Antivirus Applied to Jar Malware Detection based on Runtime Behaviors

Ricardo Pinheiro

Universidade de Pernambuco

Sidney Lima (✉ sidney.lima@ufpe.br)

Federal University of Pernambuco

Danilo Souza

Universidade de Pernambuco

Sthéfano Silva

Universidade de Pernambuco

Petrônio Lopes

Universidade de Pernambuco

Rafael Lima

Universidade de Pernambuco

Jemerson Oliveira

Universidade de Pernambuco

Thyago Monteiro

Universidade de Pernambuco

Sérgio Fernandes

Universidade de Pernambuco

Edison Albuquerque

Universidade de Pernambuco

Washington Silva

Federal University of Pernambuco

Wellington Santos

Federal University of Pernambuco

Research Article

Keywords: Antivirus, Malware , Jar Files ,Dynamic Runtime Behaviors , Artificial Neural Network, Computer Forensics

Posted Date: September 2nd, 2021

DOI: <https://doi.org/10.21203/rs.3.rs-834072/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Version of Record: A version of this preprint was published at Scientific Reports on February 4th, 2022.

See the published version at <https://doi.org/10.1038/s41598-022-05921-5>.

Antivirus Applied to Jar Malware Detection based on Runtime Behaviors

Ricardo P. Pinheiro^a, Dr. Sidney M. L. Lima^{b,*1} (Professor), Danilo M. Souza^a, Sthéfano H. M. T. Silva^a, Petrônio G. Lopes^a, Rafael D. T. de Lima^a, Jemerson R. de Oliveira^a, Thyago de A. Monteiro^a, Sérgio M. M. Fernandes^a, Edison de Q. Albuquerque^a, Washington W. A. da Silva^c and Wellington P. dos Santos^c

^a{rpp3,smll,dms2,shmts,pgl,rdtl,jro,tam,smurilo,edison}@ecomppoli.br, Department of Computing, University of Pernambuco, Recife, Brazil

^bsidney.lima@ufpe.br, Electronics and Systems Department, Federal University of Pernambuco, Recife, Brazil

^c{washington.silva,wellington.silva}@ufpe.br, Biomedical Engineering Department, Federal University of Pernambuco, Recife, Brazil

ARTICLE INFO

Keywords:

Antivirus

Malware

Jar Files

Dynamic Runtime Behaviors

Artificial Neural Network

Computer Forensics

ABSTRACT

Background and Objective: Java vulnerabilities correspond to 91% of all exploits observed on the World Wide Web. Then, this present work aims to create an antivirus software with machine learning and artificial intelligence, master in Java malwares detection..

Methods: Within the proposed methodology, the suspect Jar sample is executed in order to infect, intentionally, Windows OS monitored in a controlled environment. In all, our antivirus monitors and ponders, statistically, 6,824 actions that the suspected Jar file can do when executed.

Results: Our antivirus achieves an average performance of 91.58% in the distinction between benign and malwares Jar files. Different initial conditions, learning functions and architectures of our antivirus are investigated in order to maximize their accuracy.

Conclusions: The limitations of commercial antiviruses can be supplied by intelligent antiviruses. Instead of blacklist-based models, our antivirus allows Jar malware detection in a preventive way and not in a reactive manner as Oracle's Java and traditional antivirus modus operandi.

1. Introduction

Nowadays, through the use of applications in social networks, global computer networks have been widely spread, and their applications are not only concerned with entertainment and leisure, but also within the work. Large parcel of the content provided by the global computer network is developed using Java technology. In the United States, Java applications are present in 97% and 89% of company computers and personal computers, respectively¹. From servers to personal computers to mobile and dedicated devices, Java applications have been used in various computers [23].

The goal of Java technology is to focus on the creation of applications that can be executed on various OS (operating systems) and computer-controlled devices. Unlike machine language instructions, Java applications are OS-independent. Then, Java applications are portable, so they can be executed in any OS that includes a JVM (Java Virtual Machine) without modification or extension. It should be emphasized that JVM has been implemented in thousands of OS. The conclusion is that Java applications have good portability. They can work on about 3 billion devices, including servers, personal computers, printers, smart phones, tablets, smart credit cards, household appliances, etc.².

*Corresponding author

sidney.lima@ufpe.br (S.M.L. Lima)

ORCID(s): 0000-0002-4350-9689 (S.M.L. Lima)

¹Java Technology. Available at: <https://www.java.com>. Accessed on June 2020.

²JavaOne 2012 Review: Make the Future Java. Available at: <http://www.oracle.com/technetwork/articles/Java/Javaone12review-1863742.html>. Accessed on June 2020.

Contrary to its popularity, incidents of targeted attacks using Java-based malware have been on the rise in recent years [4][7]. The reason is: between 2012 and 2013, the number of vulnerabilities exploiting Java increased by more than 300% [7]. Java vulnerabilities usually target personal computers technically named endpoints. Then, once on an endpoint, Java malware maliciously exploits daily activities, such as navigating the Internet or using utilities. Therefore, bank passwords, social networks, photos or private videos may be stolen due to cyber-infections.

One of the reasons for the failure of Java Security Manager is the retard in the catalog of new virtual plagues by its manufacturer. The modus operandi of Java Security Manager is mainly to identify malicious files based on signatures³. Like Oracle Java, commercial antivirus software can also identify malicious files with signature features. Then, this paper proposes to study (i) 86 kinds of commercial antivirus software to identify Java malware files, especially Jar files. Jar malware detection range is from 0% to 99.10%, depending on the antivirus software. On average, 34.95% of virtual plagues were detected.

In our paper, about 31.39% of antivirus software cannot diagnose any kind of malicious samples. It is worth pointing out that in our research, the malicious behavior of the Jar malware analyzed has been recorded by incident responders. Even so, more than one-third of evaluated commercial antivirus software is not aware of the existence of the Jar malware file under investigation. Note the impediment of conventional antiviruses as for the vigor of large-scale and

³Findar: jar files database. Available at: <http://findar.com/>. Accessed on June 2020.

real-time administrations.

The state-of-the-art antivirus recommends extracting features of suspicious files in a preventive manner before executing [10]. The executable file will be submitted to the reverse engineering process. Then, the assembly code associated with the executable file can be studied, so the malicious intent of the suspicious file can be investigated. The features from assembly code are used as input attributes of the artificial neural network used as the classifier. Neural network-based antivirus accomplishes an average performance of more than 90% in distinguishing benign and malware samples [10].

As a side effect, static analysis presents severe deficiencies when submitted to obfuscated malware, although several sophisticated static analysis techniques have been developed which show promising results [20]. As a digital anti-forensics strategy, malware employs runtime code packaging and obfuscation. Therefore, the original instructions, made by the server, are different from those executed at runtime on the personal computer [20]. It is concluded that the static features research can be easily bypassed by obfuscation strategies [20].

The constraint of the static feature methodology to precisely identify new types of malicious applications moved the focal point of malware research to the exploration of features that can recognize malicious behavior as a process, named dynamic approach, instead of a static approach. Then, our authorial antivirus performs the dynamic analysis of Java malwares. Our features extraction concerns the traces of calls made by all processes generated by the malware, files being created, deleted and downloaded by the malware while it is running, memory dumps of the malware processes, and tracking of network traffic in PCAP format. The main advantage of the dynamic approach is that behavioral features are insensitive to low-level (bytecode) mutation techniques, such as runtime packing or obfuscation [20]. To the whole, our dynamic extraction of features monitors 6824 behaviors that the suspect Jar file can make when executed.

Then, the malicious behaviors, originating from the suspect files, serve as input attributes of neural networks. In this work, we employ mELM (morphological ELM) neural network, an ELM with a hidden layer kernel, which is inspired by Erosion and Dilation image processing morphological operators. The proposed paper claims that the morphological kernel can fit any critical decision. Mathematical morphology refers to studying the shape of objects in an image by using the mathematical theory of intersection and union of the set [19]. Then, the morphological operation naturally processes the detection of the shape of the object present in the image [19]. Authorial antivirus achieves an average performance of 91.58% in distinguishing between benign and malware Java applications accompanied by an average training time of 52.36 seconds.

This work is organized as follows: in section 2 we present the limitations of commercial antiviruses. In section 3, we discuss the state-of-the-art regarding artificial intelligence antiviruses; in section 4, we present the proposed methodology; in section 5, we make a comparison between the au-

thorial ELM network and classic ELM networks; in section 6, we show the results and some discussions. Finally, in section 7, we make the general conclusions and discuss the perspectives of our work.

