Hindawi
Security and Communication Networks
Volume 2021, Article ID 3578695, 12 pages
https://doi.org/10.1155/2021/3578695

Research Article

Detecting Portable Executable Malware by Binary Code Using an Artificial Evolutionary Fuzzy LSTM Immune System

Jian Jiang and **Fen Zhang**

College of Computer and Electrical Engineering, Hunan Arts and Science University, Changde 415000, China

Correspondence should be addressed to Jian Jiang; jianjiang211@yahoo.com

Received 24 May 2021; Revised 17 June 2021; Accepted 22 June 2021; Published 8 July 2021

Academic Editor: Konstantinos Demertzis

As the planet watches in shock the evolution of the COVID-19 pandemic, new forms of sophisticated, versatile, and extremely difficult-to-detect malware expose society and especially the global economy. Machine learning techniques are posing an increasingly important role in the field of malware identification and analysis. However, due to the complexity of the problem, the training of intelligent systems proves to be insufficient in recognizing advanced cyberthreats. The biggest challenge in information systems security using machine learning methods is to understand the polymorphism and metamorphism mechanisms used by malware developers and how to effectively address them. This work presents an innovative Artificial Evolutionary Fuzzy LSTM Immune System which, by using a heuristic machine learning method that combines evolutionary intelligence, Long-Short-Term Memory (LSTM), and fuzzy knowledge, proves to be able to adequately protect modern information system from Portable Executable Malware. The main innovation in the technical implementation of the proposed approach is the fact that the machine learning system can only be trained from raw bytes of an executable file to determine if the file is malicious. The performance of the proposed system was tested on a sophisticated dataset of high complexity, which emerged after extensive research on PE malware that offered us a realistic representation of their operating states. The high accuracy of the developed model significantly supports the validity of the proposed method. The final evaluation was carried out with in-depth comparisons to corresponding machine learning algorithms and it has revealed the superiority of the proposed immune system.

1. Introduction

Critical sectors, such as transport, energy, health, education, and the financial sector, are increasingly dependent on digital technologies for their core business functionalities [1]. Although digitalization offers enormous opportunities and solutions to many of the challenges of modern society, it significantly exposes the economy and society to widespread cyberthreats, most of which are implemented with specialized forms of malware [2].

Malware development is quite organized with constant innovation, and sophisticated techniques are constantly being developed to bypass even the most advanced digital security systems. Due to the great popularity of the Windows operating system, Portable Executable (PE) files have been at the center of the efforts of organized cybercrime groups for several years now [3]. PEs are executable file formats or object code such as .exe, .dll, .sys, .ocx, and .drv, used in 32/64-bit versions of the Windows operating system. Their format is essentially a data structure that encapsulates all the information required by the Windows loader to manage and execute the executable code contained in each file.

The PE archetype consists of a set of headers and segments of the dynamic linker on assigning the file to memory. An executable string consists of several regions, each of which has different memory protection requirements [4]. Figure 1 shows the basic structure of PE programs.

Since the PE format was not designed to be resistant to modification, it is relatively easy to tamper them for malicious or improper use. Malware developers usually use sophisticated polymorphism and metamorphism techniques to obscure their malicious intentions. The main difference

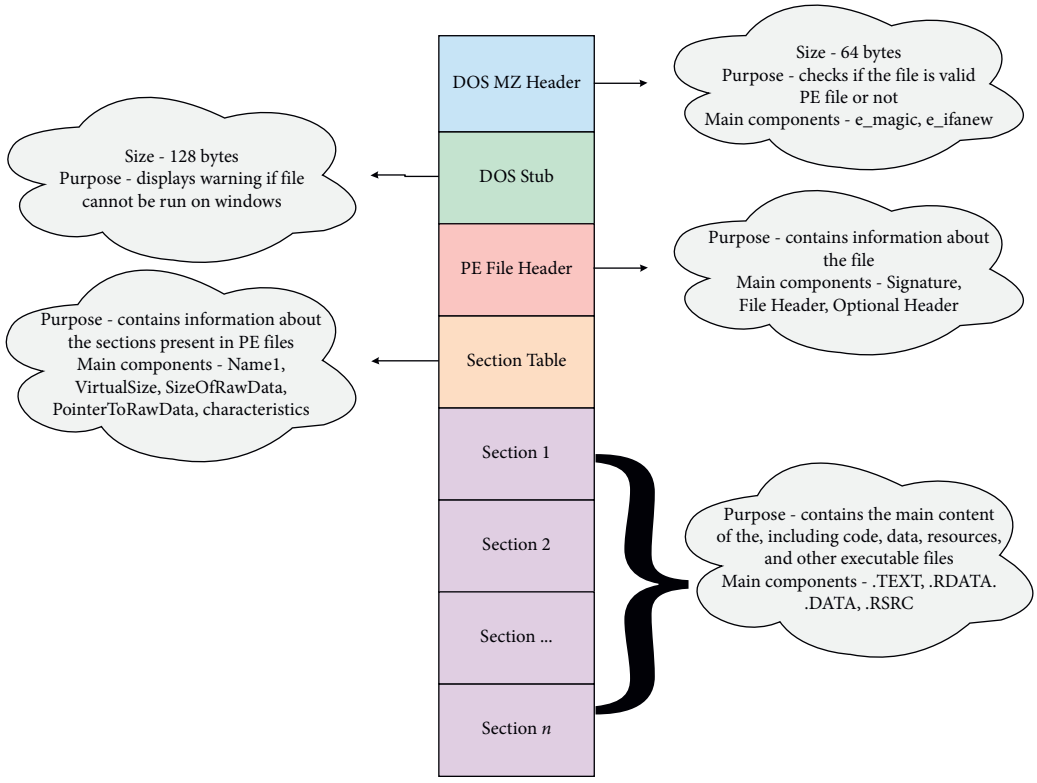

Figure 1: Basic structure of PE file (https://malware.news/).

between polymorphic and metamorphic viruses is that the polymorphic virus is encrypted using a variable encryption key so that each copy of the virus looks different, while the metamorphic virus rewrites its code to make each copy different, without the use of an encryption key [5]. Packing or obfuscation techniques are also widely used to greatly complicate the analysis of infected PE files with polymorphic or metamorphic viruses.

To investigate possible infected PE files, either static analysis, i.e., examination of the file without being executed, or dynamic analysis, i.e., execution of the file to extract information and reveal its behavior, is performed. After analyzing an executable PE and extracting appropriate attributes, special techniques must be applied to detect the intent of the file, so that it can be properly categorized. The various methods for the above detection are through either a signature-based process for comparing and detecting distinct patterns in an updated database of known malware or detection based on a behavior-based process, thus calculating behavioral parameters including elements such as sender addresses and recipient, attachment types, and various other measurable statistical features [6].

Signature-based processes are considered obsolete and only used as an auxiliary method while achieving efficient detection of malicious PE files is equal to the process of analyzing a huge amount of data to identify the behavioral patterns of each malware family, to group them in separate similar categories. This categorization with clearly defined and sufficient criteria is of particular interest, as the detection is more difficult and complex and also requires advanced technical knowledge and experience to understand the malicious behavior of the infected files [7]. Therefore, a significant part of the research community of information systems security and machine learning has turned its attention to malware classification using specialized methodologies and advanced techniques for modeling PE file behavior.

The rest of the work includes Section 2 that gives a detailed description of the proposed Artificial Evolutionary Fuzzy LSTM Immune System, a related work section, and Methodology section which describes in detail the methodology of the proposed system, while Experiments section explains the data used and the scenarios taken into account for the implementation of the proposed system. Finally, Conclusions section summarizes the research conducted and presents the future objectives that extend it.

