F-measure(F1) | 0.7262 | 0.6930 | 0.6574 | 0.8921 | 0.7422 | + 14.99 | + 23.47 |
| Accuracy(Acc) | 0.8959 | 0.7863 | 0.8403 | 0.9415 | 0.8660 | + 7.55 | + 10.12 |
| $S_{Roc}$ | 0.7852 | 0.8582 | 0.7699 | 0.9614 | 0.8437 | + 11.77 | + 19.15 |

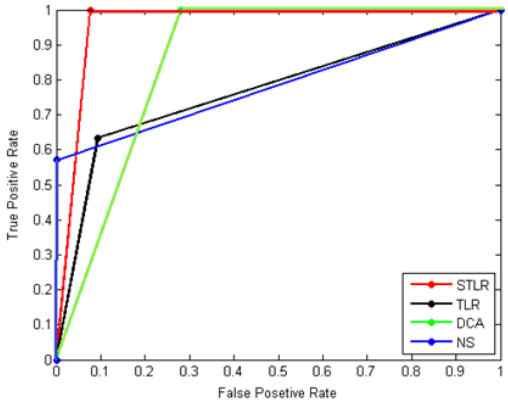


**Figure 9**. The ROC graph of NS, DCA, TLR and STLR algorithms in *Structural Syscall* dataset.

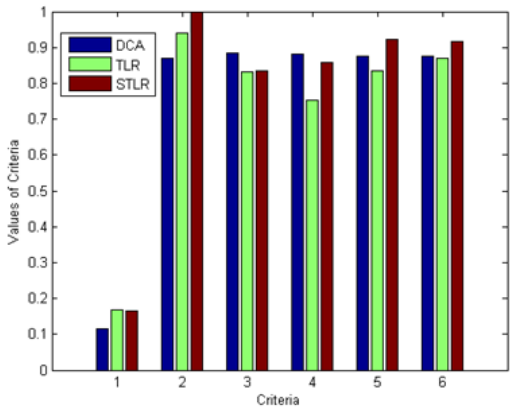


**Figure 10**. The comparison between DCA, TLR and STLR algorithms in NSL_KDD dataset.

STLR improvement versus Average values in terms of different criteria. Eventually, since STLR is a modified version of TLR algorithm, the last column of this table presents the quantitative comparison between these two algorithms to show the improvement.

Finally, Figure 9 compares the ROC graph of four algorithms in *Structural Syscall* dataset. As discussed previously, a large amount of the area under the graph is the indication of better classification. According to Figure 9 and Table 5, this area is 0.7852, 0.8582, 0.7699 and 0.9614 for NS, DCA, TLR and STLR algorithms, respectively. This also confirms that our proposed method performs much better than the others.

Table 6 presents a quantitative comparison of applying four mentioned algorithms on NSL-KDD dataset and Figure 10 visually compares the power of danger theory algorithms in terms of criteria described in Section 6.2. Moreover Figure 11 compares the ROC graph of four algorithms in NSL-KDD dataset. According to Figure 11 and Table 6, this area is 0.8287, 0.8766, 0.8932 and 0.9171 for NS, DCA, TLR and STLR algorithms, respectively.

Totally, the results of TLR and STLR algorithms on two mentioned datasets show that although the num-