2. Commercial Antiviruses Limitation

Although it has been questioned for more than a decade, the modus operandi of antiviruses is based on signatures when the suspect file is consulted on datasets named blacklist. Therefore, it is enough that the hash of the investigated file not to be in the blacklist of the antivirus in order to malware not to be detected. The hash functions as a unique identifier for a given file.

Then, given the limitations of commercial antiviruses, it is not a difficult task to develop and to distribute variants of a malicious application. To do this, it is enough to make small alterations in the original malware with routines that, effectively, do not have any usefulness like repetition loops and conditional branches without instructions in their scopes. These alterations without usefulness, however, they turn the hash of the modified malware different from the hash of the original malware. Consequently, malware, augmented with null routines, will not be recognized by the antivirus that recognized the initial malware. It should emphasize the existence of botnets responsible for creating and distributing, in automated form, variants of the same original malware. It is concluded that antiviruses, based on signatures, have null effectiveness when submitted to variants of the same malware [17][9].

Through the VirusTotal platform, this proposed work explores 86 commercial antiviruses with their respective results presented in Table 1. We utilized 998 Jar malicious obtained from the dataset REJAFADA [20]. The objective of the work is to verify the amount of malicious samples cataloged by antivirus. The motivation is that the acquisition of new malwares is primordial in order to combat malicious activities.

Then, as larger is the dataset, named blacklist, better tends to be the defense given by the antivirus. Fig. 1 appears the diagram of the methodology proposed in a block diagram. At first, Jar malwares are sent to the server belonging to the VirusTotal platform. At this point, the Jar files are analyzed by VirusTotal's 86 commercial antiviruses. Then, the antiviruses give their diagnostics for Jar samples submitted to the server. VirusTotal allows the issuance of emission of three different types of diagnostics: benign, malware, and omission.

As for the first possibility of VirusTotal, the antivirus detects the maliciousness of the suspicious file. Within the proposed experimental environment, all submitted samples are malware documented by incident responders. The antivirus hits when it recognizes the malignity of the investigated file. Malware detection shows that the antivirus offers a robust support against digital invasions. In the second possibility, the antivirus certifies the benignity of the examined file. Then, in the proposed study, when the antivirus

claims the file is benign, it is a false negative, as all the samples sent are malicious. In other words, the investigated file is malware, however, the antivirus mistakenly attests to benignity. Within the third possibility, the antivirus does not give an analysis about the suspect application. The omission shows that the investigated file has never been evaluated by the antivirus so little it has the strength to evaluate it in real time. The omission of diagnosis by antivirus points to its limitation on large-scale services.

Table 1 shows the results of the 86 antivirus products evaluated. The McAfee-GW-Edition antivirus achieved the best performance by being able to detect 99.10% of the investigated malwares. One of the largest adversities in combating malicious applications is the fact that antivirus manufacturers do not share their malware blacklists due to commercial disputes. Through the analysis of Table 1, the proposed work points to an aggravating factor of this adversity: the same antivirus manufacturer does not share their databases among their different antiviruses. Observe, for example, that McAfee-GW-Edition and McAfee antiviruses belong to a same company. Their blacklists, though robust, are not shared amongst themselves. Therefore, the commercial strategies, of a same company, disturb the confrontation against malwares. It complements that antivirus manufacturers are not necessarily concerned in avoiding cyber-invasions, but in optimizing their business incomes.

Malware identification ranged from 0% to 99.10%, depending on the antivirus. Overall, the 86 antiviruses were able to identify 34.95% of the examined malwares, with a standard deviation of 40.92%. The elevated standard deviation shows that the recognition of malicious files can change abruptly depending on the chosen antivirus. The protection against digital intrusions is in the function of choosing a vigorous antivirus with an expansive and upgraded blacklist. Overall, antiviruses certified false negatives in 33.90% of the cases, with a standard deviation of 40.45%. To attest the benignity of malware can implicate in unrecoverable damages. A person or institution, for instance, would start to trust a certain malicious application when, in fact, it is a malware. Still as an unfavorable aspect, about 31.39% of antiviruses did not express an opinion on any of the 998 malicious samples. On average, the antiviruses were omitted in 31.15% of the cases, with a standard deviation of 45.61%. The omission of the diagnosis focuses on the constraint of antivirus in recognizing malware in real time.

As difficulty in the combat to malicious applications, the fact of the commercial antiviruses do not have a pattern in classification of malwares as found in Table 2. We have chosen 3 of the 998 Jar malwares in order to exemplify the miscellaneous of classifications given by antivirus commercial activities. As there is no pattern, the antiviruses give the names that they want, for example, a company can identify a Jar malware as "Android: RuFraud-I" and a second company identify it as "Artemis! 9EF6966B98A5". Therefore, the lack of a pattern disturbs the cyber-security strategies since each category of malware must have different treatments (vaccines). It is concluded that it is impracticable for

supervised machine learning to adopt pattern recognition as for categories of Jar malwares. Due to this confusing tangle of MultiClass Classification, provided by specialists (antiviruses) as seen in Table 2, it is statistically improbable that any machine learning technique will acquire generalization capability.

3. State-of-the-art

Jar files are collections of individually compressed class files. They are about half the size of the original class files [23]. A Java archive (JAR) file encapsulates Java classes and may also contain other resources such as a digital signature or pictures [14]. Jar file is primarily designed to provide a reliable environment for running small programs embedded in web pages known as applets [14].

The modus operandi of commercial antivirus software and Oracle Java (especially Java Security Manager) is mainly used to identify Jar malware with signature basis. The main problem with this strategy is: for a new virtual plague to assign, it must be detected that certain computers have been infected. Taking into account the limitations of commercial antivirus software and Java Security Manager, the latest technology proposes the function of extracting and analyzing files through a statistical learning machine. Artificial intelligence can automate many tasks by analyzing thousands of files, extracting their functions, and classifying them.

LIMA, *et al.* (2021) created an antivirus capable of detecting PE file (Windows) malwares with an average accuracy of 98.32% [10]. The executable file will be submitted to the disassembly process. Then, the executable file can be studied, so the malicious intent of the file can be investigated. Analysis of LIMA *et al.* (2021) extracted 630 features of each executable file. These features are the input neurons of the artificial neural network. The classification of neural networks aims to divide executable files of 32-bit architecture into two categories: benign and malicious software. Antivirus made by LIMA, *et al.* (2021) use shallow neural networks.

On the other side, deep nets-based antiviruses have also achieved excellent accuracies. VINAYAKUMAR, R. *et al.* (2019) achieve an average accuracy of 98.90% in order to detect PE files malwares [22]. The deep network structure has 34 layers⁴. The network is trained with 500 epochs with a training batch size of 64 and learning rate 0.01.

SU, J. *et al.* (2018) achieve an average accuracy of 94.00% in order to detect IoT (Internet of Things) malwares [21]. The deep network structure has 6 layers. There are 3 layers with learnable weights: 2 convolutional layers, and 1 fully connected layer. The network is trained with 5000 iterations with a training batch size of 32 and learning rate 0.0001.

The antivirus made by HARDY, W. *et al.* (2016) aim to detect PE files (Windows) malwares by employing Stacked Autoencoders deep networks [5]. The decoder attempts to

⁴DeepMalNet layers. Available at: <https://github.com/vinayakumarr/dnn-ember/blob/master/DNN-info.pdf>. Accessed on February 2020.

Table 1

Results of commercial antiviruses. Expanded results of 86 worldwide commercial antiviruses are in the authorial repository [16].

Antivirus	Detection (%)	False Negative (%)	Omission (%)
McAfee-GW-Edition	99.10	0.90	0.00
NANO-Antivirus	97.70	2.20	0.10
AegisLab	97.60	2.10	0.30
Kaspersky	96.80	2.90	0.30
ZoneAlarm	96.70	2.90	0.40
Avast	96.60	3.30	0.10
AVG	96.60	3.30	0.10
ESET-NOD32	95.90	4.10	0.00
McAfee	95.60	4.40	0.00
Avira	94.80	3.30	1.90
Cylance	0.20	0.00	99.80
WhiteArmor	0.20	91.30	8.50
Alibaba	0.20	98.50	1.30
ALYac	0.10	95.80	4.10
Bkav	0.10	97.60	2.30
Paloalto	0.00	0.00	100.00
SentinelOne	0.00	0.00	100.00
Endgame	0.00	0.00	100.00
CrowdStrike	0.00	0.00	100.00
Agnitum	0.00	0.00	100.00

Table 2

Result of the submission of three malware to VirusTotal. Expanded results of 86 worldwide commercial antiviruses are in the authorial repository [16].