2. The Proposed Artificial Evolutionary Fuzzy LSTM Immune System

As mentioned, malware detection from the current generation of antimalware products typically uses a signature-based approach, where a set of rules attempts to detect different groups of known types of malware. These rules are very specific; they are generally fragile and usually cannot detect new or transformed malware even if it uses the same functionality. Instead, the proposed architecture introduces an advanced methodology for distinguishing between benign and malicious PE executable files for Windows OS,

taking as input only the raw byte sequence of the files under investigation [3, 4, 8].

This approach has several practical advantages as it does not require complex hand-crafted features or specialized knowledge of how it is used to compile the way the malware is working. This means that, from the point that the model is properly trained, it can generalize new threats and at the same time be resistant to variants of malware that may result from polymorphism or metamorphism. Also, the computational complexity depends linearly on the length of the examined sequence (binary size), which means that the classification can be done relatively quickly and can work even in very large files [9]. It is also interesting that the analysis can be done in sections or subsections of the binary code, which makes the approach adaptable to new or similar file formats, which may come from different compilers and implementation architectures.

But the most basic and essential feature in dealing with polymorphic malicious files is the fact that the contents of a binary code at the operational level can be arbitrarily rearranged with small effort, but there is a complex spatial correlation between their functions due to system call functions and jump commands [10, 11]. Thus, this analysis can lead to the detection and successful categorization of code that has undergone polymorphism or metamorphism techniques that are used by malware developers and are particularly difficult to detect by existing methodologies.

The main innovation of the proposed immune system is the fact that it can only be trained from raw bytes of an executable file to determine if the file is malicious. However, there are many additional challenges. Specifically, treating each byte as a unit in an input sequence means that a sequence classification problem of the order of thousands to millions of time steps is created. This goes far beyond the length of data entry into sequence classifiers. Also, bytes in malware can have a lot of information details. Any byte received could encode the human-readable text, binary code, or arbitrary objects such as images, audio, etc. In addition, some of this content may be encrypted [12, 13].

But the most important problem is that sequence allocation in individually processed cases will not work, as malware indexes can be sparse and distributed throughout the file, so there is no way to map global tags for a training set (file) in later phases without importing too much noise. In addition, having only one label for thousands or millions of time steps of an input sequence with sparse distinctive features creates an extremely difficult machine learning problem due to the very weak training signal [14].

To address the above challenges in this work, an innovative Artificial Evolutionary Fuzzy LSTM Immune System is proposed, which is inspired by the way the body reacts to the appearance of a pathogen and mimics, at a higher, abstract level, the general framework of the immune system, combining evolutionary intelligence, medium-term memory, and fuzzy knowledge to detect Portable Executable Malware (PEM).

In artificial intelligence, Artificial Immune Systems (AIS) are a class of computationally intelligent, rule-based machine learning systems inspired by the principles and processes of the vertebrate immune system. The algorithms are typically modeled after the immune system's characteristics of learning and memory, for use in problem solving. AIS are distinct from computational immunology and theoretical biology that is concerned with simulating immunology using computational and mathematical models towards better understanding the immune system, although such models initiated the field of AIS and continue to provide a fertile ground for inspiration. In any case, a detailed explanation of how exactly the vertebrate immune system operates, is necessary in order to understand the proposed system.

When a virus enters the human cells, some of its protein fragments (peptides) bind to the Major Histocompatibility Complexes (MHC) molecular system. MHC genes are highly polymorphic and encode cell membrane protein molecules (antigens), which show structural and functional similarities. Lymphocytes, as specific cells of the immune system, undertake the task to recognize the virus. To achieve the identification of virus-infected cells, lymphocytes must have specific receptors to bind to the antigens (peptides) that bind to the MHC, so that at their cross-linking, they produce an immune response which translates into specific cytotoxic processes that kill infected cells.

The immune response focuses on the production of specific antibodies that are produced by a chemical immune response, while at the same time clones of specific lymphocytes are produced that activate the cell-mediated immune response. Both antibodies and lymphocytes recognize certain virus proteins (antigens), bind to them, and either inactivate the virus itself (neutralizing antibodies) or kill the virus-infected cells [15].

The proposed algorithm does not attempt to model exactly the above mechanism of the immune system, but borrows some of its features, in particular, the theory of clone selection and immune network. The recursion process will allow detecting polymorphism and metamorphism malware.

It establishes the idea that it is worth cloning only the lymphocytes that better recognize the pathogen, to create a large number of antibodies that will largely match specific antigens, significantly enhancing the role of memory antibodies. Antibodies are considered to be the possible solutions, antigens are the test data, and the degree of similarity between an antibody and an antigen represents the quality of the solution.

3. Literature Review

The basic principles of inspiration that AIS [16, 17] try to simulate, are the ability of the natural immune system to acknowledge normal cells, to distinguish the normal from the foreign, to be able to accurately characterize whether a foreign cell is harmful or not, to use lymphocyte cloning and mutation to adapt to the foreign cells that the body is dealing with, and to react directly to foreign molecules expressed by a pathogen that triggers the immune system response (antigens) that the body has already experienced, an action which is due to memory cells [18].

Also, a very important feature that provides inspiration and tries to be modeled by AIS concerns the multiple levels, the defense-in-depth, and the cover overlap of defense of the natural immune systems. A simple example of capturing these characteristics is the way the skin of living organisms' works [17]. The first line of defense is the skin, nasal hairs, etc., which essentially block the absorption of pathogens such as foreign particles, viruses, bacteria, fungi, etc. This zone is reinforced by feedback mechanisms like tears, saliva, sweat, and tears which strengthen the normal defense, by removing pathogens from the body or containing digestive enzymes [19].

Another important feature that AIS tries to model is the combination of innate and acquired immunization [20]. The innate immune system uses several molecular patterns to identify pathogens; it exists from birth and does not adapt during the life of living organisms. The acquired immune system, on the other hand, is the creation of the body's exposure to pathogens and the retrieval of the history of invaders and how they can be treated. In case a pathogen tries to invade the organism, a combined action takes place between the innate and the acquired immune system to deal with the invasion [16, 21].

The immensely valuable physical ability of the immune system to distinguish between different cells and locate and often eradicate the infected has inspired researchers in the field of information systems security to create corresponding mechanisms that could diversely enhance the active security of these systems [22].

A summarization with the most well-known immune methods that can be extracted from literature is presented in Table 1.

Over the past years, researchers have tried to combine the features of Artificial Immune Systems (AIS) with cybersecurity and more specifically to find malware. Also malware detection and more specifically Portable Executable Malware and the process of differentiating it from benign programs pose a significant research field for security researchers. In this section, we present some studies in both fields [23].

Fernandes et al. [21] made a survey of the applications of AIS to computer security. The article introduces the principles of Artificial Immune Systems and surveys several works applying such systems to computer security problems. This work pointed to the open issues afterward, elaborating on the novel applicability of these systems to cloud computing environments. Also, Aldhaheri et al. [10] proposed a novel Deep Learning and Dendritic Cell Algorithm based IDS framework (DeepDCA), to identify IoT intrusion and minimize the false alarm generation. In addition, Tabatabaefar et al. [22] proposed an AIS based intrusion detection system to achieve higher precision in intrusion detection. In this scheme two sets of antibodies—positive and negative—are generated for normal and attack samples, respectively, using negative selection and positive selection theories in primary detectors' generation. The simulation showed that the proposed algorithm achieved 99.1% true positive rate while the false positive rate is 1.9%.

