A Sense of 'Danger' for Windows Processes*

Salman Manzoor, M. Zubair Shafiq, S. Momina Tabish, Muddassar Farooq

Next Generation Intelligent Networks Research Center (nexGIN RC)
FAST National University of Computer & Emerging Sciences (NUCES)
Islamabad, 44000, Pakistan
{salman.manzoor,zubair.shafiq,momina.tabish,muddassar.farooq}@nexginrc.org

Abstract. The sophistication of modern computer malware demands run-time malware detection strategies which are not only efficient but also robust to obfuscation and evasion attempts. In this paper, we investigate the suitability of recently proposed Dendritic Cell Algorithms (DCA), both classical DCA (cDCA) and deterministic DCA (dDCA), for malware detection at run-time. We have collected API call traces of real malware and benign processes running on Windows operating system. We evaluate the accuracy of cDCA and dDCA for classifying between malware and benign processes using API call sequences. Moreover, we also study the effects of *antigen multiplier* and *time-windows* on the detection accuracy of both algorithms.

Key words: API Call Sequence, Artificial Immune System, Dendritic Cell Algorithm, Malware Detection

1 Introduction

The sophisticated computer malware is becoming a serious threat to the information technology infrastructure, which is the backbone of modern e-commerce systems [2]. A recent outbreak of `Conficker` malware affected more than 9 million computers including those of Ministry of Defence, United Kingdom [3]. This incident has proved that commercial anti-virus software, even with updated malware definitions, are incapable of safeguarding our information technology infrastructure. In [7], the authors have shown that commercial anti-virus software are easily befooled using evasion attempts, such as code obfuscation, encryption and polymorphic transformations. Therefore, security experts are now focusing their attention to robust *run-time* malware detection techniques that analyze API call sequence of a process to classify it as *benign* or *malicious*. Intuitively, such *dynamic* techniques are resilient to the above-mentioned evasion attempts because malware has to eventually execute the malicious activity.

Artificial Immune Systems (AIS) have served as a natural source of inspiration to develop dynamic systems for process classification. The field of AIS was initially dominated by the *self/nonself* theory, which models the working

* Apologies to Forrest et al. [9].

of *adaptive* immune system. Forrest et al. initially used the idea of *self/nonself* to develop the *negative selection algorithm* (NSA) [9]. Initially, NSA was used to classify a computer process as benign or malicious. NSA has been incrementally improved and several advanced versions are now available, such as the real-valued NSA [10], the *randomized* real-valued NSA [11], and the real-valued NSA with variable sized detectors [17]. However, Stibor et al. carried out several experiments to evaluate the appropriateness of NSA for anomaly detection. The authors showed that negative selection algorithm is not suitable for higher dimensional datasets [20], [21].

AIS research community has recently turned its attention to a new generation of immune-inspired AIS algorithms which mimic the working model of the *innate immune system* [4]. The fundamental principle of such algorithms is that the innate immune system responds to 'danger' instead of 'nonself'. Aickelin et al. proposed a new AIS algorithm called *Dendritic Cell Algorithm* (DCA) to overcome the above-mentioned shortcomings in NSA [6], [12].

The classical DCA (cDCA) consists of a number of context specific stochastic variables which makes it difficult to systematically analyze a given task. Consequently, Greensmith et al. [15] proposed a simplified and more predictable version of DCA which is called deterministic DCA (dDCA). Since its original inception, two major improvements are proposed for DCA namely *antigen multiplier* and *time-windows*. Gu et al. have initially investigated the usefulness of these concepts for DCA [16].

In this study, we investigate the relative merits/de-merits of cDCA and dDCA, coupled with *antigen multiplier* and *time-windows* concepts, for malware detection. In order to ensure real-world relevance, we have collected API call traces by running 100 benign and 416 malicious Windows executables in a virtual environment.[1] The malicious executables include trojans, viruses and worms. We quantify the efficacy of DCA in terms of its detection accuracy.

The remaining paper is organized as follows: Section 2 presents a summary of related work. Section 3 provides an overview of the DCA and its variations. In Section 4, we explain the collection process of API call traces for real-world malware and benign processes. Section 5 describes our experimental setup and presents the detailed discussions on empirical results. In Section 6, we briefly discuss major limitations of DCA and their potential countermeasures. Finally, we conclude our paper in Section 7.