Antivirus	VirusShare _A	VirusShare _B	VirusShare _C
McAfee-GW-Edition	Artemis!Trojan	Artemis	PWS-Zbot.gen.jr
NANO-Antivirus	Trojan.Android.SMSSend.numyx	Trojan.Android.Opfake.oefcg	Trojan.Java.CVE20113544.cspflc
AegisLab	Troj.Sms.Androidos!c	SUSPICIOUS	Troj.W32.Generic!c
Kaspersky	HEUR:Trojan-SMS.	HEUR:Trojan-SMS.	HEUR:Trojan.
	AndroidOS.Fakelogo.a	AndroidOS.Fakelogo.a	Win32.Generic
ZoneAlarm	HEUR:Trojan-SMS.	HEUR:Trojan-SMS.	HEUR:Trojan.
	AndroidOS.Fakelogo.a	AndroidOS.Fakelogo.a	Win32.Generic
Avast	Android:RuFraud-I	Android:RuFraud-I	Java:CVE-2011-3544-BD
AVG	Android:RuFraud-I	Android:RuFraud-I	Java:CVE-2011-3544-BD
ESET-NOD32	Android/TrojanSMS.Agent.K	Android/TrojanSMS.Agent.K	a variant of Java/ Exploit.CVE-2011-3544.DF
McAfee	Artemis!9EF6966B98A5	Artemis!BEE5A7C75B6A	RDN/Generic
Avira	ANDROID/SmsAgent.CQ.Gen	ANDROID/SmsAgent.CQ.Gen	EXP/CVE-2011-3544
Sophos	Andr/Jifake-B	Andr/Opfake-A	Mal/Generic-S
Symantec	Android.Fakemini	Android.Fakemini	Trojan.MalJava
IkarusV	Trojan.AndroidOS.FakeInst	Trojan.AndroidOS.FakeInst	Java.CVE
MAX	Malware	malware	Malware
TrendMicro-HouseCall	Suspicious_GEN.F47V0322	AndroidOS_OPFAKE.A,	Suspicious_GEN.F47V0322
Emsisoft	Android.Trojan.FakeInst.CB	Android.Trojan.FakeInst.CB	Gen:Variant.Barys.841
GData	Android.Trojan.FakeInst.CB	Android.Trojan.FakeInst.CB	Gen:Variant.Barys.841
BitDefender	Android.Trojan.FakeInst.CB	Android.Trojan.FakeInst.CB	Gen:Variant.Barys.841
Tencent	Trojan.Android.FakeLogo.aa	Trojan.Android.FakeLogo.aa	Win32.Trojan.Jorik.Hvje
Arcabit	Android.Trojan.FakeInst.CB	Android.Trojan.FakeInst.CB	Trojan.Barys.841

map this representation back to the original input. The deep learning model is trained with 3 hidden layers and 100 neurons at each hidden layer. The encoder maps the input to a hidden representation. HARDY, W. *et al.* (2016) achieve an average accuracy of 96.85%.

The disadvantage of the deep net is the long training

time. As an aggravating factor, deep networks have lower parallel capabilities because these layers are sequential. Therefore, this layer can only be executed after the upper layer has completed its work. In applications that require frequent training (learning) of antivirus software, this fact may become an obstacle, because on average 8 (eight) new mal-

wares are created every second [8]. In syntheses, there should be no difference in the learning time of antivirus software compared with the rate of new malware generation worldwide.

Deep learning made by SANTOS, *et al.* (2019) has training time compatible with applications that often require training. The deep network presents a single convolutional layer, and there is no data backpropagation. Therefore, since the supercomputer has sufficient computing resources (memory), deep learning made by SANTOS, *et al.* (2019) has major parallel capabilities. The convolution layer uses 30,000 convolution filters at the same time.

Instead of random convolutional filters, Deep Learning made by SANTOS, *et al.* (2019) develops PCA⁵. filters aimed at extracting the main components of the regions of interest related to the data input vector. Altogether, the technique employs 30 thousand convolutional filters. Each filter is converted to a Toeplitz matrix denoted by $W_{detect} \in \mathbb{R}^{L^2 \times R^2}$, with $L = (J + W - 1)$ for a $L \times L$ size feature map. W and J refer to filter size and pixels, respectively [18].

Herein this work we replicate the deep net made by SANTOS, *et al.* (2019) because its training time is compatible with applications that need training frequently as antivirus. Deep Learning made by SANTOS, *et al.* (2019) is not aimed at the detection of malware, but at the optical character recognition [18]. Instead of digital image processing, the input attributes concern the extraction of malware features.

Due to the excellent results obtained by Deep Learning techniques, a common sense has been created that Deep Learning is capable of providing the best accuracy in any type of application, in fact, this consideration is untrue. Deep neural networks, specifically, convolutional networks are based on linear filter convolution. Although it has a fundamental role in computer applications, filter convolution is limited to applications when vector flow gradient is formed.

Consider, for example, biomedical images from mammography devices. The images are full of noises that hinder the recognition of the breast lesion [13]. Therefore, convolution of filters is essential in order to eliminate noise and, therefore, discard small irregularities in the finding corresponding to potential cancer. Convolutional techniques, such as Gaussian filters, are essential in order to reduce noise in biomedical images [13].

As a counter-example, consider the repository illustrated in Table 3. The features are completely disconnected from each other despite belonging to the same neighborhood. An application suspected of trying to check Wi-fi data has no correlation with accessing the victim's image gallery or browser. Then, when applying the linear convolution of filters in the repository, illustrated in Table 3, accessing the browser, containing the value 0, would be treated as noise. The explanation is that its neighborhood has positive values. In synthesis, the suspect application would be accused of accessing the victim's browser even the extraction of features having audited the inverse. Then, convolutional tech-

niques suffer a disadvantage when applied to malware pattern recognition.

In order to prove our theoretical background, the authorial antivirus employs shallow morphological neural networks instead of deep convolutional networks. As experiments, the authorial antivirus has its accuracy compared to next-generation antiviruses based on both shallow and deep neural networks. Our antivirus can combine high precision with reduced learning time. In order to avoid unfair comparisons, the feature extraction stage is standardized by monitoring 6,824 behaviors that the suspicious Jar file can do when executed purposely.

Table 3

Example of a statistical repository based on malware detection.

Features		
Check Wi-fi data	Access the Browser	Access Image Gallery
1	0	1

4. Materials and Methods

The present paper aims to elaborate the REJAFADA (Retrieval of Jar Files Applied to Dynamic Analysis), a dataset which allows the classification of files with Jar extension between benign and malwares. REJAFADA consists of 998 malware Jar samples and 998 other benign Jar samples. REJAFADA dataset, consequently, is suitable for learning endowed with AI (Artificial Intelligence), considering that the Jar files presented the same amount in the different classes (malware and benign). The goal is that classifiers that are biased towards a particular class do not have their success rates favored.

In relation to virtual plagues, REJAFADA extracted malicious Jar files from VirusShare which is a repository of malware samples to provide security researchers, incident responders, forensic analysts, and the morbidly curious access to samples of live malicious code⁶. With respect to benign Jar files, the catalog was given from application repositories such as Java2s.com⁷, and findar.com⁸. All of the benign files have been audited by VirusTotal. Then, the benign Jar files, contained in REJAFADA, had their benevolence attested by the main commercial antiviruses of the world. The obtained results corresponding to the analyses of the benign and malware Jar files, resulting from the VirusTotal audit, are available for consultation at the virtual address of REJAFADA [16]. The purpose of creating the REJAFADA dataset is to give full possibility for the proposed methodology to be replicated by others in future work. Then, RE-

⁶VirusShare: malware files database. Available at: <https://virusshare.com>. Accessed on June 2020.

⁷Java2s: jar files database. Available at: <http://Java2s.com/>. Accessed on June 2020.

⁸Findar: jar files database. Available at: <http://findar.com/>. Accessed on June 2020

⁵PCA: Principal Component Analysis.

JAFADA makes available, freely, of all their samples such as benign as malwares:

- VirusTotal audits;
- Dynamic analysis of Cuckoo Sandbox.