Kumar et al. [4] proposed a novel derived feature engineering technique that improves the performance of a machine learning-based classifier for malicious PE file detection [24, 25]. The proposed technique used static analysis techniques to extract the features which have lower time and resource requirement than dynamic analysis. And finally, Vyas et al. [8] investigated static feature-based malware detection by using different supervised learning algorithms and proposed a network malware detection process for real-time malware detection on the network. They targeted malicious PE file detection with a small number of features and investigated how much they could push the supervised learning techniques towards malware detection while minimizing the computational cost for network malware detection. This research explored four supervised techniques: Decision Tree, k-NN, SVMs, and Random Forests for malware detection using the constructed 28 static features. Techniques were evaluated on four types of malware: backdoor, virus, trojan, and worm.

Table 1: Immune methods in literature review.

ID	Method	References
1	Lymphocyte cloning and mutation	[16–18]
2	Skin defense	[17, 19]
3	Immunization	[16, 20, 21]
4	Cell defense	[10, 22]
5	Antibodies	[22]

4. Methodology

This paper proposes a novel method to understand the polymorphism and metamorphism mechanisms used by malware developers and how to effectively address them. The forecasting approach provides insights of the way of the evolution of malware practices and can facilitate decision-making and management of security strategies. The determination achieved by the proposed model is indicative of its effectiveness and reliability to the extent that it incorporates fitting techniques of high resolution with latent information being visible after transforming the PE file into a raw code. The proposed Artificial Evolutionary Fuzzy LSTM Immune System is presented in Figure 2.

The flow procedure is generally described as follows [15–17, 21, 22, 26, 27].

4.1. Initialization. During initialization, all the elements of the data set that the algorithm receives as input are normalized in such a way that the Euclidean distance between any two elements of the data set is in the interval [0, 1]. Let D be the set containing the data to be classified and $x, y \in D$ where $x, y \in R$; then, the distance is defined as

$$dist_{\text{Euclidean}}\left(x_0, x_j\right) = \sqrt{\sum_{i=1}^{n}\left(x_0^i - x_j^i\right)^2} \text{ so that } d(x, y) = \|x - y\|_{\text{norm}} \leq 1, \forall x, y \in D. \quad (1)$$

The data set D consists of x classes of size W, with the training set T being a subset of a class of D and used to train the elements of this class so that

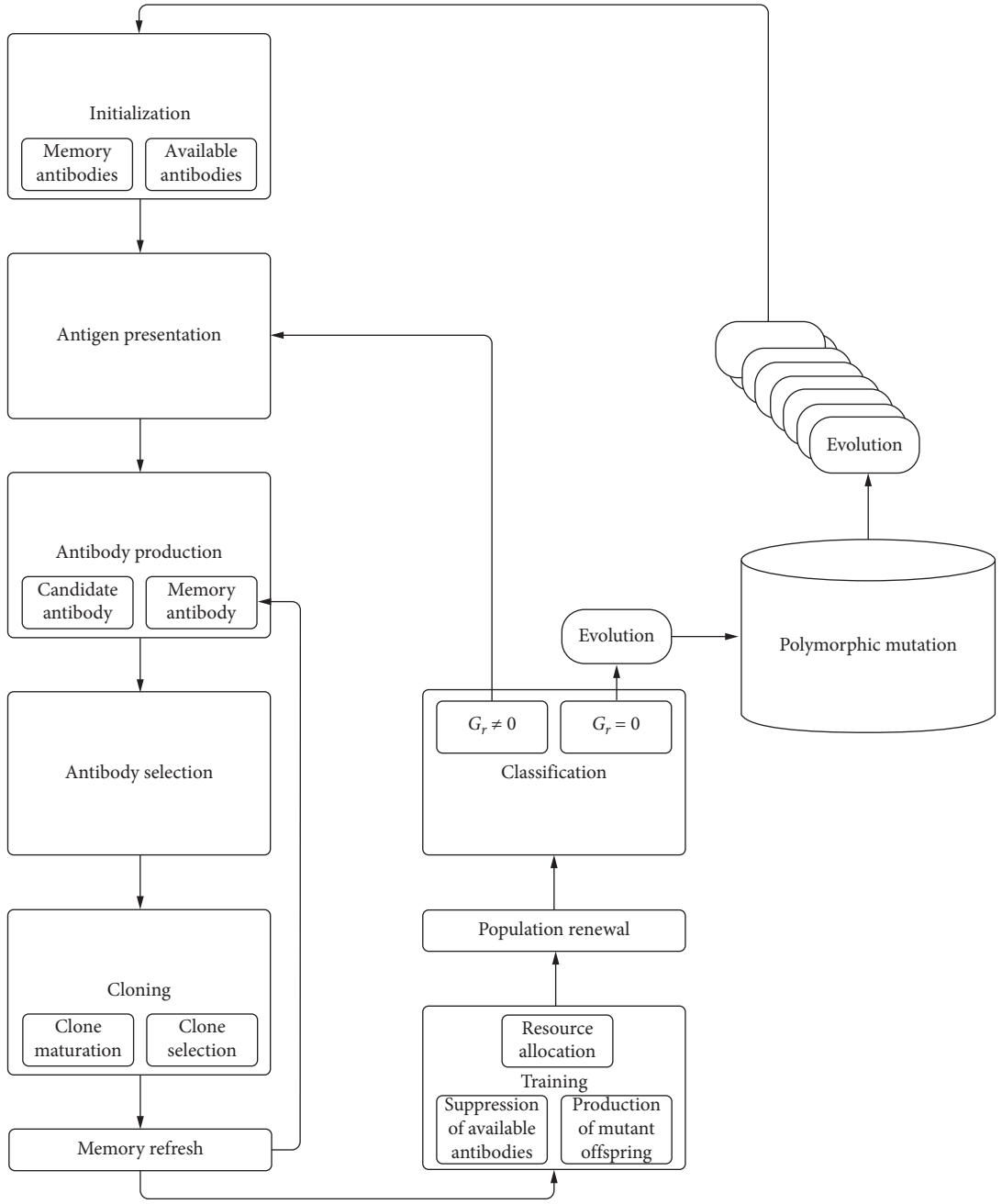

Figure 2: Structure of the proposed immune system.

$$T_i \subseteq W_i \subseteq D, i \in \{1, 2, \ldots, x\}. \quad (2)$$

The algorithm then calculates the affinity threshold, i.e., the average value of the distances between the elements of the training set, as follows:

$$\mathrm{Aff}_{\mathrm{Thr}} = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{\mathrm{aff}\left(ag_i, ag_j\right)}{(n(n-1))/2}, \text{whereaff}\,(x, y) = \|x - y\|_{\mathrm{norm}}. \quad (3)$$

The final stage of this phase is the initialization of the set of memory antibodies and the set of available antibodies.

4.2. Antibody Set Initialization. For each $a_i \in P, 1 \leq i \leq |P|$, antibody a random sequence of $s_i \in L$ symbols is selected and assigned to it, $a_i \leftarrow s_i$. The set $G_r \in L$: $G_r = G$ is also defined. The set of memory antibodies for each class is initialized to the current antigenic template from the same template class or a set of antigenic templates from the same template class.

4.3. Antigen Presentation. An $ag \in G_r, 1 \leq i \leq |G_r|$ antigen is randomly selected and presented to the population, while at the same time the binding function f is calculated for each antibody in the population. The following set is thus obtained: $V\{v_j\colon v_j = f(a_j, ag_i), 1 \leq j \leq |P|\}$ which describes

the degree of binding of each antibody in the population with the ag antigen. The ag antigen is removed from G_r so $G_r \leftarrow G_r - \{ag\}$.