2 Related Work

AISs have served as a natural source of inspiration for designing anomaly detection systems. To maintain focus, we only discuss the most relevant research.

Classical AIS algorithms are inspired by the working of adaptive immune system which follows principles of the self/nonself theory. In this paradigm, NSA has attained the status of a defacto standard. It was proposed by Forrest et al. for

[1] The datasets used in this paper are available at `http://www.nexginrc.org`.

classification of anomalous processes in a computer system [9]. Several advanced versions of NSA have been proposed to date which include but are not limited to the real-valued NSA [10], the *randomized* real-valued NSA [11], and the real-valued NSA with variable sized detectors [17]. The advanced versions of NSA improve its scalability, space coverage, convergence time and formal treatment. Even with the above-mentioned improvements, NSA has been widely criticized for poor scalability behavior especially at higher dimensions [20], [21].

Danger theory proposed by Matzinger [19] claims that immune system works by sensing 'danger'. In [6], the authors investigated the feasibility of using danger theory to develop a new paradigm of AIS algorithms for network intrusion detection. In [12], Greensmith et al. developed a novel DCA based on the concepts of *danger theory*. The authors successfully applied cDCA for classification of breast cancer dataset. In [13] and [14], the authors used cDCA for SYN scan detection.

Since the seminal work of Greensmith et al., several variations of DCA have been proposed. In [16], the authors enhanced DCA with two additional features, called *antigen multiplier* and *time-windows*. DCA relies on aggregate sampling of the antigens for eventual classification; therefore, *antigen multiplier* was used to improve sampling process. Each antigen was multiplied 10, 50 and 100 times to study the effect of multiple sampling. The authors also used *time-windows* to study the aggregate effect of signals. They used fixed time-windows of 2, 3, 5, 7 and 10 instances. They also compared DCA with NSA and C4.5 decision tree for benchmark comparison.

In [15], the authors proposed the dDCA. Several stochastic variables of cDCA were removed to understand the merits/demerits of the core algorithm. Three relevant modifications introduced in dDCA were: (1) a simple signal processing procedure, (2) context evaluation based on one factor, ($\bar{k}$), which was used to ultimately calculate an anomaly score K_α, and (3) a new metric (T_k) was defined to determine threshold for K_α. The authors evaluated the detection accuracy of dDCA using the PING scan dataset. In the next section, we provide a detailed introduction of DCA and its variations.

3 Dendritic Cell Algorithm and its variations

Dendritic cells (DCs), of the innate immune system, are the core component of DCA. They have the ability to sense the internal conditions of a tissue by detecting various signals. A *safe* signal is produced in an event of natural cell death (apoptosis), which reflects the normal environment of a tissue. On the contrary, unnatural death of cells (necrosis) because of injury or pathogenic infection leads to the release of *danger* signals. Another strong indicator of potentially harmful environment is *pathogen associated molecular pattern* (PAMP).

Newly born DCs are in an immature state and scour a tissue for antigens (suspect) and signals (evidence). Antigens and signals together evaluate the context of a tissue as *benign* or *potentially malicious*. DCs distinguish between contexts by taking different pathways to their maturity. A *fully matured* state of a DC

is the result of exposure to higher concentration of danger and PAMP signals. Likewise, *semi-matured* state of a DC depicts exposure to higher concentration of safe signals. A collective assessment of DC population activates or suppresses the immune response. We now explain the details of different variations of DCA.

3.1 Classical DCA (cDCA)

In cDCA, proposed by Greensmith et al. [12], a population of 100 DCs is maintained. Each DC is assigned a random migration threshold which limits the amount of time it spends in a tissue. A subset of population is randomly sampled to form a sampling pool of antigens. The selected DCs spend time in a tissue to collect antigens and signals. The input signals are multiplied with predefined weights to calculate output signals. In this paper, for cDCA, we have used same weight values as proposed by the authors in [12]. Three output signals (O_0, O_1, O_2) are calculated for each DC as: $O_i = \sum_{j=0}^{j=2} W_{i\,j}\, S_j, \forall\, i$, where i refers to the category of output signal, j refers to the category of input signal, W is the weight matrix, S is the input signal vector and O is the output signal vector. O_0 is costimulatory signal (csm) and it migrates to the lymph node if the value of csm exceeds assigned migration threshold. In order to derive a context, DC computes two more outputs: (1) the semi-mature context (O_1), and (2) the mature context (O_2). The values are compared with one another and the overall context is termed as safe if O_1 is greater than O_2, and vice-versa.