REJAFADA also makes available in its virtual address, its 998 benign Jar files. In addition, our dataset displays the list of all other 998 Jar files, this time, malwares. Then, there is the possibility of acquiring all the malware employed by REJAFADA by establishing agreements and submitting to ViruShare's rules of use [16]. We conclude that our REJAFADA dataset provides transparency and impartiality to the research, and demonstrates the veracity of our results. Therefore, it is expected that REJAFADA will serve as a basis for the creation of new scientific works aiming at Next-Generation antivirus.

All experiments were carried out on a cloud supercomputer equipped with 250 GB of RAM, 8 processors and 300 GB of mass storage. So that there are no unfair comparisons, state-of-the-art antiviruses are trained and tested on the same supercomputer used by the copyrighted antivirus. It is emphasized that the acquisition of a supercomputer was due to replication and comparison with state-of-the-art works. Authorial antivirus requires low processing and storage capacity. It is emphasized that the authorial antivirus could be used on any conventional desktop computer.

5. Proposed Methodology

Fig. 1 shows the diagram of the methodology proposed in the block diagram. Initially, the Jar file, originating from the REJAFADA dataset, is executed in order to verify the attempt to corrupt the JVM and, in sequence, Windows 7 audited by the Cuckoo Sandbox. The dynamic features are synthesised in section 5.1. Then, the dynamic characteristics of Jar files are stored in the format of machine learning repository.

As a method of feature mining, some behaviors, audited by the Sandbox, are despised. The adopted criterion of mining refers to features elimination which concern a single Jar file, for example process ids, process names, md5, sha, among others. After mining features, the relevant behaviors of the Jar samples serve as machine learning input attributes, specifically, the artificial neural networks employed as classifiers. The goal is to group the Jar samples into two classes: benign and malware. The classification stage is explained in detail in section 5.2. Classification results are described in Chapter 6.

As for materials, all experiments were performed on a cloud supercomputer with 250 GB of RAM, 8 processors and 300 GB of mass storage. In order to avoid unfair comparisons, the state-of-the-art antiviruses are trained and tested on the same supercomputer used by the authorial antivirus. It is emphasized that the acquisition of a supercomputer was due to the replica and the comparison with the state-of-the-art works. The authorial antivirus requires low processing

and storage capacity. It is emphasized that the authorial antivirus could be used in any conventional desktop computer.

5.1. Dynamic Feature Extraction

The features of Jar files originate through the dynamic analysis of suspicious files. Therefore, in our methodology, the malware is executed in order to infect, intentionally, the JVM installed in Windows 7 audited, in real time (dynamic), by the Cuckoo Sandbox⁹. In total, 6824 features are generated of each Jar file, regarding the monitoring of the suspect file in the proposed controlled environment. Next, the groups of features are detailed:

- Features related to virtual machines. The goal is to verify that the audited file searches for detect whether Bochs, Sandboxie, VirtualBox, VirtualPC, VMware, Xen or Parallels virtual machines are being used through the presence of registry keys (regedit), files, instructions, and device drivers used by them;
- Features related to malware. Checks whether the audited file tries create Mutexes (single name files, with a function to set a lock / unlock state, which ensures that only one process at a time uses the resources);
- Features related to Bitcoin. Examines whether the tested file attempts to install the OpenCL library, Bitcoins mining tool.
- Features related to bots (machines that perform automatic network tasks, malicious or not, without the knowledge of their owners):
- Features related to browsers. Checks if the file tried tries:
 - Modify browser security settings;
 - Modify the browser home page;
 - Acquire private information from locally installed internet browsers.
- Features related to Firewall. The proposed digital forensics audits if the file tries to modify local firewall policies and settings.
- Features related to cloud computing. The file is audited when it tries to connect to storage services and / or files from Dropbox, Google, MediaFire, MegaUpload, RapidShare, Cloudflare and Wetransfer.
- Features that seek to disable features of Windows 7 OS and other utilities. The audit checks to see if the file tries to:
 - Modify system policies to prevent the launch of specific applications or executables;
 - Disable browser security warnings;

⁹Cuckoo: Automated Malware Analysis. Available at: <https://cuckoosandbox.org/>. Accessed on Nov. 2020.

Figure 1: Diagram of the proposed methodology.

- Disable Windows security features and properties;
- Features associated with network traffic hint Windows 7 OS in PCAP format. Explicar.
- Features related to Windows 7 OS (Regedit):
 - Changes in associations between file extensions and software installed on the machine;
 - Changes to the current user information;
 - Driver corruption;
 - Changes to the Windows appearance settings and settings made by users, such as wallpaper, screen-saver, and themes;
 - Changes to Hardware Settings.
- Features related to the use of sandboxes. The digital forensics examines whether the file tries to turn off Windows functions monitored by the cuckoo sandbox.
- Features related to Ransomware (a type of malware that, by means of encryption, leaves the victim's files unusable, then request a redemption in exchange for the normal use later of the user's files, a redemption usually paid in a non-traceable way, such as bitcoins)
- Features related to exploit-related features which constitute malware attempting to exploit known or unacknowledged vulnerabilities, faults or defects in the system or one or more of its components in order to cause unforeseen instabilities and behavior on both your hardware and in your software.

- Features related to Infostealers, malicious programs that collect confidential information from the affected computer.

Besides the detection of suspicious behaviors, such as API calls, dynamic analysis also allows the reconstitution (cleaning) of the OS (Operating System) by auditing of the malefactions promoted by the malicious file in the OS assuming that the harm is not statistically irreversible. It should be noted that the reconstitution of OS, technically named vaccine, is important because it is not enough to detect and eliminate the malware in order to the victim becoming free of its actions. Besides the elimination of malware, it is necessary to undo all of its malfeasances as, for instance, having disabled Java Security Manager from the JVM. Then, if there was no auditing provided by the dynamic analysis, it would fall to the cyber-watcher to monitor, manually, any OS changes that would turn the slow and stressful process.

5.2. Classifiers

Our antivirus employs artificial neural networks as classifiers. In order to choose the best setting of the neural network architecture, we employ different learning functions and initial configurations to require a larger volume of calculations, such as multiplying the quantity of neurons in the intermediary layer. The architectures of the neural networks have an input layer containing a number of neurons regarding the vector of extracted features from the Jar file monitoring in a controlled environment. Therefore, the employed classifiers must have an input layer containing 6,824 neurons. They concern the features of auditing of the Jar file. The output layer has two neurons, corresponding to benign and malware samples.

The proposed work resulted in an antivirus composed of ELMs (Extreme Learning Machines) neural networks in order to detect malwares preventively. ELMs are powerful and flexible kernel-based learning machines whose main characteristics are fast training and robust classification performance [6]. The ELM network is a single hidden layer network, not recurrent, based on an analytical method to estimate the network output weights, in any random initialization of input weights.

The ELMs have been widely applied in several areas such as Biomedical Engineering [13][12][11] [15][1][2][3]. ELMs networks can greatly contribute to the advancement of digital security of devices. The proposed paper applies the ELMs in the area of information security specifically in the recognition of malware patterns.

Mathematically, in ELM neural network the input attributes x_{ik} correspond to the set $\{x_{it} \in \mathbb{R}; i = 1, \dots, n; t = 1, \dots, v\}$. Therefore, there are n features extracted from the application and v training data vectors. The hidden layer h_j , consisting of m neurons, is represented by the set $\{h_j \in \mathbb{R}; j \in \mathbb{N}^*; j = 1, \dots, m\}$. The ELM training process is fast because it is composed of only a few steps. Initially, the input weights w_{ji} and bias b_{jt} are defined in a random generation. Given an activation function $f: \mathbb{R} \rightarrow \mathbb{R}$, the learning process is divided into three steps:

- Random generation of weight w_{ji} , corresponding to the weights between the input and the hidden layers, and bias b_{jt} .
- Calculate the matrix H, which corresponds to the output of the neurons of the hidden layer.
- Calculate the matrix of the output weights $= H^\dagger Y$, where $H^\dagger$ is the generalized Moore-Penrose inverse matrix of the matrix H, and Y corresponds to the matrix of desired outputs s.

