Revista de Sistemas de Informação da FSMA
n. 27 (2021) pp. 10-28

SISTEMAS DE INFORMAÇÃO
http://www.fsma.edu.br/si/sistemas.html

TRILHA PRINCIPAL

Antivírus da Próxima Geração aplicado à Detecção de *Malwares* Java

Ricardo Paranhos Pinheiro, *Departamento de Computação – Universidade de Pernambuco (UPE)*
Sidney Marlon Lopes de Lima, *Departamento de Eletrônica e Sistemas (DES) - Universidade Federal de Pernambuco (UFPE)*
Sthéfano Henrique Mendes Tavares Silva, *Departamento de Computação – Universidade de Pernambuco (UPE)*
Rafael Diniz Toscano de Lima, *Departamento de Computação – Universidade de Pernambuco (UPE)*
Danilo Monteiro Souza, *Departamento de Computação – Universidade de Pernambuco (UPE)*
Jemerson Rodrigues de Oliveira, *Departamento de Computação – Universidade de Pernambuco (UPE)*
Thyago de Amorim Monteiro, *Departamento de Computação – Universidade de Pernambuco (UPE)*
Sérgio Murilo Maciel Fernandes, *Departamento de Computação – Universidade de Pernambuco (UPE)*
Edison de Queiroz Albuquerque, *Departamento de Computação – Universidade de Pernambuco (UPE)*
Washington Wagner Azevedo da Silva, *Departamento de Engenharia Biomédica - Universidade Federal de Pernambuco (UFPE)*
Wellington Pinheiro dos Santos, *Departamento de Engenharia Biomédica - Universidade Federal de Pernambuco(UFPE)*

Resumo. **Vulnerabilidades em Java correspondem a 91% de todos as cyber-infecções monitoradas na rede mundial de computadores. O presente trabalho cria um Antivírus da Próxima Geração, dotado de aprendizado estatístico e Inteligência Artificial, especializado na detecção de malwares Java. O antivírus proposto monitora e pondera, estatisticamente, 6.824 ações que os arquivos suspeitos possam fazer quando executados. O antivírus proposto alcança um desempenho médio de 91,58% na distinção entre arquivos benignos e malwares. No lugar de modelos baseados em listas negras, o antivírus proposto permite a detecção de malwares de forma preventiva, e não de maneira reativa, como acontece com antivírus comerciais.**

Palavras-Chave*— Malwares; Antivírus; Redes Neurais Artificiais; Detecção de Malwares em tempo real; Forense computacional.*

***Abstract*—Vulnerabilities in Java language answers for 91% of all threats monitored on the world wide web.The present paper aims to create a Next Generation Antivirus, endowed with machine learning and artificial intelligence, specialized in Java malware detection. The proposed antivirus monitors and statistically ponders 6,824 actions that suspicious files can take when executed. The proposed antivirus achieves an average performance of 91.58% in distinguishing between benign files and malware. Instead of blacklist-based models, the proposed antivirus allows the detection of malware preventively, and not reactively, as with commercial antivirus.**

***Index Terms* — Malwares; Antivirus; Artificial Neural Networks; Real-time malware detection; Computer Forensics.**

I. INTRODUÇÃO

Atempos contemporâneos, há a ampla difusão da rede mundial de computadores com aplicações que não focam somente na diversão e no lazer, mas também no trabalho. Uma parcela significante do conteúdo, provido pela rede mundial de computadores, é desenvolvido empregando a tecnologia Java [9]. Os aplicativos Java estão presentes em 97% e em 89% dos computadores corporativos e dos computadores pessoais, respectivamente, nos EUA [9]. Os aplicativos Java possuem grande portabilidade e podem funcionar em cerca de 3 bilhões de dispositivos, inclui-se servidores, computadores pessoais, impressoras, *smartphones*, *tablets*, maquinetas de cartão de crédito, eletrodomésticos dentre outros [16].

Devido à sua popularidade, o número de *cyber*-ataques direcionados à tecnologia Java está em ascensão nos últimos anos [7]. A quantidade de vulnerabilidades exploradas em Java cresceu mais de 300% entre os anos de 2012 e 2013 [7]. As vulnerabilidades comumente tem como alvo computadores pessoais. Então, atividades cotidianas, como navegar na internet ou o uso de programas utilitários, são exploradas de forma maliciosa pelos malwares visando a tecnologia Java. "*Malware*" é uma junção dos termos "malicioso" e "software". O *malware* tem como principal objetivo acessar um dispositivo alheio sem permissão explícita de seu proprietário [10]. Logo, senhas bancárias, redes sociais, fotos ou vídeos íntimos podem ser furtados a partir da *cyber*-infecção por malwares.

No tocante à tecnologia Java, malwares visam corromper a JVM[1] e consequentemente afetar a *Java Security Manager* [7]. A partir do momento em que o gerenciador de segurança é burlado, nada impede que o malware faça operações críticas que não deveriam ser executadas. A razão é que o *Java Security Manager* gerencia o limite externo da JVM controlando como o aplicativo Java, executado dentro da JVM, pode interagir com recursos fora da JVM. Então, quando o *Java Security Manager* é corrompido, o malware passa a ter privilégios irrestritos porque atividades maliciosas ganham aspecto de legítimas. Vulnerabilidades em Java

Autor correspondente: Mr. Ricardo Paranhos Pinheiro rpp3@ecomp.poli.br

[1] JVM – *Java Virtual Machine* – Máquina Virtual Java

correspondem a 91% de todas as *cyber*-infecções monitoradas. Na sequência, Microsoft Word, Adobe Reader, Microsoft Excel e Microsoft Power Point correspondem a 3%, 3%, 2% e 1% das *cyber*-infecções monitoradas [4].

Cabe ressaltar que as *cyber*-infecções em Java não visam somente corromper os computadores pessoais. Os *cyber*-ataques também têm como objetivo afetar aplicações webs corporativas. De acordo com dados estatísticos, provenientes de mais de 1 milhão de transações bancárias *on-line*, mais da metade de todas as vulnerabilidades exploradas visam a tecnologia Java [7]. De todas as ameaças catalogadas na rede mundial de computadores, as vulnerabilidades na linguagem de programação Java continuam a ser o alvo mais frequente [4]. Dada a grande incidência de vulnerabilidades em Java, uma das soluções, indicada pela Cisco, diz respeito a desabilitar o Java nos navegadores sempre que possível [4]. Conclui-se que, implicitamente, a Cisco recomenda não usar Java caso haja algum outro aplicativo equivalente desenvolvido em outra tecnologia.

Uma das razões desse insucesso da tecnologia Java diz respeito ao retardo no catálogo das novas pragas virtuais por sua fabricante, a *Oracle Java.* O *modus operandi* da tecnologia Java é majoritariamente a identificação de arquivos maliciosos com base em assinaturas [7]. Isso quer dizer que o aplicativo suspeito é comparado a uma lista negra confeccionada a partir de denúncias prévias. O grande problema dessa abordagem é que a aplicação mal-intencionada terá que ser conhecida antes (e isso requer que algumas máquinas já estejam infectadas) para que a *Oracle Java* possa acrescentar esse malware, recém-descoberto, em sua lista negra. Apesar de haver iniciativas com resultados promissores, o *modus operandi* da *Oracle Java* é majoritariamente reativo e não preventivo.

Assim como o *Oracle Java*, os antivírus comerciais visam a identificação de arquivos maliciosos com base em assinaturas. Então, o trabalho proposto investiga os principais 86 antivírus comerciais mundiais quanto à identificação de arquivos malwares Java, especificamente, arquivos JAR. A detecção de malwares variou entre 0% a 99,10% a depender do antivírus. Em média, houve a detecção de 34,95% das pragas virtuais. Como aspecto desfavorável, os antivírus, em média, atestaram falsos negativos e foram omissos em 33,90% e 31,15% dos malwares, respectivamente. Além disso, cerca de 31,39% dos antivírus não foram capazes de diagnosticar qualquer uma das amostras maliciosas. Cabe salientar que em nosso estudo, os malwares analisados têm as suas atuações maliciosas documentadas pelos institutos de pesquisa [27]. Mesmo assim, mais da terça parte dos antivírus comerciais avaliados não tinham qualquer conhecimento sobre as existências dos arquivos malwares investigados. Nota-se, a limitação dos antivírus comerciais quanto à robustez de serviços em tempo real e em larga escala.

O estado-da-arte recomenda extrair características de aplicativos suspeitos de forma preventiva antes de executá-los [11]. O arquivo executável é submetido a um processo de engenharia reversa. O código de montagem (*assembly*) referente ao arquivo executável é estudado de modo que a intenção maliciosa do aplicativo suspeito possa ser investigada. Os recursos do código de montagem[2] são usados como atributos de entrada da rede neural artificial usada como classificador. Essa metodologia, nomeada análise estática, é capaz de alcançar um desempenho médio superior a 90% na distinção entre executáveis benignos e malwares [11].

Como efeito colateral, a análise estática pode apresentar deficiências quando submetida a malwares obfuscados. Como estratégia de antiforense digital, os malwares Java empregam empacotamento e obfuscação de código em tempo de execução. Logo, as instruções originais do aplicativo no servidor são diferentes daquelas executadas em tempo de execução no computador pessoal [24]. Conclui-se que a abordagem de características estáticas pode ser burlada por métodos de obfuscação [24].

A análise forense digital se desmembra em uma abordagem denominada extração dinâmica [24]. Ao invés de análise estática, o antivírus autoral realiza a análise dinâmica dos *malwares* Java. Então, a nossa extração de características diz respeito ao comportamento do sistema quando o arquivo suspeito é propositalmente invocado em ambiente controlado. A principal vantagem da abordagem dinâmica é que a análise comportamental é capaz de detectar técnicas de mutação de baixo nível (*bytecode*), como empacotamento ou obfuscação em tempo de execução [20]. Ao todo, nossa extração dinâmica de características monitora 6.824 comportamentos que o arquivo JAR suspeito posso fazer quando executado.

Os comportamentos maliciosos oriundos dos arquivos investigados servem como atributos de entrada das redes neurais. No presente artigo, são empregadas as mELMs (ELMs[3] morfológicas), ou seja, ELM com núcleos de camada oculta inspirados em operadores morfológicos de processamento de imagem de *Erosão* e *Dilatação* [23]. A Morfologia Matemática diz respeito ao estudo das formas dos corpos presentes nas imagens através do uso da teoria matemática de intersecção, diferença e união de conjuntos. Logo, as operações morfológicas lidam naturalmente com a detecção das formas dos corpos presentes nas imagens ao se interpretar a fronteira de decisão de uma rede neural como uma imagem *n*-dimensional, onde *n* diz respeito à quantidade de características extraídas. O antivírus autoral alcança uma acurácia média de 91,58% na distinção entre aplicativos Java benignos e malwares.

Este trabalho está organizado da seguinte forma: na seção 2 apresentamos as limitações dos antivírus comerciais. Na seção 3, discutimos o estado-da-arte quanto aos antivírus dotados de inteligência artificial; na seção 4, apresentamos a metodologia proposta; na seção 5, fazemos um comparativo entre a rede ELM autoral e as redes ELMs clássicas; na seção 6, mostramos os resultados e algumas discussões. Por fim, na seção 7, fazemos as conclusões gerais e discutimos as perspectivas do nosso trabalho.

[2] Por recursos de código *assembly* se denota o repertório de instruções e APIs (*Application Programming Interface* - Interface de Programação de Aplicações)

[3] ELM: *Extreme Learning Machine* – Máquina de Aprendizado Extremo.

II. Limitação dos Antivírus Comerciais

Apesar de ser questionado há mais de uma década, o modus operandi dos antivírus é baseado em assinaturas quando o arquivo suspeito é consultado em bases de dados nomeadas de lista negra [21]. Por consequência, antivírus baseados em assinaturas têm efetividade nula quando submetidos a variantes de um mesmo malware ou pragas virtuais inéditas [10][21]. Por intermédio da plataforma VirusTotal, o trabalho proposto investiga os principais antivírus comerciais mundiais com seus respectivos resultados apresentados na Tabela 1 [28]. Os resultados expandidos estão em nosso repositório autoral [19]

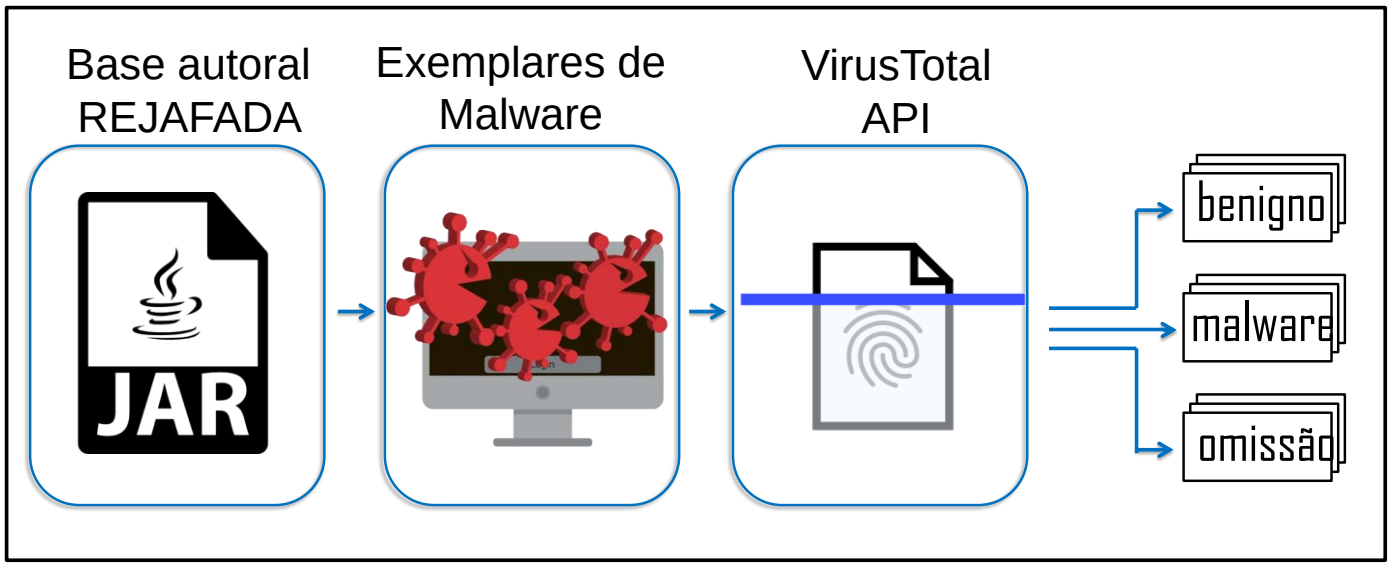

Figura 1. Diagrama da API VirusTotal.