4.4. Determination of Compatible Memory Antibody. The algorithm is one-shot; i.e., it examines one element (antigen) at a time. The first step is to identify a compatible memory antibody from the set of memory antibodies. Let ag be an antigen from the training set; identify the mc_{match} memory antibody that exhibits the greatest degree of stimulation relative to the current ag antigen. $mc_{\text{match}} = \text{argmax}_{mc} \in MC_{ag \cdot c}\ stim(ag, mc) \acute{o} \pi ovstim(x, y) = 1 - \|x - y\|_{\text{norm}}$.

Thus mc_{match} is the memory antibody that is less distant from the ag antigen. If the set of memory antibodies of this template class is empty, i.e., $MC_{ag \cdot c} = \varnothing$, then the $mc_{\text{match}} \leftarrow ag$; i.e., the mc_{match} is the antigenic template itself and thus is placed inside the set of memory antibodies.

4.5. Identification of Candidate Antibody. The candidate vector for memory is the characteristic vector that exhibits the greatest degree of stimulation relative to the current antigenic pattern, called $mc_{\text{candidate}}$.

4.6. Antibody Production. The mc_{match} memory antibody that exhibits the highest degree of stimulation to the current antigenic standard ag is used as the archetype to produce a set of mutated versions of the original. These antibodies will be included in all available antibodies to address the polymorphism and metamorphism mechanisms used in malware development. The rate of mutation is inversely proportional to the degree of stimulation to the current antigenic pattern.

4.7. Antibody Selection. Based on the data of a set V, the n_b antibodies are selected that indicated the best binding quality and now constitute the set $B, |B| = n_b$.

4.8. Amplification/Cloning. Based on the quality of its binding to the antigen ag, each antibody of set B is cloned, with each antibody yielding more clones depending on its quality. A new set C includes the resulting clones.

4.9. Clone Maturation. Each element c_j of set C changes at an a_j rate which depends on the degree of binding of clone c_j to the antigen ag. The better the binding quality, the lower the rate of mutation so that no reversible changes are made to the antibody [28, 29]. The set of mutant clones composes the set C_m.

4.10. Clone Selection and Memory Refresh. The function f is applied to each element of the set C_m and the set V' is obtained which contains the connection quality of each mutated clone, $V' = \{v'_j : v'_j = f(c'_j, g_i), 1 \le j \le |C_m|\}$. Based on V' the n_m best clones are selected which constitute the set B'. Imaging K is then applied to the g_i antigen to obtain the M_i set of memory antibodies that are candidates for replacement. Based on the memory renewal policy followed by the algorithm, a final set of M'_i cells is obtained such that $n_m = |M'_i| \le |M_i|$. The memory cells of the set M'_i will be replaced by other selected cells if and only if these cells show a better quality of connection, which means that the condition $f(m, g_i) < f(a, g_i), m \in M'_u, a \in B'$, must apply [30].

4.11. Introduction of Memory Antibodies. Affinity threshold is used as a criterion for placing $mc_{\text{candidate}}$ in the set of memory antibodies if its degree of stimulation, in terms of the current antigenic standard, is higher than that of mc_{match}. If this is the case, then

$$\text{aff}(mc_{\text{candidate}}, mc_{\text{match}}) = \left\| mc_{\text{candidate}} - mc_{\text{match}} \right\|_{\text{norm}}, \quad (4)$$

and then $mc_{\text{candidate}}$ is placed in the memory antibody set and replaced by mc_{match}.

4.12. Training Procedure. The training procedure is repeated until the average degree of stimulation of all available antibodies is less than a predetermined value. This step of the algorithm aims to generate antibodies that better recognize the current antibody.

4.12.1. Resource Allocation. For each element of the set of available antibodies, a portion of the total system resources is committed depending on the degree of stimulation of the current antigenic pattern.

4.12.2. Suppression of Available Antibodies. Those antibodies that bound the smallest part of the total system resources are deleted.

4.12.3. Production of Mutant Offspring. The subset of available antibodies that have secured most of the system's resources has an additional opportunity to produce mutant progeny.

4.13. Population Renewal. To maintain population diversity, either n_t cells are selected from the set V' and introduced into the population replacing some others, or n_d worse cells from the P population are selected and replaced with completely new ones.

4.14. Classification. The k-NN classifier with Self-Adjusting Memory (k-NN SAM) is used for classification [31–33]. The k-NN SAM algorithm is inspired by the field of human memory research and specifically by the dual model of short-term and long-term memory (STM & LTM). The information that reaches the STM through the sensory organs is accompanied by relevant knowledge derived from the LTM. The information that receives a lot of attention and is considered important is transferred to LTM in the form of Synaptic Consolidation. STM capacity is quite limited and

information is retained for a very short time, unlike LTM, which can retain information for several years. A typical example of how human memory works in this field is the fact that we never forget the way we ride a bike, no matter how many years have passed since our last bike ride. The architecture of k-NN SAM is partly inspired by this model, presenting proportions such as the obvious separation of short-term and long-term memory, the different retention times between memories, and the transfer of knowledge from STM to LTM and vice versa. The implementation of this algorithm as a categorization model is based on the general assumption that the new data is more relevant to the current predictions, but prior knowledge is also required for their correct classification. The optimal combination of the two processing levels can minimize errors and increase categorization accuracy. Memories are represented by sets of short-term memory (M_{ST}), long-term memory (M_{LT}), and merged memory (M_M). Each memory is a subset of $R^n \times \{1, \ldots, c\}$ of different lengths, which fluctuates during the adjustment process. M_{ST} represents the current idea and is a dynamic slider containing the latest m data flow examples:

$$M_{ST} = \{(x_i, y_i) \in R^n \times \{1, \ldots, c\} \vee i = t - m + 1, \ldots, t\}. \tag{5}$$

M_{LT} retains all former information, which does not conflict with that of M_{ST}. Unlike M_{ST}, M_{LT} is not a continuous part of the data stream, but a set of points p:

$$M = \{(x_i, y_i) \in R^n \times \{1, \ldots, c\} \vee i = 1, \ldots, p\}. \tag{6}$$

The association of both memories is the M_M memory:

$$MM = M_{ST} \cup M. \tag{7}$$

Each set includes the weighted k-NN classifier:

$$R^n \times \{1, \ldots, c\}, \; kNN_{M_{ST}}, kNN_M, kNN_{M_M}. \tag{8}$$

The k-NN function assigns a label to a given point x based on a set $Z = \{(x_i, y_i) \in R^n \times \{1, \ldots, c\} \vee i = 1, \ldots, n\}$:

$$kNN_Z(x) = \operatorname{argmax}\left\{ \sum_{x_i \in N_k(x,Z) \vee y_i = \widehat{c}} \frac{1}{d(x_i, x)} \vee \widehat{c} = 1, .., c \right\}, \tag{9}$$

where $d(x_i, x)$ is the Euclidean distance between two points and the $N_k(x, Z)$ returns the set k of x' s nearest neighbors to Z.

Generally speaking, the LSTM function is capable of learning order dependence in sequence prediction patterns [34]. Each one of the 3 types of the gate in a LSTM cell, forget gate, input gate, and output gate (M_{ST}, M_{LT}, and M_M), will decide what portion of the older data have to be forgotten, what portion of newer data have to be remembered, and what portion of the memory has to be given out correspondingly [35]. The main reason we used the LSTM function is that the contents of a binary at the function level can be arbitrarily rearranged with little effort in cases of polymorphism and metamorphism, but there is always a complicated spatial correlation across functions due to function calls and jump commands which can be identified by a recurrent model.