The output of the hidden layer neurons, corresponding to the matrix H, is computed by the kernel K , dataset inputs and weights between the input and the hidden layers shown in Eq. (1).

$$H_{jt} = \begin{bmatrix} K(1,1) & K(1,2) & \dots & K(1,n) \\ K(2,1) & K(2,2) & \dots & K(2,n) \\ \vdots & \vdots & \ddots & \vdots \\ K(v,1) & K(v,2) & \dots & K(v,n) \end{bmatrix} \quad (1)$$

Instead of using conventional kernels, authoring kernels will be used for ELMs. We employ mELMs (morphological ELMs), ELMs with hidden layer cores based on the morphological operators of Erosion and Dilation image processing. Kernels are mathematical functions employed as a method for learning neural networks. This learning method enables the creation of non-linear data mapping. Thus, there is no need to increase the number of adjustable parameters, as in the learning rate used in networks with backward propagation. There are two fundamental morphological operations,

Erosion and Dilation. The theory of Mathematical Morphology can be considered constructive, since all operations are built based on Erosion and Dilation. Mathematically, Erosion and Dilation are characterized according to Eq. (2) and Eq. (3), respectively:

$$\epsilon_g(f)(u) = \bigcap_{v \in S} f(v) \vee \bar{g}(u - v) \quad (2)$$

$$\delta_g(f)(u) = \bigcup_{v \in S} f(v) \wedge g(u - v) \quad (3)$$

Where $f : S \rightarrow [0, 1]$ and $g : S \rightarrow [0, 1]$ normalized images in the form of a matrix named S format, where $S \in \mathbb{N}^2$. pixel is defined by the Cartesian pair $(u, f(u))$, where u is the position related to value $f(u)$. v is the matrix of $f(u)$, encompassed by g . The operators $\vee$ and $\wedge$ are associated with the maximum operation, while $\bar{g}$ and g are associated with the minimum operation. g is the structuring element for both Erosion and Dilation [19]. $\bar{g}$ is the negation of g .

In Eq. (2) initially occurs the negation of the structuring element $\bar{g}$. Then, it happens the operation of maximum $\vee$ denoted by $f(v) \vee \bar{g}(u - v)$, where $f(v)$ refers to the original image matrix f covered (matched) by $\bar{g}$. $f(v)$ is technically named the active region of the image. Finally, the value $\epsilon_g(f)(u)$, in the position u , of the eroded image receives the minimum value between the maximums, via the operator. $\epsilon_g(f)(u)$ gets the value 0 associated with absolute black. Erosion overlays $\bar{g}$ to the original image f . The goal is that areas similar to $\bar{g}$ expand [19]. By associating 1's with absolute white and 0's with absolute black, Erosion enhances darker areas and eliminates regions with high intensity [19].

Eq. (3) shows the performance of the morphological dilation operation. Due to mathematical precedence, the minimum $\wedge$ operation denoted by $f(v) \wedge g(u - v)$, occurs, where $f(v)$ refers to the original image matrix f covered (matched) by g . Therefore, the value $\delta_g(f)(u)$, at the u position of the expanded image receives the maximum value between the minimums through the $\cup$ operator. Dilatation superimposes the structuring element g on the original image f . The goal is that areas similar to g expand. By associating 1's with absolute white and 0's with absolute black, the dilation increases the areas with more intense tonality and eliminates the dark regions [19].

Our antivirus utilizes mELMs (Morphological Extreme Learning Machines). They are inspired by Mathematical Morphology based on nonlinear Erosion and Dilation operators. As indicated by Eq. (2) concerning the Erosion image operator, the Erosion ELM kernel can be defined together with Eq. (4), where $\{i \in \mathbb{N}^*, i = 1, \dots, n\}$, $\{j \in \mathbb{N}^*, j = 1, \dots, m\}$, $\{t \in \mathbb{N}^*, t = 1, \dots, v\}$. So there are n neurons in the entry layer (without the bias), m neurons in the hidden layer, and v training data vectors.

$$K_e(t,i) = (x_{it} \vee \bar{w}_{ji}) + b_{jt} \quad (4)$$

Figure 2: (a) Successful performance of the kernel compatible with dataset. (b) Inaccurate classification of the Linear kernel in a non-linearly separable distribution. (c - d) Successful performances by Dilation and Erosion kernels.

Similar to the Erosion kernel, Eq. (5) defines the Dilation kernel inspired by Eq. (3) and refers to morphological operator of Dilation.

$$K_{\delta}(t,i) = (x_{it} \wedge w_{ji}) + b_{ji} \quad (5)$$

In order to achieve good performance in ELMs, it is necessary to choose a kernel that is able to optimize the decision boundary for the presented problem as seen in Figure 2 (a). A Linear kernel gets great results when used to solve a linearly separable problem. However, when used to solve non-linearly separable problems as shown in Figure 2 (b) for a sigmoid distribution, it does not perform satisfactorily. A great generalization capability of the neural network may depend on a fine-tuned kernel choice. The best kernel may be subordinate to the problem to be resolved.

As a side impact, investigating different kernels is often a stressful affair that involves cross-validation combined with different random initial conditions. However, the investigation of distinct kernels may be necessary, otherwise the neural network composed, by a mismatched kernel, for generating unsatisfactory results.

Figure 2 (c), Figure 2 (d) show the performance of the

mELM kernel Erosion and Dilation, with the respective accuracies of 93.07% and 95.05%. It is possible to notice when analyzing the figures that the mELMs have the capacity to accurately map the different distributions referring to different problems.

The effectiveness of our morphological neural networks is due to their ability to adapt to any type of distribution, since their mapping does not obey any conventional geometric figure. The mapping of decision borders is made by their own training data, the very position in n-dimensional space that will determine whether that surrounding region belongs to class 1 or class 2. The n represents the number of neurons in the input layer. Therefore, our mELM kernel is able to naturally detect and model the n-dimensional regions divided into different classes by using Math Morphology.

6. Results of ELM Networks

We employ seven different kernel types for the ELMs neural networks. In the state-of-the-art, three of these kernels are described by HUANG, et al, (2012), they are; Wavelets Transform, Hard Limit and Tribas (Triangular Base Function) [6]. In addition, authorial kernels are employed: Fuzzy-

Dilation, Fuzzy-Erosion, Dilation and Erosion.

The fmELMs have been successful in the treatment of biomedical images, specifically in the detection and classification of breast cancer [1][2]. The fmELMs constitute linear functions, inspired by the Mathematical Morphology, with optimized learning time compared to the mELMs. Despite the low learning time, fmELMs are not completely suitable for non-linear distributions such as Sigmoid and Sinusoidal distribution.

The Wavelets kernel has no hidden layer [6]. The calculations are based on the transformation of the input data and can work nearly to kernels containing architectures with hidden layers [6]. A good generalization capability of these kernels depends on an adjusted choice of parameters (C, γ) [6]. The cost parameter C refers to a reasonable equilibrium point between the hyperplane margin width and the classification error minimization in relation to the training set. The kernel parameter γ controls the decision boundary in function of classes [6]. There is no universal method in the sense of choosing the parameters (C, γ).

In the present work, there is the investigation of the parameters (C, γ) inspired by the method proposed by HUANG, et al (2012), which consists of training increasing sequences of C and γ , mathematically, 2^n , where $n = \{-24, -10, 0, 10, 25\}$ [6]. The hypothesis is to verify if these parameters with values different from the standards; ($C = 1, \gamma = 1$), generate better results. In the Linear kernel, there is only the investigation of the cost parameter C , it is not possible to explore the kernel parameter γ [6].

Each combination employs cross-validation through the k-fold method, where $k=10$. The goal is that the results achieved do not be influenced by training and test sets. For this, the total of images is divided into ten parts. In the first iteration, the first part is destined to the test set, while the others are reserved for the training. This rotation happens for ten cycles until all ten parts have been applied to the test phase. The accuracy of the ELM is the arithmetic mean of the hit rate obtained in the ten cycles. As previously mentioned, in the ELM network there is no backpropagation of data. Therefore, the objective of the k-fold cross-validation method is not to establish a stopping criterion to avoid overfitting (excess training), but to verify that the classifier undergoes abrupt changes in its accuracy depending on the sets destined for training and testing. There is moreover the methodological care to randomly select benign and malwares samples for each fold. The objective is that biased classifiers, in relation to a given class, do not have their accuracies rates favored.

Table 4 details the results obtained by the ELMs neural networks with Wavelets kernel. Each row in this table contains 10 executions referent to cross-validation of the k-fold method, where $k=10$. In relation to precision in the test phase, the maximum average performance was 56.01% in the distinction between benign and malware cases with the parameters (C, γ) = ($2^{-24}, 2^0$). In Table 4, there are only the best and worst case descriptions, in this order, for each ELM kernel.

The Hard Limit, Tribas (Triangular Basis Function), Fuzzy-Dilation, and Fuzzy-Erosion, Dilation and Erosion kernels employ hidden-layer architectures. At this point, there is the investigation regarding the number of neurons in the hidden layer of these kernels. The hypothesis is to verify whether architectures that require a higher volume of calculations, such as multiplying the number of neurons in the hidden layer, are able to produce better accuracies rates compared to architectures that demand less calculations. There is the evaluation of two architectures; they employ 100 and 500 neurons in their respective hidden layers. These architectures have a background of excellent accuracy in the application of ELM networks in the area of Biomedical Engineering [10].