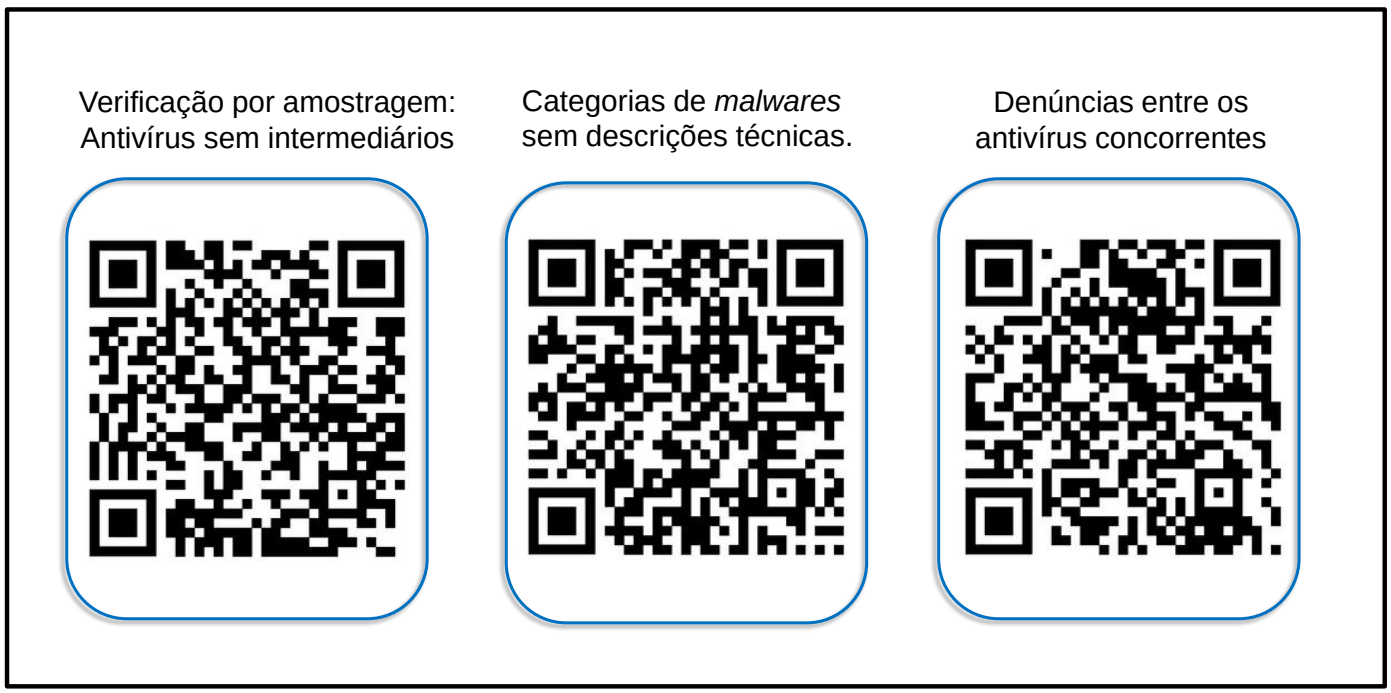

Figura 2. Os QR Codes são links para referências criadas ao longo do texto. a) Categorias de malwares sem descrições técnicas. b) Verificação por amostragem: Antivírus sem intermediários. c) Um antivírus comercial pode denunciar o instalador do antivírus concorrente.

Tabela 1. Resultado dos principais antivírus comerciais. Resultados expandidos dos 86 antivírus comerciais estão no repositório autoral [19].

Antivírus	**Detecção (%)**	**Falso negativo (%)**	**Omissão (%)**
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%
Sophos	94,70%	5,00%	0,30%
Symantec	94,10%	5,90%	0,00%
Ikarus	93,60%	0,70%	5,70%
MAX	91,00%	8,80%	0,20%
ViRobot	3,30%	96,70%	0,00%
Rising	2,90%	96,10%	1,00%
TheHacker	1,10%	98,80%	0,10%
Kingsoft	0,70%	99,20%	0,10%
Invincea	0,60%	0,10%	99,30%
Zoner	0,60%	97,10%	2,30%
Baidu	0,60%	99,00%	0,40%
VIPRE	0,50%	99,50%	0,00%
Malwarebytes	0,30%	94,10%	5,60%
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%

Em nosso experimento, foram empregados 998 arquivos maliciosos de extensão JAR. O objetivo do trabalho é verificar a quantidade de pragas virtuais catalogadas pelos antivírus comerciais. A motivação é que a aquisição de novas pragas virtuais assume papel importante no combate a aplicações mal-intencionadas. Logo, quanto maior for a base de dados de *malwares*, chamada de lista negra, melhor tende a ser a defesa provida pelo antivírus. A Figura 1 exibe o diagrama da metodologia proposta em diagrama de blocos. Inicialmente, os malwares são enviados ao servidor pertencente à plataforma VirusTotal. Após isso, os arquivos são analisados pelos 86 antivírus comerciais vinculados ao VirusTotal. A seguir, os antivírus provêm seus diagnósticos para os arquivos submetidos à plataforma. O VirusTotal permite a possibilidade de emissão de três tipos diferentes de diagnósticos; *malware*, benigno e omissão.

Quanto à primeira possibilidade do VirusTotal, o antivírus detecta a malignidade do arquivo suspeito. No ambiente experimental proposto, todos os arquivos submetidos são *malwares* documentados por instituto de pesquisa [27]. Logo, o antivírus acerta quando detecta a malignidade do arquivo investigado. A detecção do *malware* indica que o antivírus provê um serviço robusto contra *cyber*-invasões. Na segunda possibilidade, o antivírus atesta a benignidade do arquivo investigado. Logo, no estudo proposto, quando o antivírus alega a benignidade do arquivo, trata-se de um caso de falso negativo visto que todas as amostras são maliciosas. Isso quer dizer, o arquivo investigado é *malware* e, no entanto, o antivírus atesta benignidade, de forma equivocada. Na terceira possibilidade, o antivírus não emite opinião sobre o arquivo suspeito. A omissão indica que o arquivo investigado jamais foi avaliado pelo antivírus e esse tampouco possui robustez para avaliá-lo em tempo real. A omissão do diagnóstico, por

parte do antivírus, aponta a sua limitação quanto a serviços em larga escala.

A Tabela 1 exibe os resultados alcançados pelos principais 86 antivírus comerciais avaliados. O antivírus McAfee-GW-Edition obteve o melhor desempenho sendo capaz de detectar 99,10% dos malwares investigados. Uma grande adversidade, no combate a aplicações mal-intencionadas, é o fato dos fabricantes dos antivírus não compartilharem, entre elas, as suas respectivas listas negras de *malwares* devido a disputas comerciais.

Como cuidado metodológico, o presente trabalho instalou alguns antivírus com taxas de detecção baixas. A hipótese foi verificar algum problema de integração entre o VirusTotal e os antivírus investigados. Os *malwares* listados na Tabela 1 foram empregados, dessa vez através do próprios antivírus sem intermediários. Por amostragem, os resultados foram igualmente ruins (conforme visto no vídeo apontado pelo *QR Code* da Fig. 2 a).

Ainda no tocante à Tabela 1, o trabalho proposto aponta para um agravante dessa adversidade; um mesmo fabricante de antivírus sequer compartilha as suas bases de dados entre seus distintos antivírus. Por exemplo, os antivírus McAfee-GW-Edition e McAfee pertencem a uma mesma empresa. As suas listas negras, apesar de robustas, não são compartilhadas entre si. Logo, as estratégias comerciais, de uma mesma empresa, atrapalham o enfretamento a *malwares*. Complementa-se que as fabricantes de antivírus estão preocupadas em evitar as *cyber*-invasões assim como também em otimizar seus rendimentos comerciais.

Tabela 2: Miscelânea de classificações providas pelos antivírus comerciais. O total de classificações está no repositório autoral [19].

Antivírus	**VirusShare_9ef6966b98a5c9ce524bc9a24dc9c488**	**VirusShare_bee5a7c75b6a5b9a41634dce2ae21128**	**VirusShare_93fa0564cfc049a16768d11b34dd60e8**
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.AndroidOS.Fakelogo.a	HEUR:Trojan-SMS.AndroidOS.Fakelogo.a	HEUR:Trojan.Win32.Generic
ZoneAlarm	HEUR:Trojan-SMS.AndroidOS.Fakelogo.a	HEUR:Trojan-SMS.AndroidOS.Fakelogo.a	HEUR:Trojan.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
Ikarus	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
MicroWorld-eScan	Android.Trojan.FakeInst.CB	Benign	Gen:Variant.Barys.841

A detecção de *malwares* variou entre 0% e 99,10%, a depender do antivírus investigado. Em média, os 86 antivírus foram capazes de detectar 34,95% das pragas virtuais avaliadas, com desvio padrão de 40,92%. O desvio padrão elevado indica que a detecção de arquivos maliciosos pode sofrer variações abruptas a depender do antivírus escolhido. Determina-se que a proteção, contra invasões cibernéticas, está em função da escolha de um antivírus robusto dotado de uma grande e atualizada lista negra. Em média, os antivírus atestaram falsos negativos em 33,90% dos casos, com desvio padrão de 40,45%. Atestar a benignidade de um *malware* pode implicar em prejuízos irrecuperáveis. Uma pessoa ou

instituição, por exemplo, passaria a confiar em uma determinada aplicação maliciosa quando, de fato, trata-se de um *malware*. Ainda como aspecto desfavorável, cerca de 31,39% não emitiram opinião em qualquer uma das 998 amostras maliciosas. Em média, os antivírus foram omissos em 31,15% dos casos, com desvio padrão de 45,61%. A omissão do diagnóstico aponta a limitação dos antivírus quanto à detecção de *malwares* em tempo real.

Inclui-se como adversidade, no combate a aplicações mal-intencionadas, o fato dos antivírus comerciais não possuírem um padrão na classificação dos *malwares* como visto na Tabela 2. Nós escolhemos 3 dos 998 malwares de modo a exemplificar a miscelânea de classificações dadas pelos antivírus comerciais. Como não existe um padrão, os antivírus dão os nomes que desejam, por exemplo, uma empresa pode identificar um *malware* como "Android:RuFraud-I' e uma segunda empresa identificá-lo como "Artemis!9EF6966B98A5".

A falta de um padrão atrapalha as estratégias de *cyber*-vigilância visto que cada categoria de *malware* deve ter tratamentos (vacinas) distintos. Conclui-se que é complexo o aprendizado de máquina supervisionado visando reconhecimento de padrão de categorias de *malwares*. Devido a esse emaranhado confuso de classificações multi-classe, providas pelos especialistas (antivírus) como visto na Tabela 2, é improvável que alguma técnica de aprendizado de máquina adquira capacidade de generalização.

Distintos trabalhos visam elaborar estratégias de modo a agrupar as categorias providas pelos antivírus comerciais exemplificados na Tabela 2. Algumas estratégias dizem respeito a agrupar as categorias por nomes similares. Outra alternativa, com maior rigor metodológico, seria agrupar as categorias de *malwares* de acordo com as suas descrições técnicas e características. A referida estratégia se torna inviável. Como regra geral, as categorias de *malwares* disponibilizadas pelos antivírus comerciais não são definidas nos portais de seus respectivos fabricantes (como visto no vídeo apontado pelo *QR Code* da Fig. 2 b).

Após a invasão cibernética, os clientes infectados são condicionados a relatar o contágio em blogs avulsos, geralmente sem qualquer resposta oficial do fabricante. Com exceção de muito poucos antivírus, não há descrição das categorias de *malwares* nos portais dos fabricantes. Essa falta de informação dos antivírus comerciais pode ser interpretada como um desrespeito ao código de defesa do consumidor, em específico, o artigo 14. Legalmente, os antivírus não poderiam fornecer informações insuficientes ou inadequadas quanto aos riscos do serviço adquirido. Por exemplo, não se sabe os riscos de colocar na quarentena um aplicativo classificado como um *malware* de nome confuso e pouco elucidativo a exemplo dos apresentados na Tabela 2. Em acréscimo, não há qualquer descrição técnica das referidas classes de *malwares* seja na literatura acadêmica ou no portal dos fabricantes.