4.15. Termination Condition. If $G_r \neq 0$, then the algorithmic procedure is repeated from the second step of the antigen presence. Otherwise, some criterion of convergence of memory antibodies M with the antigens of set G is checked. In the case of unsuccessful convergence, $G_r \leftarrow_G$ and the algorithm is repeated from the second step of the antigen presence then, wherein in the opposite case $G_r = 0$ and thus the algorithm terminates and a generation of evolution is completed.

4.16. Polymorphic Mutation. Antibodies involved in the treatment of polymorphism and metamorphism mechanisms used by malware developers are initialized through Gibbs sampling [36, 37]. Gibbs sampling is a Markov Monte Carlo chain algorithm that takes repeated samples from the target p distribution, taking into account all other variables [38]. The basic idea is simple: instead of calculating in detail the quantities we are interested in, with complex posterior distributions, we simulate a sample of values from a suitable Markovian chain that is in equilibrium. So we can calculate the characteristics we want (average value, dispersion, etc.) through the corresponding values of the sample. The Gibbs sampler simulates observations from multidimensional target distributions through their fully bound distributions, which in our case have a known form. Thus, the problem of simulating observations from a large-dimensional target distribution is transformed into a problem of simulating observations from smaller dimensional distributions [39].

After defining the set of antibodies by the above procedure (Algorithm 1), assign each point $x^{(i)}$ of the data set to some possible solution C_k so that the function Score (C, D) is maximized or minimized on a case-by-case basis. The equation of calculating the function is given as follows:

$$\text{Score}(C, D) = \sum_{k=1}^{K} d(x, c_k), \tag{10}$$

where $c_k = (1/n_k) \sum_{x \in c_k} x$ and $d(x, y) = \|x - y\|^2$.

The probability of classification error is

$$P_B \leq P_C \leq P_B + \frac{1}{\sqrt{ke}}, \tag{11}$$

where P_B is the optimal Bayesian error which expresses the probability that c is the value of the dependent variable C based on the values $x=(x_1, x_2, ..., x_n)$ of the attributes $X=(X_1, X_2,..., X_n)$ and is given by the relation [40]

$$P(c \vee x) = P(c) \cdot \prod_{i}^{n} P(x_i \vee c). \tag{12}$$

In this way, vague sets of solutions are created. This is a more realistic categorization of elements with fuzzy boundaries, where the transition from the category of X elements belonging to the fuzzy set $\widetilde{A}$ to the category of X

Initialize $x^{(t)} = (x_1^{(t)}, \ldots, x_k^{(t)}) for t = 0$
For $t = 0, 1, \ldots$
Pick index i uniformly at random from $1, \ldots, k$
Draw a sample $a\ p(x_i' \vee x_{-i}^{(t)})$ where $x_{-i}^{(t)}$ is the set of all variables in $x^{(t)}$ except for the i^{th} variable.
Let $x^{(t+1)} = (x_1^{(t)}, x_2^{(t)}, \ldots, x_{i-1}^{(t)}, a, x_{i+1}^{(t)}, \ldots, x_k^{(t)})$
Let x_i denote the i^{th} variable and let x_{-i} denote the set of all variables except x_i. Let $Q(x_i', x_{-i} \vee x_i, x_{-i}) = 1/k p(x_i' \vee x_{-i})$. Let
$A(x_i', x_{-i} \vee x_i, x_{-i}) = \min(1, a)$ where
$a = (p(x_i', x_{-i}) Q(x_i, x_{-i} \vee x_i', x_{-i}) / p(x_i, x_{-i}) Q(x_i', x_{-i} \vee x_i, x_{-i})) \longrightarrow a = (p(x_i', x_{-i}) p(x_i \vee x_{-i}) / p(x_i, x_{-i}) p(x_i' \vee x_{-i})) \to a$
$= (p(x_i' \vee x_{-i}) p(x_{-i}) p(x_i \vee x_{-i}) / p(x_i \vee x_{-i}) p(x_{-i}) p(x_i' \vee x_{-i})) \longrightarrow a = 1$

Algorithm 1: Polymorphic mutation algorithm.

elements that do not belong to A is not abrupt-clear but is gradual-vague. Among the created fuzzy sets, operations can be performed on a case-by-case basis as follows (μ is called the membership function of the fuzzy set) [41]:

$$\begin{aligned}
\mu_{\tilde{A} \cup \tilde{B}}(x) &= \mu_{\tilde{A}}(x) \vee \mu_{\tilde{B}}(x) = \max\left[\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)\right] \forall x \in X, \\
\mu_{\tilde{A} \cap \tilde{B}}(x) &= \mu_{\tilde{A}}(x) \wedge \mu_{\tilde{B}}(x) = min\left[\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)\right] \forall x \in X, \\
\mu_{\tilde{A} \cdot \tilde{B}}(x) &= \mu_{\tilde{A}}(x) \cdot \mu_{\tilde{B}}(x) \forall x \in X, \\
\mu \Gamma_{\tilde{A}} &= 1 \mu_{\tilde{A}}(x).
\end{aligned} \tag{13}$$

The use of fuzzy sets arises from the fact that learning techniques are designed for stable environments, in which training and testing data are considered to be generated from the same (possibly unknown) distribution. A properly designed and implemented binary code corresponding to a modified pattern may come from a slightly differentiated malware and it can lead the algorithm to make a wrong classification decision. The fuzzy set theory permits the gradual assessment of the membership of elements in a set; this is described with the aid of a membership function valued in the real unit interval $[0, 1]$ [42]. From this point of view, it helps in understanding the dynamic environment and offers a range of adequate explanations that could occur as part of the human decision-making process [43].

5. Experiments

A set of 19,620 PE files was used to test and validate the proposed system, of which 11,084 were benign files from a clean install of Microsoft Windows and some commonly installed applications, while the remaining 8,536 files were PEM that came from the most updated VirusShare database [44]. All experiments were performed in the Google Colab [45] environment using a Tesla P100 GPU, using the Tensorflow library. To achieve timely model convergence, it was necessary to train the proposed system using a relatively small but at the same time satisfactory batch size, which after extensive trial and error tests resulted in 872 samples. Due to overuse of memory, this required the use of parallel model training using all available GPU memory. The results of the process are presented in the table below and the corresponding diagrams. Specifically, the most popular evaluation measures, which can clearly and objectively identify the proposed system with extensive comparison with other machine learning algorithms, are presented in Table 2 [46]:

The Correctly Classified Instances, i.e., the accuracy of the procedure, was calculated at 98.59%, which essentially expresses the percentage of classification of the plots of PE samples that were checked and that are correctly categorized. Only 276 files, i.e., 1.41%, were categorized incorrectly, a fact that is interpreted as 0.014 false positive rate, with a corresponding 0.986 true positive rate. Figure 3 depicts the confusion matrix that provides the accurate and aggregate information needed to evaluate the model [46].

In particular, information for a more complete understanding and evaluation of the process, concerning the unique number of performance measures that can be expressed about the number of true positive, true negative, false positive, and false negative classifications, is presented in Figure 4, with the display of Precision, Recall, and F1-Score for each class separately [46].

The most important measurement for evaluating the performance of the model is the ROC area, which gathers information about the prediction quality of the categorizer for different threshold values while remaining independent of the possible class imbalance in the data. The very high ROC area rating (with Weighted Average of 0.987, i.e., very close to 1), as shown in Figure 5 below, corresponds to the successful ranking of most malicious programs [46].