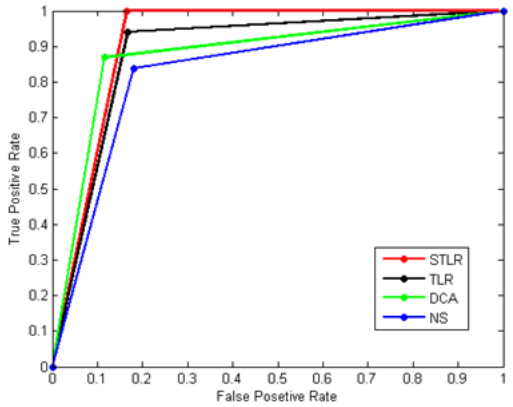


**Figure 11**. The ROC graph in NS, DCA, TLR and STLR algorithms in NSL_KDD dataset.

**Table 6**. Comparative results between STLR and TLR algorithms in NSL_KDD dataset.

| Criteria \ Algorithms | NS | DCA | TLR | STLR | Average | STLR improvement vs. Average (%) | STLR improvement vs. TLR (%) |
|---|---|---|---|---|---|---|---|
| FPR(FA) | 0.1809 | 0.1156 | 0.1675 | 0.1658 | 0.1575 | - 0.83 | + 0.17 |
| TPR(DR) | 0.8384 | 0.8687 | 0.9407 | 1.0000 | 0.9119 | + 8.81 | + 5.93 |
| TNR(Specificity) | 0.8191 | 0.8844 | 0.8325 | 0.8342 | 0.8426 | - 0.84 | + 0.17 |
| PPV(Precision) | 0.8218 | 0.8821 | 0.7520 | 0.8571 | 0.8283 | + 2.88 | + 10.51 |
| F-measure(F1) | 0.8300 | 0.8753 | 0.8358 | 0.9231 | 0.8661 | + 5.70 | + 8.73 |
| Accuracy(Acc) | 0.8287 | 0.8766 | 0.8704 | 0.9169 | 0.8732 | + 4.37 | + 4.65 |
| $S_{Roc}$ | 0.8287 | 0.8766 | 0.8932 | 0.9171 | 0.8789 | + 3.82 | + 2.39 |

ber of FA in STLR is less than that of TLR algorithm, the main difference between them is their detection rate (DR). In fact, these results are completely compatible with characteristics of these two algorithms.

As mentioned before, TLR algorithm does not utilize antigen structures and assumes that each antigen only has an identifier. This algorithm uses from *"non-self antigen identifier values"* list for anomaly detection and tries to create this list with only some identifier values that belong to abnormal antigens. But this dependency to only one parameter is the biggest issue in TLR algorithm; because there is not any identifier which has the ability to separate normal and abnormal antigens completely.

The identifier value is not unique for each antigen and there is no guarantee that the identifier values of abnormal antigens are completely different from identifier values of normal antigens. So, it is probable that TLR algorithm cannot detect an abnormal antigen because its identifier value is equal to the identifier value of a normal antigen in training phase and so this value has been removed from *"non-self antigen identifier values"* list. Because of this reason, TLR algorithm has relatively low detection rate. It is also possible that TLR algorithm detects a normal antigen as anomaly because its identifier value is not equal to any identifier values of normal antigens in training phase and so this value exists in *"non-self antigen identifier values"* list. Because of this reason, TLR algorithm does not have low false alarm rate.

STLR algorithm solves thisissue by defining a structure for each antigen (Figure 3). So, unlike TLR algorithm; each antigen in STLR is dependent to several parameters. This increases the dimension of problem space and so decreases the sensitivity to exact values of only one parameter. This algorithm uses from *"non-self region"* for anomaly detection which is defined completely out of the Self region. Here, Self region consists of putting some spheres with constant radius, rs, side by side that the centers of them are normal training samples.

So, according to (3), STLR algorithm assigns "abnormal" label to an antigen if the Euclidian distances between its location and the centers of self spheres (which are normal training data) are larger than $r_s$ or,i.e, if its location is out of the self-region. So, the value of $r_s$ has a critical role in this algorithm, because it can indicate the size of self and non-self regions indirectly. The larger values of $r_s$ can create larger self region and smaller non-self region and so they can decrease the values of false alarm and detection rate. Vice versa, the smaller values of $r_s$ can create smaller self region and larger non-self region and so they can increase the values of false alarm and detection rate.

In this paper, our goal is focused on increasing the value of detection rate and detecting all abnormal antigens. In order to make sure that this algorithm can detect all abnormal antigens perfectly, we assume that all self spheres have a constant and very small radius ($r_s = 0.001$) . For this reason, the detection rate of STLR algorithm is in an ideal level, and it is considerably higher than TLR's detection rate. But, the false alarm rate of this algorithm is a little high because of the small value of $r_s$. Here, it is notable that in this situation, the values of false alarm in TLR and STLR algorithms are almost equal, but the value of detection rate is considerably different.

As discussed before, the detection rate and false alarm rate of STLR algorithm can be flexible by changing the value of $r_s$. This flexibility is a good parameter in STLR algorithm that we intend to use it in our future works. In fact, the results of STLR algorithm can be improved by applying an optimization process on this parameter using learning automata, or by defining a variable radius for each self sphere in training phase. We have evaluated the effect of this parameter in increasing and decrising the values of false alarm and detection rate, in our other articles [69, 70].