Table 5 details the results obtained by the ELMs neural networks with the Hard Limit, Tribas (Triangular Basis Function), Fuzzy-Dilation, and Fuzzy-Erosion, Dilation and Erosion kernels. Each row in Table 5 contains 10 distinct executions referring to the k-fold method, where $k=10$. Regarding precision, the maximum average performance was 91.58% with standard deviation of 1.77% through the Dilation kernel endowed with 500 neurons in its hidden layer.

7. Results in relation to the State-of-the-Art

In this section authorial antivirus is compared with state-of-the-art antiviruses. In order to avoid unfair comparisons, the feature extraction stage is standardized by monitoring 6,824 behaviors that the suspicious Jar file can do when executed purposely. Authorial antivirus employs shallow morphological neural networks. Our antivirus is endowed with mELM Dilation kernel and contains 500 neurons in its hidden layer.

On the other hand, the antivirus made by LIMA, et al. (2021) employ shallow neural networks based on backpropagation. LIMA, et al. (2021) investigate eleven distinct learning functions in order to optimize the accuracy of their antivirus. For each learning function, LIMA, et al. (2021) explore 4 hidden layer architectures.

Our authorial antivirus is also compared to antivirus based on deep neural networks. The antivirus made by VINAYAKUMAR, R. et al. (2019) is replicated, specifically, its deep network structure containing 34 sequential convolutional layers¹⁰. The network made by VINAYAKUMAR, R. et al. (2019) is trained with 500 epochs with a training batch size of 64 and learning rate of 0.01.

Finally, the authorial antivirus is also compared to Deep Learning made by SANTOS, et al. (2019). This state-of-the-art work does not aim at malware detection, but at optical character recognition. Therefore, the deep net made by SANTOS, et al. (2019) undergoes an adaptation in its input neuron layer. Instead of digital image processing, the input attributes concern the extraction of malware features. The Deep Learning made by SANTOS, et al. (2019) presents a single convolutional layer and there is no data backpropaga-

¹⁰DeepMalNet layers. Available at: <https://github.com/vinayakumarr/dnn-ember/blob/master/DNN-info.pdf>. Accessed on February 2020.

Table 4

Result of ELM Networks. The (C, γ) parameters vary according to the set $\{2^{-24}, 2^{-10}, 2^0, 2^{10}, 2^{25}\}$. There are only the best and worst case descriptions.

kernel	(C, γ)	Train rate (%)	Test rate (%)	Train time (sec.)	Test time (sec.)
Wavelets	$(2^{-24}, 2^0)$	100.00 $\pm$ 0.00	56.01 $\pm$ 2.78	3.11 $\pm$ 0.07	0.76 $\pm$ 0.03
	$(2^{10}, 2^{25})$	67.40 $\pm$ 1.97	47.91 $\pm$ 3.76	3.12 $\pm$ 0.08	0.78 $\pm$ 0.03

Table 5

Result of ELM Networks. The number of neurons in the hidden layer varies according to the set 100, 500.

kernel	neurons	Train rate (%)	Test rate (%)	Train time (sec.)	Test time (sec.)
Hard limit	100	50.03 $\pm$ 0.00	49.75 $\pm$ 0.00	0.48 $\pm$ 0.01	0.03 $\pm$ 0.01
	500	50.03 $\pm$ 0.00	49.75 $\pm$ 0.00	2.51 $\pm$ 0.03	0.13 $\pm$ 0.02
Tribas	100	50.00 $\pm$ 0.03	50.00 $\pm$ 0.26	0.49 $\pm$ 0.02	0.02 $\pm$ 0.01
	500	50.00 $\pm$ 0.03	50.00 $\pm$ 0.26	1.76 $\pm$ 0.05	0.14 $\pm$ 0.01
Fuzzy-Dilation	500	95.71 $\pm$ 0.28	87.28 $\pm$ 2.40	2.01 $\pm$ 0.05	0.14 $\pm$ 0.02
	100	84.37 $\pm$ 0.29	81.61 $\pm$ 2.32	0.55 $\pm$ 0.02	0.03 $\pm$ 0.01
Fuzzy-Erosion	500	95.70 $\pm$ 0.33	87.72 $\pm$ 2.55	2.16 $\pm$ 0.07	0.16 $\pm$ 0.01
	100	84.67 $\pm$ 0.37	82.16 $\pm$ 1.93	0.65 $\pm$ 0.01	0.03 $\pm$ 0.01
Dilation	500	97.63 $\pm$ 0.13	91.58 $\pm$ 1.77	52.36 $\pm$ 0.57	5.68 $\pm$ 0.05
	100	81.20 $\pm$ 0.35	78.86 $\pm$ 1.18	7.34 $\pm$ 0.18	0.77 $\pm$ 0.02
Erosion	500	78.38 $\pm$ 0.24	70.58 $\pm$ 2.81	53.23 $\pm$ 2.41	5.78 $\pm$ 0.23
	100	54.99 $\pm$ 0.11	53.05 $\pm$ 1.95	8.17 $\pm$ 0.15	0.86 $\pm$ 0.03

tion. The convolutional layer employs all of its 30,000 filters simultaneously so that the training time is compatible with applications that need training frequently as antivirus.

Figure 3 and Figure 4 are graphical representations of the results described in Table 6. Figure 3 (a) presents the boxplots, from the training stage, in relation to authorial antivirus and state-of-the-art. The best average accuracy, resulting from the training, was 98.33% through antivirus made by VINAYAKUMAR, R. *et al.* (2019). Deep Learning made by SANTOS, *et al.* (2019) obtains a training accuracy of 50% on average. The antivirus made by LIMA, *et al.* (2021) obtained average accuracy of 48.82% and 98.82%, in its worst and best scenarios, respectively. These results were obtained using the learning functions "Batch training-learning rules" and "Conjugate gradient backpropagation with Fletcher-Reeves updates", respectively, with 100 neurons in their hidden layers. The authorial antivirus obtained an average performance of 97.63% with a standard deviation of 0.13%. Therefore, our antivirus has the advantage of not suffering abrupt changes due to the initial conditions (synaptic connections and cross validation).

Figure 3 (b) shows the boxplots for the best accuracy in the test phase. The best average accuracy, resulting from the test, was 96.54% through the antivirus made by VINAYAKUMAR, R. *et al.* (2019). Deep Learning made by SANTOS, *et al.* (2019) achieved an average performance of 50%. The authorial antivirus obtained an average accuracy of 91.58%. The antivirus made by LIMA, *et al.* (2021) achieved a mean accuracy of 50.26% and 95.67%, in its worst and best scenarios, respectively. Therefore, it is corroborated that neural networks based on backpropagation can suf-

fer major variations, in their accuracies, depending on their configuration parameters. Then, the decision made by LIMA, *et al.* (2021) was salutary. This state-of-the-art antivirus explores different learning functions, gradients and architectures in order to optimize the accuracy of its neural networks based on data backpropagation.

Figure 4 (a) and Figure 4 (b) present the boxplots referring to the times spent during the training and test phases, respectively. Authorial antivirus consumes only 52.36 seconds in order to conclude, on average, its training. In relation to training time, the antivirus made by VINAYAKUMAR, R. *et al.* (2019) is slower since there is the use of deep network recurrent structure. The antivirus made by VINAYAKUMAR, R. *et al.* (2019) consumes about two (2) days in order to complete its training. The Deep Learning made by SANTOS, *et al.* (2019) consumes 695.26 seconds in its training since 30,000 convolutional filters are employed in parallel and there is no data backpropagation. Antivirus made by LIMA, *et al.* (2021) concludes its training in order of seconds because they employ shallow neural networks. Regarding the time consumed during the test phase, all techniques consumed very close times without great discrepancies.

Table 7 shows the confusion matrices of the techniques presented in Table 6 in percentage terms. The confusion matrix is important in order to verify the quality of supervised learning. In Table 7, B. and M. are abbreviations of Benign and Malware. The desired classes are arranged on the vertical label while the obtained classes are on the horizontal label. On a confusion matrix, the main diagonal is occupied by cases whenever the obtained class coincides with expected class, named true positive cases. Then, a good classifier has

Table 6

Comparison among the authorial antivirus and the state-of-the-art.

