Revista de Sistemas de Informação da FSMA
n. 29 (2022) pp. 25-44

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

TRILHA PRINCIPAL

Antivírus aplicado à detecção de *malware* IoT com base em comportamentos em tempo de execução

Sthéfano H. M. T. Silva, *Departamento de Computação – Universidade de Pernambuco (UPE)*,
Sidney M. L. de Lima, *Departamento de Eletrônica e Sistemas (DES) - Universidade Federal de Pernambuco (UFPE)*,
Ricardo P. Pinheiro, *Departamento de Computação – Universidade de Pernambuco (UPE)*,
Rafael D. T. de Lima, *Departamento de Computação – Universidade de Pernambuco (UPE)*,
Liosvaldo M. S. de Abreu, *Departamento de Eletrônica e Sistemas (DES) - Universidade Federal de Pernambuco (UFPE)* e
Sérgio M. M. Fernandes, *Departamento de Computação – Universidade de Pernambuco (UPE)*

***Resumo*—Atualmente, a Internet das Coisas (IoT) tem um impacto significativo na vida das pessoas, atingindo centenas de bilhões de dispositivos conectados à Internet. Devido à popularidade dos dispositivos inteligentes, o número de ataques cibernéticos voltados para a tecnologia aumentou nos últimos anos. O constante surgimento de novos *malwares* voltados para IoT, como a *botnet*, o uso de técnicas complexas de obfuscação e evasão e, muitas vezes, a disponibilidade de grandes recursos para seu desenvolvimento, faz dele o maior vilão cibernético nos atuais cenários de IoT. O presente trabalho cria um Antivírus para Análise Dinâmica de *Malware* baseado em Redes Neurais Artificiais, equipado com aprendizado estatístico e Inteligência Artificial, especializado na detecção de *malware* a partir de arquiteturas IoT de 32 bits do tipo *Advanced RISC Machine* (ARM). Sob diferentes condições iniciais e funções de aprendizagem, nossas arquiteturas de antivírus são investigadas para maximizar sua precisão. A ausência ou limitação na detecção de software malicioso por antivírus comercial pode ser provida por um antivírus inteligente. Em vez de modelos baseados em listas negras ou heurística, nosso antivírus permite a detecção de *malware* em sistemas Linux embarcados de forma preventiva e não reativa como o *modus operandi* do Clamav e outros antivírus tradicionais.**

***Palavras-chave*—Antivírus, Malware, IoT, Arquivos, ELF ARM, Análise Dinâmica, Redes Neurais Artificiais, Computação Forense.**

Antivirus Applied to IoT Malware Detection based on Runtime Behaviors

***Abstract*—Nowadays, the Internet of Things (IoT) has a significant impact on people's lives, reaching hundreds of billions of Internet-connected devices. Due to the popularity of smart devices, the number of tech-driven cyber attacks has increased in recent years. The constant emergence of new malware aimed at IoT, such as the botnet, the use of complex obfuscation and evasion techniques, and often the availability of large resources for its development, makes him the biggest cyber villain in IoT scenarios today. The present work creates an Antivirus for Dynamic Malware Analysis based on Artificial Neural Networks, equipped with statistical learning and Artificial Intelligence, specialized in malware detection from 32-bit IoT architectures of the Advanced RISC Machine (ARM) type. Under different starting conditions and learning functions, our antivirus architectures are investigated to maximize their accuracy. The absence or limitation in the detection of malicious software by commercial antivirus can be provided by a smart antivirus. Instead of models based on blacklists or heuristics, our antivirus allows the detection of malware on embedded Linux systems in a preventive and non-reactive way like Clamav's *modus operandi* and other traditional antiviruses.**

***Index Terms*—Antivirus, Malware, IoT, ARM ELF Files, Dynamic Runtime Behaviors, Artificial Neural Network, Computer Forensics.**

Autor correspondente: Mr. Sthéfano Henrique M. T. Silva, *shmts@ecomp.poli.br*

I. Introdução

ATUALMENTE, a Internet das Coisas (IoT) impacta significativamente a vida das pessoas ao atingir centenas de bilhões de dispositivos conectados à Internet [1]. Ela também pode ser entendida como uma extensão da Internet atual. Cujo principal objetivo é transformar dispositivos do cotidiano em objetos inteligentes, proporcionando-lhes detecção, reconhecimento, processamento, sensoriamento, comunicação e conexão com a rede mundial de computadores. Com a IoT traz-se o mundo digital para perto do físico.

A IoT compreende vários domínios de aplicação, como cidades inteligentes [2], agricultura [3] e saúde [4]. Para Gubbi *et al.* (2013) [5], a IoT consiste em objetos físicos, geralmente conhecidos como coisas (dispositivos) que detectam, coletam, por meio de sensores, e podem processar informações. Sensores, em cenários de IoT, podem coletar informações ambientais como umidade, temperatura, geolocalização, frequência cardíaca e imagens. As informações são enviadas por uma rede de dados, geralmente sem fio, para a Internet, onde são processadas e armazenadas em um serviço de nuvem.

Presente na Indústria 4.0, alguns autores apontam que a IoT será a nova revolução na tecnologia da informação [6] [7]. Entretanto essas possibilidades apresentam riscos. Embora esta seja uma área promissora, realizar a visão da

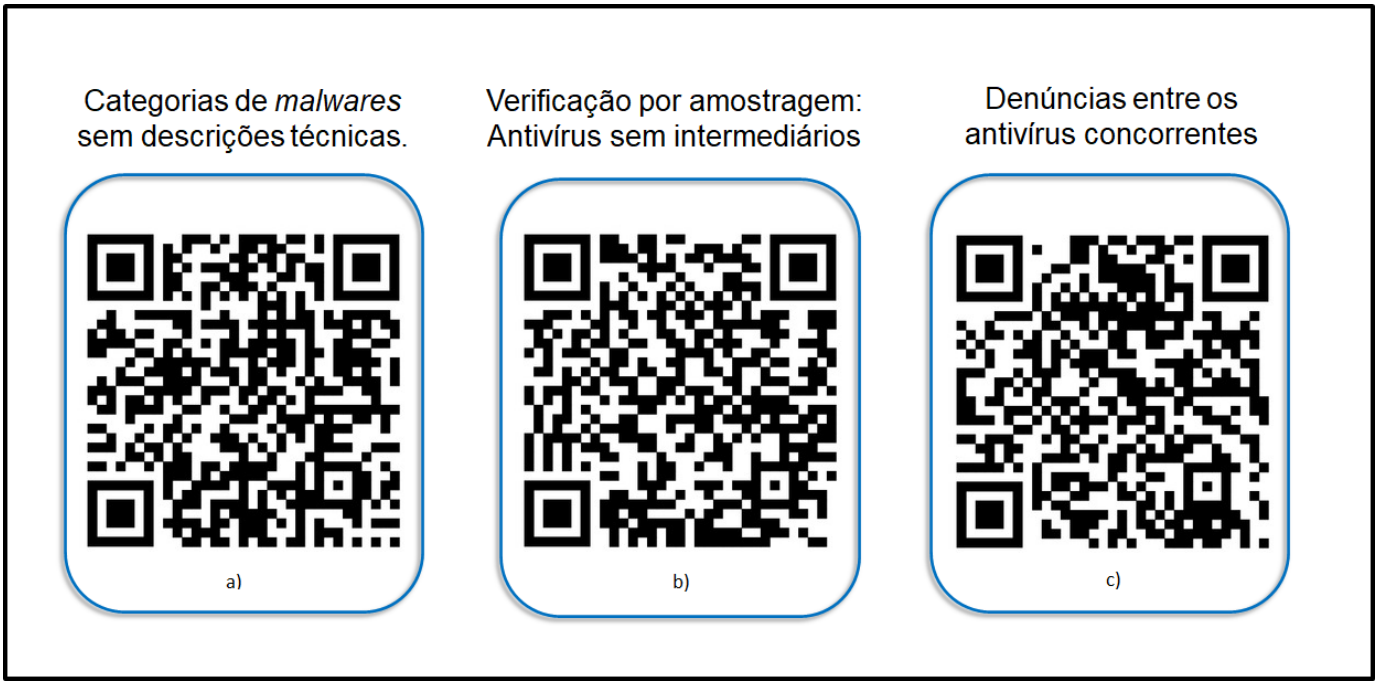

Fig. 1. QR Codes de links para referências criadas ao longo do texto sobre algumas limitações dos antivírus comerciais. a) Categorias de *malwares* sem descrições técnicas. b) Verificação por amostragem: Antivírus sem intermediários. c) Um antivírus comercial pode denunciar o instalador do antivírus concorrente.

IoT requer muitos desafios, como disponibilidade, confiabilidade, escalabilidade, desempenho, mobilidade, interoperabilidade e segurança [8] [9]. Dentre tantos desafios, a segurança e a confiabilidade se destacam essencialmente, tendo em vista que uma falha pode colocar pessoas em perigo, resultar em prejuízo financeiro ou dano ambiental [10], por exemplo.

Diversas pesquisas estão sendo desenvolvidas nessas áreas, como utilização de criptografia [11] e novos protocolos seguros [12], com o objetivo de aprimorar técnicas tradicionais já utilizadas em sistemas embarcados. No entanto, considerando a heterogeneidade e os recursos limitados dos dispositivos (processamento, memória, conectividade, bateria, etc.), é essencial conhecer as soluções que abordam a segurança em cenários de IoT.

Malware é um termo amplo que se refere a qualquer software intencionalmente projetado para danificar a funcionalidade normal de um computador ou rede [13]. De acordo com [14], um tipo de *malware* que representa uma séria ameaça à segurança da IoT são as *botnets* [15] [16]. Porém grande parte dos pesquisadores está focada em abordar a segurança da camada de comunicação que só fica comprometida após uma infecção no dispositivo [17] [18]. Em regra geral, os trabalhos na literatura que lidam com detecção de intrusão, principalmente para IoT, não se concentram em uma arquitetura específica de computador ou tipo de dispositivo para testar suas abordagens.

Ferramentas de segurança cibernética abstratas geralmente são baseadas em modelos que podem ser baseados em listas negras, heurística, técnicas empíricas dentre outros. Com suas limitações, cujas descrições detalhadas podem ser obtidas nos links embutidos nos QR Codes da Figura 1, tais ferramentas permitem investigar atitudes suspeitas de aplicativos IoT. Em contrapartida, nossa abordagem busca fornecer um modelo de segurança intra-dispositivo, voltado para a arquitetura ARM 32-Bits[19], por meio de um mecanismo de detecção baseado no processamento da máquina com análise das características do dispositivo.

Portanto, nós optamos por estudar o sistema GNU/Linux sob uma arquitetura mais utilizada em cenários de IoT: a arquitetura ARM de 32 bits. O *kernel* Linux está presente em um grande número de sistemas com diferentes arquiteturas de computador. Esse sistema pode ser encontrado em tudo, de automóveis a foguetes, relógios a TVs e laptops aos supercomputadores mais rápidos. GNU/Linux representa apenas uma porcentagem relativamente pequena dos sistemas operacionais encontrados em computadores pessoais, no entanto, tem amplo uso em servidores, dispositivos de IoT[20], equipamentos de rede, *smartphones* e muitos outros dispositivos, mesmo aqueles baseados na nuvem.

Para a remoção de software malicioso em sistema GNU/Linux, é essencial construir uma abordagem capaz de detectar e neutralizar anomalias antes que o sistema tenha seu comportamento comprometido. Assim, o sistema artificial de detecção de *malware* consiste em técnicas de aprendizado de máquina embutidas em sistemas IoT para manter o funcionamento correto do sistema quando exposto a um arquivo malicioso. A definição do modelo das técnicas de aprendizagem adotado é fundamental para o correto desenvolvimento de uma solução de segurança da informação voltada para sistemas embarcados.

A Inteligência Artificial (IA) pode ser definida como sis-

temas ou máquinas que imitam a inteligência humana para executar tarefas e que podem se aprimorar iterativamente com base nas informações que coletam[21]. A IA pode ocorrer de várias maneiras, considerando complexidade e objetivos. Já no Aprendizado de Máquina, subcampo da IA usado neste trabalho, é o processo pelo qual um sistema tecnológico aprende como se comportar e interpretar o conteúdo [22].

Quanto aos resultados, nosso antivírus monitora e pondera estatisticamente 2.793 ações que o arquivo suspeito pode realizar quando executado. Nosso antivírus atinge um desempenho médio de 98,75% na distinção entre aplicativos ARM benignos e *malware* de 32 bits. Assim, o presente trabalho demonstra que a IA pode ser uma boa alternativa para os fabricantes de dispositivos IoT e indica que as limitações dos antivírus comerciais podem ser supridas por nosso antivírus autoral. Ao invés de modelos baseados em abstração, nosso antivírus emprega ciência de dados avançada, aprendizado de máquina e inteligência artificial para identificar o comportamento malicioso de aplicativos IoT.

Este trabalho está organizado da seguinte forma: na seção II apresentamos as limitações dos antivírus comerciais. Na seção III, discutimos o estado-da-arte em relação aos antivírus de inteligência artificial; na seção IV realizamos um estudo preliminar sobre as redes ELMs; na seção V apresentamos o método proposto; na seção VI, fazemos uma comparação entre a rede ELM autoral e as redes ELM clássicas; na seção VII, mostramos os resultados e algumas discussões. Por fim, na seção VIII, fazemos as conclusões gerais e discutimos as perspectivas de nosso trabalho na seção IX.

II. Limitação dos Antivírus Comerciais

A detecção e mitigação do *malware* moderno é crítico para o funcionamento normal de uma organização [13]. Embora tenha sido questionado por mais de uma década, o *modus operandi* do antivírus comercial ainda é baseado em assinaturas quando o arquivo suspeito é consultado em conjuntos de dados nomeado de lista negra [23]. Portanto, para que o *malware* não seja detectado, basta que o *hash* do arquivo investigado não esteja contido na lista negra do antivírus ou este não possua assinaturas de comportamentos detectáveis. Nesse caso, o *hash*, também conhecido como criptografia unidirecional, serve como um identificador exclusivo para um arquivo específico.