DCs that have lived their allotted span migrate to the lymph node. The antigens and their corresponding contexts are saved to a log file. Each antigen is sampled multiple times so that it can appear in different contexts in a log file. In order to detect potentially malicious antigens, they are tagged with a mature context antigen value ($MCAV$). $MCAV$ for a particular antigen i, ($MCAV_i$), is derived by dividing the number of times that antigen (Ag_i) has appeared in the danger context (N_{di}) by total number of appearances (N_i). Mathematically, $MCAV_i = \frac{N_{di}}{N_i}$.

A threshold (T) is applied to $MCAV$ to make the final classification decision. The antigens with $MCAV$ higher than T are termed *malicious*, and vice-versa. Let ζ_m be the number of anomalous instances and ζ be the total number of instances in a dataset. We can define $T = \frac{\zeta_m}{\zeta}$.

3.2 Deterministic Dendritic Cell Algorithm (dDCA)

The DCA has provided promising classification accuracy results on a number of benchmark datasets [12], [13]. However, the basic DCA uses several stochastic variables which make its systematic analysis very difficult. In order to mitigate this problem, the authors in [15] have proposed some changes in cDCA. The new variation of DCA, called dDCA, has following enhanced features:

- Three input signal categories are reduced to two, i.e. danger and safe signal;
- Random migration threshold is replaced with uniform distribution of lifespan values in a population;

- Dedicated storage and sampling of antigens is replaced with sampling of all antigens by DCs;
- Instead of forming a sampling pool, the signals' data is processed by all DCs. As a result, output signals are calculated once for population of DCs;
- Only one factor ($\bar{k}$) is calculated for each DC to arrive at a context. Negative values of $\bar{k}$ reflect a benign context and positive values indicate a malicious context.

Signal processing is simplified by reducing the number of input signals and using a weight assigning scheme. Two outputs are calculated: (1) accumulation of signals (csm), and (2) score ($\bar{k}$), to which the threshold is applied for classification. csm is defined as $csm = D - S$, and $\bar{k} = D - 2S$, where D and S are values of danger and safe signals respectively. A new parameter K_α is defined using the values of $\bar{k}$. Its purpose is to provide real-valued scores. K_α is defined as $K_\alpha = \frac{\sum_m k_m}{\sum_m \alpha_m}$, where k_m is the $\bar{k}$ value for DC_m, and α_m is the number of antigens of type α presented by DC_m. Moreover, a threshold parameter (T_k) is also defined. The values of K_α greater than the value of T_k depict malicious context and smaller values indicate benign behavior. T_k is defined as $T_k = \frac{S_k \cdot \bar{i}}{I_s}$, where I_s is the total number of instances in a dataset, $\bar{i}$ is the mean number of iterations per incarnation of a DC, and $S_k = \sum_{I_s} D - 2\sum_{I_s} S$.

3.3 Antigen Multiplier

DCA has been mostly utilized for data mining problems. Most of the datasets used for data mining contain only one copy of each instance (or antigen). In order to assess the type of an antigen, it should be presented multiple times so that $MCAV$ value can be generated for it. The concept of antigen multiplier caters for this requirement [16]. Each antigen is copied multiple times in the tissue antigen vector. The classification decision is now averaged over the replicated population. Intuitively, replicating an antigen should help in improving the classification accuracy.

3.4 Moving Time-Windows

The signals in our body do not die suddenly; rather, they fade slowly over a period of time. This temporal effect of signals is captured by introducing the concept of moving time-windows in DCA [16]. New signals are computed using: $N_{i\,j} = \frac{1}{w}\sum_{n=i}^{i+w} O_{n\,j}, \forall\, j$, where $N_{i\,j}$ is new signal value of i^{th} antigen of j^{th} category, w is the window size and $O_{i\,j}$ is original signal of i^{th} antigen and j^{th} category. New signals (N) are the average of old signals (O) in a particular time-window. Intuitively speaking, averaging of signals reduces the noise in input signals.

4 Dataset

In this section, we provide statistics of the benign and malware executable files used in our study. We also describe the commercial API call tracer, API Monitor 1.5, used for logging API traces.