Technique	Train rate (%)	Test rate (%)	Train time (sec.)	Test time (sec.)
Authorial Antivirus	97.63 $\pm$ 0.13	91.58 $\pm$ 1.77	52.36 $\pm$ 0.57	5.68 $\pm$ 0.05
Antivirus made by LIMA, <i>et al.</i> (2021), worst c. [10]	50.26 $\pm$ 0.89	50.26 $\pm$ 0.71	9.67 $\pm$ 1.76	0.44 $\pm$ 0.14
Antivirus made by LIMA, <i>et al.</i> (2021), best c. [10]	96.71 $\pm$ 2.10	95.67 $\pm$ 1.85	580.07 $\pm$ 228.44	0.46 $\pm$ 0.19
Antivirus made by SU, <i>et al.</i> (2018) [21]	78.37 $\pm$ 0.47	78.31 $\pm$ 3.37	3337.15 $\pm$ 19.08	3.53 $\pm$ 0.14
Antivirus made by VINAYAKUMAR, R. <i>et al.</i> (2019) [22]	98.33 $\pm$ 5.16	96.54 $\pm$ 4.84	149968.09 $\pm$ 33112.72	71.64 $\pm$ 23.13
Antivirus made by HARDY, <i>et al.</i> (2016) [5]	99.57 $\pm$ 1.17	96.49 $\pm$ 1.89	6854.74 $\pm$ 300.59	0.14 $\pm$ 0.01
Deep Learning made by SANTOS, <i>et al.</i> (2019) [18]	50.00 $\pm$ 0.00	50.00 $\pm$ 0.00	695.26 $\pm$ 29.67	7.90 $\pm$ 2.56

the main diagonal occupied by high values and other elements have low values.

Table 7 shows main diagonals emphasized in bold. Our antivirus, in the test phase, mistakenly classified on average 10.79% of cases as benign when they were malware cases (false negative). Following the same thinking, there was a mean classification of 5.49% of cases said to be malware when they were benign applications (false positive).

Still regarding Table 7, sensitivity and specificity refer to the ability of the antivirus to identify malware and benign applications, respectively. The proposed work presents the confusion matrix in percentage terms in order to facilitate the interpretation of sensitivity and specificity. In synthesis, the sensitivity and specificity are presented in the confusion matrix itself, described in Table 7. For example, the proposed antivirus averages 94,51% with respect to both sensitivity and true positives. Following the same reasoning, authorial antivirus obtains, on average, 89,21% for both specificity and true negatives.

Table 8 shows the parametric t-students and non-parametric Wilcoxon hypothesis tests between our antivirus and the state-of-the-art. It is possible to conclude that our authorial antivirus is statistically different in comparison to all other samples. The explanation is that in both the parametric t-students and the non-parametric Wilcoxon tests, the null hypothesis were rejected. Therefore, the samples are statistically distinct.

Authorial antivirus demonstrated a major advantage when compared to the state-of-the-art. Our antivirus achieves an average performance of 99.80% within an average training time of 9.32 seconds. The average relationship between percentage accuracy and training time in reverse order is employed in Biomedical Engineering [11]. It is admitted that the establishment of this relationship assumes an important function in Information Security since 8 (eight) new malwares are released every second [8]. So, paradoxically, a newly launched antivirus may already be obsolete and require new training through a newly discovered vulnerability. In syntheses, the learning time of an antivirus should not be discrepant in comparison to the rate of new malware creation worldwide.

8. Conclusion

Each year thousands of malwares are developed in growth and continuous proportions [7]. Therefore, it is of vital importance that malware detection platforms provide cyber-surveillance mechanisms that meet customer demands in a preventive manner. Otherwise, in scenarios where there is a failure to identify the malicious application, sensitive customer data is likely to be made available to unauthorized persons. Worldwide, vulnerabilities in Java account for 91% of all monitored cyber-infections [4]. In addition to personal computers, Java exploits are capable of corrupting corporate web applications being responsible for most of the web-based threats [4].

Thus, it is inferred that the selection of the antivirus has an important role in the combat to virtual plagues. In our evaluation, the variation in Jar malwares detection was from 0% to 99.10%, depending on which commercial antivirus was chosen. The present paper performed the analysis of 86 antiviruses commercially available. On average, they were able to detect 34.95% of Jar malwares. From the analysis of the samples, it was possible to identify that the antiviruses, on average, reported false negatives and they were omitted in 33.90% and 31.15% of the cases, respectively. In the present paper, the VirusTotal platform was used to submit, in an automated way, the malwares to the antiviruses. It should be emphasized that in VirusTotal, there is not the possibility of choosing the shareware version of antiviruses. Then, it was not possible to do comparisons among the free and commercial resources of the same antivirus. It is deduced that the services offered in the shareware versions perform significantly inferior than the full versions.

On average, 31.39% of the paid antivirus (full), submitted to our evaluation, they were not able to detect any of the Jar malwares. It is verified, the flaw in the effectiveness of the commercial antiviruses concerning large-scale, and real-time services. It is noteworthy that in our study, the Jar malwares analyzed are of public domain, employed in malicious activities, and with their performances classified by incident responders. Even so, more than one-third of the evaluated commercial antiviruses had no knowledge about the existence of Jar malwares.

In order to supply the limitations of commercial antiviruses, artificial intelligence-based antivirus are capable to audit thousands of malwares and learn, statistically,

Figure 3: Boxplots referring to the accuracies of the authorial antivirus and the state-of-the-art.

Figure 4: Boxplots regarding the processing times of the authorial antivirus and the state-of-the-art.

what they are their malicious characteristics. Therefore, after learning, intelligent antivirus can identify and classify newly created malware according to the comparison between their features and those cataloged during their learning phase. Hence, there would be no need to wait for a client to be contaminated, and subsequently report a suspicious attitude, as

if it were at that moment, the antivirus takes some action with respect to the discovery of a new malware. Intelligent antivirus enable preventive detection of virtual threats in a controlled environment before they reach customers' machines.

We create an antivirus capable to classify Jar files be-

Table 7
Confusion matrix of the authorial antivirus and the state-of-the-art. (%)

Technique		Train		Test	
		M.	B.	M.	B.
Authorial Antivirus	M.	99.06 ± 0.25	0.94 ± 0.25	94.51 ± 1.83	5.49 ± 1.83
	B.	3.72 ± 0.16	96.28 ± 0.16	10.79 ± 3.25	89.21 ± 3.25
Antivirus made by LIMA, <i>et al.</i> (2021), worst conf. [10]	M.	42.76 ± 49.24	57.24 ± 49.24	42.56 ± 49.07	57.44 ± 49.07
	B.	42.60 ± 49.51	57.40 ± 49.51	42.64 ± 49.45	57.36 ± 49.45
Antivirus made by LIMA, <i>et al.</i> (2021), best conf. [10]	M.	94.27 ± 2.62	5.73 ± 2.62	93.42 ± 1.99	6.58 ± 1.99
	B.	0.84 ± 1.65	99.16 ± 1.65	2.06 ± 1.77	97.94 ± 1.77
Antivirus made by SU, <i>et al.</i> (2018) [21]	M.	74.86 ± 0.65	25.14 ± 0.65	74.86 ± 4.09	25.14 ± 4.09
	B.	16.86 ± 1.89	83.14 ± 1.89	16.86 ± 4.49	83.14 ± 4.49
Antivirus made by VINAYAKUMAR, R. <i>et al.</i> (2019) [22]	M.	97.63 ± 7.39	2.37 ± 7.39	96.60 ± 6.96	3.40 ± 6.96
	B.	0.46 ± 1.33	99.54 ± 1.33	3.12 ± 2.11	96.88 ± 2.11
Antivirus made by HARDY, <i>et al.</i> (2016) [5]	M.	99.92 ± 0.22	0.08 ± 0.22	98.01 ± 2.03	1.99 ± 2.03
	B.	0.75 ± 2.02	99.25 ± 2.02	4.90 ± 2.30	95.10 ± 2.30
Deep Learning made by SANTOS, <i>et al.</i> (2019) [18]	M.	100.00 ± 0.00	0.00 ± 0.00	100.00 ± 0.00	0.00 ± 0.00
	B.	100.00 ± 0.00	0.00 ± 0.00	100.00 ± 0.00	0.00 ± 0.00

Table 8
T-students and Wilcoxon hypothesis test of the authorial antivirus and the state-of-the-art.