Considerando as limitações dos antivírus comerciais, não é uma tarefa difícil desenvolver e distribuir variantes de um aplicativo malicioso. Os mecanismos de defesa tradicionais estão se tornando cada vez mais ineficazes devido às técnicas usadas pelos invasores, como obfuscação de código, metamorfismo e polimorfismo, que fortalecem a resiliência do *malware*. É suficiente fazer pequenas alterações no *malware* original com rotinas que efetivamente não têm uso, como *loops* repetidos e desvios condicionais sem instruções em seus escopos. Essas alterações até mesmo inúteis tornam o *hash* do *malware* modificado diferente do *hash* do *malware* original. Como resultado dessa ação, o *malware*, incrementado com rotinas nulas (por exemplo, NOP - *No Operation*), não será detectado pelo antivírus que catalogou o *malware* original. Ressalta-se a existência de *botnets* e *exploits* responsáveis por criar e distribuir, de forma automatizada, variantes do mesmo *malware* original. Portanto, conclui-se que os atuais sistemas antivírus baseados em assinaturas têm baixa eficácia quando submetidos a variantes do mesmo *malware* [23] [24].

Por meio da plataforma VirusTotal[1], este artigo investiga 68 antivírus comerciais com seus respectivos resultados apresentados na Tabela 1. Resultados resumidos dos antivírus comerciais mundiais estão no repositório autoral [25]. Usamos 1.000 arquivos maliciosos ELF para arquitetura ARM obtidos de nosso banco de dados [25]. O objetivo do trabalho é verificar a quantidade de pragas virtuais catalogadas pelos principais antivírus comerciais. A motivação é que a aquisição do novo *malware* desempenha um papel importante no combate a aplicativos maliciosos. Portanto, quanto maior o conjunto de dados, a lista negra, melhor tende a ser a defesa fornecida pelo antivírus. Inicialmente, o *malware* ELF ARM é examinado pelo servidor pertencente à plataforma VirusTotal, com o envio feito por sua respectiva API. Posteriormente, os arquivos ELF ARM de 32 bits foram analisados pelos 68 antivírus comerciais do VirusTotal. Logo depois, os antivírus fornecem seus diagnósticos para arquivos ELF enviados ao servidor. É permitido pelo VirusTotal emitir três tipos de diagnósticos: *malware*, benigno e omissão.

Considerando o risco primário do VirusTotal, o antivírus detecta o comportamento malicioso do arquivo suspeito enviado. Dentro da configuração experimental arquitetada, todos os arquivos carregados medem o *malware* documentado por meio do correspondente de incidentes. Então, o *malware* é indentificado assim que a malignidade do arquivo investigado é detectada. A detecção de software malicioso indica que o antivírus fornece um serviço robusto contra intrusão cibernética. Em um segundo cenário, o antivírus testa a benignidade do arquivo investigado. Portanto, dentro do estudo construído, quando o antivírus afirma que o arquivo é benigno, trata-se de um caso de falso negativo porque todas as amostras enviadas têm ação maliciosa confirmada. Em outras palavras, o arquivo investigado é considerado *malware*, no entanto, o antivírus atesta sua benignidade de forma bastante errada. Dentro do terceiro cenário, o antivírus não emite opinião sobre o arquivo suspeito. Em caso de omissão, o arquivo investigado nunca foi avaliado pelo antivírus, portanto, há muito pouco poder de julgá-lo em tempo real. A omissão de identificação por antivírus aponta para sua limitação em serviços de grande escala.

A Tabela 1 mostra os resultados dos 68 antivírus comerciais avaliados. O antivírus ESET-NOD32 foi o que obteve melhor desempenho, pois conseguiu detectar 82,6%

[1]VirusTotal: Serviço gratuito de análise de conteúdo malicioso em arquivos e URLs. Disponível em: https://www.virustotal.com/. Acessado em junho de 2021.

TABELA 1
Resultados de antivírus comerciais na detecção de malwares obtidos do VirusShare. Resultados resumidos de 68 antivírus comerciais mundiais estão no repositório autoral [25].

Posição	Antivírus	Detecção (%)	Falso Negativo (%)	Omissão (%)
1	ESET-NOD32	82.6%	17.3%	0.1%
2	MicroWorld-eScan	79,9%	20,1%	0%
3	Ad-Aware	79,9%	19,9%	0,2%
4	BitDefender	79,8%	19,9%	0,3%
5	Emsisoft	79,2%	20,5%	0,3%
6	NANO-Antivirus	79,2%	20,8%	0%
7	FireEye	79,1%	9,8%	1,1%
8	GData	78,9%	19,3%	1,8%
9	Tencent	78,8%	21%	0,2%
10	Zillya	78,7%	20,1%	1,2%
20	ClamAV	76,7%	22,3%	1%
58	Kingsoft	0%	96,9%	3,1%
59	TACHYON	0%	100%	0%
60	CMC	0%	100%	0%
61	SUPERAntiSpyware	0%	99,9%	0,1%
62	Baidu	0%	99,1%	0,9%
63	Malwarebytes	0%	99,9 %	0,1%
64	TotalDefense	0%	89,8 %	10,2%
65	F-Prot	0%	0,3%	99,7%
66	eGambit	0%	5,8%	94,2%
67	CrowdStrike	0%	0,4%	99,6%
68	Invincea	0%	0,1 %	99,9%

TABELA 2
Resultado do envio de três malwares ao VirusTotal obtidos do VirusShare. Resultados resumidos de 68 antivírus comerciais mundiais estão no repositório autoral [25].

Antivírus	VirusShare_A	VirusShare_B	VirusShare_C
AVG	ELF:DDoS-S [Trj]	ELF:Gafgyt-DO [Trj]	ELF:DDoS-S [Trj]
Ad-Aware	Trojan.Linux.Generic.69575	Gen:Variant.Backdoor.Linux.Tsunami.1	Trojan.Linux.GenericA.73451
AhnLab-V3	Linux/Mirai.Gen6	Linux/Tsunami.Gen	Linux/Gafgyt.Gen
Avast	ELF:DDoS-S [Trj]	ELF:Gafgyt-DO [Trj]	ELF:DDoS-S [Trj]
Avast-Mobile	ELF:DDoS-S [Trj]	ELF:Tsunami-CR [Trj]	ELF:DDoS-S [Trj]
Avira	LINUX/Gafgyt.opnd	LINUX/Tsunami.wkuvt	LINUX/Gafgyt.opnd
BitDefender	Trojan.Linux.Generic.69575	Gen:Variant.Backdoor.Linux.Tsunami.1	Trojan.Linux.GenericA.73451
BitDefenderTheta	Gen:NN.Mirai.34608	Gen:NN.Mirai.34608	Gen:NN.Mirai.34608
ClamAV	Unix.Malware.Agent-6769357-0	Unix.Trojan.Mirai-5607483-0	Unix.Trojan.Gafgyt-111
Comodo	Malware@#3o8smwc385jui	Malware@#3j0a9lm761bhc	Malware@#1vbephabp2pmb
DrWeb	Linux.BackDoor.Fgt.1755	Linux.BackDoor.Tsunami.485	Linux.BackDoor.Fgt.9
ESET-NOD32	a variant of Linux/Mirai.AE	a variant of Linux/Tsunami.NBV	a variant of Linux/Gafgyt.C
F-Secure	Malware.LINUX/Gafgyt.opnd	Malware.LINUX/Tsunami.wkuvt	Malware.LINUX/Gafgyt.opnd
Fortinet	ELF/Mirai.AE!tr	ELF/Tsunami.NDJ!tr	ELF/Gafgyt.UN!tr.bdr
GData	Linux.Trojan.Gafgyt.A	Linux.Trojan.Gafgyt.A	Linux.Trojan.Gafgyt.B
Ikarus	Trojan.Linux.Mirai	Trojan.Linux.Fgt	Trojan.Linux.Fgt
Kaspersky	HEUR:Backdoor.Linux.Gafgyt.a	HEUR:Backdoor.Linux.Tsunami.bh	HEUR:Backdoor.Linux.Gafgyt.ac
McAfee	Linux/Mirai.g	Linux/Gafgyt.h	Linux/Gafgyt.r
McAfee-GW-Edition	Linux/Mirai.g	Gen:Variant.Backdoor.Linux.Tsunami.1	Linux/Gafgyt.r
Microsoft	DDoS:Linux/Gafgyt.YA!MTB	Trojan.ElfArm32.Tsunami.ejtrus	Backdoor:Linux/Gafgyt.AF!MTB
TrendMicro	Backdoor.Linux.BASHLITE.SMJC	Backdoor.Linux.BASHLITE.SMJC	Backdoor.Linux.BASHLITE.SMJC

dos *malwares* investigados. O Clamav, um dos mais utilizados entre os sistemas GNU/Linux, teve seu desempenho abaixo do esperado, ficando aquém de outros menos conhecidos, ocupando a 20ª posição no ranking com a base de dados estudada com 76,7% de detecção de *malware*. Um dos maiores empecilhos observados no combate ao *malware* é o fato de os fabricantes de antivírus não compartilharem suas lbase de dados de software malicioso devido a disputas comerciais. Por meio da análise da Tabela 1, o trabalho proposto aponta para um agravante da referida adversidade: o mesmo desenvolvedor de antivírus não compartilha seus bancos de dados entre seus diferentes antivírus. Observe o exemplo do antivírus McAfee-GW-Edition e McAfee, ambos pertencentes à mesma empresa. Suas respectivas base de dados não são compartilhadas. Portanto, as estratégias comerciais de uma mesma empresa dificultam o combate ao *malware* [23]. Complementa-se o fato de que os fabricantes de antivírus estão preocupados somente em prevenir intrusões cibernéticas, mas também otimizar a receita de seus negócios. O que, talvez, não torne atraente o mercado de IoT até agora.

Em análise, a detecção de *malware* presente nos resultados obtidos variou de 0% a 82,6%, dependendo do antivírus investigado. Em média, os 68 antivírus foram capazes de detectar 46,08% das pragas virtuais avaliadas, com um desvio padrão de 35,61%. O alto desvio padrão obtido indica que a detecção de arquivos IoT maliciosos pode sofrer variações abruptas dependendo do antivírus selecionado. A

Fig. 2. Modelos Arquiteturais em IoT: a) Três Camadas. b) Cinco camadas. c) Sete Camadas.

proteção contra intrusões cibernéticas está em função de escolher o antivírus robusto dotado de uma base de dados ampla e atualizada. Os antivírus confirmaram uma média de falsos negativos em 44,85% dos casos, com um desvio padrão de 36,26%.

Atestar a benignidade do software malicioso voltado para a IoT pode implicar em danos irrecuperáveis. Uma pessoa ou instituição, por exemplo, passaria a confiar em um determinado aplicativo malicioso quando ele é *malware*. Ainda como aspecto desfavorável, cerca de 5 dos antivírus, ou 7,35%, não opinaram sobre as 1.000 amostras maliciosas. Em média, os antivírus se omitiram no julgamento em 8,82% dos casos, com um desvio padrão de 25,76%. A falha identificada no diagnóstico indica a limitação dos antivírus em detectar *malware* em tempo real.

Inclui-se como adversidade, no combate às aplicações maliciosas, o fato dos antivírus comerciais não possuírem uma padronização quanto à classificação de *malware*, como pode ser verificado na Tabela 2. Entre os 1.000 *malwares* ELF ARM de 32 bits analisados, escolhemos 3 para ilustrar a confusão de classificações fornecidas pelos antivírus comerciais líderes. Se não houver um padrão, os desenvolvedores de antivírus darão os nomes que quiserem; por exemplo, uma empresa pode identificar um *malware* ELF como "Trojan.Linux.Generic.Linebreak 69575" e uma segunda empresa como "ELF: DDoS-S [Trj]". Conclui-se, portanto, que a falta de um padrão degrada os meios de segurança cibernética, uma vez que cada classe de *malware* deveria ser tratada de forma completamente diferente (vacinas) em diferentes sistemas anti-*malware*. Conclui-se ainda que não é prático para o aprendizado de máquina monitorado adotar o reconhecimento de padrões para categorias de *malware* ELF ARM de 32 bits. Devido a esse emaranhado complexo da Classificação Multiclasse, fornecida por especialistas (antivírus) como visto na Tabela 2, é estatisticamente improvável que qualquer técnica supervisionada de aprendizado de máquina adquira boa generalização.

III. Estado da Arte

NO levantamento do estado da arte, é possível identificar o crescimento das pesquisas relacionadas à segurança da informação em cenários de IoT ao longo dos anos. Trabalhos com ênfase em soluções de segurança nas seguintes categorias: dispositivo, conectividade, borda/névoa, nuvem e *blockchain*. Tradicionalmente, os sistemas IoT usam plataformas de computação em nuvem para facilitar a implantação [26]. No entanto, grande parte dos pesquisadores se concentra em abordar a segurança em uma camada de comunicação que é comprometida após uma infecção no dispositivo [27] [28].

A. Internet das Coisas

Trabalhos na literatura, que lidam com detecção de intrusão principalmente para IoT, costumam não focar em um dispositivo específico ou arquitetura específica para testar suas abordagens. Esse fato é explicado pela definição não padronizada da própria IoT. Em regra geral, os

TABELA 3
Resumo das principais técnicas de antivírus de última geração.