III. Estado-da-Arte

Arquivos de extensão JAR são coleções de arquivos de classe compactados individualmente. Em regra geral, quando compactados, tais arquivos ocupam metade do tamanho dos arquivos originais [29]. Um arquivo JAR encapsula classes Java e também pode conter outros recursos como assinatura digital ou imagens [17]. O arquivo JAR foi projetado visando fornecer um ambiente confiável para execução de pequenos programas incorporados a páginas da web nomeadas de *applets* [17].

O *modus operandi* dos antivírus comerciais e do *Oracle Java* é majoritariamente a identificação de *malwares* com bases em listas negras. Dada as limitações dos antivírus comerciais e do *Oracle Java*, o estado-da-arte propõe extrair e analisar as características dos arquivos através de máquinas de aprendizados estatísticos. A inteligência artificial consegue automatizar muitas tarefas, analisando milhares de arquivos, extraindo suas características e os classificando.

LIMA, *et al*. (2021 a) criou um antivírus capaz de detectar malwares em aplicativos Windows[4] com uma precisão média de 98,32% [11]. O arquivo executável é submetido ao processo de engenharia reversa. Então, o aplicativo pode ser estudado de modo que a intenção maliciosa do arquivo possa ser investigada. O antivírus de LIMA, *et al*. (2021 a) extraiu 630 atributos de cada arquivo executável. Esses atributos servem como parâmetros para os neurônios de entrada das redes neurais artificiais baseadas em retropropagação. A base de dados empregada por LIMA, et al (2021 a) utiliza 6632 aplicativos balanceados entre duas categorias: benignos e maliciosos. O antivírus confeccionado por LIMA, *et al*. (2021 a) emprega redes neurais rasas.

Por outro lado, antivírus baseados em redes profundas (*Deep Learning*) também alcançam excelentes acurácias. VINAYAKUMAR, R. *et al*. (2019) obtém uma precisão média de 98,90% visando detectar malwares Windows [26]. A estrutura da rede profunda apresenta 34 camadas sequenciais[5]. A base de dados para treinamento contém 25035 arquivos benignos e outros 24965 arquivos malwares para treinamento. A rede é treinada com 500 épocas com um tamanho de lote de treinamento de 64 e taxa de aprendizado de 0,01.

O antivírus de SU, *et al*. (2018) alcançou uma precisão média de 94,00% para detectar *malwares* de IoT (*Internet of Things* – Internet das Coisas) [25]. A estrutura de rede profunda possui 6 camadas. Há 3 camadas sequenciais de processamento: 2 camadas convolucionais e 1 camada totalmente conectada. A base de dados contém 365 arquivos benignos e outros 500 *malwares*. A rede é treinada com 5.000 iterações com um tamanho de lote de treinamento de 32 e taxa de aprendizado de 0,0001.

O antivírus de HARDY, W. *et al.* (2016) visa detectar *malwares* para Windows através de redes profundas dotadas de *Stacked Autoencoders* [5]. A referida rede profunda tenta mapear o reconhecimento de padrão final de volta à entrada original. O modelo de aprendizado profundo é treinado com 3

[4] PE (*Portable Executable* – Portáveis Executáveis) se referem a arquivos executados diretamente no sistema operacional Windows. Os arquivos PE são; processos (.exe), bibliotecas (.dll) e serviços (.sys).

[5] Camadas DeepMalNet. Disponível em: https://github.com/vinayakumarr/dnn-ember/blob/master/DNN-info.pdf. Acesso em fevereiro de 2020.

camadas ocultas e 100 neurônios em cada camada oculta. A base de dados contém 22.500 arquivos benignos e outros 22.500 *malwares*. HARDY, W. *et al.* (2016) obteve uma acurácia média de 96,85%.

Redes profundas são modelos de processamento de dados os quais empregam um grafo profundo. Tal estrutura é convencionalmente construída contendo múltiplas camadas sequenciais. Arquiteturas de redes profundas do estado-da-arte empregam camadas contendo distintos tipos de processamento. Costumeiramente, as diferenciações entre as camadas dizem respeito às funções de ativação, normalização, convolução e redução de dimensionalidade[6].

Redes neurais profundas empregam largamente a convolução linear de filtros. A convolução tem o processo básico de deslocamento dos filtros sobre a imagem original. Os filtros lineares são pequenas matrizes bidimensionais cujos valores de seus coeficientes determinam o objetivo a ser alcançado durante o processamento. Os filtros lineares geram medias ponderadas das regiões da imagem original abrangidas (sobrepostas) a sua matriz.

A desvantagem da rede profunda é o longo tempo de treinamento. Como agravante, as redes profundas apresentam baixa capacidade de paralelismo porque as camadas de processamento são sequenciais. Logo, uma camada só pode ser executada após a camada imediatamente anterior ter concluído o seu trabalho. Isso pode ser um obstáculo em aplicativos que precisam de treinamento frequente como os antivírus pois em média 8 (oito) novos *malwares* são criados a cada segundo [8]. Em síntese, não deve haver discrepâncias no tempo de aprendizagem do antivírus em comparação com a taxa de nova geração de *malware* mundial.

A rede profunda confeccionada por SANTOS, *et al.* (2019) possui tempo de treinamento compatível com aplicações que requerem treinamento frequente. A referida rede profunda apresenta uma única camada de processamento e não há retropropagação de dados. Portanto, uma vez que o supercomputador possua recursos computacionais suficientes (memória), a rede profunda feita por SANTOS, *et al.* (2019) possui larga capacidade de paralelismo. A camada de processamento emprega 30.000 filtros convolucionais lineares simultaneamente.

Ao invés de filtros convolucionais aleatórios, a rede neural profunda de SANTOS, *et al.* 2019 desenvolve filtros PCA[7] visando a extração de componentes principais das regiões de interesse referentes ao vetor de entrada de dados. Cada filtro é convertido em uma matriz de Toeplitz denotada por $W_{detect} \in \mathbb{R}^{L^2 x J^2}$, com $L = (J + W - 1)$ para um mapa de características de tamanho LxL, onde $We\ J$ se referem ao tamanho dos filtros e o número de pixels, respectivamente [22].

Neste trabalho replicamos a rede profunda feita por SANTOS, *et al.* 2019, pois seu tempo de treinamento é compatível com aplicações que necessitam de treinamento frequente como os antivírus. A rede profunda de SANTOS, *et al.* 2019 não visa a detecção de *malware*, mas sim o reconhecimento óptico de caracteres [22]. Ao invés de processamento digital de imagem, os neurônios de entrada dizem respeito à extração de características dos malwares.

Devido aos excelentes resultados obtidos por técnicas de rede profunda, criou-se um senso comum de que redes profundas são capazes de fornecer a melhor precisão em qualquer tipo de aplicação; na verdade, essa consideração é falsa. Enfatiza-se que redes profundas empregam largamente camadas convolucionais em seus grafos arquiteturais. Embora tenha um papel fundamental em aplicações computacionais, a convolução de filtros é limitada a aplicações nas quais o gradiente de fluxo vetorial é formado [23].

Considere, por exemplo, as imagens biomédicas oriundas de mamógrafos. As imagens são repletas de ruídos que atrapalham o reconhecimento da lesão mamária [14]. Logo, a convolução de filtros é fundamental no sentido de eliminar os ruídos e, portanto, descartar pequenas irregularidades no achado correspondente ao câncer em potencial. Técnicas convolucionais como, por exemplo, filtros gaussianos são essenciais na redução de ruídos em imagens biomédicas [14].

Como contra-exemplo, considere o repositório ilustrado na Tabela 3. Os atributos estão completamente desconectados uns dos outros, apesar de pertencerem à mesma vizinhança. Um aplicativo suspeito de tentar verificar os dados de Wi-Fi não tem correlação com o acesso à galeria de imagens da vítima ou ao navegador. Ao aplicar a convolução linear dos filtros no repositório ilustrado na Tabela 3, o acesso às imagens contendo o valor 0 seria tratado como ruído. A explicação é que sua vizinhança possui valores positivos. Em síntese, o aplicativo suspeito seria acusado de acessar a galeria de imagens da vítima até mesmo quando a extração de atributos tivesse auditado o inverso.

Em análises estáticas, o *malware* costuma ser convertido em imagem de modo a servir como atributo de entrada das redes profundas. Essa solução permite o surgimento de um gradiente vetorial visto que os aplicativos possuem uma estrutura pré-definida específica. Por outro lado, o trabalho proposto aqui aplica análise dinâmica. A cadeia de eventos invocada pelo aplicativo suspeito pode não seguir um gradiente conforme ilustrado na Tabela 3. De modo a provar nosso embasamento teórico, o antivírus autoral emprega redes neurais morfológicas rasas em detrimento de redes profundas contendo camadas convolucionais.

Como experimentos, o antivírus autoral tem sua acurácia comparada a antivírus de última geração baseados tanto em redes neurais rasas quanto em profundas. Nosso antivírus pode combinar alta precisão com tempo de aprendizagem reduzido. A fim de evitar comparações injustas, a etapa de extração de características é padronizada através do monitoramento de 6.824 comportamentos que o arquivo JAR suspeito posso fazer quando executado propositalmente.

[6] Exemplo de arquitetura de rede neural profunda. Disponível em: https://se.mathworks.com/help/deeplearning/gs/create-simple-image-classification-network-using-deep-network-designer.html. Acesso em maio de 2021.

[7] PCA: *Principal Component Analysis* – Análise de Componentes Principais.

Tabela 3. Ilustrativo de repositório de aprendizado estatístico visando o reconhecimento de *malwares*.

Atributos		
capturar informações de rede	acessar a galeria de imagens	acessar o e-mail da vítima
1	0	1

IV. Estudos Preliminares: Redes Neurais ELM

O trabalho proposto resulta em um antivírus composto por redes neurais ELMs (*Extreme Learning Machines*) visando a detecção preventiva de malwares. ELMs constituem máquinas de aprendizado estatístico baseadas em *kernels* poderosos e flexíveis, cujas características principais são treinamento rápido e desempenho de classificação robusto [6]. A rede ELM é uma rede de camada única oculta, não recorrente, baseada em um método analítico para estimar os pesos de saída da rede, em qualquer inicialização aleatória de pesos de entrada.

As ELMs têm sido amplamente aplicadas em diversas áreas como Engenharia Biomédica [1][2][3][13][14][15][18]. As redes de ELMs podem contribuir muito para o avanço da segurança digital dos dispositivos como exemplificado no trabalho de LIMA, *et al.* 2020 b que as aplica [12]. O presente trabalho aplica as ELMs na área de segurança da informação, especificamente no reconhecimento de padrões de malware Java.

O processo de aprendizagem da rede ELM é baseado na inversa generalizada de Moore-Penrose (pseudo-inversa), com a qual são calculados os pesos entre a camada escondida e a camada de saída (HUANG, ZHOU, *et al.*, 2012). Matematicamente, na rede ELM os atributos de entrada x_{ik} correspondem ao conjunto $\{x_{it} \in R;\ i \in N^*,\ i = 1, \dots, n;\ t \in N^*,\ t = 1, .. v\}$. Logo, há n características extraídas da aplicação e v vetores de dados de treinamento. A camada escondida h_j, constituída por m neurônios, é representada pelo conjunto $\{h_j \in R;\ j \in N^*,\ j = 1, \dots, m\}$.

O processo de treinamento da ELM é rápido por ser composto por poucas etapas não iterativas. Inicialmente, os pesos de entrada w_{ji} e $biasb_{jt}$ são definidos de maneira aleatória. Dada uma função de ativação $f: \mathbb{R} \to \mathbb{R}$, o processo de aprendizagem é dividido em três passos:

1. Atribuição aleatória de pesos w_{ji}, correspondente aos pesos entre a camada de entrada e a camada escondida, e *bias* b_{jk}.
2. Calcular a matriz H, que corresponde à saída dos neurônios da camada escondida.
3. Calcular a matriz dos pesos de saída $\beta = H^{\dagger}Y$, onde $H^{\dagger}$ é a matriz inversa generalizada de Moore-Penrose da matriz H, e Y corresponde à matriz de saídas desejadas *s*.

Assim como na MLP[8], a saída dos neurônios da camada escondida, correspondente à matriz H, é calculada através do *kernel* K, entradas e pesos da camada escondida, conforme mostra a Equação (1).

$$H_{jt} = \begin{bmatrix} K(11) & \cdots & K(1N) \\ \vdots & \ddots & \vdots \\ K(V1) & \cdots & K(VN) \end{bmatrix} \quad (1)$$

Ao invés de *kernels* convencionais, são empregados *kernels* autorais para as ELMs. No presente artigo, são empregadas as mELMs (ELMs morfológicas), que consistem em ELM com núcleos de camada oculta inspirados em operadores morfológicos de Erosão e Dilatação, usados em processamento de imagens. *Kernels* são funções matemáticas empregadas como método de aprendizagem de redes neurais. Este método de aprendizagem permite a criação de mapeamento não linear de dados. Assim, não há necessidade de aumentar o número de parâmetros ajustáveis a exemplo de taxa de aprendizado empregada em redes baseadas em retropropagação.