After above discussion and comparison between

TLR and STLR algorithms in terms of detection rate, false alarm and accuracy, here we want to compare these two algorithms in terms of computational complexity for intrusion detection. According to Section 4 and Section 5, iDCs in both of these algorithms remain in the tissue for a certain time to capture antigens and signals using their receptors. For simplicity, we name this certain time, $T_{tissue}$. After that, iDCs turn to mDC or smDC and migrate to lymph node to present their captured antigens to nTC storage (in TLR algorithm) or non-self region (in STLR algorithm) for binding process. Elapsed time for binding process has an important effect on computational complexity of these two algorithms that we name it, $T_{bind}$, for simplicity. So, $T_{total}$, the total time for assigning "normal" or "abnormal" label to a test data (an entered antigen) in both TLR and STLR algorithms can be calculated by (11).

$$T_{total} = T_{bind} + T_{tissue} \tag{11}$$

Where $T_{bind}$ is calculated differently in TLR and STLR algorithms. Binding process is based on exact matching between antigens and nTC receptors in TLR algorithm. If nTC storage consists of NnTC detectors (or nTCs) and each detector has $N_{receptor}$ receptors, then $T_{bind}$ for TLR algorithm can be calculated using (12).

$$T_{bind} = N_{nTC} \times N_{receptor} \times T_{comp} \tag{12}$$

Where $T_{comp}$ is elapsed time for comparing the value of an nTC receptor with the value of presented antigen. On the other side, binding process in STLR algorithm is done by checking the ability of binding between antigens and non-self region using (3). According to (3), $T_{bind}$ for STLR algorithm can be calculated using (13).

$$T_{bind} = Y \times \left( T_{dist} + T_{comp} \right) \tag{13}$$

Where $Y$ is the total size of normal training dataset, $T_{dist}$ is elapsed time for calculating the Euclidian distance between a normal training data and presented antigen and finally, $T_{comp}$ is elapsed time for comparing the Euclidian distance between a normal training data and presented antigen with $r_s$ (the constant radius of all self spheres) for binding.

According to (12) and (13), if $Y = N_{nTC} \times N_{receptor}$ then STLR algorithm has more computational complexity than TLR algorithm because of $T_{dist}$ component. In fact this time overhead is acceptable for our proposed method to have more accurate results.

## 7 Conclusion

Danger Theory inspired algorithms are a new generation of AIS algorithms which are trying to model the interaction of adaptive and innate biological immune system. In this paper we proposed an algorithm called STLR which is an extended form of TLR algorithm.

We applied NS, DCA, TLR and STLR algorithms on two separate datasets (*Structural Syscall* and NSL_KDD) to evaluate their power of detection rate and finally we showed that using the structure of antigens as the input to the system, STLR algorithm can gain less false alarm rate, better accuracy and greater detection rate compared to other algorithms specially TLR algorithm which uses exact matching and ignores the structure of antigens.

## Acknowledgements

We would like to start by acknowledging the help and support we received from our colleagues in Operating System Security Lab (OSSL) of Alzahra University, Tehran, Iran. We would like to thank Mrs. Reyhaneh Salkhi and Mrs Fahimeh Soltani-Nejad for their efforts and helps in creation of *"structural syscall"* dataset. Finally, it is notable that the work described in this paper has been supported in part by the Iran Telecommunication Research Center (ITRC) under contract /500/6085.

## References

[1] J. Twycross, "Immune Systems, Danger Theory and Intrution Detection", Symposium on The Immune System and Cognition, (2004).

[2] Mu. Chengpo, "A Survey of Intrusion-Detection Alert Aggregation and Correlation Techniques", Journal of Computer Research and Development, 43(1), (2006).

[3] S. X. Wu, W. Banzhaf, The use of computational intelligence in intrusion detection systems: A review, Applied Soft Computing, vol. 10, pp. 1-35, (2010).

[4] L. Ruochen, D. Haifeng and J. Licheng, Immunity Clonal Strategies, 5th InternationalConference on Computational Intelligence and Multimedia Applications IEEE Computer Society, (2003).

[5] S. A. Hofmeyr, An immunological model of distributed detection and its application to computer security, PhD thesis, University Of New Mexico, (1999).

[6] N. Jerne, Towards a network theory of the immune system, Annals of Immunology (Paris), 125 (1-2), pp. 373389, (1974).

[7] P. Matzinger, Tolerance, danger and the extended family, Annual Reviews in Immunology, Vol. 12, pp. 9911045, (1994).