Módulos	Consumo	Técninca de Rede Neural	Dispositivos	Acucácia
Antivírus Autoral	Shallow neural network	Morphological ELM (mELM)	ARM para IoT	98.75%
FARUKI, P. *et al.* (2019) [29]	Deep Learning	Rectified Linear Unit (ReLU)	Smartphone	98.65%
KALASH, *et al.* (2018) [30]	Deep Learning	VGG-16	Computador Pessoal	98.52%
LIMA, *et al.* (2021), [31]	Shallow neural network	Multi-Layer Perceptron (MLP)	Computador Pessoal	98.32%
HARDY, W. *et al.* (2016) [32]	Deep Learning	Stacked Autoencoders	Computador Pessoal	96.85%
MANIATH, S. *et al.* (2017) [33]	Deep Learning	Long-Short Term Memory (LSTM)	Computador Pessoal	96.67%
SU, *et al.* (2018) [34]	Deep Learning	Training batch (CNN)	x86 para IoT	94.00%

trabalhos do estado-da-arte não lançam luz sobre os mais recentes tipos de dispositivos, mais ainda, na arquitetura destes.

Embora existam várias propostas para arquiteturas em camadas para IoT, o consenso sobre um modelo de referência ainda não foi alcançado. O modelo de arquitetura IoT, Figura 2, pode ser composto por três ou cinco camadas [8] [35] [36], embora um modelo apresentando sete camadas tenha sido descrito pela *Cisco Networks* [1]. Neste trabalho, um modelo de três camadas foi adotado para fins didáticos, pois trata de níveis inferiores com foco na arquitetura do dispositivo em sensor/atuador, embora possa ser aplicado em neblina/computação.

A IoT surgiu como um componente chave de todas as infraestruturas críticas avançadas [27]. Observa-se que uma importante questão de pesquisa na área é desenvolver métodos e abordagens leves, sem abrir mão do nível necessário de segurança e resiliência. A maioria dos trabalhos sobre segurança IoT se concentra na camada de comunicação da arquitetura por meio de uma análise de pacotes de rede [27], os quais, após a captura com *sniffers* como *tcpdump*, são convertidos em um conjunto de dados de atributos para envio às redes neurais. Outros trabalham na camada de aplicação, na computação em nuvem, provendo mecanismos de segurança bem distantes dos dispositivos. Soluções como *blockchain* [17], seja em *edge*, *fog* ou *cloud computing*, têm sido usadas em ambas as abordagens e criptografia, mas são mais complexas, computacionalmente caras e às vezes não seguras, além de falharem na detecção.

B. Sistema de Antivírus Inteligente

O antivírus é o dispositivo mais popular para detectar *malware*. O *modus operandi* do software antivírus comercial consiste principalmente na identificação de um *malware* ELF com base em sua assinatura O principal problema com essa abordagem é: para que uma nova praga cibernética seja atribuída, ela deve detectar que certos dispositivos foram infectados. A estratégia do antivírus comercial é esperar que um usuário seja infectado e, em seguida, relatar o comportamento anormal de seu dispositivo para que o fabricante do antivírus possa agir.

Levando em conta as limitações do software antivírus comercial, a tecnologia mais recente propõe a função de extrair e examinar arquivos por meio de uma máquina de aprendizagem estatística.

É importante observar também que a detecção e eliminação do executável malicioso não é suficiente para que a vítima esteja livre de suas ações. Além de remover o *malware*, é necessário desfazer todas as suas irregularidades, como reabilitar os mecanismos de defesa da vítima, incluindo firewall, *plug-ins* de segurança e o próprio antivírus. A IA pode ser usada para detectar e neutralizar ataques ao ciberespaço. Automatizando muitas tarefas, analisando milhares de arquivos, extraindo suas funções e analisando-as [23]. Na tabela 3, é demonstrado os trabalhos que figuram no estado da arte da área, suas respectivas técnicas e acurácias obtidas.

No trabalho de LIMA *et al.* (2021) foi criado um antivírus capaz de detectar *malware* em arquivos do Windows (*Portable Executable*, PE) com uma precisão média de 98,32% [31]. O arquivo executável do sistema operacional da Microsoft passa pelo processo de desmontagem(*disassembly*). Em seguida, o arquivo PE pode ser estudado para que a intenção maliciosa do arquivo possa ser analisada. A primeira arquitetura possui uma única camada oculta contendo 100 neurônios. A segunda arquitetura possui duas camadas ocultas, cada uma com 100 neurônios. A terceira arquitetura emprega uma camada oculta; no entanto, 500 neurônios são usados. Por fim, a quarta arquitetura possui quatro camadas ocultas, cada uma com 1.261 neurônios. Dentro da quarta arquitetura, o número de 1261 neurônios se deve à aplicação da fórmula empírica de Hecht-Nielsen. Hecht-Nielsen (1987) apontou que a rede neural pode ser configurada como uma camada oculta com exatamente $2n+1$ nós, onde n é o número de nós de entrada. Em sua análise [37] extrai 630 atributos de cada arquivo. Esses recursos são os neurônios de entrada da rede neural artificial. No antivírus feito por LIMA *et al.* (2021) redes neurais rasas são usadas. A classificação por meio de redes neurais rasas visa dividir os arquivos executáveis da arquitetura de 32 bits em duas categorias: software malicioso e software benigno.

Por outro lado, os antivírus baseados em *Deep Learning* também alcançaram excelente precisão. SU, J. *et al.* (2018) alcançou uma precisão média de 94,00% para detectar *malware* de IoT [34]. A estrutura de rede profunda tem 6 camadas. Existem 3 camadas com pesos aprendíveis: 2 camadas convolucionais e 1 camada totalmente conectada. A rede é treinada com 5.000 iterações com um tamanho de lote de treinamento de 32 e taxa de aprendizado de 0,0001.

FARUKI, P. *et al.* (2019) alcançou uma precisão média de 98,65% para detectar *malware* Android [29] utilizando

uma rede profunda ReLU (*Rectified Linear Unit*) dotada de técnica de *dropout*. As camadas ReLU executam a operação de limite para cada elemento da entrada, onde qualquer valor menor que zero é definido como zero.

MANIATH, S. *et al.* (2017) criaram um antivírus para detectar *ransomware* empregando redes profundas LSTM (*Long-Short Term Memory*) [33]. A rede de treinamento consiste em 3 camadas com 64 nós LSTM em cada camada. A rede profunda é treinada para 500 épocas com um tamanho de lote de 64. MANIATH, S. *et al.* (2016) alcançam uma precisão média de 96,67%.

O antivírus feito por HARDY, W. *et al.* (2016) visa detectar *malware* em arquivos PE (Windows) empregando redes profundas de *Stacked Autoencoders* [32]. O decodificador tenta mapear essa representação de volta para a entrada original. O modelo de *Deep Learning* é treinado com 3 camadas ocultas e 100 neurônios em cada camada oculta. O codificador mapeia a entrada para uma representação oculta. HARDY, W. *et al.* (2016) alcançou uma precisão média de 96,85%.

KALASH, M. *et al.* (2018) propõem um *Deep Learning* empregando VGG-16 para classificação de *malware* [30]. KALASH, M. *et al.* (2018) alcançam uma precisão média de 98,52%. A rede tem um tamanho de entrada de imagem de 224 por 224 *pixels*. O VGG-16 é uma rede neural convolucional com 16 camadas de profundidade. A rede pré-treinada pode classificar imagens em 1000 categorias de objetos, como teclado, mouse, lápis e muitos animais. A última camada de processamento contendo 1.000 neurônios é substituída por uma camada completamente conectada softmax contendo 2 neurônios (benigno, *malware*). No presente trabalho, replicamos o antivírus feito por KALASH, M. *et al.* (2018) por utilizar uma rede convolucional, presente no estado da arte em diversas áreas, e obter alta acurácia na resolução de diversos problemas. A entrada da imagem é criada por meio de um gráfico de superfície feito por nossa extração de recursos dinâmicos. As superfícies geradas estão em nosso repositório [25].

A desvantagem da *Deep Learning* é o longo tempo de treinamento. Como um fator agravante, a *Deep Learning* tem baixa capacidade de paralelismo porque as camadas são sequenciais. Portanto, esta camada só pode ser executada após a camada superior ter concluído seu trabalho. Em aplicações que requerem treinamento frequente (aprendizado) de software antivírus, este fato pode se tornar um obstáculo, pois em média 8 (oito) novos *malwares* são criados a cada segundo [38]. Em síntese, não deve haver diferença no tempo de aprendizagem do software antivírus em comparação com a taxa de nova geração de *malware* em todo o mundo.

Devido aos excelentes resultados obtidos pelas técnicas de *Deep Learning*, criou-se um senso comum de que esta é capaz de fornecer a melhor precisão em qualquer tipo de aplicação,mas, de fato, essa consideração é falsa. Redes neurais profundas, especificamente, redes neurais convolucionais (Convolution Neural Networks - CNNs) são baseadas na convolução de filtro linear. Embora tenha um papel fundamental em aplicações de computador, a convolução do filtro é limitada a aplicações quando o gradiente de fluxo vetorial é formado.

Considere, por exemplo, imagens biomédicas de dispositivos de mamografia. As imagens estão repletas de ruídos que dificultam o reconhecimento da lesão mamária [39]. Portanto, a convolução dos filtros é essencial para eliminar ruídos e, portanto, descartar pequenas irregularidades no achado correspondentes ao potencial câncer. Técnicas convolucionais, como os filtros gaussianos, são essenciais para reduzir o ruído em imagens biomédicas [39].

Como contra-exemplo, considere o repositório ilustrado na Tabela 4. Os recursos estão completamente desconectados um do outro, apesar de pertencerem originalmente à mesma aplicação. 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. Então, ao aplicar a convolução linear dos filtros no repositório, ilustrada na Tabela 4, o navegador de acesso, 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 o navegador da vítima até mesmo a extração de recursos mesmo que o inverso tenha sido o resultado da auditoria.. Então, as técnicas convolucionais sofrem uma desvantagem quando aplicadas ao reconhecimento de padrões de *malware*.

TABELA 4
Exemplo de um repositório estatístico baseado na detecção de *malware*.

Atributos		
Verificação dos dados Wi-Fi	Acesso ao Navegador	Acesso à Galeria de Imagens
1	0	1

A fim de provar nosso embasamento teórico, o antivírus autoral emprega redes neurais morfológicas rasas em vez de redes convolucionais profundas. Como experimentos, o antivírus autoral tem sua precisão em comparação com os antivírus de última geração baseados em redes neurais rasas e profundas. Nosso antivírus pode combinar alta precisão com tempo de aprendizagem reduzido. Para evitar comparações injustas, o estágio de extração de recursos é padronizado monitorando 2.793 comportamentos que o arquivo ELF ARM de 32 bits suspeito pode fazer quando executado propositalmente.

IV. Estudos Preliminares: *Extreme Learning Machines*

O presente trabalho resulta em um antivírus composto por redes neurais do tipo Máquina de Aprendizado Extremo (*Extreme Learning Machines* - ELMs) visando a detecção preventiva de softwares maliciosos. A redes ELMs são constituídas de aprendizado estatístico baseados em *kernels* flexíveis e poderosos, cujas principais características são baixo tempo de treinamento e robusto desempenho de classificação [40]. Os *kernels* são funções matemáticas

utilizadas como método de aprendizado das redes neurais ELMs.

O algoritmo proposto por Huang *et al.* [41] [40] é formado por uma rede de camada única oculta, não recorrente, baseada em método analítico para estimar os pesos de saída da rede, em qualquer inicialização aleatória de pesos de entrada. O princípio de funcionamento da rede é o mesmo de uma rede neural artificial, no entanto, a metodologia de treinamento de uma ELM não é baseada em gradiente descendente. Com isso, o algoritmo evita os principais problemas dos algoritmos de retroprogração como convergência lenta ou para mínimos locais.

As redes ELMs têm sido amplamente aplicadas em diversas áreas, tais como a Engenharia Biomédica [42] [43] [44] [39] [45] [46] [47]. As ELMs podem contribuir bastante para o avanço da segurança digital em dispositivos inteligentes. O trabalho descrito aqui aplica as ELMs na área de segurança da informação, mais especificamente no reconhecimento de padrões em *malware* IoT.

Matematicamente, na rede neural ELM, os atributos de entrada x_{it} correspondem ao conjunto $\{x_{it} \in \mathbb{R};\ i = 1, ...n;\ t = 1, ...v\}$. Portanto, existem n recursos extraídos do aplicativo e v vetores de dados de treinamento. A camada oculta h_j, consiste em m neurônios representados pelo conjunto $\{h_j \in \mathbb{R};\ j \in N^*;\ j = 1, ...m\}$. O processo de treinamento da ELM é rápido porque é composto de poucas etapas. Inicialmente, os pesos de entrada w_{ji} e *bias* b_{jt} são definidos em uma geração aleatória. Dada uma função de ativação f: $\mathbb{R} \rightarrow \mathbb{R}$, o processo de aprendizagem é dividido em três etapas:

- Geração aleatória do peso w_{ji}, correspondendo aos pesos entre a entrada e as camadas ocultas, e *bias* b_{jt}.
- Cálculo da matriz H, que corresponde à saída dos neurônios da camada oculta.
- Cálculo da 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.

A saída dos neurônios da camada oculta, correspondente à matriz H, é calculada pelo *kernel* φ, entradas do conjunto de dados e pesos entre a entrada e as camadas ocultas mostradas na Eq. (1).

$$H_{jt} = \begin{bmatrix} \varphi(1,1) & \varphi(1,2) & \cdots & \varphi(1,v) \\ \varphi(2,1) & \varphi(2,2) & \cdots & \varphi(2,v) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi(m,1) & \varphi(m,2) & \cdots & \varphi(m,v) \end{bmatrix}_{m \times v} \quad (1)$$

O aprendizado das redes ELMs é baseado em *kernels*. Este tipo de aprendizado oferece a possibilidade de criar um mapeamento não linear de dados sem a necessidade de aumentar o número de parâmetros ajustáveis, como a taxa de aprendizado comumente usada em redes neurais baseadas em retropropagação. Eq. (2) descreve um *kernel* φ Sigmoide de uma rede ELM com os resultados mostrados na Figura 3 (a).

$$\varphi(t,i) = sigmoide\Big(x_{ti}.w_{ji} + b_{jt}\Big), \quad onde\ sigmoide(\xi) = \frac{1}{1 + e^{-\xi}} \quad (2)$$

Ao contrário das redes de retropropagação, nas redes ELM, não há necessidade de definir critérios de parada para treinamento ou fazer estratégias para garantir que a rede neural não perca a capacidade de generalização. A razão é que a ELM emprega apenas uma iteração. Então, não há necessidade de separar conjuntos de dados em treinamento, validação e teste. Basta dividir o conjunto de dados em treinamento e teste. Em comparação com redes neurais baseadas em retropropagação, esses dois conjuntos podem fornecer mais amostras.