A. Morfologia Matemática

Há duas operações morfológicas fundamentais: Erosão e Dilatação [23]. A Morfologia Matemática pode ser considerada uma teoria construtiva porque todas as operações são construídas tendo como base Erosões e Dilatações. Matematicamente, Erosão e Dilatação são formalizadas de acordo com a Equação (2) e a Equação (3), respectivamente:

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

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

onde $f: S \to [0,1]$ e $g: S \to [0,1]$ são imagens normalizadas em forma de matriz nomeada de formato S, onde $S \in \mathbb{N}^2$. f diz respeito à imagem original. O pixel é definido através do par cartesiano $(u, f(u))$, onde u é a posição associada ao valor $f(u)$. v é a matriz de *f(u)*, abrangida por g. Os operadores $\cup$ e $\vee$ estão associados à operação de máximo, enquanto $\cap$ e $\wedge$ estão associados à operação de mínimo. g é o elemento estruturante tanto para Erosão quanto para Dilatação[23]. $\bar{g}$ é a negação de g.

Na Equação (2) inicialmente ocorre a negação do elemento estruturante $\bar{g}$. Logo, acontece a operação de máximo $\vee$ denotada por $f(v) \vee \bar{g}(u - v)$, onde $f(v)$ diz respeito à matriz da imagem original f abrangida (casada) por $\bar{g}$. $f(v)$ é nomeada tecnicamente de região ativa da imagem. Por fim, o

[8] MLP: *Multi Layer Perceptron* – Rede *Perceptron* Multicamadas.

valor $\varepsilon_g(f)(u)$, na posição u, da imagem erodida recebe o mínimo valor entre os máximos, através do operador $\cap$. $\varepsilon_g(f)(u)$ obtém o valor 0 associado ao preto absoluto. A Erosão sobrepõe $\bar{g}$ à imagem original f. O objetivo é que as áreas similares ao $\bar{g}$ se expandam. Ao se associar o 1's ao branco absoluto e 0's ao preto absoluto, a Erosão aumenta as áreas mais escuras e elimina as regiões com maior intensidade[23].

A Equação (3) exibe a atuação da operação morfológica de Dilatação. Por precedência matemática, ocorre a operação de mínimo $\wedge$ denotada por $f(v) \wedge g(u-v)$, onde $f(v)$ diz respeito à matriz da imagem original f abrangida (casada) por g. Logo, o valor $\delta_g(f)(u)$, na posição u, da imagem dilatada recebe o máximo valor entre os mínimos, através do operador $\cup$. A Dilatação sobrepõe o elemento estruturante g à imagem original f. O objetivo é que as áreas similares ao g se expandam. Ao se associar o 1's ao branco absoluto e 0's ao preto absoluto, a dilatação aumenta as áreas com tonalidade mais intensas e elimina as regiões escuras [23].

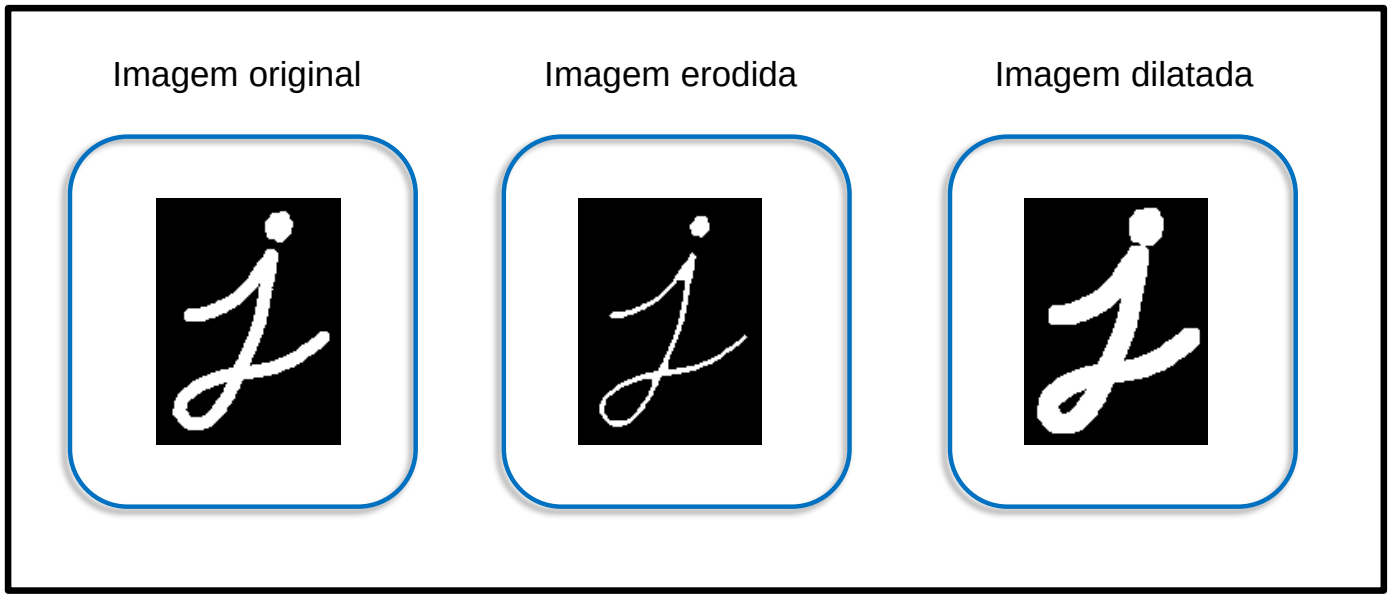

Figura 3 .a) Imagem original. b) Imagem erodida. c) Imagem dilatada. Figura extraída da biblioteca gráfica OpenCV

A. Redes Neurais ELM Morfológicas

O trabalho proposto emprega as mELMs (*Morphological Extreme Learning Machines*). Como estudo de caso, as mELMs são aplicadas na distinção entre executáveis benignos e *malwares* Java. As mELMs são inspiradas na Morfologia Matemática tendo como base os operadores não-lineares de Erosão e Dilatação. Dada a Equação (2), referente ao operador de imagem erosão, o *kernel* ELM de Erosão pode ser definido de acordo com a Equação (4), onde $\{i \in N^*,\ i = 1, \dots, n;\ j \in N^*,\ j = 1, \dots, m,;\ t \in N^*,\ t = 1, .. v\}$. Logo, há *n* neurônios na camada de entrada (sem o *bias*), *m* neurônios na camada escondida e *v* vetores de dados de treinamento.

$$K_\varepsilon(t,i) = \bigcap_{i=1}^{n} (x_{it} \vee \overline{w}_{ji}) + b_{jt} \quad (4)$$

Similarmente ao *kernel* Erosão, a Equação (5) define o *kernel* Dilatação inspirado na Equação (3) e referente ao operador morfológico de Dilatação.

$$K_\delta(t,i) = \bigcup_{i=1}^{n} (x_{it} \wedge w_{ji}) + b_{jt} \quad (5)$$

Um dos grandes desafios, em redes neurais artificiais, diz respeito a encontrar um *kernel* de modo que otimize a fronteira de decisão entre as classes de uma dada aplicação. Seguindo o raciocínio, um *kernel* sigmoidal é capaz de resolver um problema separável por uma função sigmoide como visto na Figura 4 (a). De maneira oposta, um *kernel* linear é incapaz de resolver um problema não-linearmente separável como exibido na Figura 4 (b).

Então, uma boa capacidade de generalização da rede neural pode depender de uma escolha ajustada do *kernel*, o que implica em que o melhor é aquele que estiver subordinado ao problema a ser resolvido. Como efeito colateral, a investigação de diferentes *kernels* é geralmente um processo custoso envolvendo validação cruzada combinada com diferentes condições iniciais aleatórias. A investigação de distintos *kernels*, no entanto, pode ser necessária, caso contrário a rede neural composta, por um *kernel* desajustado, por gerar resultados não satisfatórios.

A Figura 4 (c) e a Figura 4 (d) exibem as atuações dos *kernel* autorais de Erosão e de Dilatação em uma mesma distribuição sigmoide, com as respectivas precisões de 93,07% e 95,05%. Visualmente, é possível observar que os *kernel* autorais mapeiam distintas distribuições, referentes a diferentes problemas, de forma satisfatória. Cabe ressaltar que os dois atributos (características) estão normalizados sobre um mesmo limite inferior e superior.

A explicação do sucesso dos *kernels* mELMs diz respeito à sua capacidade de modelar qualquer fronteira de decisão, visto

que o seu mapeamento não obedece às superfícies geométricas convencionais como elipse e hipérbole. O mapeamento da fronteira de decisão, realizado pelos *kernels* mELMs, emprega as coordenadas no espaço *n*-dimensional das amostras reservadas ao treinamento, onde *n* diz respeito à quantidade de características extraídas. Logo, as nossas mELMs são capazes de naturalmente detectar e modelar as regiões *n*-dimensionais referentes às distintas classes por empregar a Morfologia Matemática, que lida naturalmente com a detecção das formas dos corpos presentes nas imagens [23].

O antivírus autoral emprega dados de entrada contendo atributos completamente desconexos entre si conforme ilustrado na Tabela 3. Os *kernels* morfológicos autorais apresentam uma importante relação com o referido repositório. A justificativa é de que a Morfologia Matemática é capaz de detectar e segmentar os limites dos objetos alvos preservando as relações dos corpos através do uso da teoria matemática de intersecção, união e diferença de conjuntos [23]. Ao se considerar o exemplo contido na Tabela 3, os *kernels* morfológicos autorais são capazes de processar regiões totalmente segregadas preservando seus limites. Por regiões, denota-se uma área contendo valores congruentes continuadamente.

Por outro lado, a convolução linear de filtros, convencionalmente empregada pelas redes profundas, geram médias ponderadas das regiões de interesse descartando (embaçando) pequenas irregularidades [14]. Como consequência, um aplicativo suspeito poderia ser acusado de cometer uma atividade ilícita mesmo quando a extração de atributos tivesse auditado o inverso conforme explicado na seção 0. Para tal basta que o atributo monitorado pertença a uma região de vizinhança contendo valores positivos.

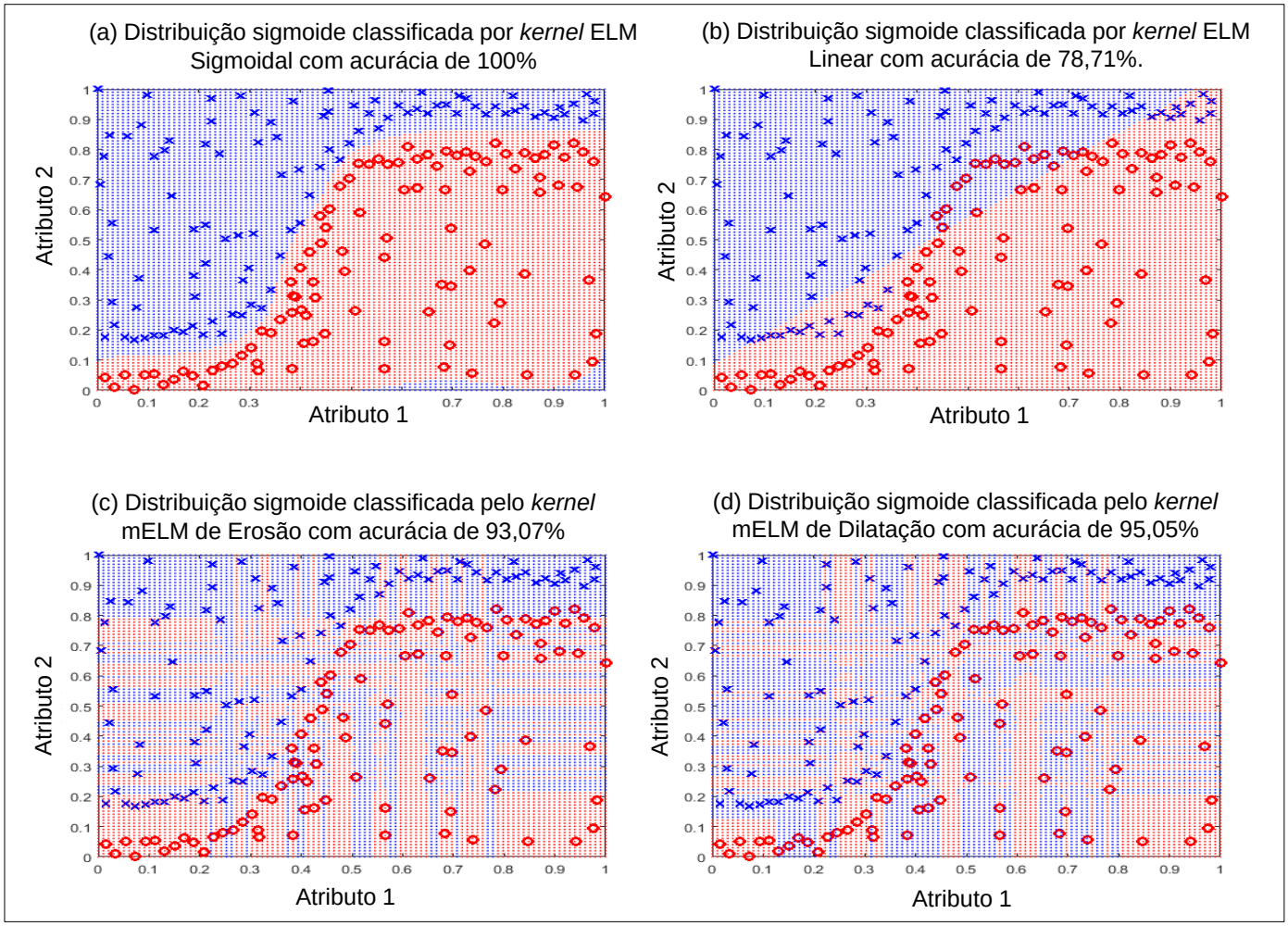

Figura 4. (a) Atuação bem-sucedida do *kernel* compatível com o conjunto de dados. (b) Classificação imprecisa do *kernel* Linear em uma distribuição não-linearmente separável. (c - d) Atuações bem-sucedidas dos *kernels* autorais de Erosão e Dilatação. Figura extraída de LIMA, *et al*. (2020) b [12].