Table 1. Statistics of the Data used in this Study

Executable Type	**Quantity**	**Avg. Filesize** (Kilo Bytes)	**Min. Filesize** (Kilo Bytes)	**Max. Filesize** (Kilo Bytes)
Benign	100	1,263	4	104,588
Trojan	117	270	1	9,277
Virus	165	234	4	5,832
Worm	134	176	3	1,301
Total	516	50	2	1,332

4.1 Benign and Malware executables

We have collected a set of 416 malware and 100 benign Windows executables. The malware executables consist of trojans, viruses and worms. All of them are in `Win32` portable executable (PE) format. The benign executables are obtained from a freshly installed copy of Windows XP and application installers. The malware executables are obtained from VX Heavens virus collection which is publicly available [5]. Table 1 provides statistics of the executables used in our study.

4.2 API Call Tracer

We have used API Monitor 1.5 to log the API call sequences of Windows processes. It captures these API calls and stores them in `apm` file format [1]. It has an *API* and a *process* filter. The API filter gives us the option of filtering unnecessary API calls by category. In the API filter, we can select the calls of following categories: (1) Dynamic-Link Libraries, (2) Memory Management, (3) Network Management, (4) Processes and Threads, (5) Registry, and (6) Socket. The process filter allows us to filter API calls made by different processes. We have captured API calls of all system-wide processes because some malware in our study use Windows processes like `explorer.exe` to carry out malicious activities. Therefore, it is not possible for us to exactly pin-point a set of processes for monitoring.

We install API call tracer on a fresh virtual machine of Microsoft Windows XP and also create its backup. After execution of each malware or benign executable, we replace the virtual machine source with the original backup. We capture API calls from the start of execution of a process till it finishes.

4.3 Feature Selection and Extraction

We use $n-$gram analysis for feature extraction. $n-$gram of a sequence is the normalized frequency histogram of n successive elements of the sequence [8]. $n-$grams computed with very less value of n contain insufficient information and those with very large values of n incur unacceptably high processing overheads. So, we have to choose a suitable value of n to get sufficient information from the $n-$grams while incurring reasonable processing overheads. In this study, we have used the value of $n = 4$. Each API function is mapped to a unique random variable. We extract the most informative $4-$grams from all dataset files by ranking them according to their *information gain*, also known as *average mutual information*. The information gain of a feature i is defined as [18]:

$$I(Y; X) = H(Y) - H(Y|X),$$

where X is an input attribute, Y is a class attribute, $H(Y)$ is the entropy of the class attribute variable Y and $H(Y|X)$ is the conditional entropy of Y with respect to X. Therefore, information gain of an input attribute quantifies the reduction in uncertainty of the class attribute given that we know the value of input attribute. We have selected top 500 $4-$grams sequences by applying threshold to information gain values. 500 $4-$grams are selected to ensure that adequate amount of relevant information is selected for the signal calculation process, which is explained in the next section.

For feature extraction, we check the log file of each executable file for presence or absence of the selected $n-$grams. We place 1 if the $n-$gram is present and 0 otherwise. Each executable log is mapped to a $500-$dimensional binary string. All strings generated for benign executables are placed in a separate file and the same is done for trojans, viruses and worms. We then combine the separate files to create three datasets: benign-trojan, benign-virus and benign-worm. In the next section, we explain the detection accuracy of different variations of DCA using each of the above-mentioned datasets.

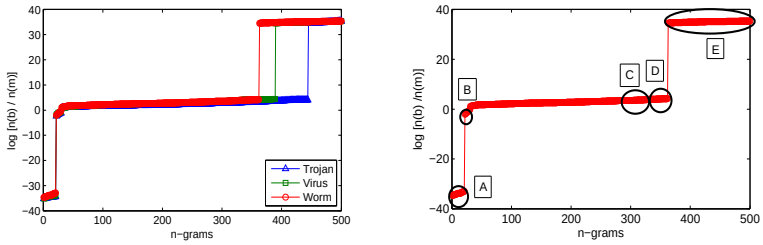
5 Experimental Analysis

5.1 Signals and Antigens

We have used boolean information about presence or absence of top 500 $n-$grams, which characterize the activities of each executable.