Após o treinamento da rede ELM, os padrões de teste são apresentados juntamente com a saída desejada. A rede não fará mais alterações e apenas calculará os resultados obtidos para cada novo conjunto de testes. Ao comparar os padrões esperados e obtidos, a precisão do ELM é calculada.

V. Máquinas Morfológicas de Aprendizado Extremo

Ao invés de usar *kernels* convencionais, *kernels* autorais serão usados para ELMs. Empregamos mELMs (*Morphological Extreme Learning Machines*), ELMs com núcleos de camadas ocultas baseados nos operadores morfológicos de processamento de imagem de erosão e dilatação. Este método de aprendizagem permite a criação de mapeamento de dados não lineares em que o aprendizado se ajusta aos dados e não cria uma fronteira de decisão (Figura 3.c e Figura 3.d). Assim, não há necessidade de aumentar o número de parâmetros ajustáveis, como na taxa de aprendizado usada em redes com retropropagação.

Existem duas operações morfológicas fundamentais, erosão e dilatação. A teoria da Morfologia Matemática pode ser considerada construtiva, pois todas as operações são construídas com base na Erosão e Dilatação. Matematicamente, a erosão e a dilatação são definidas de acordo com as Eq. (3) e Eq. (4), respectivamente:

$$\epsilon_g(f)(u) = \bigcap_{v \in S} f(v) \ \vee \ \overline{g}(u - v) \quad (3)$$

$$\delta_g(f)(u) = \bigcup_{v \in S} f(v) \ \wedge \ g(u - v) \quad (4)$$

Onde $f : S \rightarrow [0,1]$ e $g : S \rightarrow [0,1]$ são imagens normalizadas na forma de uma matriz chamada S, onde $S \in \mathbb{N}^2$. *Pixel* é definido pelo par cartesiano $(u, f(u))$, onde u é a posição associada com o valor $f(u)$. v é a matriz de $f(u)$, coberto por g. Os operadores u e v estão

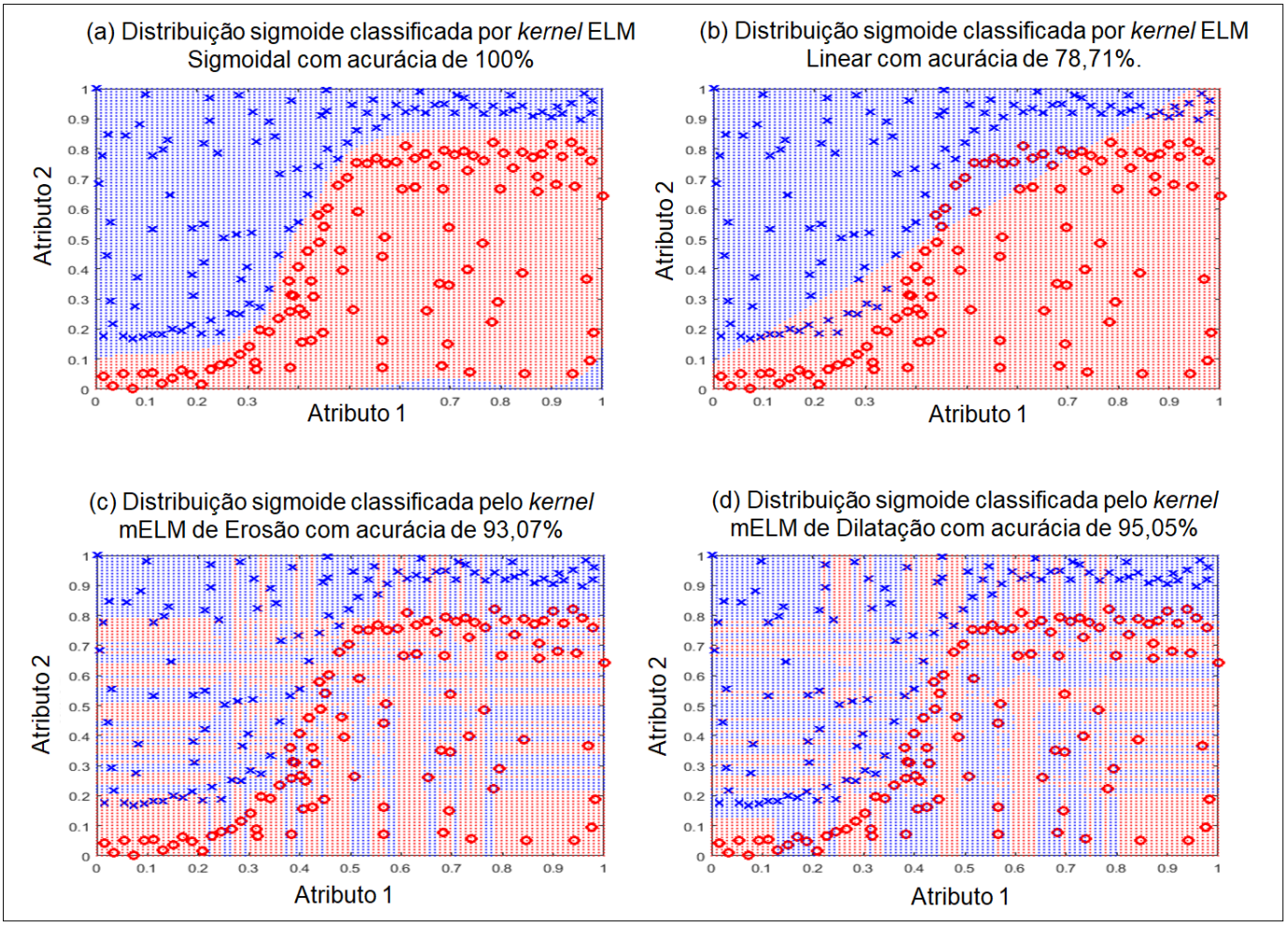

Fig. 3. (a) Desempenho bem-sucedido do *kernel* compatível com o conjunto de dados. (b) Classificação imprecisa do *kernel* Linear em uma distribuição não linearmente separável. (c - d) Desempenhos de sucesso por núcleos de dilatação e erosão.

associados à operação máxima, enquanto $\cap$ e $\wedge$ estão associados à operação mínima. g é o elemento estruturante para erosão e dilatação [48]. $\overline{g}$ é a negação ou complemento de g.

Na Eq. (3) ocorre inicialmente a negação do elemento estruturante $\overline{g}$. Então, acontece a operação de máxima $\vee$ denotado por $f(v) \vee \overline{g}(u-v)$, onde $f(v)$ refere-se à matriz da imagem original f coberto por $\overline{g}$. $f(v)$ é tecnicamente chamada de região ativa da imagem. Finalmente, o valor $\epsilon_g(f)(u)$, na posição u, da imagem erodida recebe o valor mínimo entre os máximos, através do operador $\cap$. $\epsilon_g(f)(u)$ obtém o valor 0 associado ao preto absoluto que são sobreposições de erosão $\overline{g}$ para a imagem original f. O objetivo é que áreas semelhantes a $\overline{g}$ expanda [48]. Ao associar 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 [48].

Eq. (3) mostra o desempenho da operação de dilatação morfológica. Devido à precedência matemática, o mínimo $\wedge$ operação denotada por $f(v) \wedge g(u-v)$, ocorre, onde $f(v)$ refere-se à matriz de imagem original f coberta (combinada) por g. Portanto, o valor $\delta_g(f)(u)$, na posição u, da imagem expandida, recebe o valor máximo entre os mínimos, por meio do operador $\cup$. A dilatação sobrepõe o elemento estruturante g na imagem original f. O objetivo é que áreas semelhantes a g se expandam. Ao associar 1's com branco absoluto e 0's com preto absoluto, a dilatação aumenta as áreas com tonalidade mais intensa e elimina as regiões escuras [48].

De acordo com a Eq. (2) em relação ao operador de imagem de erosão, o *kernel* ELM de erosão pode ser definido de acordo com a Eq. (4), onde $\{i \in \mathbb{N}^*, i = 1, \ldots, n\}$, $\{j \in \mathbb{N}^*, j = 1, \ldots, m\}$, $\{t \in \mathbb{N}^*, t = 1, ..v\}$. Portanto, há n neurônios na camada de entrada (sem o *bias*), m neurônios na camada oculta e v vetores de dados de treinamento.

$$\varphi_{\epsilon}(t,i) = \bigcap_{i=1}^{n}(x_{ti} \vee \overline{w}_{ji}) + b_{jt} \tag{5}$$

Semelhante ao *kernel* de erosão, Eq. (5) define o *kernel* de dilatação inspirado na Eq. (4) e referindo-se ao operador morfológico de dilatação.

$$\varphi_{\delta}(t,i) = \bigcup_{i=1}^{n}(x_{ti} \wedge w_{ji}) + b_{jt} \tag{6}$$

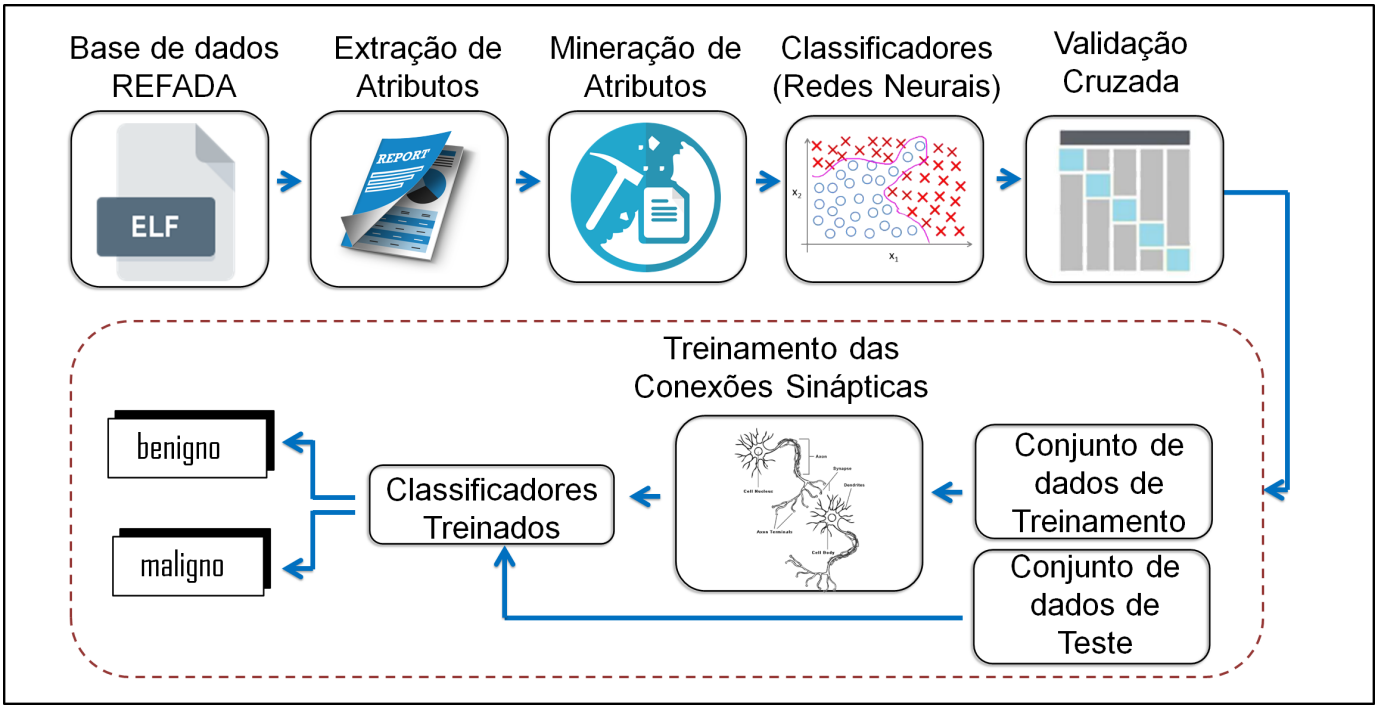

Fig. 4. Diagrama da metodologia proposta.

Para se obter um bom desempenho em ELMs, é necessário escolher um *kernel* que seja capaz de otimizar a fronteira de decisão para o problema apresentado como visto na Figura 3 (a). Um *kernel* linear obtém ótimos resultados quando usado para resolver um problema separável linearmente. No entanto, quando usado para resolver problemas separáveis não linearmente como mostrado na Figura 3 (b) para uma distribuição sigmoide, ele não tem um desempenho satisfatório.

Então, uma boa capacidade de generalização da rede neural pode depender de uma escolha ajustada de *kernel*. O melhor *kernel* pode estar subordinado ao problema a ser resolvido. Como efeito colateral, a investigação de diferentes *kernels* é geralmente um assunto desgastante que envolve validação cruzada combinada com diferentes condições iniciais aleatórias. Assim, a investigação de *kernels* distintos pode ser necessária, caso contrário, a rede neural composta, por um *kernel* incompatível, pode gerar resultados insatisfatórios.

Logo, as redes neurais rasas, proposta por Huang *et al.* (2012) [40], e suas morfologias, apresentam baixa complexidade, baixa necessidade de processamento, baixo tempo de treinamento e boa capacidade de classificação. Tal fato a torna ideal para o uso em ambiente IoT, pois estes possuem limitações computacionais.