V. MÉTODO PROPOSTO

A Figura 5 exibe o diagrama da metodologia proposta em diagrama de blocos. Inicialmente, o arquivo suspeito, oriundo da base de dados REJAFADA, é executado com o objetivo de verificar a tentativa de corromper a JVM e, em sequência, o Windows 7. O motivo da escolha do Windows 7 se dá porque é a versão padrão do *CuckooBox*. A referida *SandBox* diz respeito ao ambiente controlado empregado pelo trabalho proposto visando o monitoramento do aplicativo suspeito.

As características dinâmicas estão sintetizadas na subseção V-B. Então, as características dinâmicas dos arquivos são armazenadas no formato de repositório de aprendizado estatístico. Após a mineração de características, os comportamentos relevantes servem como atributos de entrada das máquinas de aprendizado estatístico, especificamente, redes neurais artificiais empregadas como classificadores. O objetivo é agrupar os arquivos investigados em duas classes; benignos e malwares. A etapa de classificação é explicada, em detalhes, na subseção V-D. Os resultados da classificação estão descritos na seção VI.

Quanto aos materiais, todos os experimentos foram realizados em um supercomputador em nuvem dotado de 250 GB de memória RAM, 8 processadores e 300 GB de armazenamento em massa. De modo a não haver comparações injustas, os antivírus do estado-da-arte são treinados e testados no mesmo supercomputador empregado pelo antivírus autoral. Enfatiza-se que a aquisição de um supercomputador foi devido à réplica e à comparação com os trabalhos do estado-da-arte. O antivírus autoral exige baixa capacidade de processamento e de armazenamento. Enfatiza-se que o antivírus autoral poderia ser usado em qualquer computador de mesa convencional.

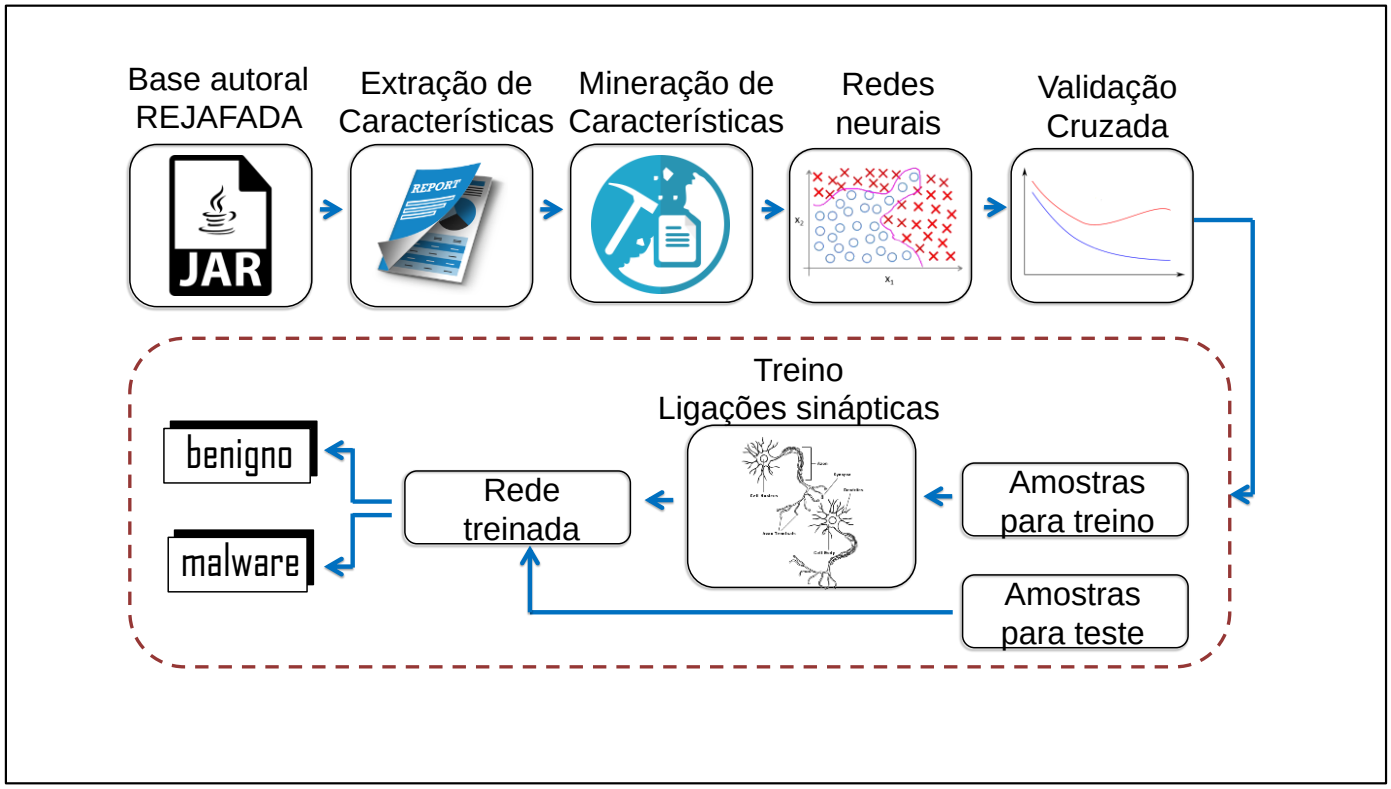

Figura 5. Metodologia Proposta.

A. *Base de Dados Autoral*

O presente trabalho emprega a REJAFADA (*Retrieval of JAR Files Applied to Dynamic Analysis* – Redistribuição de Arquivos JAR aplicados à Análise Dinâmica). Uma base de dados autoral que permite a classificação de arquivos de extensão JAR entre maliciosos e benignos. A REJAFADA é composta de 998 arquivos JAR *malwares* e outros 998 arquivos JAR benignos. A base de dados REJAFADA, consequentemente, é adequada ao aprendizado dotado de Inteligência Artificial, tendo em vista que os arquivos JAR apresentaram o mesmo montante nas duas classes (benignos e *malwares*). O objetivo é que classificadores tendenciosos, em relação a uma determinada classe, não tenham suas taxas de acerto favorecidas.

Em relação às pragas virtuais, a REJAFADA extraiu arquivos JAR maliciosos do VirusShare o qual é um repositório de malwares tendo como público alvo cientistas, analistas forenses e demais entusiastas da área de segurança da informação [27]. No que tange aos arquivos JAR benignos, o catálogo foi dado a partir de repositórios de aplicações como Java2s.com e findjar.com. Todos os arquivos benignos foram submetidos à auditoria do VirusTotal. Logo, os arquivos Jar benignos, contidos na REJAFADA, tiveram sua benevolência atestada por todos os antivírus comerciais mundiais. Isso posto, alguns dos aplicativos desprezados na confecção da base autoral podem ser falsos positivos. Tais aplicativos podem possuir um comportamento anômalo, porém sem realizar atividades de *malware*.

Cabe enfatizar que a nota de periculosidade do *CuckooBox* está em fase de desenvolvimento conforme a própria documentação da referida *SandBox*, Isso posto, o *CuckooBox* determina que um aplicativo suspeito não deve ser condenado em função apenas da sua nota de periculosidade emitida pela própria ferramenta. O *CuckooBox* não se propõe a ser uma unidade certificadora de aplicativos benignos ou de *malwares*.

O objetivo da referida *SandBox* é não se ater a eventos individuais e, portanto reconstruir a cadeia de eventos invocada pelos aplicativo suspeito.

Não há uma organização internacional, aos moldes da Interpol, de modo a ser uma unidade certificadora de *malwares* e consequentemente de aplicativos benignos. Em decorrência desse fato, os antivírus comerciais podem classificar como *malware* um determinado aplicativo sem maiores justificativas. Tal circunstância faz com que um antivírus comercial possa denunciar o instalador do antivírus concorrente como *malware* (conforme visto no vídeo apontado pelo *QR Code* da Fig 2 c). Esse fato atrapalha a construção de bases de dados dos antivírus inteligentes pois não há como determinar se o aplicativo é realmente um *malware* ou se trata de sabotagem industrial por parte de alguma fabricante de antivírus comercial.

Os resultados obtidos correspondentes às análises dos arquivos benignos e malwares, resultante da auditoria do VirusTotal, estão disponibilizados para consulta no endereço virtual da REJAFADA [19]. O objetivo da criação da base de dados REJAFADA é dar total possibilidade da metodologia proposta ser replicada, por terceiros, em trabalhos futuros. Logo, o REJAFADA disponibiliza, livremente, de todas as suas auditorias tanto benignas quanto *malwares*:

- auditorias do *VirusTotal*,
- análises dinâmicas do *Cuckoo Sandbox*,

A REJAFADA também disponibiliza, em seu endereço virtual, seus 998 arquivos JAR benignos. Além disso, a nossa base exibe a relação de todos os outros 998 arquivos JAR, dessa vez, *malwares*. Então, há a possibilidade da aquisição de todos os *malwares*, empregados pela REJAFADA, através do estabelecimento de acordo e submissão às normas de uso do VirusShare [27]. Conclui-se que a nossa base de dados REJAFADA viabiliza transparência e imparcialidade à pesquisa, além de demonstrar a veracidade dos resultados alcançados. Espera-se que o REJAFADA sirva de base para a criação de novos trabalhos científicos visando antivírus.

B. Mineração de Características

A mineração de dados está dentre os diversos desafios quanto à confecção de um repositório de aprendizado estatístico. Dados oriundos de uma aplicação real são altamente suscetíveis a aumentar demasiadamente a dimensionalidade da aplicação. Diversas técnicas podem ser usadas para superar esse desafio, visando melhorar a qualidade dos dados que é definida em termos da acurácia, completude, consistência, credibilidade e interpretabilidade.

No presente trabalho, os atributos auditados muitas vezes podem não fomentar capacidade de generalização aos classificadores. Como método de mineração de características, alguns comportamentos auditados pela Sandbox são desprezados. O critério adotado de mineração diz respeito à eliminação de características as quais dizem respeito a um único arquivo, com por exemplo; identificadores e nomes de processo, *md5*, *sha*, dentre outros. Dessa maneira, há a redução do volume de processamento sem haver a perda da capacidade de reconhecimento de padrão dos *malwares*.

C. Extração Dinâmica de Características

A seguir, são detalhados os grupos de características referentes ao monitoramento, em ambiente controlado, dos arquivos investigados. No total, são monitorados 6.824 comportamentos que o arquivo JAR suspeito possa fazer quando executado propositalmente em ambiente controlado (*Cuckoo Sandbox*).

- ✓ Características relacionadas ao tráfego de rede.
- ✓ Características relacionadas ao Registro (Regedit) do Windows 7.
  - Mudanças nas associações entre extensões de arquivos e *softwares* instalados na máquina.
  - Modificações nas informações sobre o usuário atual.
  - Corrupção do funcionamento dos *drivers*.
  - Alterações nas configurações de aparência do Windows e as configurações efetuadas pelos usuários, como papel de parede, protetor de tela e temas.
  - Mudanças nas Configurações de hardware.
- ✓ Características relacionadas a *Backdoos*. Programa que permite o retorno de um invasor a um computador comprometido, por meio da inclusão de serviços criados
- ✓ Características relacionadas a ameaças bancárias. Malware visando obter acesso a informações confidenciais e/ou materiais armazenadas ou processadas por meio de sistemas bancários online.
- ✓ Características relacionadas a Bitcoin. Examina-se caso o arquivo testado tenta instalar a biblioteca *OpenCL*, ferramenta para minerar Bitcoins.
- ✓ Características relacionadas a *Bots* (máquinas que desempenham tarefas automáticas em rede, maliciosas ou não, sem o conhecimento de seus proprietários).
- ✓ Características relacionadas a navegadores. É verificado se o arquivo testado tenta:
  - instalar um objeto *Browser Helper* (geralmente um arquivo DLL que adiciona novas funções ao navegador) com a finalidade de deixar a experiência de navegação prejudicada de alguma forma;
  - modificar as configurações de segurança do navegador;
  - modificar a página inicial do navegador;
  - adquirir informações privadas de navegadores de internet instalados localmente.
- ✓ Características relacionadas à computação em nuvem. O arquivo é auditado quando tenta se conectar aos serviços de armazenamento e/ou arquivos do *Dropbox*, *Google*, *MediaFire*, *MegaUpload*, *RapidShare*, *Cloudflare* e *Wetransfer*.
- ✓ Características relacionadas a ataques de DDoS (*Distributed Denial of Service* – Ataque Distribuído de Negação de Serviço).
- ✓ Características relacionadas a *Ransomware*; tipo de malware que por meio de criptografia, deixa inutilizados os arquivos da vítima, para depois solicitar um resgate em troca da normal utilização posterior dos arquivos do usuário, resgate este geralmente pago de forma não rastreável, como *bitcoins*.