To map information from boolean feature vectors to signals in a systematic manner, we propose an intuitive procedure. Firstly, we count the number of times a given feature ($n-$gram sequence) is present in benign and malicious logs. We term these counts as $n(b)$ for benign logs and $n(m)$ for malware logs. We then compute the *kappa metric* (κ) for every $n-$gram as:

$$\kappa = \log_e \left(\frac{n(b)}{n(m)} \right)$$



(a) κ distribution plot of n-grams

(b) Regions of interest for benign-worm dataset

Fig. 1. κ distribution plot of n-grams for all datasets

Figure 1(a) shows the plot of κ for every $n-$gram in all datasets. An interested reader will appreciate the peculiar nature of the plot, which can be helpful in selecting $n-$grams for signal calculation. Figure 1(b) shows the plot of only benign-worm dataset where potentially interesting $n-$gram regions are marked with black circles. Intuitively, $n-$grams with positive values of κ represent benign behavior and vice-versa. We now detail the formation of all types of signals: (1) safe, (2) PAMP, and (3) danger.

For safe signal, we are interested in $n-$grams which are mostly present in the traces of benign processes (region D), and those which are *never* present in benign traces (region A). For PAMP signal, we are interested in $n-$grams which are mostly present in malware traces (region B), and those which are *never* present in the traces of malware processes (region E). To derive the magnitude of safe and PAMP signals, we add a predefined value to the signal value if we are: (1) able to find $n-$gram that is supposed to be present, and (2) unable to find $n-$gram that is supposed to be absent. Finally, we normalize the value signal in a desired range.

To extract danger signal, we are interested in $n-$grams belonging to region C. These $n-$grams have high probability of being present in benign traces, but not as high as required for deriving safe signal. We use the absence of these $n-$grams as the evidence of danger. Remember that large deviations from benign behavior reflect associated danger. In order to create antigens, we label each instance of all datasets with an integer value.

5.2 Experimental Setup

We now describe the experimental setup used in our study. We plan to examine the effect of antigen multiplier and moving time-windows on the performance of cDCA and dDCA in terms of detection accuracy. We perform independent experiments using all three datasets—benign-trojan, benign-virus, benign-worm.

We have designed three sets of experiments for each algorithm. First two experiments determine the effect of varying the amount of antigen multiplier value and sizes of moving time-windows independently on the detection accuracy. The objective of first two experiments is to determine the best antigen multiplier value and the best size of time-windows. In the third experiment, we analyze the combined effect of both techniques on the detection accuracy of both algorithms.

- **E.0**: Effect of antigen multiplier value on the detection accuracy of cDCA and dDCA.
- **E.1**: Effect of size of moving time-windows on the detection accuracy of cDCA and dDCA.
- **E.2**: Combined effect of antigen multiplier value and size of moving time-windows on the detection accuracy of cDCA and dDCA.

In our experimentation, the number of DCs are kept constant at 100 for both algorithms. All datasets are ordered, i.e., benign class followed by malicious class. The experiments are conducted on a 2.2 GHz Dell Vostro 1510 Core 2 Duo.

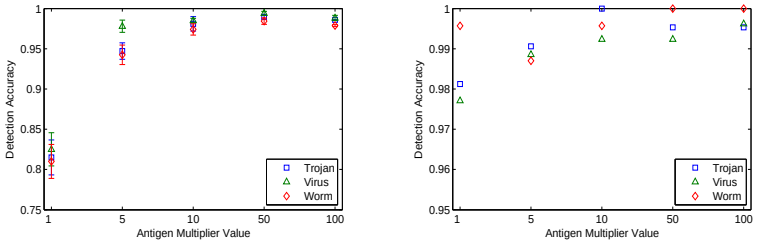
Parameters of cDCA. The thresholds values used in this study are derived from distributions of the datasets. We divide the number of anomalous instances with the total number of instances in a dataset to compute threshold. As a result, we have determined threshold values of 0.547, 0.630 and 0.580 for benign-trojan, benign-virus and benign-worm datasets respectively.

Danger and PAMP signals are normalized within a range of 0 to 100 while safe signal is normalized within a range of 0 to 66. The migration threshold is kept between 300 to 500. The use of high values ensures that the classification of current instance is affected by its neighboring instances. Each experiment is repeated 10 times and the averages, along with standard deviations, of these runs are plotted in Figures 2, 3 and 4.

Parameters of dDCA. For dDCA, we use same values of PAMP and safe signals as that of cDCA. Lifespans are uniformly distributed between a range of 300 to 500 across the DC population. The increments are computed by dividing the range of lifespan with the number of used DCs. As before, the higher values include the effect of neighboring instances on the classification of current instance, which tends to reduce the error because of noise during signal calculation.