[8] P. Matzinger, The danger model in its historical

context, Scandinavian Journal of Immunology, 54 (1-2), pp. 4-9, (2001).
[9] Danger Theory Project Website. Retrieved January 26, 2012, from http://www.dangertheory.com/.
[10] J. Greensmith, Dendritic Cell Algorithm, Ph.D. Thesis, The University of Nottingham, (2007).
[11] J. Greensmith, U. Aickelin, Dendritic cells for real-time anomaly detection, the Workshop on Artificial Immune Systems and Immune System Modelling (AISB06), pp. 7-8, (2006).
[12] J. Greensmith, U. Aickelin, Dendritic cells for syn scan detection, the Genetic and Evolutionary Computation Conference (GECCO07), pp. 49-56, (2007).
[13] J. Greensmith, U. Aickelin, S. Cayzer, Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection, 4th International Conference on Artificial Immune Systems (ICARIS05), Springer, Berlin/Heidelberg, pp. 153-167, (2005).
[14] J. Greensmith, U. Aickelin, G. Tedesco, Information fusion for anomaly detection with the dendritic cell algorithm, Information Fusion, 11(1), pp. 21-34, (2010).
[15] J. Greensmith, U. Aickelin, J. Twycross, Detecting danger: Applying a novel immunological concept to intrusion detection systems, 6th International Conference in Adaptive Computing in Design and Manufacture (ACDM04), (2004).
[16] J. Greensmith, J. Twycross, U. Aickelin, Dendritic cells for anomaly detection, the IEEE Congress on Evolutionary Computation (CEC06), IEEE Press, pp. 664-671, (2006).
[17] J. Twycross, integrated innate and adaptive artificial immune systems applied to process anomaly detection, PhD Thesis, The University of Nottingham, January, (2007).
[18] J. Twycross, U. Aickelin, Detecting anomalous process behaviour using second generation artificial immune systems, the International Symposium on Recent Advances in Intrusion Detection (RAID), (2008).
[19] J. Twycross, U. Aickelin, An immune-inspired approach to anomaly detection, Handbook of Research on Information Assurance and Security, Information Science Reference, pp. 109121, chapter X, (2007).
[20] D. Mutz, F. Valeur, G. Vigna, Christopher Kruegel, Anomalous system call detection, ACM Transactions on Information and System Security (TISSEC), 9(1), pp. 61-93, (2006).
[21] C. Ko, M. Ruschitzka, K. Levitt, Execution Monitoring of Security-Critical Pro- grams in Distributed Systems: A Specification-based Approach, IEEE Symposium on Security and Privacy, pp.175-187, (1997).
[22] M. Bernaschi, E. Gabrielli, and L. V. Mancini, Remus: A Security-Enhanced Operating System. ACM Transactions on Information and System Security, 5(1), pp. 36-61, (2002).
[23] S. N. Chari, P. C. Cheng, Bluebox: A policy-driven, host-based intrusion detection system, ACM Transaction on Information and System Security (TISSEC), 6(2), pp. 173-200, (2003).
[24] D. Wagner, D. Dean, Intrusion detection via static analysis, IEEE Symposium on Research in Security and Privacy, (2001).
[25] I. Goldberg, D. Wagner, R. Thomas, E. A. Brewer, A secure environment for untrusted helper applications, 6th Usenix Security Symposium, (1996).
[26] N. Provos, Improving host security with system call policies, SSYM03: 12th Conference on USENIX Security Symposium, USENIX Association, pp. 257272, (2003).
[27] C. Warrender, S. Forrest, B. Pearlmutter, Detecting intrusionsusing system calls: Alternative data models, IEEE Symposium on Security and Privacy, IEEE ComputerSociety, pp. 133-145, (1999).
[28] S. A. Hofmeyr, S. Forrest, A. Somayaji, Intrusion detectionusing sequences of system calls, Journal of ComputerSecurity, 6(3), (1998).
[29] S. Forrest, S. A. Hofmeyr, A. Somayaji, T. A. Longstaff, A sense of self for Unix processes, IEEE Symposium on Security and Privacy, Los Alamitos, CA, USA, IEEE Computer Society Press, pp. 120128, (1996).
[30] A. Somayaji, S. Forrest, Automated response usingsystem-call delays, 9th USENIX SecuritySymposium, August (2000).
[31] J. T. Giffin, S. Jha, B. Miller, Automated discovery of mimicry attacks, 9th International Symposium on Recent Advances in Intrusion Detection (RAID 06), (2006).
[32] H. G. Kayacik, M. Heywood, N. Zincir-Heywood, Onevolving buffer overflow attacks using genetic programming, the 8th annual conferenceon Genetic and evolutionary computation, pp. 1667-1674, (2006).
[33] C. Kruegel, E. Kirda, D. Mutz, W. Robertson, G. Vigna, Automating mimicry attacks using static binary analysis, 14th Annual Usenix Security Symposium, (2006).
[34] C. Kruegel, D. Mutz, F. Valeur, G. Vigna, On the detectionof anomalous system call arguments, ESORICS 2003, pp. 326-343, (2003).
[35] A. P. Kosoresow, S. A. Hofmeyr, Intrusion detection viasystem call traces, IEEE Software, 14(5), pp. 35-42, (1997).
[36] C. Marceau, Characterizing the behavior of a