VI. Método Proposto

A Figura 4 mostra o diagrama em blocos da metodologia proposta. Inicialmente, o arquivo ARM de 32 bits, proveniente do conjunto de dados autoral, é executado para verificar a tentativa de corromper o o GNU/Linux e, na sequência, o sistema é auditado pelo *Cuckoo Sandbox*. Os recursos dinâmicos são sintetizados na seção VI.B. Em seguida, as características dinâmicas dos arquivos ELF para a arquitetura ARM de 32 bits são armazenadas no formato de repositório de aprendizado de máquina.

Todos os experimentos foram realizados em um servidor Dell modelo PowerEddge T440 equipado com processador Xeon Bronze 1.70GHz, memória de 32GB de RAM e 2TB de armazenamento em massa. Para que não haja comparações injustas, os antivírus de última geração são treinados e testados no mesmo servidor utilizado pelo antivírus autoral. Ressalta-se que a aquisição de um servidor se deu para replicação e comparação com obras de última geração. O antivírus autoral requer baixa capacidade de processamento e armazenamento. Salienta-se ainda que o antivírus autoral pode ser usado em qualquer computador *desktop* convencional.

A. Base de dados Autoral

O presente trabalho emprega a base de dados REFADA (*A Retrieval of ELF Files ARM to Dynamic Analysis* – Redistribuição de Arquivos ELF ARM para Análise Dinâmica). A base de dados mencionada visa permitir a classificação de arquivos de arquivos ELF para arquituras ARM de 32 bits entre benignos e maliciosos, sendo composta de 1000 arquivos ELF benignos e outros 1000 outros *malwares* ELF. Tendo em vista a distribuição igualitária, a mesma quantidade em ambas as classes (benignos e *malwares*), a REFADA torna-se adequada ao aprendizado dotado de IA. O objetivo do balanceamento entre as classes é evitar o favorecimento na taxa de acerto em relação a uma determinada classe.

Em relação às pragas virtuais, nosso conjunto de dados extraiu arquivos maliciosos ELF ARM do VirusShare[2], que é um repositório de amostras de *malware* para fornecer amostras de código malicioso vivo a pesquisadores de segurança, respondentes de incidentes e analistas forenses. Com relação aos arquivos ELF benignos, as amostras foram extraídas da imagem de um distribuição GNU/Linux Fedora para arquitetura ARM de 32 bits, presente no site oficial da desenvolvedora. Além disso, bancos de dados de outras plataformas de executáveis benignos também foram usados: GNU/Linux Debian para ARM e amostras compiladas. Para evitar a repetição dos exemplos, foram utilizados *scripts* autoriais codificados em python. O *script* lê os arquivos buscados e exclui suas cópias.

A construção do ambiente de teste para o experimento foi realizada utilizando o emulador de software livre QEMU[3]. Ao contrário de outras soluções de virtualização mais populares, como o Oracle VirtualBox, o emulador é capaz de instalar sistemas operacionais em máquinas com processadores divergentes do *host*, como as arquiteturas IoT ARM e RISC-V, que diferem das arquiteturas voltadas para *desktop*. O sistema convidado, arquitetura ARM de 32 bits, foi configurado e as distribuições GNU/Linux Debian e Fedora foram instaladas para compilação e obtenção de amostras para análise em *sandbox*.

Todos os arquivos benignos foram auditados pelo VirusTotal. Então, os arquivos ELF ARM benignos, contidos em nosso banco de dados, tiveram sua benevolência atestada pelos principais antivírus comerciais do mundo. Os resultados obtidos correspondentes às análises dos ficheiros ELF benignos e malignos, resultantes da auditoria do VirusTotal, estão disponíveis para consulta no endereço virtual da nossa base de dados [25].

O objetivo da criação do nosso conjunto de dados é dar plena possibilidade de replicacão metodologia proposta por terceiros em trabalhos futuros. Então, nosso banco de dados disponibiliza, gratuitamente, de todas as suas amostras, tanto benignas quanto malignas:

- Auditorias VirusTotal;
- Análise dinâmica do Cuckoo Sandbox.

Nosso conjunto de dados também disponibiliza em seu endereço virtual, 1.000 amostras de arquivo ELF benigno para ARM de 32 bits. Além disso, nosso conjunto de dados exibe a lista de todos os outros 1.000 arquivos ARM de 32 bits, desta vez, *malware*. Assim, existe a possibilidade de aquisição de todo o *malware* utilizado por nosso banco de dados, através do estabelecimento de convênios e submissão às regras de uso do VirusShare [25]. Conclui-se que nosso conjunto de dados confere transparência e imparcialidade à pesquisa, além de demonstrar a veracidade dos resultados alcançados. Portanto, espera-se que nosso conjunto de dados sirva de base para a criação de novos trabalhos científicos visando antivírus de próxima geração.

[2]VirusShare: banco de dados de arquivos maliciosos. Disponível em: https://virusshare.com. Acessado em junho de 2021.

[3]QEMU. Disponível em: https://www.qemu.org. Acessado em junho de 2021

B. Extração Dinâmica de Atributos

Os atributos do arquivo ELF para a arquitetura ARM de 32 bits originam-se da análise dinâmica de arquivos suspeitos. Portanto, em nossa metodologia, o *malware* é executado para infectar, intencionalmente, o GNU/Linux instalado na arquitetura ARM de 32 bits auditada, em tempo real (dinâmico), pelo *Cuckoo Sandbox*[4]. No total, são gerados 2.793 atributos de cada arquivo ELF ARM, referentes ao monitoramento do arquivo suspeito no ambiente controlado proposto. Para facilitar o entendimento dos neurônios da camada de entrada, nosso repositório apresenta, além as amostras e análises, a descrição dos atributos auditados pelo antivírus autoral [25]. A seguir, os grupos de recursos são detalhados:

- Recursos relacionados a máquinas virtuais. O objetivo é verificar se o arquivo auditado busca detectar se as máquinas virtuais Bochs, Sandboxie, VirtualBox, VirtualPC, VMware, Xen ou Parallels estão sendo utilizadas por meio da presença de arquivos, instruções e *drivers* de dispositivos por elas utilizados;
- Recursos relacionados a *malware*. Verifica se o arquivo auditado tenta criar *Mutexes* (arquivos de nome único, com uma função para definir um estado de bloqueio/desbloqueio, o que garante que apenas um processo por vez use os recursos);
- Recursos relacionados ao *Bitcoin*. Examina se o arquivo testado tenta instalar a biblioteca OpenCL, ferramenta de mineração *bitcoins*;
- Recursos relacionados a *bots* (máquinas que realizam tarefas automáticas de rede, maliciosas ou não, sem o conhecimento de seus proprietários);
- Recursos relacionados a navegadores. Verifica se o arquivo tentou:
  - Modificar as configurações de segurança do navegador;
  - Modificar a página inicial do navegador;
  - Obter informações privadas de navegadores de Internet instalados localmente.
- Recursos relacionados ao Firewall. A proposta de auditoria forense digital se o arquivo tentar modificar as políticas e configurações locais do firewall;
- Recursos relacionados à computação em nuvem. O arquivo é auditado quando tenta se conectar a serviços de armazenamento e/ou arquivos do Dropbox, Google, MediaFire, MegaUpload, RapidShare, Cloudflare e Wetransfer;
- Recursos associados à sugestão de tráfego de rede do sistema operacional GNU/Linux em formato PCAP(*Packet Capture*)[5];
- Recursos relacionados ao *Ransomware* (tipo de *malware* que, por meio de criptografia, deixa os arquivos da vítima inutilizáveis, em seguida, solicita um resgate em troca do uso normal posterior dos arquivos

[4]Cuckoo: Automated Malware Analysis. Disponível em: https://cuckoosandbox.org/. Acessado em novembro de 2021.

[5]PCAP: Formato de arquivo que armazena a interceptação de pacotes de dados que trafegam em uma rede.

do usuário, um resgate geralmente pago de forma indetectável, como *bitcoins*);
- Recursos relacionados à exploração que constituem *malware* que tenta tirar proveito de vulnerabilidades conhecidas ou não agrupadas, falhas ou defeitos no sistema ou um ou mais de seus componentes, a fim de causar instabilidades e comportamento imprevisto em seu hardware e software;
- Recursos relacionados a *Infostealers*, programas maliciosos que coletam informações confidenciais do computador afetado.

Além da detecção de comportamentos suspeitos, como chamadas API, a análise dinâmica também permite a reconstituição (limpeza) do Sistema Operacional (SO) por meio da auditoria dos malefícios promovidos pelo arquivo malicioso no SO assumindo que o dano não é irreversível. Ressalta-se que a reconstituição do SO, tecnicamente denominado vacina, é importante porque não basta detectar e eliminar o *malware* para que a vítima fique livre de suas ações. Além da eliminação do *malware*, é necessário desfazer todas as suas malfeitorias, como ter desabilitado o firewall, abrindo brechas para *sockets* usados por *backdoors*. Então, se não houvesse auditoria fornecida pela análise dinâmica, caberia ao cybervigilante monitorar, manualmente, qualquer mudança de SO que tornaria o processo lento e estressante.

C. Mineração de Características

Dentre os diversos desafios quanto à confecção de um repositório de aprendizado estatístico está a mineração de atributos. Dados oriundos de uma aplicação real, como a auditoria de sistemas GNU/Linux, são altamente suscetíveis a aumentar demasiadamente a dimensionalidade da aplicação. Visando melhorar a qualidade dos dados, diversas técnicas podem ser utilizadas para superar esse desafio, sendo definida em termos da precisão, completude, consistência, credibilidade e interpretabilidade.

No presente trabalho, os atributos auditados pelo Cuckoo Sanbox muitas vezes podem não fomentar a capacidade de generalização aos classificadores estudados. Como método de mineração de atributos, alguns comportamentos, auditados pelo *sandbox*, são desprezados. O critério adotado de mineração refere-se à eliminação de recursos que dizem respeito a um único arquivo ELF para arquitetura ARM de 32 bits, por exemplo *process_id*, *process_name*, *md5*, *sha*, entre outros. Inicialmente, a base de dados é constituída de 16.383 características extraídas pelo *sandbox* e, após a transformação, passa a ser composta de 2.793 atributos. Dessa maneira, há a redução na dimensionalidade, e consequente volume de processamento, sem haver a perda da capacidade de reconhecimento de padrão dos *malwares* e posterior generalização por parte dos classificadores.

A engenharia de atributos provida a partir das análises do Cuckoo Sandbox é fundamental para a rede neural autoral adquirir capacidade de generalização durante a etapa de classificação. Ao invés de se ater a eventos individuais, a referida *sandbox* é capaz de reconstruir a cadeia de eventos invocada pelo aplicativo suspeito. Tal fato propicia que o aplicativo auditado não seja condenado em função apenas de um recurso de forma isolada (eg. acessar o Wi-Fi). Logo, um bom classificador, auxiliado pela mineração de atributos, apresenta a excelente característica de aprender milhares de comportamentos suspeitos em questões de segundos e torna capaz de um antivírus ser prototipado em qualquer dispositivo com baixa capacidade de processamento e armazenamento.

Após a mineração de recursos, os comportamentos relevantes dos arquivos ELF servem como atributos de entrada para as máquinas de aprendizagem estatística, especificamente, redes neurais artificiais usadas como classificadores. O objetivo é agrupar arquivos ARM ELF em duas classes: benignos e *malware*. A etapa de classificação é explicada, em detalhes, na seção subsequente. Os resultados da classificação são descritos no seção VIII.

D. Classificadores

Quanto ao reconhecimento de padrão em arquivos maliciosos, uma tarefa essencial diz respeito à atribuição de um rótulo (classe) a cada arquivo investigado a partir de suas características. Nosso antivírus emprega redes neurais artificiais como classificadores. 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 obtidas durante a etapa de treinamento.

O objetivo do classificador composto por redes neurais é obter uma função de separação entre as classes do antivírus (*malware*, benigno). Dessa forma, ao ser apresentado a um arquivo inédito, a função é aplicada e, então, atribui uma classe na qual esse arquivo suspostamente pertence. Matematicamente, $c = f(x)$, onde $x = x_1, x_2, ..., x_t$ um vetor características extraídas do arquivo investigado, corresponde às 2.793 características dinâmicas, c é a classe e, por fim, f é a função de mapeamento do classificador.

O antivírus proposto emprega redes neurais extremas como classificadores [40]. As arquiteturas das redes neurais possuem uma camada de entrada contendo uma série de neurônios em relação ao vetor de recursos extraídos do monitoramento de arquivo ARM de 32 bits em um ambiente controlado. Portanto, os classificadores empregados devem ter uma camada de entrada contendo 2.793 neurônios. Eles dizem respeito aos recursos de auditoria do arquivo ARM de 32 bits. A camada de saída possui dois neurônios, correspondentes a amostras benignas e de *malware*.

Em vez de usar *kernels* convencionais, *kernels* autorais são utilizados para ELMs. Empregamos mELMs com *kernels* inspirados na morfologia matemática baseada em operadores não lineares de erosão e dilatação. Os mELMs têm a capacidade de mapear com precisão as diferentes distribuições referentes a diferentes problemas.

TABELA 5
RESULTADOS DAS REDES ELM. OS PARÂMETROS (C, γ) VARIAM DE ACORDO COM O CONJUNTO $\left\{2^{-24}, 2^{-10}, 2^{0}, 2^{10}, 2^{25}\right\}$. EXISTEM APENAS AS DESCRIÇÕES DOS MELHORES E DOS PIORES CASOS.

kernel	(C, γ)	Treinamento (%)	Teste (%)	Tempo Treinamento (s)	Tempo Teste (s)
Wavelets	$(2^0, 2^{25})$	52.65 ± 0.43	**52.20 ± 3.50**	1.83 ± 0.03	0.35 ± 0.03
	$(2^{-10}, 2^{10})$	**63.26 ± 0.57**	49.10 ± 4.01	2.00 ± 0.04	0.36 ± 0.02

A eficácia das nossas redes neurais morfológicas deve-se à sua capacidade de adaptação a qualquer tipo de distribuição, uma vez que o seu mapeamento não obedece a nenhuma figura geométrica convencional.