- ✓ Características relacionadas aos arquivos executáveis. A forense digital proposta verifica se o arquivo suspeito tenta:
  - utilizar a ferramenta *BITSAdmin* (ferramenta de linha de comando originalmente utilizada para fazer download e upload de arquivos, assim como acompanhar o progresso desta transferência, mas que hackers utilizam de forma maliciosa) para baixar um arquivo qualquer;
  - travar, ao menos, um processo durante a sua execução;
  - executar a instrução *WaitFor* (executável presente no Windows desde sua versão 7, originalmente tem a função de sincronizar eventos entre computadores em rede, mas que malfeitores usam de maneiras prejudiciais), possivelmente para sincronizar atividades maliciosas.
- ✓ Características relacionadas a *exploits* os quais se constituem como malwares que tentam se utilizar de vulnerabilidades, falhas ou defeitos conhecidos e ainda não sanados do sistema ou de um ou mais de seus componentes, a fim de causar instabilidades e comportamentos imprevistos tanto em seu hardware como em seu software.
- ✓ Características relacionadas a *Infostealers*, programas maliciosos que coletam informações confidenciais do computador afetado.

Além da detecção de comportamentos suspeitos, como chamadas a APIs, a análise dinâmica também permite a reconstituição (limpeza) do SO (Sistema Operacional) através da auditoria das malfeitorias promovidas pelo arquivo malicioso partindo do princípio que o malefício não é irreversível. Cabe ressaltar que a reconstituição do SO, tecnicamente nomeada de vacina, é importante porque não basta apenas a detecção e eliminação do malware para que a vítima esteja livre de sua atuação. Além da eliminação do *malware*, é necessário desfazer todas as suas malfeitorias como, por exemplo, ter desabilitado *Java Security Manager*. Então, caso não houvesse a auditoria, provida pela análise dinâmica, caberia ao cyber-vigilante monitorar, manualmente, qualquer mudança no SO o que tornaria o processo moroso e estressante.

D. Classificadores: Redes Neurais ELMs

Quanto ao reconhecimento de padrão de *malwares*, uma tarefa essencial diz respeito à atribuição de uma classe (rótulo) a cada arquivo investigado a partir de suas características. Então, com base em um conjunto de arquivos, chamado de conjunto de treinamento, é possível formular uma hipótese sobre as distintas classes atreladas ao antivírus. Logo, cabe ao classificador, estimar a classe de um arquivo inédito através da comparação entre as características do seu comportamento auditadas em tempo real e aquelas captadas durante a etapa de treinamento.

O objetivo do classificador é obter uma função de separação entre as classes do nosso antivírus (e.g.: *malware*, benigno). Dessa forma, ao ser apresentado a um aplicativo inédito, a função é aplicada e, então, atribui uma classe na qual esse arquivo suspostamente pertence. Matematicamente, $c = f(x)$, onde $x = x_1, x_2, \dots, x_t$ é um vetor características extraídas do arquivo investigado, t corresponde às 7.690 características dinâmicas, c é o classe (rótulo) e, por fim, f é a função de mapeamento do classificador.

O nosso antivírus proposto aqui emprega redes neurais extremas como classificadores [6]. As arquiteturas das redes neurais têm uma camada de entrada contendo uma quantidade de neurônios referentes ao vetor de características extraídas do monitoramento do arquivo JAR em ambiente controlado. Logo, os classificadores empregados devem possuir uma camada de entrada contendo 6.824 neurônios. A camada de saída possui dois neurônios correspondentes a casos benignos e malwares.

Há a investigação quanto à quantidade de neurônios da camada escondidas das redes ELMs. A hipótese é verificar se arquiteturas que exijam um maior volume de cálculos, como por exemplo, aumentar a quantidade de neurônios na camada escondida, são capazes de gerar taxas de acertos superiores em comparação com arquiteturas que exijam uma menor quantidade de cálculos. Há a avaliação de duas arquiteturas, elas empregam 100 e 500 neurônios em suas respectivas camadas escondidas. Tais arquiteturas possuem lastro de excelentes acurácias na aplicação de redes ELM na área de Engenharia Biomédica [14].

E. Validação Cruzada

São investigados 10 *folds*, referentes à validação cruzada do método *k-fold*, para cada *kernel*. O objetivo é que os resultados alcançados não sejam influenciados pelos conjuntos destinados ao treinamento e teste. Para isso, o total de aplicativos é dividido em dez partes. Na primeira execução, a primeiro parte é destinada ao conjunto de teste, enquanto as demais são reservadas ao treinamento. Essa alternância ocorre por dez execuções até que todas as dez partes tenham sido aplicadas à fase de teste. A acurácia do ELM é a média aritmética da taxa de acerto obtida nas dez iterações para cada combinação.

Como dito anteriormente, na rede ELM não há retropropagação de dados. Logo, o objetivo do método de validação cruzada *k-fold* não é estabelecer um critério de parada para evitar o *overfitting (treinamento em excesso)*, mas sim verificar se o classificador sofre mudanças abruptas em suas acurácias a depender dos conjuntos destinados ao treinamento e teste. Também há o cuidado metodológico de selecionar equitativamente, de forma randômica, exemplares benignos e malwares para cada *fold*. O objetivo é que classificadores tendenciosos, em relação a uma determinada classe, não tenham suas taxas de acerto favorecidas.

VI. Resultados das Redes ELMs

O trabalho proposto utiliza sete redes ELMs cada uma dotada de um tipo de *kernel* distinto. No estado-da-arte, três desses *kernels* são descritos por HUANG, et al, (2012), são eles; Transformada de Wavelets, *Hard Limite e* Função de Base Triangular [6]. Além disso, são empregados quatro *kernels* autorais; *Fuzzy*-Dilatação, *Fuzzy*-Erosão, Dilatação e Erosão.

As fmELMs (*fuzzy*-morfológica ELMs) têm obtido sucesso no tratamento de imagens biomédicas, especificamente, na

detecção e classificação de câncer de mama [1]. A solução autoral diz respeito à criação de aproximações lineares dos operadores morfológicos clássicos de forma que seus tempos de execução e andamentos se mantenham uniformes independente dos valores dos dados de entrada. Logo, os desvios condicionais são substituídos por operações aritméticas que apresentam tempo de execução uniforme além de computacionalmente menos onerosas do que os desvios condicionais. Apesar de treinamento rápido, as fmELMs não são completamente adaptáveis a distribuições não-lineares como as distribuições sigmoide ou senoidal.

O *kernel* Wavelets não possui camada escondida [6]. Os cálculos são baseados na transformação dos dados de entrada e podem trabalhar de maneira aproximadamente similar aos *kernels* contendo arquiteturas dotadas de camadas escondidas [6]. Uma boa capacidade de generalização desses *kernels* depende de uma escolha ajustada de parâmetros (C, γ). O parâmetro de custo C se refere a um ponto de equilíbrio razoável entre a largura da margem do hiperplano e a minimização do erro de classificação em relação ao conjunto de treinamento. O parâmetro do kernel γ controla o limite de decisão em função das classes [6]. Não existe um método universal no sentido de escolher os parâmetros (C, γ). No presente trabalho, os parâmetros C e γ variam exponencialmente em sequencias crescentes, matematicamente de acordo com a função 2^n, onde $n = \{-24, 10, 0, 10, 25\}$. A referida distribuição apresenta excelentes acurácias tanto para regressão quanto reconhecimento de padrões em obras do estado-da-arte [6]. A hipótese é verificar se esses parâmetros distintos dos padrões; $(C = 1, \gamma = 1)$, são capazes de gerar melhores acurácias.

A Tabela 4 detalha os resultados obtidos pelas redes ELMs com o *kernel* Wavelets. Cada linha da Tabela 4 contém 10 execuções referentes à validação cruzada do método *k-fold*, onde $k = 10$. Em relação à precisão na fase de teste, o máximo desempenho médio foi de 56,01% na distinção entre casos benignos e *malwares* através do *kernel* Polinomial dotado dos parâmetros $(C, \gamma) = (2^{-24}, 2^0)$. Na Tabela 4, há apenas as descrições do melhor e pior caso, nessa ordem.

A Tabela 5 detalha os resultados obtidos pelas redes ELMs com os *kernels Hard* Limite, Função de Base Triangular, *Fuzzy*-Dilatação, *Fuzzy*-Erosão, Dilatação e Erosão. Todos os kernels mencionados empregam arquiteturas dotadas de camada escondida. Cada linha da Tabela 5 contém 10 execuções distintas referentes ao método *k-fold*, onde $k = 10$. Em relação à acurácia, o máximo desempenho médio foi de 91,58% com desvio padrão de 1,77% através do mELM *kernel* Dilatação dotado de 500 neurônios na sua camada escondida.

Tabela 4: Resultados das redes ELMs. Os parâmetros (C, γ) variam de acordo com o conjunto $\{2^{-24}, 2^{-10}, 2^0, 2^{10}, 2^{25}\}$.

Kernel	(C, γ)	***Acerto treino (%)***	***Acerto teste (%)***	***Tempo treino (seg.)***	***Tempo teste (seg.)***
Wavelets	$(2^{-24}, 2^0)$	**100,00 ± 0,00**	**56,01 ± 2,78**	3,11 ± 0,07	0,76 ± 0,03
	$(2^{10}, 2^{25})$	67,40 ± 1,97	47,91 ± 3,76	3,12 ± 0,08	0,78 ± 0,03

Tabela 5: Resultados das redes ELMs. A quantidade de neurônios na camada escondida varia de acordo com o conjunto $\{100, 500\}$.

Kernel	**Neurônios**	***Acerto treino (%)***	***Acerto teste (%)***	***Tempo treino (seg.)***	***Tempo teste (seg.)***
Hard limite	100	50,03 ± 0,00	49,75 ± 0,00	0,48 ± 0,01	0,03 ± 0,01
	500	50,03 ± 0,00	49,75 ± 0,00	2,51 ± 0,03	0,13 ± 0,02
Base triangular	100	50,00 ± 0,03	50,00 ± 0,26	0,49 ± 0,02	0,02 ± 0,01
	500	50,00 ± 0,03	50,00 ± 0,26	1,76 ± 0,05	0,14 ± 0,01
Fuzzy-Dilatação	500	95,71 ± 0,28	87,28 ± 2,40	2,01 ± 0,05	0,14 ± 0,02
	100	84,37 ± 0,29	81,61 ± 2,32	0,55 ± 0,02	0,03 ± 0,01
Fuzzy-Erosão	500	95,70 ± 0,33	87,72 ± 2,55	2,16 ± 0,07	0,16 ± 0,01
	100	84,67 ± 0,37	82,16 ± 1,93	0,65 ± 0,01	0,03 ± 0,01
Dilatação	500	**97,63 ± 0,13**	**91,58 ± 1,77**	52,36 ± 0,57	5,68 ± 0,05
	100	81,20 ± 0,35	78,86 ± 1,18	7,34 ± 0,18	0,77 ± 0,02
Erosão	500	78,38 ± 0,24	70,58 ± 2,81	53,23 ± 2,41	5,78 ± 0,23
	100	54,99 ± 0,11	53,05 ± 1,95	8,17 ± 0,15	0,86 ± 0,03

VII. Resultados em Comparação ao Estado-da-Arte

Nesta seção, o antivírus autoral é comparado com os antivírus do estado-da-arte. A fim de evitar comparações injustas, a etapa de extração de características é padronizada através do monitoramento de 6.824 comportamentos que o arquivo JAR suspeito posso fazer quando executado propositalmente. O antivírus autoral emprega redes neurais

morfológicas rasas. O nosso antivírus é dotado do *kernel* mELM Dilatação e contém 500 neurônios em sua camada escondida.

Por outro lado, o antivírus confeccionado por LIMA, *et al*. (2021) emprega redes neurais rasas baseadas em retropropagação. LIMA, *et al.* (2021) averigua onze distintas funções de aprendizado com o objetivo de otimizar a precisão do seu antivírus. Para cada função de aprendizado, LIMA, *et al.* (2021) explora 4 arquiteturas de camadas escondidas. O antivírus autoral também é comparado com antivírus baseados em redes neurais profundas convolucionais [5][25][26].

Por fim, o antivírus autoral também é comparado à rede neural profunda de SANTOS, *et al*, 2019. Tal obra do estado-da-arte não visa a detecção de *malwares* e, sim, o reconhecimento óptico de caracteres. Logo, a técnica de SANTOS, *et al*, 2019 sofre uma adaptação em sua camada de neurônios de entrada. Ao invés de processamento digital de imagem, os atributos de entrada dizem respeito à extração de características dos *malwares*. A rede neural profunda de SANTOS, *et al.* 2019 apresenta uma única camada convolucional e não há retropropagação de dados. A camada convolucional emprega todos seus 30 mil filtros simultaneamente de modo que o tempo de treinamento é compatível com aplicativos que precisam de treinamento frequentemente como os antivírus.