program usingmultiple-length n-grams, the New Security Paradigms Workshop 2000, Association for Computing Machinery, (2000).
[37] R. Sekar, M. Bendre, P. Bollineni, D. Dhurjati, A fastautomaton-based method for detecting anomalous programbehaviors, the 2001 IEEE Symposium on Security and Privacy, (2001).
[38] D. Gao, M. K. Reiter, D. Song, Behavioral distancemeasurement using hidden markov models , In D. Zamboniand C. Kruegel, editors, Research Advances in Intrusion Detection, LNCS 4219, pp. 19-40, Berlin Heidelberg, Springer-Verlag, (2006).
[39] A. K. Ghosh, A. Schwartzbard, M. Schatz, Learning program behaviour profiles for intrusion detection, 1st USENIX Workshop on Intrusion Detection and Network Monitoring, (1999).
[40] N. Darvishzadeh, R. Azmi, "Intrusion Detection Based on System calls Analysis", 13th International Iranian Conference of Computer Society, Tehran, Iran, pp. 200-205, (2007), (Persian).
[41] A. K. Ghosh, A. Schwartzbard, A study in using neuralnetworks for anomaly and misuse detection, 8th USENIX Security Symposium, (1999).
[42] D. Endler, Intrusion detection:Applying machine learning toSolaris audit data, IEEE Annual Computer Security Applications Conference, pp. 268279. SocietyPress, (1998).
[43] H. Nemati, R. Azmi, A. R. Ghahremanian, M. T. Mir Mohammad Rezaei, "Intrusion Detection Using System call Auditing in Virtual Machine Monitor Space", 7th International Iranian ISC Conference On Information Security and Cryptology, pp. 187-193, (2010), (Persian).
[44] R. H. C. Sufatrio. Yap, Improving host-based ids with argument abstraction to prevent mimicry attacks, the International Symposium on Recent Advances in Intrusion Detection (RAID), pp. 146164, (2006).
[45] D. Gao, M. Reiter, D. Song, Gray-box extraction of executiongraphs for anomaly detection, 11th ACM Conference on Computer and Communications Security, pp. 318-329, (2004).
[46] Y. Liao, V. Rao Vemuri, Use of k-nearest neighbor classifier for intrusion detection, Computer & Security, 21(5), pp.439448, Oct (2002 ).
[47] N. Almassian, R. Azmi, Aidslk:an anomaly based intrusion detection system in linux kernel. Information Systems Technology and Management, pp. 232243, (2009).
[48] R. Azmi. B. Pishgoo, H. Nemati, "Hypervisor-based Intrusion Detection Using Artificial Immune Systems", 8th International Iranian ISC Conference on Information Security and Cryptology, pp. 147-153, (2011), (Persian).
[49] http://linux.die.net/man/8/auditd
[50] http://linux.die.net/man/7/audit.rules
[51] http://ipsecs.com/web/?p=277
[52] M. Tavallaee, E. Bagheri, W. Lu, A. Ghorbani, A Detailed Analysis of the KDD CUP 99 Data Set , Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), pp. 1-6, (2009).
[53] A. Frank, A. Asuncion, UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science. (2010).
[54] S. D. Bay, D. Kibler, M. J. Pazzani, P. Smyth, The UCI KDD archive of large data sets for data mining research and experimentation. ACM SIGKDD Explorations Newsletter, 2(2), pp. 81-85, (2000).
[55] D. M. Farid, J. Darmont, N. Harbi, H. H. Nguyen, M. Z. Rahman, Adaptive Network Intrusion Detection Learning: Attribute Selection and Classification, International Conference on Computer Systems Engineering (ICCSE09), (2009).
[56] F. Soltani Nejad, R. Salkhi, R. Azmi, B. Pishgoo, Structural TLR Algorithm for Anomaly Detection Based on Danger Theory, 9th International Iranian ISC conference on information security and cryptology, Tabriz, Iran, (2012).
[57] Z. Ji, D. Dasgupta, Augmented negative selection algorithm with variable-coveragedetectors, the IEEE Congress on Evolutionary Computation(CEC04), IEEE Press, pp. 10811088, (2004).
[58] Z. Ji, D. Dasgupta, Real-valued negative selection using variable-sized detectors, the Genetic and Evolutionary ComputationConference (GECCO04), Springer, Berlin/ Heidelberg, pp.287-298, (2004).
[59] J. Greensmith and U. Aickelin. The deterministic dendritic cell algorithm. In Proceedings of the 7th International Conference on Artificial Immune Systems (ICARIS 2007), pages 291302, (2008).
[60] J. Greensmith and U. Aickelin. Human-Centric Information Processing Through Granular Modelling, chapter Artificial Dendritic Cells: Multifaceted Perspectives, pages 375395. Springer, (2009).
[61] J. Greensmith, U. Aickelin, and J. Twycross. Articulation and clarification of the dendritic cell algorithm. In Proceedings of the 5th International Conference on Artificial Immune Systems (ICARIS 2006), pages 404417, (2006).
[62] T. Stibor, R. Oates, G. Kendall, and J. M. Garibaldi. Geometrical insights into the dendritic cell algorithms. In Proceedings of the 11th Annual conference on Genetic and evolutionary computation, (2009).