E. Validação Cruzada

No trabalho proposto são investigados 10 *folds*, referentes à validação cruzada do método *k-fold*, para cada *kernel*. O objetivo é que os resultados obtidos não sejam influenciados pelos conjuntos de treinamento e teste. Para tal, o total de arquivos é dividido em dez partes. Na execução inicial, a primeira parte é destinada ao conjunto de teste, enquanto as demais são reservadas ao treinamento. A alternância ocorre por dez vezes até que todas as dez partes tenham sido submetidas à fase de teste. A acurácia do ELM é obtida pela média aritmética da taxa de acerto nas dez iterações para cada combinação. Como mencionado anteriormente, na rede ELM não há retropropagação de dados. Portanto, o objetivo do método de validação cruzada *k-fold* não é estabelecer um critério de parada para evitar o treinamento em excesso (*overfitting*), mas sim verificar se o classificador sofre mudanças abruptas em suas acurácias a depender dos conjuntos destinados ao treinamento e teste. Há ainda o cuidado metodológico de selecionar de forma igualitária, e aleatória, exemplares benignos e *malwares* para cada *fold*. Tal fato, objetiva que classificadores tendenciosos, em relação a determinada classe, não tenham favorecimento em suas taxas de acerto.

VII. RESULTADOS DAS REDES ELM

EMPREGAMOS sete tipos de *kernel* diferentes para as redes neurais ELMs. No estado da arte, cinco desses *kernels* são descritos por HUANG, *et al*, (2012), eles são; *Wavelets* Transformada, Sigmoide, Seno, Limite Rígido e Tribas (Função de Base Triangular) [40]. Além disso, os núcleos autorais são empregados: Dilatação e Erosão.

O *kernel Wavelets* não possui uma camada oculta [40]. Os cálculos são baseados na transformação dos dados de entrada e podem funcionar quase como *kernels* contendo arquiteturas com camadas ocultas [40]. Uma boa capacidade de generalização desses *kernels* depende de uma escolha ajustada de parâmetros (C, γ) [40]. O parâmetro de custo C refere-se 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 [40]. Não existe um método universal no sentido de escolher os parâmetros (C, γ).

No presente trabalho, há a investigação dos parâmetros (C, γ) inspirado no método proposto por HUANG, *et al.* (2012), que consiste em treinar sequências crescentes de C e γ, matematicamente, 2^n, onde $n = \left\{-24, -10, 0, 10, 25\right\}$ [40]. A hipótese é verificar se esses parâmetros apresentam valores diferentes dos padrões; $(C = 1, \gamma = 1)$, gerar melhores resultados [40]. Cada combinação emprega validação cruzada por meio do método *k-fold*, onde $k = 10$. A precisão da ELM é a média aritmética da taxa de acerto obtida nas dez iterações para cada combinação.

A Tabela 5 detalha os resultados obtidos pelas redes neurais ELMs com *kernel Wavelets*. Cada linha nesta tabela contém 10 execuções referentes à validação cruzada do método *k-fold*, onde $k = 10$. Em relação à precisão na fase de teste, o desempenho médio máximo foi de 52,20% na distinção entre casos benignos e *malwares* com os parâmetros $(C, \gamma) = (2^0, 2^{25})$. Na Tabela 5, existem apenas as descrições do melhor e do pior resultado da aplicação ao conjunto de teste, nesta ordem, para o *kernel Wavelets*.

A Tabela 6 apresenta os resultados alcançados pela rede ELM com os núcleos Sigmoide, Seno, Limite Rígido, Tribas (Função de Base Triangular), Dilatação e Erosão. Eles empregam arquiteturas de camadas ocultas. Em seguida, há a investigação em relação à quantidade de neurônios na camada oculta desses *kernels*. A hipótese é verificar se arquiteturas que requerem maior volume de cálculos, como dobrar o número de neurônios na camada oculta, são capazes de gerar taxas de acurácia melhores em comparação com arquiteturas que requerem menor quantidade de cálculos. Existe a avaliação de duas arquiteturas; eles empregam 100 e 500 neurônios em suas respectivas camadas ocultas. Estas arquiteturas possuem antecedentes de excelente precisão na aplicação de redes ELM na área de Engenharia Biomédica [31]. Cada linha na Tabela 6 contém 10 execuções distintas referentes ao método *k-fold*, onde k=10. Em relação à precisão, o desempenho médio máximo no conjunto de teste foi de 98,75% com desvio padrão de 0,89% por meio do *kernel* de Dilatação dotado de 500 neurônios em sua camada oculta.

VIII. RESULTADOS EM RELAÇÃO AO ESTADO DA ARTE

NESTA seção, o antivírus autoral é comparado com os antivírus de última geração. Para evitar comparações injustas, o estágio de extração de recursos é padronizado monitorando 2.793 comportamentos, após a mineração de atributos, que o executável ARM suspeito pode fazer quando executado propositalmente. O antivírus autoral emprega redes neurais morfológicas rasas. Nosso antivírus

TABELA 6
RESULTADO DAS REDES ELM. O NÚMERO DE NEURÔNIOS NA CAMADA OCULTA VARIA DE ACORDO COM O CONJUNTO 100, 500.

kernel	neurônios	Taxa de Treinamento (%)	Taxa de Teste (%)	Tempo Treinamento (s)	Tempo Teste (s)
Sigmoide	100	50.30 ± 0.05	50.15 ± 0.24	0.34 ± 0.01	0.01 ± 0.00
	500	50.30 ± 0.05	50.15 ± 0.24	1.68 ± 0.07	0.06 ± 0.01
Seno	500	74.54 ± 0.64	54.75 ± 2.42	1.31 ± 0.03	0.07 ± 0.01
	100	59.84 ± 0.45	51.75 ± 2.92	0.34 ± 0.01	0.01 ± 0.01
Limite Rígido	100	50.00 ± 0.00	50.00 ± 0.00	0.35 ± 0.02	0.01 ± 0.01
	500	50.00 ± 0.00	50.00 ± 0.00	1.86 ± 0.03	0.07 ± 0.01
Tribas	100	50.05 ± 0.02	50.00 ± 0.00	0.34 ± 0.01	0.01 ± 0.02)
	500	50.05 ± 0.02	50.00 ± 0.00	1.09 ± 0.01	0.06 ± 0.01
Dilatação	500	**99.93 ± 0.03**	**98.75 ± 0.89**	23.31 ± 0.07	2.46 ± 0.02
	100	99.03 ± 0.10	98.20 ± 0.63	3.33 ± 0.02	0.33 ± 0.01
Erosão	500	97.62 ± 0.10	96.15 ± 0.75	24.44 ± 0.09	2.59 ± 0.02
	100	92.48 ± 0.21	91.40 ± 1.61	4.20 ± 0.04	0.42 ± 0.01

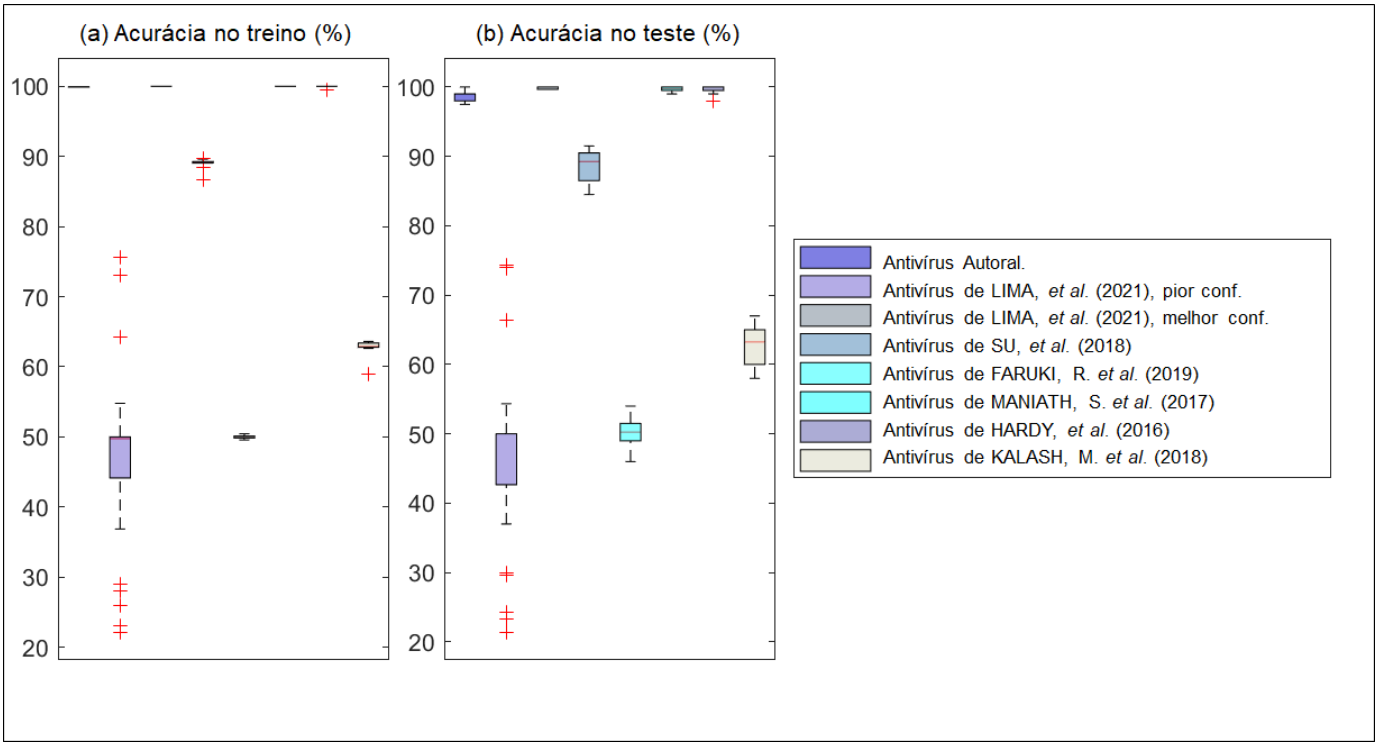

Fig. 5. *Boxplots* referentes à precisão do antivírus autoral e ao estado da arte.

é dotado de *kernel* mELM de Dilatação e contém 500 neurônios em sua camada oculta.

Por outro lado, o antivírus feito por LIMA, *et al.* (2021) [37] emprega redes neurais rasas baseadas em retropropagação. LIMA, *et al.* (2021) investigou onze funções de aprendizagem distintas a fim de otimizar a precisão de seu antivírus. Para cada função de aprendizagem, LIMA, *et al.* (2021) explora 4 arquiteturas de camadas ocultas.

Nosso antivírus autoral também é comparado a antivírus baseado em *Deep Learning*. No presente trabalho, eles são replicados os antivírus feitos por SU, *et al.* (2018), FARUKI, *et al.* (2019), MANIATH, *et al.* (2017), HARDY, *et al.* (2016), and KALASH, M. *et al.* (2018). Nós replicamos esses trabalhos usando nosso conjunto de dados.

A Figura 5 e a Figura 6 são representações gráficas dos resultados descritos na Tabela 7. A Figura 5 (a) mostra os *boxplots* para a melhor precisão no treinamento. O antivírus autoral obteve uma precisão média de 99,93%. O antivírus da LIMA, *et al.* (2021) obteve acurácia média de 47,20% e 100,00%, em seus piores e melhores cenários, respectivamente. Esses resultados foram obtidos por meio das funções de aprendizagem "Treinamento em lote com peso e regras de aprendizagem de viés" e "retropropagação de gradiente conjugado com reinicialização de Powell-Beale", respectivamente. A melhor e a pior arquitetura têm uma única camada oculta contendo 500 neurônios.

A Figura 5 (b) apresenta os *boxplots*, em fase de teste, em relação ao antivírus autoral e o estado da arte. O antivírus autoral obteve desempenho médio de 98,75% com desvio padrão de 0,89%. O antivírus da LIMA, *et al.* (2021) obteve acertos médios de 47,02% e 99,87%, em seus piores e melhores cenários, respectivamente. Portanto, é corroborado que as redes neurais baseadas em retropropagação podem sofrer grandes variações, em suas precisões,

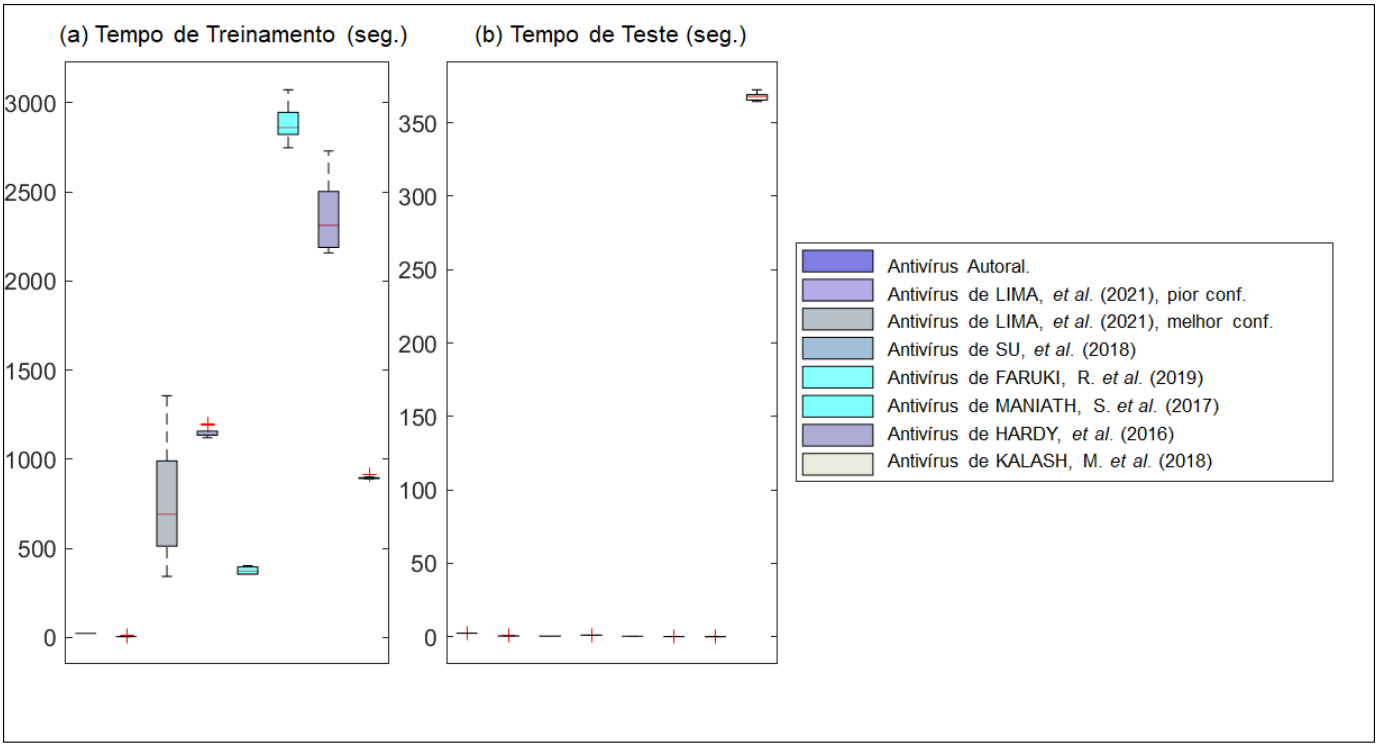

Fig. 6. *Boxplots* sobre os tempos de processamento do antivírus autoral e o estado da arte.

dependendo de seus parâmetros de configuração. Então, a decisão de LIMA, *et al.* (2021) foi salutar. Este antivírus de última geração explora diferentes funções de aprendizagem, gradientes e arquiteturas para otimizar a precisão de suas redes neurais com base na retropropagação de dados.