A Figura 6 e a Figura 7 são representações gráficas dos resultados descritos na Tabela 6. A Figura 6 (a) apresenta os *boxplots*, da etapa de treinamento, tanto do antivírus autoral quanto do estado-da-arte. A melhor acurácia média, resultante do treinamento, foi de 98,33% através do antivírus de VINAYAKUMAR, R. *et al.* (2019). A rede neural profunda de SANTOS, *et al.* (2019) obtém uma acurácia de treino de 50,00% em média. O antivírus de LIMA, *et al.* (2021) obteve precisão média de 48,82% e 98,82%, no seu pior e melhor cenário, respectivamente. Tais resultados foram obtidos usando as funções de aprendizado "*Batch training–learning rules*" e "*Conjugate gradiente backpropagation with Fletcher-Reeves updates",* respectivamente, com 100 neurônios em suas camadas ocultas. O antivírus autoral obteve um desempenho médio de 97,63% com desvio padrão de 0,13%. Logo, o antivírus criado apresenta a vantajosa característica de não sofrer mudanças abruptas em função das condições iniciais (*k-fold*).

A Figura 6 (b) exibe os *boxplots* referentes às melhores precisões na fase de teste. A melhor precisão média, resultante do teste, foi de 96,54% através do antivírus VINAYAKUMAR, R. *et al.* (2019). O antivírus autoral obteve uma acurácia média de 91,58%. A rede neural profunda de SANTOS, *et al* (2019) alcançou um desempenho médio de 50%. O antivírus de LIMA, *et al* (2021) obteve precisão média de 50,26% e 95,67%, no seu pior e melhor cenário, respectivamente. Logo, corrobora-se que redes neurais baseadas em retropropagação podem sofrer variações abruptas, em suas acurácias, a depender dos seus parâmetros de configurações. Então, foi salutar a decisão de LIMA, *et al.* (2021) em explorar distintas funções de aprendizado, gradientes e arquiteturas de modo a otimizar a precisão de suas redes neurais baseadas em retropropagação de dados.

A Figura 7 (a) e Figura 7 (b) apresentam os *boxplots* referentes aos tempos gastos durante a fase de treinamento e de teste, respectivamente. O antivírus autoral consome apenas 52,36 segundos para concluir, em média, o seu treinamento. Em relação ao tempo de treinamento, o antivírus de VINAYAKUMAR, R. *et al.* (2019) é mais lento em comparação aos demais visto que há a utilização de hierarquia de rede profunda. O antivírus de VINAYAKUMAR, R. *et al.* (2019) leva cerca de 2 (dois) dias para concluir seu treinamento. A rede neural profunda de SANTOS, *et al* (2019) consome 695,26 segundos em seu treinamento visto que são empregados 30.000 filtros convolucionais de forma paralela e sem retropropagação de dados. O antivírus de LIMA, *et al.* (2021) conclui seu treinamento na ordem de segundos porque são empregadas redes neurais rasas. Em relação ao tempo consumido durante a fase de teste, as aplicações consumiram tempos bastante próximos sem grandes discrepâncias.

Tabela 6: Comparação entre o antivírus proposto e o estado-da-arte.

Técnica	***Acerto treino (%)***	***Acerto teste (%)***	***Tempo treino (seg.)***	***Tempo teste (seg.)***
Antivírus autoral	97,63 ± 0,13	91,58 ± 1,77	52,36 ± 0,57	5,68 ± 0,05
Antivírus de LIMA, *et al* (2021), pior conf.	50,26 ± 0,89	50,26 ± 0,71	9,67 ± 1,76	0,44 ± 0,14
Antivírus de LIMA, *et al* (2021), melhor conf.	96,71 ± 2,10	95,67 ± 1,85	580,07 ± 228,44	0,46 ± 0,19
Antivírus de SU, *et al.* (2018)	78,37 ± 0,47	78,31 ± 3,37	3337,15 ± 19,08	3,53 ± 0,14
Antivírus de VINAYAKUMAR, *et al.* (2019)	**98,33 ± 5,16**	**96,54 ± 4,84**	149968,09 ± 33112,72	71,64 ± 23,13
Antivírus de HARDY, *et al.* (2016)	99,57 ± 1,17	96,49 ± 1,89	6854,74 ± 300,59	0,14 ± 0,01
Deep Learning de SANTOS, *et al* (2019)	50,00 ± 0,00	50,00 ± 0,00	695,26 ± 29,67	7,90 ± 2,56

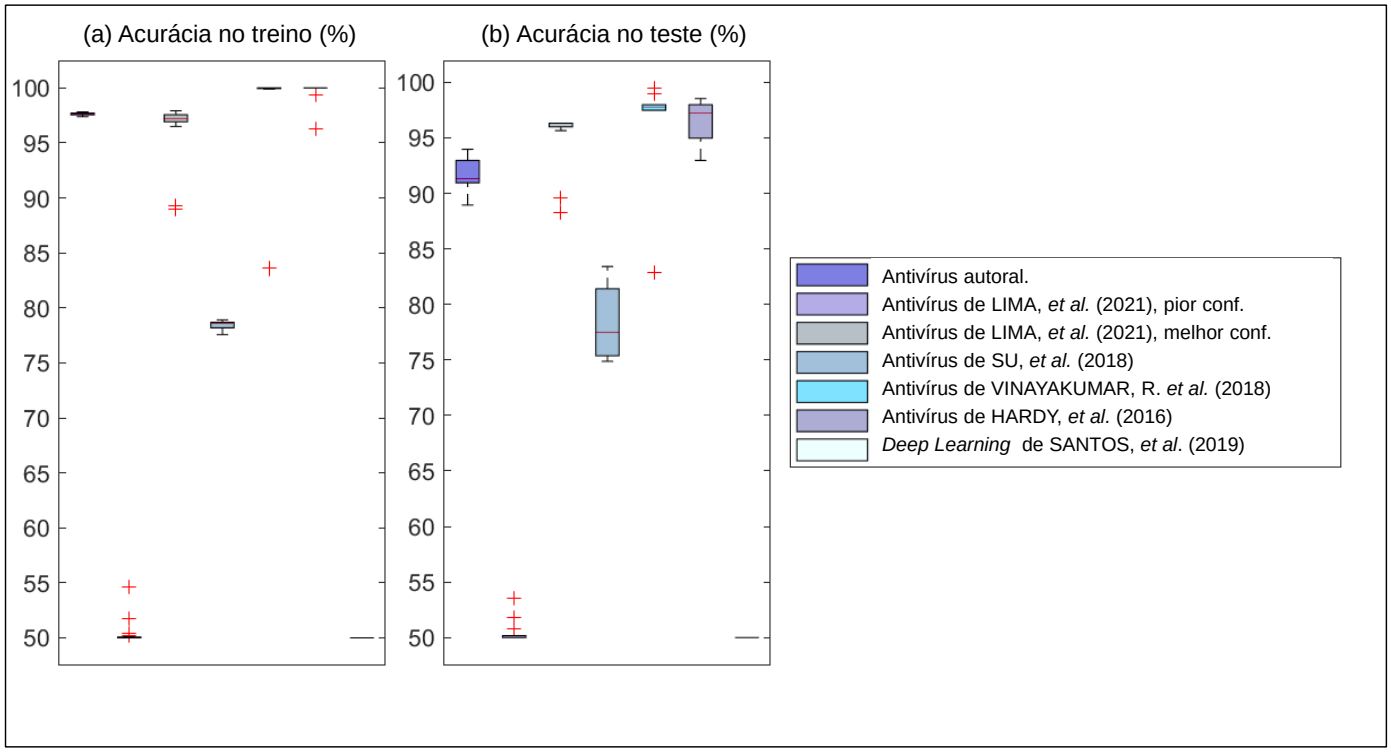

Figura 6. Boxplots referentes às acurácias do antivírus autoral e dos antivírus do estado-da-arte.

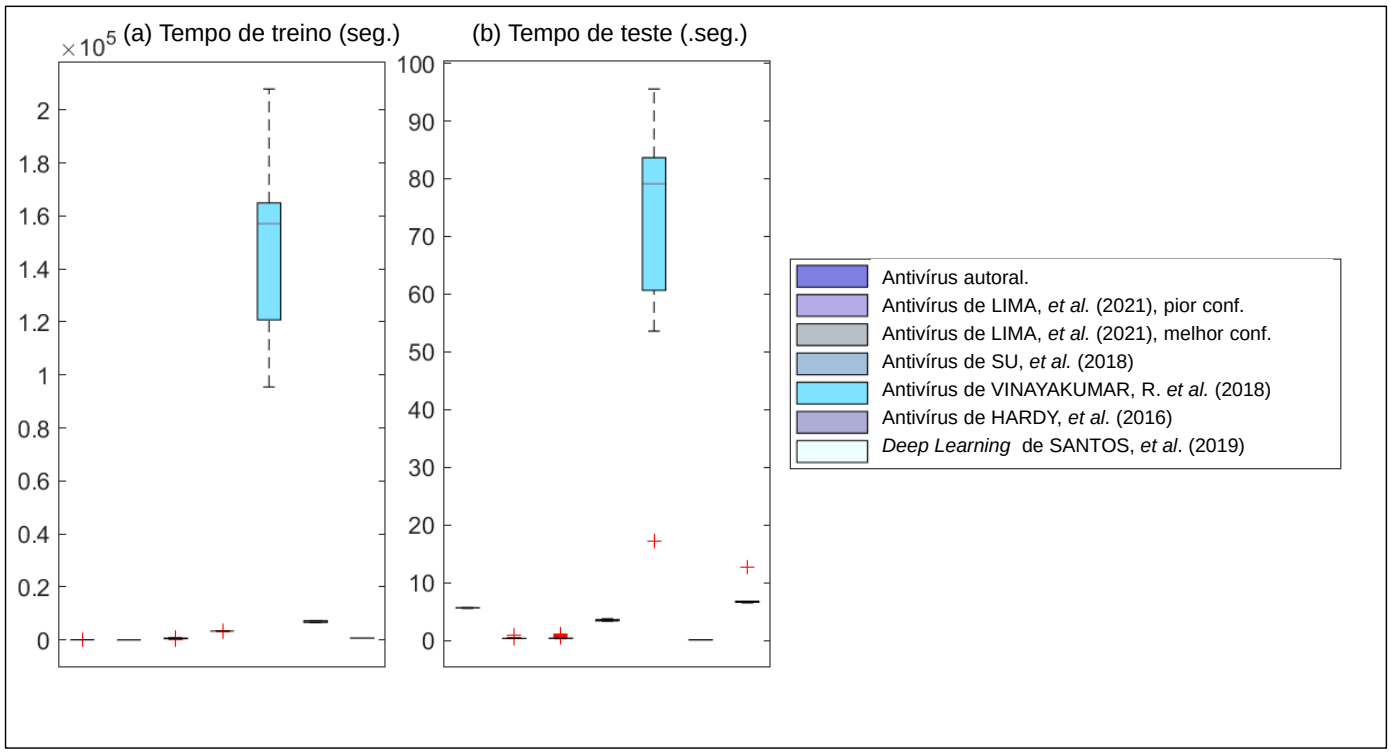

Figura 7. Boxplots referentes aos tempos consumidos pelo antivírus autoral e pelos antivírus do estado-da-arte.

A Tabela 7 exibe as matrizes de confusão das técnicas apresentadas na Tabela 6 em termos percentuais. A matriz de confusão é importante para a verificação da qualidade de um aprendizado supervisionado. Na Tabela 7, B. e M. são abreviaturas de Benigno e *Malware*. As classes desejadas estão dispostas no rótulo vertical enquanto as classes obtidas estão no rótulo horizontal.

Na Tabela 7, por exemplo, o antivírus proposto, na fase de teste, classificou em média, de maneira equivocada, 10,79% casos como benignos quando se tratavam de casos malignos

(falso negativo). Seguindo o mesmo raciocínio, houve a classificação média de 5,49% casos ditos como *malwares* quando se tratava de aplicações benignas (falso positivo). Na matriz de confusão, a diagonal principal é ocupada por casos nos quais a classe obtida coincide com a classe desejada. Então, um bom classificador deve ter uma diagonal principal ocupada por valores altos enquanto as demais posições devem possuir valores baixos. Na Tabela 7, as diagonais principais estão em negrito.

Ainda quanto à Tabela 7, sensibilidade e especificidade dizem respeito à capacidade do antivírus identificar os aplicativos *malwares* e benignos, respectivamente. O trabalho proposto apresenta a matriz de confusão em termos percentuais de modo a facilitar a interpretação da sensibilidade e especificidade. Em síntese, a sensibilidade e a especificidade estão apresentadas na própria matriz de confusão, descrita na Tabela 7. Por exemplo, o antivírus proposto alcança, em média, 94,51% tanto em relação à sensibilidade quanto a verdadeiros positivos. Seguindo o mesmo raciocínio, o antivírus autoral obtém, em média, 89,21% tanto para especificidade quanto para verdadeiros negativos.

Na perícia forense digital, um falso positivo implicaria em um aplicativo benigno impedido de ser executado pelo sistema. Um falso negativo, no entanto, pode implicar em um *malware* que não foi detectado. Vale salientar que os *malwares* podem gerar malefícios irreversíveis e irrecuperáveis para toda a rede mundial de computadores. Isso posto, um falso negativo pode implicar na perda da dignidade, das finanças e da saúde mental da vítima. Enfatiza-se que o antivírus autoral apresenta o percentual médio de falsos negativos de apenas 5,49%.

A

Tabela **8** exibe os testes de hipótese paramétrico *t-students* e não-paramétrico *Wilcoxon* entre o antivírus criado e o estado-da-arte. É possível concluir que o antivírus autoral é estatisticamente distinto em comparação a todas as demais amostras. A explicação é que, tanto no teste paramétrico *t-students* quanto no teste não-paramétrico *Wilcoxon*, a hipótese nula foi rejeitada. Logo, as qualidades dos antivírus propostos são estatisticamente distintas.