5.3 Discussions on Results

E.0: Effect of antigen multiplier on cDCA and dDCA. Remember that by using multiplier, each antigen is copied several times into the tissue antigen vector. More than one presentations of the same antigen allows multiple DCs to evaluate its context. Intuitively speaking, the collective assessment by DC population should provide more accurate prediction since final context is less vulnerable to the wrong judgment of a single DC.



(a) Accuracy plot for basic DCA (b) Accuracy plot for dDCA

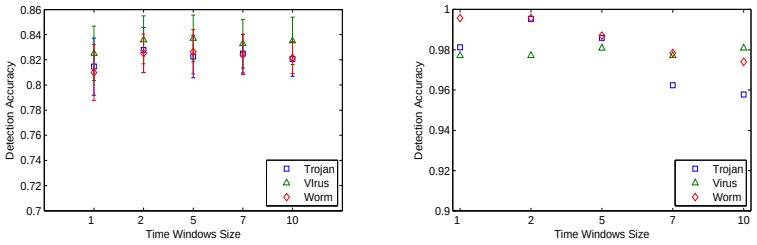
Fig. 2. Accuracy results for varying Antigem Multiplier

The need for antigen multiplier is justified where insufficient antigens are available for doing classification. This is a common situation in data mining applications. For example, KDD 99 dataset used by Gu et al. in [16] and breast cancer dataset used by Greensmith et al. in [12] have only one copy of each antigen. Therefore, antigen multiplier can play an important role in improving the classification accuracy in such applications. We have used antigen multiplier values of 1, 5, 10, 50 and 100. Note that the multiplication factor of 1 refers to the case when antigen multiplication is not applied.

The results of cDCA are shown in Figure 2(a). DCA achieves poor classification accuracy when antigen multiplication is 1 because there is just one antigen of each type which may get picked by a DC having either a very high or a very low migration threshold. In the former case, it may get associated with the instances of a wrong class, and in the later case the neighboring instances would hardly have any effect because the DC would migrate to a lymph node in a single iteration. As expected, the detection accuracy of the algorithm significantly increases with an increase in the antigen multiplication factor.

The results of varying antigen multiplier in dDCA are shown in Figure 2(b). The classification accuracy trend is similar to the one observed in cDCA. However, the classification accuracy of dDCA, even without antigen multiplication, is significantly better compared with the cDCA. This improvement can be attributed to the use of T_k, which caters for different ranges of normalization of signals and their effect on the migration threshold. As a result, the algorithm becomes more adaptive to variations of parameters. Finally, we observe that the antigen multiplication factor of 100 for *dDCA* achieves better classification accuracy than all antigen multiplier configurations of cDCA.

E.1: Effect of Moving Time-Windows on cDCA and dDCA. When we use the concept of moving time-windows, we actually take average of the magnitude of signals present in that window. This effectively reduces the signal's noise by including the effect of neighboring signals. The signals are also averaged



(a) Accuracy plot for basic DCA (b) Accuracy plot for dDCA

Fig. 3. Accuracy results for varying size of time-windows

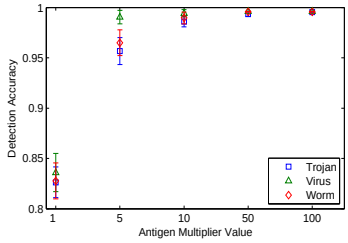
near class boundaries. This helps in reducing the possibility of false predictions. We have used time-windows of sizes 1, 2, 5, 7 and 10 in our study.

The results of moving time-windows for cDCA are plotted in Figure 3(a). The poor accuracy of cDCA at a time-window of size 1 stems in the same reasons already explained for antigen multiplication factor of 1. However, it is interesting to note that increasing the size of time-windows has almost negligible effect on accuracy of the algorithm. This behavior is due to the fact that noise in the obtained signals is already fairly low, which leaves little room for removing the noise or making other relevant improvements.

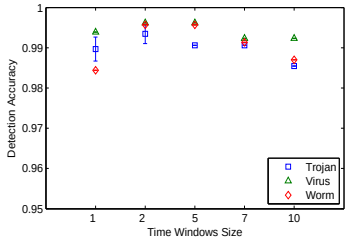
The results of moving time-windows for dDCA are shown in Figure 3(b). It is interesting to observe that the accuracy drops with an increase in the size of time-window. We contemplate on a possible explanation that unrelated signals and antigens get associated with each other for larger time-windows. This effect is more evident at the boundary of two classes. If we take the example of the last entry of a benign class, most of the signals in the time-window would belong to the malware class. Now if we take average, the signal value is biased towards the malware class. dDCA still achieves better classification accuracy for all respective time-windows sizes than cDCA.