[63] P. Matzinger. Tolerance, danger, and the extended family. Annual Review of Immunology, 12:9911045, (1994).

[64] F. Gu, J. Greensmith, U. Aickelin, "Theoretical formulation and analysis of the deterministic dendritic cell algorithm", Biosystems, 111(2), pp. 127-135, (2013)

[65] Y. Al-Hammadi, U. Aickelin, J. Greensmith, "Performance Evaluation of DCA and SRC on a Single Bot Detection", Journal of Information Assurance and Security, 5 (1), pp. 265-275, (2010)

[66] J. Greensmith, J. Feyereisl, U. Aickelin, "The DCA: SOMe Comparison A comparative study between two biologically-inspired algorithms", Evolutionary Intelligence, 1 (2), pp. 85-112, (2008)

[67] F. Gu, J. Feyereisl, R. Oates, J. Reps, J. Greensmith, U. Aickelin, "Quiet in Class: Classification, Noise and the Dendritic Cell Algorithm", Proceedings of the 10th International Conference on Artificial Immune Systems (ICARIS 2011), LNCS Volume 6825, Cambridge, UK, pp. 173-186, (2011)

[68] F. Gu, J. Greensmith, U. Aickelin, "The Dendritic Cell Algorithm for Intrusion Detection", Bio-Inspired Communications and Networking, IGI Global, pp. 84-102, (2011).

[69] N. Afzali, R. Azmi, B. Pishgoo, Radius Regularization using Learning Automata for Spherical Intelligent Anomaly Detectors, 5th international conference on information security and cryptology (ISCTurkey), Ankara, Turkey, pp. 23-27, (2012).

[70] N. Afzali, R. Azmi, B. Pishgoo, "A new clonal selection algorithm based on Radius Regularization of Anomaly Detectors", 16th CSI international symposium on Artificial intelligence and signal processing (AISP2012), Shiraz, Iran, pp. 497-502, (2012).

**Reza Azmi** received his B.S. degree in Electrical Engineering from Amirkabir University of Technology, Tehran, Iran in 1990 and his M.S. and Ph.D. degrees in Electrical Engineering from Tarbiat Modares University, Tehran, Iran in 1993 and 1999 respectively. Since 2001, he has joined Alzahra University, Tehran, Iran. He was an expert member of Security Information Technology and Systems working groups in ITRC (From 2006 to 2008). He was Project Manager and technical member of many industrial projects. Dr. Azmi is founder and director of Operating System Security Lab (OSSL) in Alzahra University.

**Boshra Pishgoo** received her B.S. and M.S. degrees in Computer Engineering from Alzahra University, Tehran, Iran in 2009 and 2012 respectively. Since 2013, she is a Ph.D. student at Iran University of Science and Technology (IUST), Tehran, Iran. She has many research papers and projects in security and intrusion detection fields. Her research interests concern different aspects of operating system security, mainly intelligent host based intrusion detection systems. She is currently, head and expert member of Operating System Security Lab (OSSL) in Alzahra University.