O antivírus autoral demonstrou uma grande vantagem quando comparado ao estado-da-arte. O nosso antivírus foi capaz de alcançar a melhor precisão média com 91,58% de acurácia acompanhada de um tempo de treinamento correspondente a 52,36 segundos. A relação entre a acurácia percentual e tempo de treinamento em ordem inversa é empregada em Engenharia Biomédica [14]. Admite-se que o estabelecimento de tal relação assume papel importante na área de Segurança da Informação visto que são lançados 8 (oito) novos *malwares* por segundo [8]. Então, paradoxalmente um antivírus recém-lançado já pode estar obsoleto e necessitar de um novo treinamento mediante uma vulnerabilidade recém-descoberta. Em síntese, o tempo de aprendizado de um antivírus não deve ser discrepante em comparação à taxa de criação de novos *malwares* mundialmente.

Tabela 7: Matrizes de confusão [9] das técnicas apresentadas na Tabela 6 (%).

Técnica		***Treino***		***Teste***	
		M.	B.	M.	B.
Antivírus autoral	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**
Antivírus de LIMA, *et al* (2021), pior conf.	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**
Antivírus de LIMA, *et al* (2021), melhor conf.	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**
Antivírus de SU, *et al.* (2018)	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**
Antivírus de VINAYAKUMAR, *et al.* (2019)	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**
Antivírus de HARDY, *et al.* (2016)	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 de SANTOS, *et al* (2019)	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**

[9] Matriz de confusão. Disponível em: https://www.mathworks.com/help/stats/confusionmat.html. Acesso em maio de 2020.

Tabela 8: Teste de hipótese t-students e Wilcoxon entre o antivírus criado e o estado-da-arte.

Comparação	***t-students(teste paramétrico)***		***Wilcoxon (teste não-paramétrico)***	
	Hipótese	***Valor p***	***Hipótese***	***Valor p***
Antivírus autoral *vs* LIMA, *et al* (2019), pior conf.	1	4,2134e-41	1	1,30487e-11
Antivírus autoral *vs* LIMA, *et al* (2019), melhor conf.	1	3,81625e-09	1	2,86398e-09
Antivírus autoral *vs* Antivírus de SU, *et al.* (2018)	1	6,621e-19	1	2,5046e-11
Antivírus autoral *vs* Antivírus de VINAYAKUMAR, *et al.* (2019)	1	3,803e-06	1	8,83703e-08
Antivírus autoral *vs* Antivírus de HARDY, *et al.* (2016)	1	1,33009e-11	1	5,13671e-10
Antivírus autoral *vs* *Deep Learning* de SANTOS, *et al* (2019)	1	5,42386e-42	1	1,02645e-12

VIII. CONCLUSÃO

A cada ano milhares de *malwares* são identificados em crescimento e proporções contínuas [7]. Portanto, é de vital importância que as plataformas de detecção de *malwares* forneçam mecanismos de *cyber*-vigilância que atendam as demandas dos clientes de forma preventiva. Caso contrário, nos cenários em que ocorrem falhas na identificação de aplicativos maliciosos, há a iminência dos dados confidenciais dos clientes sejam disponibilizados a pessoas não autorizadas. Mundialmente, as vulnerabilidades em Java correspondem a 91% de todas as *cyber*-infecções monitoradas [4]. Além de computadores pessoais, as vulnerabilidades em Java são capazes de corromper aplicações webs coorporativas sendo responsáveis pela maioria das *cyber*-infecções na rede mundial de computadores [4].

Infere-se que a seleção do antivírus possui um importante papel no combate a pragas virtuais. Na nossa avaliação, a variação quanto à detecção *malwares* foi de 0% e 99,10%, a depender de qual antivírus comercial escolhido. O presente artigo realizou a análise de 86 antivírus comercialmente disponibilizados. Na média, eles conseguiram detectar 34,95% dos *malwares* averiguados. Feita a análise das amostras, foi possível identificar que os antivírus, em média, reportaram falsos negativos e omitiram-se em 33,90% e 31,15% dos casos, respectivamente. No presente trabalho, usou-se a plataforma VirusTotal para submeter, de forma automatizada, os *malwares* aos antivírus comerciais. Cabe salientar que no VirusTotal, não há a possibilidade de escolha da versão *shareware* dos antivírus. Então, não foi possível fazer comparações entre os recursos comercias e gratuitos de um mesmo antivírus. É razoável supor que os serviços ofertados nas versões *shareware* apresentam um desempenho significativamente inferior às versões completas.

Na média, 31,39% dos antivírus pagos (completos), submetidos à nossa avaliação, não foram capazes de detectar nenhum dos *malwares*. Constata-se, a falha na eficácia dos antivírus comerciais no tocante a serviços em larga escala e em tempo real. Ressalta-se que em nosso estudo, os *malwares* analisados são de domínio público, empregados em atividades maliciosas, e com as suas atuações catalogadas por institutos de pesquisa [27]. Mesmo assim, mais da terça parte dos antivírus comerciais avaliados não tinham qualquer conhecimento sobre as existências dos arquivos *malwares* investigados.

Visando suprir as limitações dos antivírus comerciais, os antivírus baseados em inteligência artificial são capazes de averiguar milhares de *malwares* e aprender, de forma estatística, quais são as suas características maliciosas. Portanto, após o aprendizado, os antivírus inteligentes podem identificar e classificar *malwares* recém-criados de acordo com a comparação entre suas características e as aquelas catalogadas durante a sua fase de aprendizado. Em síntese, pode-se tornar autônoma a aprendizagem do comportamento dos *malwares*. Sendo assim, não haveria a necessidade de esperar que um usuário seja contaminado, e posteriormente, denuncie uma atitude suspeita, para, só então tomar alguma atitude em relação à descoberta de um malware. Os antivírus possibilitam a detecção preventiva das ameaças virtuais, em ambiente controlado, antes de alcançarem as máquinas dos clientes.

Este trabalho cria um antivírus capaz de classificar arquivos suspeitos entre benignos e *malwares*. Ao todo, nosso antivírus monitora e pondera, estatisticamente, ações que o arquivo suspeito possa fazer quando executado na JVM contida no Windows 7. Em ambiente controlado, o nosso antivírus monitora alterações no Registro (Regedit), vestígios de chamadas realizadas por todos os processos gerados pelo malware, arquivos sendo criados, excluídos e baixados pelo malware durante sua execução, forenses de memória e rastreamento de tráfego de rede. O reconhecimento de padrão, quanto às 6.824 ações suspeitas, é feito por redes neurais extremas. As condições iniciais das redes neurais e suas

arquiteturas foram alternadas com o objetivo de maximizar a acurácia da classificação.

Ao invés de *kernels* convencionais, são empregados *kernels* autorais para as redes neurais extremas. As redes extremas têm como principal característica a velocidade de treinamento e predição de dados quando comparadas às redes neurais baseadas em retropropagação. O *kernel* autoral de Dilatação consegue distinguir aplicativos *malwares* dos benignos em 91,58% dos casos de teste acompanhado de um tempo de treinamento de 52,36 segundos.

O nosso antivírus pode ser estendido no sentido de prover *cyber*-proteção a outros sistemas operacionais dotados de JVM. Então, a intenção é estender o nosso antivírus a outros sistemas operacionais além do Windows. A meta futura é aplicar a nossa metodologia ao sistema Android visto que *smartphones* e *tablets* estão gradativamente se tornando indispensáveis na sociedade contemporânea. Também como objetivo futuro, o nosso antivírus visa auditar SOs, dotados de JVM, especializados em transações financeiras como cartões de créditos, passes de transporte e terminais lotéricos.

REFERENCES

[1] AZEVEDO, W. W. E. A. 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, 2015 a.

[2] AZEVEDO, W. W. E. A. 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, 2015 b.

[3] AZEVEDO, W. W. et al. 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. doi: https://doi.org/10.1016/B978-0-12-819295-5.00003-2, 2020.

[4] CISCO. Cisco 2014 Annual Security Report, Disponível em: http://www.efocus.sk/images/uploads/Cisco_2014_ASR.pdf.Acesso em junho de 2019, 2014.

[5] HARDY, W.; LINGWEI, C. DL 4 MD: A Deep Learning Framework for Intelligent Malware Detection, In Int'l Conf. Data Mining. pág. 61-67, 2016.

[6] HUANG, G. B. et al. Extreme Learning Machine for Regression and MultiClass Classification. IEEE Transactions on Systems, Man, and Cybernetics. Volume 42, número 2, pág. 513-529: [s.n.]. 2012.

[7] IBM. 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, 2014.

[8] INTEL. McAfee Labs: Threat Report. , Disponível em: https://www.mcafee.com/ca/resources/reports/rp-quarterly-threats-mar-2018.pdf. Acesso em março de 2020, 2018.

[9] JAVA. Java Technology, Disponível em: https://www.Java.com. Acesso em junho de 2021, 2019.

[10] LIMA, S. M. L. 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, 2021 b.

[11] LIMA, S. M. L.; SILVA, H. K. D. L. Artificial intelligence-based antivirus in order to detect malware preventively. Progress in Artificial Intelligence.10, 1–22 (2021) doi: https://doi.org/10.1007/s13748-020-00220-4, 2021 a.

[12] LIMA, S. M. L.; SILVA, H. K. L. E. A. Antivírus dotado de Máquina Morfológica de Aprendizado Extremo., SISTEMAS DE INFORMAÇÃO (MACAÉ), v. 26, p. 31-50, 2020 b.

[13] LIMA, S. M. L.; SILVA-FILHO, A. G.; DOS SANTOS, W. P. 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, 2014.

[14] LIMA, S. M. L.; SILVA-FILHO, A. G.; SANTOS, W. P. Detection and classification of masses in mammographic images in a multi-kernel approach, Computer Methods and Programs in Biomedicine.doi: https://doi.org/10.1016/j.cmpb.2016.04.029, 2016.

[15] LIMA, S. M. L.; SILVA-FILHO; SANTOS, W. P. Morphological Decomposition to Detect and Classify Lesions in Mammograms.In: Wellington Pinheiro dos Santos; Maíra Araújo de Santana; Washington Wagner Azevedo da Silva. (Org.). Understanding a Cancer Diagnosis, New York: Nova Science. https://novapublishers.com/shop/understanding-a-cancer-diagnosis/, 2020 a.

[16] ORACLE. JavaOne 2012 Review: Make the Future Java, Disponível em: http://www.oracle.com/technetwork/articles/Java/Javaone12review-1863742.html. Acesso em junho de 2019, 2012.

[17] PAUL, N.; EVANS, D. Comparing Java and.NET security: Lessons learned and missed, Computers & Security. Volume 25, pág. 338-350, 2016.

[18] PEREIRA, J. M. S. E. A. Method for Classification of Breast Lesions in Thermographic Images Using ELM Classifiers. In: Wellington Pinheiro dos Santos; Maíra Araújo de Santana; Washington Wagner Azevedo da Silva. (Org.). Understanding a Cancer Diagnosis, New York: Nova Science. doi: https://novapublishers.com/shop/understanding-a-cancer-diagnosis/, 2020.

[19] REJAFADA. A Retrieval of Jar Files Applied to Malware Dynamic Analysis - Redistribuição de Arquivos Jar aplicados à Análise Dinâmica de Malwares, Disponível em: https://github.com/rewema/rejafada. Acesso em junho de 2019, 2019.

[20] SALEHI, Z.; SAMI, A.; GHIASI, M. Using features generation from API calls form malware detection, Computer Fraud & Security. Volume 9, pág. 9 -18, 2014.

[21] SANS. SANS Institute InfoSec Reading Room. Out with The Old, In with The New: Replacing Traditional Antivirus, Disponível em: https://www.sans.org/reading-room/whitepapers/analyst/old-new-replacing-traditional-antivirus-37377. Acesso em junho de 2019, 2016.

[22] SANTOS, M. M. *et al*. Deep convolutional extreme learning machines: Filters combination and error model validation, Neurocomputing. doi: https://doi.org/10.1016/j.neucom.2018.10.063, 2019.

[23] SANTOS, W. P. Mathematical Morphology In Digital Document Analysis and Processing, New York: Nova Science. Capítulo 8, pág 159-192., 2011.

[24] SHAHZAD, F.; SHAHZAD, M.; F., M. In-execution dynamic malware analysis and detection by mining information in process control blocks of Linux OS, Information Sciences. Volume 231, pág. 45–63, 2013.

[25] SU, J.; VASCONCELLOS, D. Lightweight Classification of IoT Malware Based on Image Recognition, 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC). https://doi.org/10.1109/COMPSAC.2018.10315, 2018.

[26] VINAYAKUMAR, R.; SOMAN, K. P. DeepMalNet: Evaluating shallow and deep networks for static PE malware detection, ICT Express. doi: https://doi.org/10.1016/j.icte.2018.10.006, 2018.

[27] VIRUSSHARE. VirusShare: malware files database, Disponível em: https://virusshare.com. Acesso em junho de 2019 , 2019.

[28] VIRUSTOTAL. VirusTotal: Serviço online quanto à identificação de malwares pelos principais antivírus comerciais, Disponível em: https://www.virustotal.com/. Acesso em maio de 2021, 2021.

[29] WAGNER, G.; GAL, A.; FRANZ, M. "Slimming" a Java virtual machine by way of cold code removal and optimistic partial program loading, Science of Computer Programming. Volume 76, pág.1037-1053, 2011.