E.2: Combined effect of both antigen multiplier and moving time-windows on cDCA and dDCA. From the results of previous two experiments, it can be deduced that antigen multiplier values of 50 and 100 provide the best classification accuracy results for cDCA and dDCA respectively. In case of time-windows, the window size of 2 has yielded best average accuracy results for both algorithms.

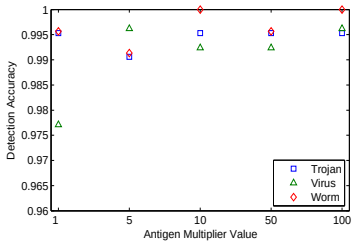
Figures 4(a) and 4(c) respectively show the results of varying antigen multiplier while keeping the window size fixed at 2 and varying the window size while keeping the multiplier value fixed at 50 for cDCA. Similarly, Figures 4(b) and 4(d) show the results of varying antigen multiplier while keeping the window size fixed at 2 and varying the window size while keeping the multiplier value at



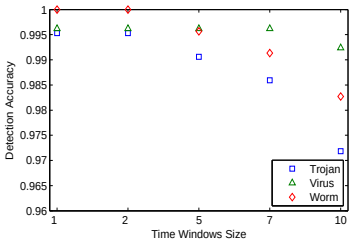
(a) Accuracy plot for varying Antigen Multiplier with Time-Windows fixed at 2 for basic DCA



(b) Accuracy plot for varying Time-Windows with Antigen Multiplier fixed at 50 for basic DCA



(c) Accuracy plot for varying Antigen Multiplier with Time-Windows fixed at 2 for dDCA



(d) Accuracy plot for varying Time-Windows with Antigen Multiplier fixed at 100 for dDCA

Fig. 4. Accuracy results for cascading antigen multiplier and time-windows

100 respectively for dDCA. It is evident that the detection accuracies are better compared with the results of **E.0** and **E.1** experiments of cDCA.

The results of dDCA in Figure 4(c) show an increase in the classification accuracy reflecting an increase in antigen multiplier value. Likewise, Figure 4(d) shows decrease in the classification accuracy, similar to the trends shown in Figure 3(b), due to the increase in the size of time-windows. The results of dDCA, in Figure 4(c) with antigen multiplier value set to 100 and in Figure 4(d) with the value of time-windows set to 1 and 2, show better classification accuracy than all other settings for dDCA.

From our results, we can conclude that the use of antigen multiplication is highly recommended. While, the effect of time-windows on results of cDCA is minimal and the accuracy drops as the size of time-window increases for dDCA.

6 Limitations and Potential Solutions

In this section, we briefly present the limitations of DCA which we have observed for the presented problem. We also discuss potential solutions of these limitations.

A basic problem with DCA is that it requires adjustment of several parameters and there are no rules of thumb to determine their optimal values. This limitation introduces additional design dimensions which are to be explored for obtaining the best results. For example, there is no standard procedure to determine the weights used in DCA. Intuitively, the optimal values of these weights vary with respect to different properties of the dataset.

DCA also lacks an automated module for signal computation from high-dimensional data. Signal computation strategies vary significantly from one application to another. In this paper, we have introduced a systematic and intuitive method to group and transform high-dimensional input data for typical $2-$class problems.

DCA decisions are based on the aggregate sampling of each antigen. This demands multiple presentations of every antigen. Hence, DCA does not work well in situations where insufficient number of antigens are available (such as data-mining). Our study has shown that antigen multiplication is an important concept which can be used to overcome this limitation.

DCA performs a temporal correlation between antigens and the signals for classification. It distinguishes between normal and potentially malicious antigens on the basis of neighboring antigens. This feature can be exploited by crafty attackers (via mimicry attacks) to evade detection by DCA. The malicious entities may remain undetected by wilfully mimicking benign behavior intermittently. This vulnerability is also observable at the class boundaries.

7 Conclusions

In this study, we have analyzed the feasibility of using two variations of DCA—cDCA and dDCA—for run-time detection of malware. We have also investigated the effect of antigen multiplier and moving time-windows on the accuracy of both algorithms. The results of our experiments highlight the promise of DCA in malware detection applications and relevant $2-$class problems.