A visualization of the Precision, Recall, and F1-Score, concerning the classifier discrimination threshold, is shown in Figure 6. The discrimination threshold depicts how the system ranks a PE in the positive order versus the negative order. Generally, this is usually set at 50%, but in this case, the threshold was set to 48% to increase the sensitivity to false positives based on the queue rate, i.e., the percentage of files to be checked [46].

Finally, additional diagrams showing the quality of the proposed model are presented in Figures 7–9 [46].

Making a general assessment of the process proposed and evaluated in this study, we demonstrated the categorizer's ability to differentiate between benign and malicious PE files with high accuracy and with the same importance given to each one, without any unwanted bias, which is most often the result of bad categorizers that cannot generalize. It is also important to note that very accurate process

Table 2: Performance metrics.

Correctly classified instances										19344	98.59%
Incorrectly classified instances										276	01.41%
					Weighted average						
Method	*Accuracy*	*TP rate*	*Precision*	*Recall*	*F1-score*	*MCC*	*ROC area*	*PRC area*	*MAE*	*RMSE*	*K stats*
Proposed	98.59%	0.986	0.986	0.986	0.986	0.971	0.987	0.981	0.0141	0.1167	0.9714
SVM	92.91%	0.929	0.930	0.930	0.930	0.935	0.960	0.965	0.0205	0.1233	0.9398
NaBayes	88.38%	0.884	0.884	0.885	0.885	0.884	0.890	0.890	0.0318	0.2034	0.8904
k-NN	90.63%	0.906	0.900	0.900	0.910	0.900	0.900	0.950	0.0297	0.1293	0.9008
RF	97.03%	0.970	0.970	0.970	0.970	0.965	0.970	0.972	0.0170	0.1195	0.9682

SVM = support vector machines; NaBayes = naïve Bayes k-NN = k nearest neighbor; RF = random forest; TP rate = true positive rate; FP rate = false positive rate; MCC = Matthews correlation coefficient; ROC area = receiver operating characteristic area; PRC area = Precision-Recall curve area; MAE = mean absolute error; RMSE = root mean square error.

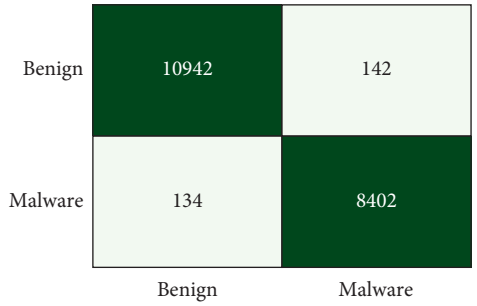

Figure 3: Confusion matrix.

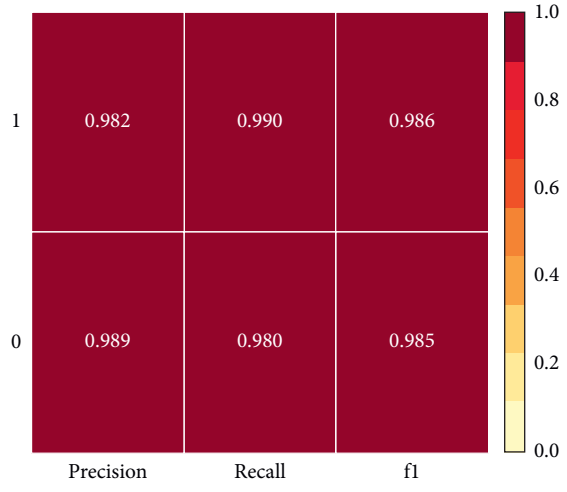

Figure 4: Precision, Recall, and F1-Score by class.

predictions encourage the use of the model, as the manual analysis of a single binary PE file by a dedicated malware researcher can take more than 10 hours. Thus, in the proposed way, the process is significantly simplified and accelerated, which makes this method capable of being used in forensic investigations, where a fast and valid assessment of malicious actions is required.

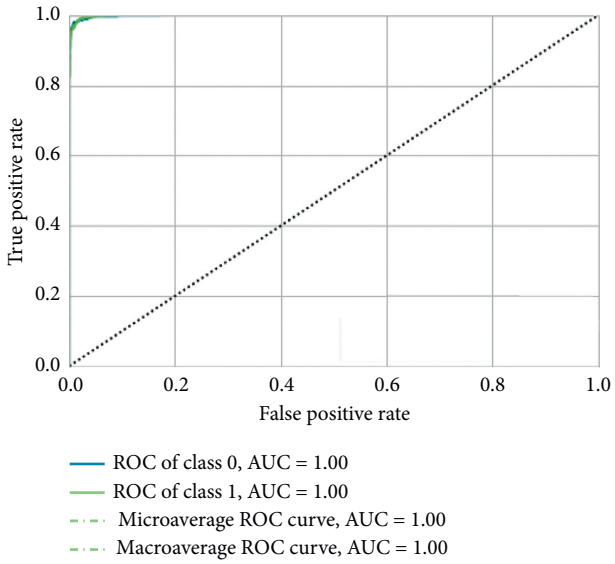

Figure 5: ROC curves.

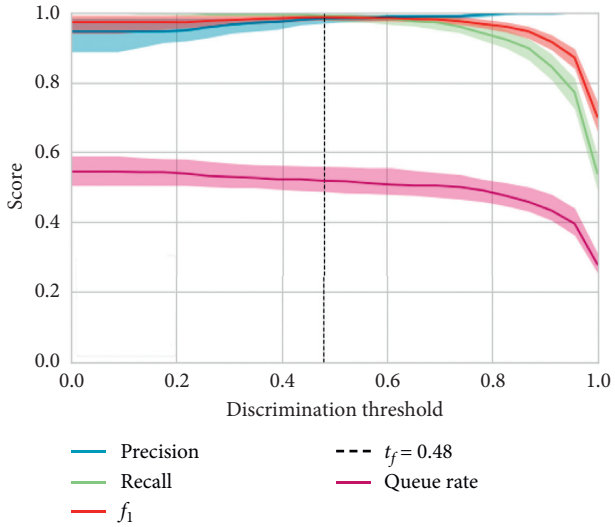

Figure 6: Threshold plot.

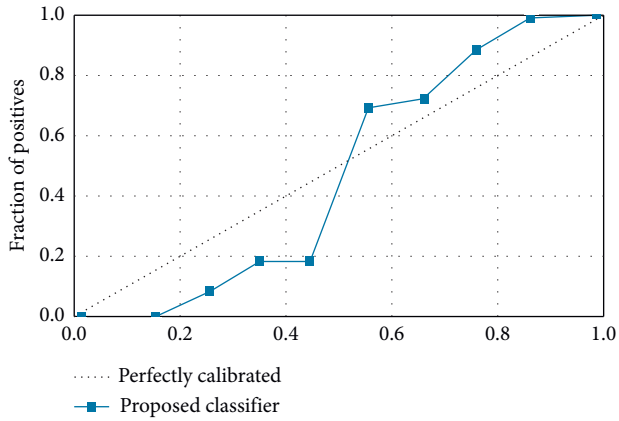

Figure 7: Reliability curves.

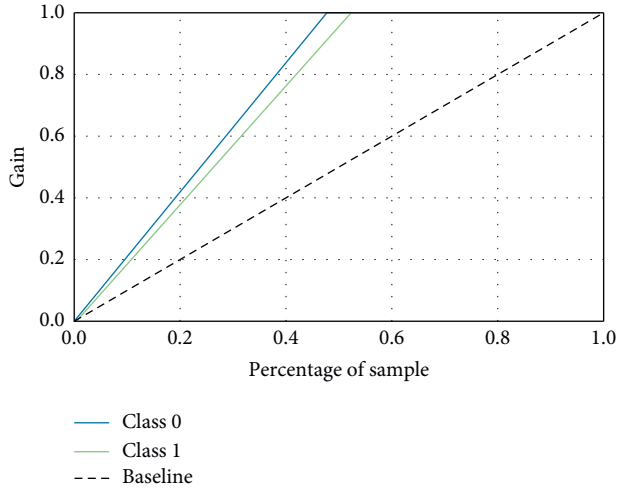

Figure 8: Gains curves.