A Figura 6 (a) e a Figura 6 (b) apresentam os *boxplots* referentes aos tempos despendidos nas fases de treinamento e teste, respectivamente. O antivírus autoral consome apenas 23,31 segundos para concluir, em média, seu treinamento. Em relação ao tempo de treinamento, o antivírus da HARDY, *et al.* (2016) é o mais lento. Em relação ao tempo consumido durante a fase de teste, todas as técnicas consumiram tempos muito próximos sem grandes discrepâncias.

O tempo de teste é importante em aplicações convencionais, tais como processamento de imagens, engenharia biomédica e etc. A explicação é que, em regra geral, não surgem novas variantes de anomalias biológicas em questão de segundos. Por outro lado, na aplicação em estudo, o sistema proposto requer treinamento constante, visto que a cada segundo são lançados 8 novos *malwares*. Logo, o tempo de treinamento assume papel fundamental de forma que o antivírus inteligente não se torne obsoleto mesmo que recém lançado.

A Tabela 8 mostra as matrizes de confusão das técnicas apresentadas na Tabela 7 em termos percentuais.

A matriz de confusão é importante para verificar a qualidade da aprendizagem supervisionada. As classes desejadas são organizadas na etiqueta vertical, enquanto as classes obtidas estão na etiqueta horizontal. Na matriz de confusão, a diagonal principal é ocupada por casos sempre que a classe obtida coincide com a classe esperada, denominados casos positivos verdadeiros. Então, um bom classificador tem a diagonal principal ocupada por valores altos e outros elementos têm valores baixos.

A Tabela 8 mostra ainda as principais diagonais destacadas em negrito. Nosso antivírus, em fase de teste, classificou erroneamente em média 2,20% dos casos como benignos quando eram casos de *malware* (falso negativo). Seguindo o mesmo raciocínio, houve uma classificação média de 0,30% dos casos ditos *malware* quando eram aplicativos benignos (falso positivo).

Ainda em relação à Tabela 8, a sensibilidade e a especificidade referem-se à capacidade do antivírus em identificar *malware* e aplicativos benignos, respectivamente. O trabalho proposto apresenta a matriz de confusão em termos percentuais para facilitar a interpretação da sensibilidade e especificidade. Em síntese, a sensibilidade e a especificidade são apresentadas na própria matriz de confusão, descrita na Tabela 8. Por exemplo, o antivírus proposto tem uma média de 99,70% em relação à sensibilidade e aos verdadeiros positivos. Seguindo o mesmo raciocínio, o antivírus autoral obtém, em média, 97,87% para especificidade e verdadeiros negativos.

A Tabela 9 mostra os testes de hipóteses paramétricos *t-student* e não paramétricos de *Wilcoxon* entre nosso antivírus e o estado da arte. É possível concluir que nosso antivírus autoral é estatisticamente diferente em comparação a todas as outras amostras. A explicação é que tanto nos testes paramétricos *t-student* quanto nos não paramétricos de *Wilcoxon*, a hipótese nula foi rejeitada.

O antivírus autoral demonstrou grande vantagem

TABELA 7
Comparação entre o antivírus autoral e o estado da arte.

Técnica	Treinamento (%)	Teste (%)	Tempo Treinamento (s)	Tempo Teste (s)
Antivírus Autoral	99.93 ± 0.03	98.75 ± 0.89	23.31 ± 0.07	2.46 ± 0.02
Antivírus de LIMA, *et al.* (2021), pior conf. [31]	47.20 ± 12.51	47.02 ± 12.52	5.88 ± 2.03	0.57 ± 0.08
Antivírus de LIMA, *et al.* (2021), melhor conf. [31]	**100.00** ± 0.00	**99.87 ± 0.17**	742.32 ± 273.91	0.45 ± 0.03
Antivírus de SU, *et al.* (2018) [34]	88.93 ± 0.84	88.55 ± 2.55	1149.29 ± 26.52	1.14 ± 0.04
Antivírus de FARUKI, *et al.* (2019) [29]	50.00 ± 0.28	50.00 ± 2.54	376.12 ± 21.83	0.39 ± 0.02
Antivírus de MANIATH, *et al.* (2017) [33]	100.00 ± 0.00	99.65 ± 0.41	2883.53 ± 96.22	0.11 ± 0.01
Antivírus de HARDY, *et al.* (2016) [32]	99.96 ± 0.14	99.50 ± 0.62	2368.76 ± 220.66	0.09 ± 0.01
Antivírus de KALASH, *et al.* (2018) [30]	62.66 ± 1.33	62.65 ± 3.08	896.07 ± 6.48	367.51 ± 2.46

TABELA 8
Matriz de confusão do antivírus autoral e o estado da arte. (%)

Técnica		Treinamento		Teste	
		Malware	Benigno	*Malware*	Benigno
Antivírus Autoral	*Malware*	**99.86 ± 0.05**	**0.00 ± 0.00**	**97.80 ± 1.75**	0.30 ± 0.48
	Benigno	0.14 ± 0.05	**100.00 ± 0.00**	2.20 ± 1.75	**99.70 ± 0.48**
Antivírus de LIMA, *et al.* (2021), pior conf. [31]	*Malware*	**50.12 ± 26.99**	65.29 ± 32.47	**46.77 ± 27.16**	50.11 ± 37.37
	Benigno	43.21 ± 26.10	**24.71 ± 25.19**	43.23 ± 26.47	**23.22 ± 25.59**
Antivírus de LIMA, *et al.* (2021), melhor conf. [31]	*Malware*	**100.00 ± 0.00**	0.00 ± 0.00	**99.74 ± 0.33**	0.00 ± 0.00
	Benigno	0.00 ± 0.00	**100.00 ± 0.00**	0.26 ± 0.33	**100.00 ± 0.00**
Antivírus de SU, *et al.* (2018) [34]	*Malware*	**81.61 ± 2.31**	3.76 ± 0.80	**81.04 ± 4.51**	3.90 ± 1.59
	Benigno	18.39 ± 2.31	**96.24 ± 0.80**	18.96 ± 4.51	**96.10 ± 1.59**
Antivírus de FARUKI, *et al.* (2019) [29]	*Malware*	**0.00 ± 0.00**	0.00 ± 0.00	**0.00 ± 0.00**	0.00 ± 0.00
	Benigno	100.00 ± 0.00	**100.00 ± 0.00**	100.00 ± 0.00	**100.00 ± 0.00**
Antivírus de MANIATH, *et al.* (2017) [33]	*Malware*	**100.00 ± 0.00**	0.00 ± 0.00	**99.50 ± 0.70**	0.20 ± 0.43
	Benigno	0.00 ± 0.00	**100.00 ± 0.00**	0.50 ± 0.70	**99.80 ± 0.43**
Antivirus made by HARDY, *et al.* (2016) [32]	*Malware*	**99.93 ± 0.21**	0.02 ± 0.07	**99.40 ± 0.69**	0.40 ± 0.95
	Benigno	0.07 ± 0.21	**99.98 ± 0.07**	0.60 ± 0.69	**99.60 ± 0.95**
Antivírus de KALASH, *et al.* (2018) [30]	*Malware*	**93.93 ± 1.95**	68.62 ± 4.54	**93.70 ± 2.54**	68.40 ± 6.04
	Benigno	6.07 ± 1.95	**31.38 ± 4.54**	6.30 ± 2.54	**31.60 ± 6.04**

TABELA 9
Teste de hipótese *T-student* e *Wilcoxon* do antivírus autoral e do estado da arte

Comparação	*t-student* (teste paramétrico)		*Wilcoxon* (teste não paramétrico)	
	Hipóteses	*p-value*	Hipóteses.	*p-value*
Antivírus Autoral *vs* Antivírus de LIMA, *et al.* (2021), pior conf.	1	7.91278e-20	1	1.89556e-11
Authorial Antivirus *vs* Antivírus de LIMA, *et al.* (2021), best conf.	1	1.42716e-07	1	6.06702e-07
Antivírus Autoral *vs* Antivírus de SU, *et al.* (2018)	1	1.44266e-19	1	2.29741e-11
Antivírus Autoral *vs* Antivírus de FARUKI, *et al.* (2019)	1	2.17731e-39	1	2.2579e-11
Antivírus Autoral *vs* Antivírus de MANIATH, *et al.* (2017)	1	6.83312e-07	1	1.87413e-05
Antivírus Autoral *vs* Antivírus de HARDY, *et al.* (2016)	1	0.00348677	1	0.000300536
Antivírus Autoral *vs* Antivírus de KALASH, *et al.* (2018)	1	5.33832e-34	1	2.2579e-11

quando comparado ao estado da arte. Nosso antivírus atinge um desempenho médio de 98,75% com um tempo médio de treinamento de 23,31 segundos. A relação média entre a precisão percentual e o tempo de treinamento em ordem reversa é empregada em Engenharia Biomédica [46]. O antivírus proposto é mais de 122 vezes melhor que o estado da arte em *Deep Learning* [33]. Admite-se que o estabelecimento desta relação assume uma função importante na Segurança da Informação, uma vez que 8 (oito) novos *malwares* são lançados a cada segundo [38]. Portanto, paradoxalmente, um antivírus recém-lançado pode já estar obsoleto e exigir novo treinamento por meio de uma vulnerabilidade recém-descoberta. Em síntese, o tempo de aprendizado de um antivírus não deve ser discrepante em comparação com a taxa de criação de novos *malwares* em todo o mundo.

IX. Conclusão

A proliferação de objetos inteligentes com capacidade de detecção, processamento e comunicação tem crescido nos últimos anos. De acordo com a literatura, a IoT

e outros paradigmas são requisitos e partes do conceito visionário da Indústria 4.0 [49] [50]. O *malware* representa um desafio significativo para os pesquisadores em IoT. Além da grande velocidade de disseminação e criação de novas variantes, a pesquisa da natureza furtiva exige que se desenvolva uma ferramenta eficaz para sua detecção. Assim, infere-se que a seleção do antivírus tem um papel importante no combate às pragas virtuais.

O objetivo deste artigo é a análise por redes neurais artificiais, após uma eficiente metodologia de extração e representação de atributos, para melhorar a precisão da classificação de sistemas de detecção de *malware* existentes na IoT. Diferentemente dos métodos tradicionais, que incluem uma análise manual para geração de assinaturas, além de atuar em camadas externas ao dispositivo como comunicação e aplicação, o foco da solução proposta é uma metodologia automática intra-dispositivo utilizando *sandbox* e processamento de análises máquina. Este estudo oferece uma abordagem de sistema de detecção de *malware* para software malicioso baseado em aprendizado de máquina usando o conjunto de dados baseado em arquitetura IoT de 32 bits do tipo ARM coletado do VirusShare. Em um estágio posterior, por meio da engenharia de atributos, os atributos comportamentais mais representativos foram selecionados. Finalmente, eles foram transformados em vetores binários para treinar/testar classificadores de aprendizado de máquina usados pelo estado da arte, como MLP, VGG-16 e ELM com diferentes *kernels*.

O trabalho atual analisou 68 antivírus disponíveis comercialmente. Em nossa avaliação, o intervalo de *malware* ELF detectado para arquitetura ARMs de 32 bits foi de 0% a 82,6%, dependendo de qual antivírus comercial foi escolhido. Ao todo, nossos antivírus monitoram e avaliam estatisticamente 2.793 ações que o arquivo ELF ARM suspeito pode executar quando tem seu processamento inicializado em GNU/Linux contido em arquitetura ARM de 32 bits. Em média, eles foram capazes de detectar 46,08% do *malware* ELF ARM. Feita a análise das amostras, foi possível identificar que os antivírus, em média, relataram falsos negativos e foram omitidos em 44,85% e 8,92% dos casos, respectivamente. No presente trabalho, a plataforma VirusTotal foi utilizada para submeter, de forma automatizada, o *malware* aos antivírus. É importante ressaltar que no VirusTotal não existe a possibilidade de escolher a versão *shareware* dos antivírus. O *kernel* de dilatação autoral é capaz de distinguir *malware* ELF para arquitetura ARM de 32 bits de aplicativos benignos em 91,58% dos casos, acompanhado por um tempo de treinamento de 52,36 segundos. Conclui-se que os serviços oferecidos nas versões *shareware* têm desempenho significativamente inferior do que nas versões completas.