The important conclusions of our experiments are: (1) danger theory based DCA has the potential in the domain of run-time malware detection, (2) dDCA consistently outperforms DCA in terms of classification accuracy, and (3) antigen multiplier shows promise to improve the detection accuracy while time-windows show little relevance in improving the detection accuracy.

References

1. API Monitor, available at `http://www.rohitab.com/apimonitor`.

2. F-Secure Corporation, "F-Secure Reports Amount of Malware Grew by 100% during 2007", Press release, 2007.
3. Symantec, "Internet Security Threat Report", Vol. XIV, 2009.
4. The Danger Project, `http://www.dangertheory.com`.
5. VX Heavens Virus Collection, VX Heavens website, `http://vx.netlux.org`.
6. U. Aickelin, P. Bentley, S. Cayzer, J. Kim, J. McLeod, "Danger Theory: The Link between AIS and IDS?", International Conference on Artificial Immune Systems (ICARIS), pp. 147-155, UK, 2003.
7. M. Christodorescu, S. Jha, "Testing Malware Detectors", ACM SIGSOFT Software Engineering Notes, 29(4), pp. 34-44, ACM Press, 2004.
8. M. Damashek, "Gauging Similarity with n-Grams: Language-Independent Categorization of Text", Vol. 267, pp. 843-848, Science, 1995.
9. S. Forrest, S.A. Hofmeyr, A. Somayaji, T.A. Longstaff, "A sense of self for Unix processes", IEEE Symposium on Security and Privacy, pp. 120-128, USA, IEEE Press, 1996.
10. F. Gonzalez, D. Dasgupta, "Anomaly Detection Using Real-Valued Negative Selection", Journal of Genetic Programming and Evolvable Machines, 4(4), pp. 383-403, 2003.
11. F. Gonzalez, D. Dasgupta, L.F. Nino, "A Randomized Real-Valued Negative Selection Algorithm", International Conference on Artificial Immune Systems (ICARIS), pp. 261-272, UK, 2003.
12. J. Greensmith, U. Aickelin, S. Cayzer, "Introducing Dendritic Cells as a Novel Immune-Inspired Algorithm for Anomaly Detection", International Conference on Artificial Immune Systems (ICARIS), pp. 153-167, Springer, Canada, 2005.
13. J. Greensmith, U. Aickelin, J. Twycross, "Articulation and clarification of the dendritic cell algorithm", International Conference on Artificial Immune Systems (ICARIS), pp. 404-417, Springer, Portugal, 2006.
14. J. Greensmith, U. Aickelin, "Dendritic Cells for SYN Scan Detection", Genetic and Evolutionary Computation Conference (GECCO), pp. 49-56, ACM Press, UK, 2007.
15. J. Greensmith, U. Aickelin, "The Deterministic Dendritic Cell Algorithm", International Conference on Artificial Immune Systems (ICARIS), pp. 291-303, Springer, Thailand, 2008.
16. F. Gu, J. Greensmith, U. Aickelin, "Further Exploration of the Dendritic Cell Algorithm: Antigen Multiplier and Time Windows", International Conference on Artificial Immune Systems (ICARIS), pp. 142-153, Springer, Thailand, 2008.
17. Z. Ji, D. Dasgupta, "Real-Valued Negative Selection Using Variable-Sized Detectors", Genetic and Evolutionary Computation Conference (GECCO), pp. 287-298, Springer, USA, 2004.
18. J.Z. Kolter, M.A. Maloof, "Learning to detect malicious executables in the wild", International Conference on Knowledge Discovery and Data Mining, pp. 470-478, ACM Press, USA, 2004.
19. P. Matzinger, "Tolerance, danger and the extended family", Annual Review of lmmunology, Vol. 12, pp. 991-1045, 1994.
20. T. Stibor, J. Timmis, C. Eckert, "On the Appropriateness of Negative Selection defined over Hamming Shape Space As a Network Intrustion Detection System", IEEE Congress on Evolutionary Computation (CEC), pp. 995-1002, IEEE Press, UK, 2005.
21. T. Stibor, P. Mohr, J. Timmis, C. Eckert, "Is Negative Selection Appropriate for Anomaly Detection?", Genetic and Evolutionary Computation Conference (GECCO), pp. 321-328, USA, ACM Press, 2005.