Comparison	t-students (parametric test)		Wilcoxon (non-parametric test)	
	Hypothesis	p-value	Hypothesis.	p-value
Authorial Antivirus vs Antivirus made by LIMA, <i>et al.</i> (2021), worst conf.	1	4.2134e-41	1	1.30487e-11
Authorial Antivirus vs Antivirus made by LIMA, <i>et al.</i> (2021), best conf.	1	3.81625e-09	1	2.86398e-09
Authorial Antivirus vs Antivirus made by SU, <i>et al.</i> (2018)	1	6.621e-19	1	2.5046e-11
Authorial Antivirus vs Antivirus made by VINAYAKUMAR, R. <i>et al.</i> (2019)	1	3.803e-06	1	8.83703e-08
Authorial Antivirus vs Antivirus made by HARDY, <i>et al.</i> (2016)	1	1.33009e-11	1	5.13671e-10
Authorial Antivirus vs Deep Learning made by SANTOS, <i>et al.</i> (2019)	1	5.42386e-42	1	1.02645e-12

tween benign and malware. In all, our antivirus monitors and ponders, statistically, 6,824 actions that the suspect Jar file can do when executed in JVM contained in Windows 7. In a controlled environment, our antivirus monitors alterations in the Registry (Database) of OS, traces of calls performed by all processes spawned by the malware, files being created, deleted and downloaded by the malware during its execution, memory dumps of the malware processes, and network traffic trace. The pattern recognition, regarding the 6.824 suspicious actions, is made by extreme neural networks.

Instead of conventional kernels, authorial kernels are employed for ELMs. The ELM network has as its main characteristic the training speed and data prediction when compared to conventional neural networks. The authorial Dilation kernel is able to distinguish Java malwares from benign applications in 91.58% of cases, accompanied by a training time of 52.36 seconds.

Our antivirus can be extended to provide cyber-protection

to other operating systems equipped with JVM. Then, the intention is to extend our antivirus to other operating systems besides Windows. The future goal is to apply our methodology to the Android system since smartphones and tablets are gradually becoming indispensable in contemporary society. Also as a future objective, our antivirus aims to audit OSs, equipped with JVM, specializing in financial transactions such as smart credit card, smart transport passes and lottery terminals.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

Research Involving Human Participants and/or Animals The authors declare that no human participants were involved in this research.

Informed Consent This research did not include health-care intervention of human participants.

References

- [1] AZEVEDO, W.W. *et al.*, 2015a. Fuzzy morphological extreme learning machines to detect and classify masses in mammograms. In: 2015 IEEE International Conference on Fuzzy Systems (FUZZIEEE), Istanbul. doi:<https://doi.org/10.1109/FUZZ-IEEE.2015.7337975>.
- [2] AZEVEDO, W.W. *et al.*, 2015b. Morphological extreme learning machines applied to detect and classify masses in mammograms. In: 2015 International Joint Conference on Neural Networks (IJCNN), Killarney. doi:<https://doi.org/10.1109/IJCNN.2015.7280774>.
- [3] AZEVEDO, W.W. *et al.*, 2020. Morphological extreme learning machines applied to the detection and classification of mammary lesions. In: Tapan K Gandhi; Siddhartha Bhattacharyya; Sourav De; Debanjan Konar; Sandip Dey. (Org.). *Advanced Machine Vision Paradigms for Medical Image Analysis*. 1ed. Londres: Elsevier Science. , 1–30doi:<https://doi.org/10.1016/B978-0-12-819295-5.00003-2>.
- [4] CISCO, 2014. Cisco 2014 annual security report. URL: http://www.efocus.sk/images/uploads/Cisco_2014_ASR.pdf.
- [5] HARDY, W., LINGWEI, C.t., 2016. D14 md : A deep learning framework for intelligent malware detection. In *Int'l Conf. Data Mining* , 61–67.
- [6] HUANG, G.B. *et al.*, 2012. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics* 42(2), 513–519. doi:<https://doi.org/10.1109/TSMCB.2011.2168604>.
- [7] IBM, 2014. Ibm x-force threat intelligence quarterly 1q 2014. Explore the latest security trends—from malware delivery to mobile device risks—based on 2013 year-end data and ongoing research .
- [8] Intel, 2018. McAfee Labs. Accessed on Feb 2020. URL: <https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-mar-2018.pdf>.
- [9] LIMA, S., 2020. Limitation of COTS antiviruses: issues, controversies, and problems of COTS antiviruses. In: Cruz-Cunha, M.M., Mateus-Coelho, N.R. (eds.) *Handbook of Research on Cyber Crime and Information Privacy*, vol. 1, 1st edn. IGI Global, Hershey. doi:<http://dx.doi.org/10.4018/978-1-7998-5728-0.ch020>.
- [10] LIMA, S., SILVA, H., LUZ, J. *et al.*, 2020a. Artificial intelligence-based antivirus in order to detect malware preventively. *Progress in Artificial Intelligence* doi:<https://doi.org/10.1007/s13748-020-00220-4>.
- [11] LIMA, S., SILVA-FILHO, A.G., SANTOS, W.P., 2016. Detection and classification of masses in mammographic images in a multi-kernel approach. *Computer Methods and Programs in Biomedicine* 134, 11–29. doi:<https://doi.org/10.1016/j.cmpb.2016.04.029>.
- [12] LIMA, S.M.L., SILVA-FILHO, SANTOS, W.P., 2020b. Morphological Decomposition to Detect and Classify Lesions in Mammograms. In: Wellington Pinheiro dos Santos; Máira Araújo de Santana; Washington Wagner Azevedo da Silva. (Org.). *Understanding a Cancer Diagnosis*. <https://novapublishers.com/shop/understanding-a-cancer-diagnosis/>.
- [13] LIMA, S.M.L., SILVA-FILHO, A.G., DOS SANTOS, W.P., 2014. A methodology for classification of lesions in mammographies using zernike moments, elm and svm neural networks in a multi-kernel approach. In: 2014 IEEE International Conference on Systems, Man and Cybernetics SMC, San Diego doi:<https://doi.org/10.1109/SMC.2014.6974041>.
- [14] PAUL, N., EVANS, D., 2006. Comparing java and .net security: Lessons learned and missed. *Computers Security* 25(5), 338–350. doi:<https://doi.org/10.1016/j.cose.2006.02.003>.
- [15] PEREIRA, J.M.S. *et al.*, 2020. Method for Classification of Breast Lesions in Thermographic Images Using ELM Classifiers. In: Wellington Pinheiro dos Santos; Máira Araújo de Santana; Washington Wagner Azevedo da Silva. (Org.). *Understanding a Cancer Diagnosis*. <https://novapublishers.com/shop/understanding-a-cancer-diagnosis/>.
- [16] REJAFADA, 2019. Rejafada (a retrieval of jar files applied to dynamic analysis) URL: <https://github.com/rewema/rejafada>.
- [17] SANS, 2017. SANS Institute InfoSec Reading Room. Out with The Old, In with The New: Replacing Traditional Antivirus. Accessed on Feb 2020. URL: <https://www.sans.org/reading-room/whitepapers/analyst/old-new-replacing-traditional-antivirus-37377>.
- [18] SANTOS, M.M., SILVA FILHO, A.G., SANTOS, W.P., 2019. Deep convolutional extreme learning machines: Filters combination and error model validation. *NEUROCOMPUTING* 329, 359–369. doi:<https://doi.org/10.1016/j.neucom.2018.10.063>.
- [19] SANTOS, W.P., 2011. *Mathematical Morphology In Digital Document Analysis and Processing*. volume 8. New York: Nova Science.
- [20] SHAHZAD, F., SHAHZAD, M., F., M., 2013. In-execution dynamic malware analysis and detection by mining information in process control blocks of linux os. *Information Sciences* 231(10), 45–63. doi:<https://doi.org/10.1016/j.ins.2011.09.016>.
- [21] SU, J., VASCONCELLOS, D., t., 2018. Lightweight classification of iot malware based on image recognition. 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC) doi:<https://doi.org/10.1109/COMPSAC.2018.10315>.
- [22] VINAYAKUMAR R.; SOMAN, K., 2018. Deepmalnet: Evaluating shallow and deep networks for static pe malware detection. *ICT Express*. doi:<https://doi.org/10.1016/j.icte.2018.10.006>.
- [23] WAGNER, G.; GAL, A.F.M., 2011. “slimming” a java virtual machine by way of cold code removal and optimistic partial program loading. *Science of Computer Programming* 76, 1037–1053. doi:<https://doi.org/10.1016/j.scico.2010.04.008>.