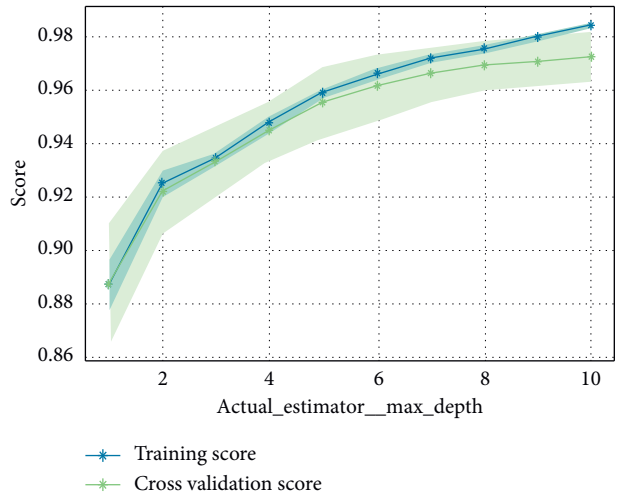

Figure 9: Training and validation curves.

6. Conclusions

The proposal of the present work is about a method of malware detection inspired by the effectiveness of the immune system. The implementation of the method is based on the fact that minimal effort has been made to utilize biologically inspired machine learning in polymorphic and metamorphic malicious classification problems. The aim of the proposed Artificial Evolutionary Fuzzy LSTM Immune System is to produce multiple identical solutions, to increase the algorithm classification accuracy into various malicious patterns, which result from polymorphism or metamorphism. It is a hybrid system that optimally combines evolutionary intelligence, medium-term memory, and fuzzy knowledge to analyze and classify Portable Executable Malware.

The proposed immune system is trained to differentiate between benign and malicious Windows executable files with only the raw byte sequence of the executable as input. This approach has several practical advantages [47]:

(1) No hand-crafted features or knowledge of the compiler used is required. This means the trained model is generalizable and robust to natural variations in malware.

(2) The computational complexity is linearly dependent on the sequence length (binary size), which means inference is fast and scalable to very large files.

(3) Important subregions of the binary can be identified for forensic analysis.

(4) This approach is also adaptable to new file formats, compilers, and instruction set architectures.

The main innovation of the proposed algorithmic method is the detectors that successfully detect malicious patterns and which are placed in long-term memory so that, in this way, the set of detectors creates a different distribution of the set of successful training. Essentially, the problem of dealing with polymorphism and metamorphism mechanisms is modeled as a problem of optimizing the distance of the set of detectors with the objects of the training set. The function to be optimized is a function of the distance of the detectors to the objects of the training set.

Similarly, a key innovation in technical implementation is the challenge of whether a machine learning system could only be trained from raw bytes of an executable file to determine if the file is malicious. This success could greatly simplify the tools used to detect malware, improve detection accuracy, and detect obscure but important malware features. We are convinced that this article proves that detecting malware from raw byte sequences has unique and challenging properties that make it a fertile research field for the machine learning community.

The algorithm implemented can be the basis for several future extensions. More specifically, some extensions and variations to the classification algorithm could be applied to investigate system behavior in cases of adversarial examples. The function could also be investigated by adding predefined weight tables containing weights depending on the weight of

the feature in the classification process, to implement the proposed system faster and more quickly. An additional feature that could be added to the classification algorithm is a function for transferring data to even larger dimensions to create different correlations between data and categorization patterns. Finally, another point of research could be the addition of a feature reduction process for the more efficient operation of the proposed Artificial Evolutionary Fuzzy LSTM Immune System.

Data Availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research work was supported by the MOE (Ministry of Education in China) Liberal Arts and Social Sciences Foundation (No. 17YJCZH157). It was also supported by the Innovation Team of Guangdong Provincial Department of Education (2018KCXTD031).

References

[1] I. F. Mikhalevich and V. A. Trapeznikov, "Critical infrastructure security: alignment of views," in *Proceedings of the 2019 Systems of Signals Generating and Processing In the Field of on Board Communications*, pp. 1–5, Moscow, Russia, March 2019.

[2] E. Raff, J. Sylvester, and C. Nicholas, "Learning the PE header, malware detection with minimal domain knowledge," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 121–132, Dallas, TX, USA, November 2017.

[3] T.-Y. Wang, C.-H. Wu, and C.-C. Hsieh, "Detecting unknown malicious executables using portable executable headers," in *Proceedings of the 2009 Fifth International Joint Conference on INC, IMS and IDC*, pp. 278–284, Seoul, South Korea, 2009.

[4] A. Kumar, K. S. Kuppusamy, and G. Aghila, "A learning model to detect maliciousness of portable executable using integrated feature set," *Journal of King Saud University Computer and Information Sciences*, vol. 31, no. 2, pp. 252–265, 2019.

[5] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "BotHunter: detecting malware infection through IDS-driven dialog correlation," in *Proceedings of the 16th {USENIX} Security Symposium ({USENIX} Security 07)*, pp. 1–16, August 2007, https://www.usenix.org/conference/16th-usenix-security-symposium/bothunter-detecting-malware-infection-through-ids-driven.

[6] L. Chen, T. Li, M. Abdulhayoglu, and Y. Ye, "Intelligent malware detection based on file relation graphs," in *Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015)*, pp. 85–92, Anaheim, CA, USA, February 2015.

[7] L. Garcia, F. Brasser, M. Cintuglu et al., "My malware knows physics! attacking PLCs with physical model aware rootkit," in *Proceedings of the 2017 Network and Distributed System Security Symposium*, March 2017.

[8] R. Vyas, X. Luo, N. McFarland, and C. Justice, "Investigation of malicious portable executable file detection on the network using supervised learning techniques," in *Proceedings of the 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 941–946, Lisbon, Portugal, May 2017.

[9] A. Borkar, A. Donode, and A. Kumari, "A survey on intrusion detection system (IDS) and internal intrusion detection and protection system (IIDPS)," in *Proceedings of the 2017 International Conference on Inventive Computing and Informatics (ICICI)*, pp. 949–953, Coimbatore, India, November 2017.

[10] S. Aldhaheri, D. Alghazzawi, L. Cheng, B. Alzahrani, and A. Al-Barakati, "DeepDCA: novel network-based detection of IoT attacks using artificial immune system," *Applied Sciences*, vol. 10, no. 6, p. 1909, 2020.

[11] M. S. Ejaz, M. R. Islam, M. Sifatullah, and A. Sarker, "Implementation of principal component analysis on masked and non-masked face recognition," in *Proceedings of the 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, pp. 1–5, Dhaka, Bangladesh, May 2019.

[12] G. Memmi, K. Kapusta, and H. Qiu, "Data protection: combining fragmentation, encryption, and dispersion," in *Proceedings of the 2015 international Conference on cyber Security of smart cities, Industrial Control System and Communications (SSIC)*, pp. 1–9, Chengdu, China, August 2015.

[13] R. Alshammari and A. Nur Zincir-Heywood, "Identification of VoIP encrypted traffic using a machine learning approach," *Journal of King Saud University Computer and Information Sciences*, vol. 27, no. 1, pp. 77–92, 2015.

[14] H. HaddadPajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, "A deep recurrent neural network based approach for internet of things malware threat hunting," *Future Generation Computer Systems*, vol. 85, pp. 88–96, 2018.

[15] G. W. Litman, J. P. Cannon, and L. J. Dishaw, "Reconstructing immune phylogeny: new perspectives," *Nature Reviews Immunology*, vol. 5, no. 11, pp. 866–879, 2005.

[16] E. Guillen and R. Paez, "Artificial immune systems—AIS as security network solution," in *Bio-Inspired Models of Network, Information, and Computing Systems*, J. Suzuki and T. Nakano, Eds., vol. 87, pp. 680-681, Springer, Berlin, Germany, 2012.

[17] M. Read, P. S. Andrews, and J. Timmis, "An introduction to artificial immune systems," in *Handbook Of Natural Computing*, G. Rozenberg, T. Bäck, and J. N. Kok, Eds., Springer, Berlin, Germany, pp. 1575–1597, 2012.

[18] H. Park, J. E. Choi, D. Kim, and S. J. Hong, "Artificial immune system for fault detection and classification of semiconductor equipment," *Electronics*, vol. 10, no. 8, p. 944, 2021.

[19] C.-Y. Chang, Y.-C. (Angel) Lu, W.-C. Ting, T.-W. D Shen, and W.-C. Peng, "An artificial immune system with bootstrap sampling for the diagnosis of recurrent endometrial cancers," *Open Medicine*, vol. 16, no. 1, pp. 237–245, 2021.

[20] S. F. Rosenblatt, J. A. Smith, G. R. Gauthier, and L. Hébert-Dufresne, "Immunization strategies in networks with missing data," *PLoS Computational Biology*, vol. 16, no. 7, Article ID e1007897, 2020.

[21] D. A. B. Fernandes, M. M. Freire, P. A. P. Fazendeiro, and P. R. M. Inácio, "Applications of artificial immune systems to

computer security: a survey," *Journal of Information Security and Applications (JISA)*, vol. 35, pp. 138–159, 2017.

[22] M. Tabatabaefar, M. Miriestahbanati, and J.-C. Gregoire, "Network intrusion detection through artificial immune system," in *Proceedings of the 2017 Annual IEEE International Systems Conference (SysCon)*, pp. 1–6, IEEE, Montreal, QC, Canada, April 2017.

[23] R. Pump, V. Ahlers, and A. Koschel, "Evaluating artificial immune system algorithms for intrusion detection," in *Proceedings of the 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pp. 92–97, IEEE, London, UK, July 2020.

[24] R. Harang and E. M. Rudd, "SOREL-20M: a large scale benchmark dataset for malicious PE detection," 2020, http://arxiv.org/abs/2012.07634.

[25] Namita and Prachi, "PE file-based malware detection using machine learning," in *Proceedings of the International Conference on Artificial Intelligence and Applications*, pp. 113–123, Singapore, 2021.

[26] A. Sharma and D. Sharma, "Clonal selection algorithm for classification," in *Proceedings of the Artificial Immune Systems—10th International Conference*, pp. 361–370, Springer, Cambridge, UK, July 2011, Lecture Notes in Computer Science.

[27] X. Wang, A. S. Deshpande, G. B. Dadi, and B. Salman, "Application of clonal selection algorithm in construction site utilization planning optimization," *Procedia Engineering*, vol. 145, pp. 267–273, 2016.

[28] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia Tools and Applications*, vol. 80, no. 5, pp. 8091–8126, 2021.

[29] Y. Yuan, W. Wang, and W. Pang, "A genetic algorithm with tree-structured mutation for hyperparameter optimisation of graph neural networks," 2021, http://arxiv.org/abs/2102.11995.

[30] J. Liu, Z. Zhang, F. Chen, S. Liu, and L. Zhu, "A novel hybrid immune clonal selection algorithm for the constrained corridor allocation problem," *Journal of Intelligent Manufacturing*, vol. 31, no. 8, 2020.

[31] V. Losing, B. Hammer, and H. Wersing, "KNN classifier with self adjusting memory for heterogeneous concept drift," in *Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM)*, pp. 291–300, IEEE, Piscataway, NJ, USA, December 2016.

[32] M. Roseberry, A. Cano, and B. Krawczyk, "Multi-label KNN classifier with self adjusting memory for drifting data streams," *ACM Transactions on Knowledge Discovery from Data*, vol. 13, no. 6, pp. 1–31, 2019.

[33] A. Abolfazli and E. Ntoutsi, "Drift-aware multi-memory model for imbalanced data streams," in *Proceedings of the 2020 IEEE International Conference on Big Data (Big Data)*, pp. 878–885, Atlanta, GA, USA, December 2020.

[34] N. S. Malinović, B. B. Predić, and M. Roganović, "Multilayer long short-term memory (LSTM) neural networks in time series analysis," in *Proceedings of the 2020 55th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST)*, pp. 11–14, IEEE, Niš, Serbia, September 2020.

[35] J. Zhang, Y. Zeng, and B. Starly, "Recurrent neural networks with long term temporal dependencies in machine tool wear diagnosis and prognosis," *SN Applied Sciences*, vol. 3, no. 4, p. 442, 2021.

[36] M. M. Hossain, M. Fotouhi, and R. Hasan, "Towards an analysis of security issues, challenges, and open problems in the internet of things," in *Proceedings of the 2015 IEEE World Congress on Services*, pp. 21–28, New York, NY, USA, June 2015.

[37] "Gibbs sampling - an overview | ScienceDirect topics." https://www.sciencedirect.com/topics/economics-econometrics-and-finance/gibbs-sampling (accessed May 24 2021)..

[38] S. Triantafillou, F. Jabbari, and G. Cooper, "Causal Markov boundaries," 2021, http://arxiv.org/abs/2103.07560.

[39] H. S. Farahani, A. Fatehi, and M. A. Shoorehdeli, "Between-domain instance transition via the process of Gibbs sampling in RBM," 2020, http://arxiv.org/abs/2006.14538.

[40] R. Van de Schoot, S. Depaoli, R. King et al., "Bayesian statistics and modelling," *Nature Reviews Methods Primers*, vol. 1, no. 1, pp. 1–26, 2021.

[41] M. Jezewski, R. Czabanski, and J. Leski, "Introduction to fuzzy sets," in *Theory and Applications of Ordered Fuzzy Numbers: A Tribute to Professor Witold Kosiński*, P. Prokopowicz, J. Czerniak, D. Mikołajewski, Ł. Apiecionek, and D. Śl□zak, Eds., Springer International Publishing, Cham, Germany, pp. 3–22, 2017.

[42] A. Imtiaz, U. Shuaib, H. Alolaiyan, A. Razaq, and M. Gulistan, "On structural properties of -complex fuzzy sets and their applications," *Complexity*, vol. 2020, Article ID e2038724, , 2020.

[43] R. Tansuchat, U. Pham, and C. Van Le, "On soft computing with random fuzzy sets in econometrics and machine learning," *Soft Computing*, vol. 25, no. 12, pp. 7745–7751, 2021.

[44] "VirusShare.com". https://virusshare.com/(accessed May 24 2021).

[45] "Google Colaboratory". https://colab.research.google.com/notebooks/(accessed May 24 2021).

[46] P. Flach, "Performance evaluation in machine learning: the good, the bad, the ugly, and the way forward," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 9808–9814, 2019.

[47] J. Barker, "Malware detection in executables using neural networks," 2017, https://developer.nvidia.com/blog/malware-detection-neural-networks/.