Criamos um antivírus capaz de classificar arquivos ELF para arquitetura ARM de 32 bits entre benigno e *malware*. Ao todo, nosso antivírus monitora e pondera, estatisticamente, 2.793 ações que o arquivo ARM ELF suspeito pode realizar quando executado em GNU/Linux contido na arquitetura ARM de 32 bits. Em um ambiente controlado, nosso antivírus monitora alterações nos rastros de chamadas realizadas por todos os processos gerados pelo *malware*, arquivos sendo criados, excluídos e baixados pelo *malware* durante sua execução, despejos de memória dos processos de *malware* e rastreamento de tráfego de rede. O reconhecimento do padrão, referente às ações suspeitas, é feito por redes neurais extremas.

No estudo proposto, em vez de *kernels* convencionais, *kernels* autorais são empregados para ELMs. A rede ELM tem como principal característica a velocidade de treinamento e previsão de dados quando comparada às redes neurais convencionais. O *kernel* autoral de Dilatação é capaz de distinguir *malware* ELF para arquitetura ARM de 32 bits de aplicativos benignos em 98,75% dos casos, acompanhado por um tempo de treinamento de 23,31 segundos.

Os resultados experimentais indicam que o modelo morfológico ELM com o *kernel* autoral produz a melhor relação entre acurácia e tempo de treinamento em comparação com outros modelos do estado da arte. As redes neurais artificiais podem extrair padrões de conjuntos de dados de alto volume que consistem em atributos de dados, conforme demonstrado principalmente por nossos *kernels* de ELM. Verifica-se uma menor eficácia dos antivírus comerciais em relação aos serviços em larga escala e em tempo real. Vale ressaltar que, em nossos experimentos, os *malwares* ELF de arquitetura ARM analisados são de domínio público, empregados em atividades maliciosas, e com seus desempenhos classificados por respondentes de incidentes. Mesmo assim, mais de um terço dos antivírus comerciais avaliados não tinha conhecimento sobre a existência de arquivos de *malware* ARM investigados.

Conclui-se portanto que, buscando superar as limitações dos antivírus comerciais, os antivírus baseados em inteligência artificial são capazes de auditar milhares de *malware* e aprender estatisticamente quais são suas características maliciosas. É de vital importância para as plataformas de detecção fornecer mecanismos de vigilância cibernética que atendam às demandas de forma preventiva. Caso contrário, em cenários onde ocorrem falhas na identificação de aplicativos maliciosos, há uma iminência de dados confidenciais de dispositivos cuja confidencialidade seja quebrada, como no *Smart Health* IoT, ou tenham seu funcionamento comprometido. Portanto, após o aprendizado, o antivírus inteligente pode identificar e classificar *malware* IoT recém-criado de acordo com uma comparação entre seus recursos e aqueles identificados durante a fase de aprendizado, extraindo suas características. Desta forma, não haveria necessidade de esperar que um usuário fosse infectado, para posteriormente relatar uma atitude suspeita, só então o antivírus tomaria alguma providência quanto à descoberta de *malware*. Os antivírus inteligentes permitem a detecção preventiva de ameaças virtuais em um ambiente controlado antes que cheguem às máquinas dos usuários.

X. Trabalhos Futuros

NOSSO antivírus autoral baseado em redes neurais extrema pode ser estendido para fornecer proteção cibernética para outras arquiteturas equipadas com arquivos ELF para GNU/Linux. Dada a previsível deficiência dos antivírus comerciais, espera-se que os *malwares* IoT provoquem grandes distúrbios ao corromper arquiteturas computacionais e aplicações disruptivas em âmbitos embarcados. Assim, a intenção é estender nosso antivírus a outros sistemas operacionais com arquiteturas divergentes, como PowerPC, MIPS e RISC-V.

O objetivo futuro imediato é aplicar nossa metodologia a sistemas GNU/Linux com arquiteturas RISC-V [51], visto que tal arquitetura gradualmente se tornou uma das mais promissoras em um futuro não muito distante. Ainda como objetivo futuro, nosso antivírus visa auditar sistemas operacionais, equipados com arquivos ELF de arquiteturas IoT, especializados nas mais diversas aplicações como redes veiculares (*Vehicular Ad hoc Networks* - VANETs), drones e veículos autônomos não tripulados (UAV).

Referências

[1] CISCO, "Cisco 2014 WhitePaper," 2014. endereço: http://cdn.iotwf.com/resources/71/IoT_Reference_Model_White_Paper_June_4_2014.pdf.

[2] S. K. TARAI e S. SHAILENDRA, "Optimal and Secure Controller Placement in SDN Based Smart City Network," em *2019 International Conference on Information Networking (ICOIN)*, 2019, pp. 254–261. DOI: 10.1109/ICOIN.2019.8718165.

[3] A. TZOUNIS, N. KATSOULAS, T. BARTZANAS e C. KITTAS, "Internet of Things in agriculture, recent advances and future challenges," *Biosystems Engineering*, v. 164, pp. 31–48, 2017, ISSN: 1537-5110. DOI: https://doi.org/10.1016/j.biosystemseng.2017.09.007. endereço: https://www.sciencedirect.com/science/article/pii/S1537511017302544.

[4] S. M. R. ISLAM, D. KWAK, M. H. KABIR, M. HOSSAIN e K.-S. KWAK, "The Internet of Things for Health Care: A Comprehensive Survey," *IEEE Access*, v. 3, pp. 678–708, 2015. DOI: 10.1109/ACCESS.2015.2437951.

[5] J. GUBBI, R. BUYYA, S. MARUSIC e M. PALANISWAMI, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, v. 29, n. 7, pp. 1645–1660, 2013, Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services Cloud Computing and Scientific pplications — Big Data, Scalable Analytics, and Beyond, ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2013.01.010. endereço: https://www.sciencedirect.com/science/article/pii/S0167739X13000241.

[6] F. WANG, L. HU, J. ZHOU e K. ZHAO, "A Survey from the Perspective of Evolutionary Process in the Internet of Things," *International Journal of Distributed Sensor Networks*, v. 11, n. 3, p. 462 752, 2015. DOI: 10.1155/2015/462752.

[7] K. ASHTON et al., "That 'internet of things' thing," 2009.

[8] A. AL-FUQAHA, M. GUIZANI, M. MOHAMMADI, M. ALEDHARI e M. AYYASH, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE communications surveys & tutorials*, v. 17, n. 4, pp. 2347–2376, 2015.

[9] M. A. RAZZAQUE, M. MILOJEVIC-JEVRIC, A. PALADE e S. CLARKE, "Middleware for internet of things: a survey," *IEEE Internet of things journal*, v. 3, n. 1, pp. 70–95, 2015.

[10] D. MACEDO, L. A. GUEDES e I. SILVA, "A dependability evaluation for Internet of Things incorporating redundancy aspects," em *Proceedings of the 11th IEEE International Conference on Networking, Sensing and Control*, IEEE, 2014, pp. 417–422.

[11] R. BEAULIEU, D. SHORS, J. SMITH, S. TREATMAN-CLARK, B. WEEKS e L. WINGERS, "The SIMON and SPECK lightweight block ciphers," em *Proceedings of the 52nd Annual Design Automation Conference*, 2015, pp. 1–6.

[12] Y. BERGUIG, J. LAASSIRI e S. HANAOUI, "Anonymous and lightweight secure authentication protocol for mobile Agent system," *Journal of Information Security and Applications*, v. 63, p. 103 007, 2021.

[13] M. ALI, S. SHIAELES, G. BENDIAB e B. GHITA, "MALGRA: Machine Learning and N-Gram Malware Feature Extraction and Detection System," *Electronics*, v. 9, n. 11, 2020, ISSN: 2079-9292. endereço: https://www.mdpi.com/2079-9292/9/11/1777.

[14] V. H. BEZERRA, V. G. T. da COSTA, R. A. MARTINS, S. B. JUNIOR, R. S. MIANI e runo Bogaz ZARPELÃO, "Providing IoT host-based datasets for intrusion detection research," em *Anais do XVIII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, Natal: SBC, 2018, pp. 15–28. endereço: https://sol.sbc.org.br/index.php/sbseg/article/view/4240.

[15] A. F. A. KADIR, N. STAKHANOVA e A. A. GHORBANI, "Android botnets: What urls are telling us," em *International Conference on Network and System Security*, Springer, 2015, pp. 78–91.

[16] R. NIGAM, "A timeline of mobile botnets," *Virus Bulletin, March*, 2015.

[17] P. N. SRINIVASU, A. K. BHOI, S. R. NAYAK, M. R. BHUTTA e M. WOŹNIAK, "Blockchain Technology for Secured Healthcare Data Communication among the Non-Terminal Nodes in IoT Architecture in 5G Network," *Electronics*, v. 10, n. 12, 2021, ISSN:

2079-9292. endereço: https://www.mdpi.com/2079-9292/10/12/1437.

[18] Q. ABU AL-HAIJA e S. ZEIN-SABATTO, "An Efficient Deep-Learning-Based Detection and Classification System for Cyber-Attacks in IoT Communication Networks," *Electronics*, v. 9, n. 12, p. 2152, 2020.

[19] D. A. Patterson e J. L. Hennessy, *Computer organization and design ARM edition: the hardware software interface*. Morgan kaufmann, 2016.

[20] A. Antony e S. Sarika, "A Review on IoT Operating Systems," *International Journal of Computer Applications*, v. 975, p. 8887,

[21] *What is AI? Learn about Artificial Intelligence*, https://www.oracle.com/artificial-intelligence/what-is-ai/, Accessed on Feb 2022.

[22] T. MITCHELL, "Machine learning," 1997.

[23] S. LIMA, *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 e Information Privacy, vol. 1, 1st edn. IGI lobal, Hershey, 2021. DOI: http://dx.doi.org/10.4018/978-1-7998-5728-0.ch020.

[24] SANS, *SANS Institute InfoSec Reading Room. Out with The Old, In with The New: Replacing Traditional Antivirus*. Accessed on Feb 2020, 2017. endereço: https://www.sans.org/reading-room/whitepapers/analyst/old-new-replacing-traditional-antivirus-37377.

[25] REFADA, "REFADA (A Retrieval of ELF Files ARM to Dynamic Analysis)," 2021. endereço: https://github.com/refade/IoT_ARM.

[26] N. MONBURINON, S. M. S. ZABIR, N. VECHPRASIT, S. UTSUMI e N. SHIRATORI, "A Novel Hierarchical Edge Computing Solution Based on Deep Learning for Distributed Image Recognition in IoT Systems," em *2019 4th International Conference on Information Technology (InCIT)*, IEEE, 2019, pp. 294–299.

[27] F. MEDJEK, D. TANDJAOUI, N. DJEDJIG e I. ROMDHANI, "Fault-tolerant AI-driven Intrusion Detection System for the Internet of Things," *International Journal of Critical Infrastructure Protection*, v. 34, p. 100 436, 2021, ISSN: 1874-5482. DOI: https://doi.org/10.1016/j.ijcip.2021.100436. endereço: https://www.sciencedirect.com/science/article/pii/S1874548221000287.

[28] M. WOZNIAK, J. SILKA, M. WIECZOREK e M. ALRASHOUD, "Recurrent Neural Network Model for IoT and Networking Malware Threat Detection," *IEEE Transactions on Industrial Informatics*, v. 17, n. 8, pp. 5583–5594, 2021. DOI: 10.1109/TII.2020.3021689.

[29] P. FARUKI e B. BUDDHADEV, "DroidDivesDeep: Android Malware Classification via Low Level Monitorable Features with Deep Neural Networks," *International Conference on Security Privacy*, 2019. DOI: https://doi.org/10.1007/978-981-13-7561-3_10.

[30] M. KALASH e M. e. a. ROCHAN, "Malware Classification with Deep Convolutional Neural Networks," *9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2018. DOI: https://doi.org/10.1109/NTMS.2018.8328749.

[31] S. LIMA, H. SILVA e J. LUZ, "Artificial intelligence-based antivirus in order to detect malware preventively," *Progress in Artificial Intelligence*, 2020. DOI: https://doi.org/10.1007/s13748-020-00220-4.

[32] W. HARDY e C. e. a. LINGWEI, "DL 4 MD : A Deep Learning Framework for Intelligent Malware Detection," *In Int'l Conf. Data Mining*, pp. 61–67, 2016.

[33] S. MANIATH e A. ASHOK, "Deep Learning LSTM based Ransomware Detection," *Recent Developments in Control, Automation Power Engineering*, 2017. DOI: https://doi.org/10.1109/RDCAPE.2017.8358312.

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

[35] M. d. CRUZ, J. RODRIGUEZ, J. AL-MUHTADI, V. KOROTAEV e V. de ALBUQUERQUE, *A reference model for internet of things middleware. Internet Things J 5: 871–883*, 2018.

[36] R. KHAN, S. U. KHAN, R. ZAHEER e S. KHAN, "Future internet: the internet of things architecture, possible applications and key challenges," em *2012 10th international conference on frontiers of information technology*, IEEE, 2012, pp. 257–260.

[37] S. M. LIMA, D. M. SOUZA, R. P. PINHEIRO et al., "Next generation antivirus endowed with bitwise morphological extreme learning machines," *Microprocessors and Microsystems*, v. 81, p. 103 724, 2021, ISSN: 0141-9331. DOI: https://doi.org/10.1016/j.micpro.2020.103724. endereço: https://www.sciencedirect.com/science/article/pii/S0141933120308693.

[38] INTEL, *McAfee Labs*. Accessed on Feb 2020, 2018. endereço: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-mar-2018.pdf.

[39] S. M. L. LIMA, A. G. SILVA-FILHO e W. P. DOS SANTOS, "A methodology for classification of lesions in mammographies using Zernike Moments, ELM and SVM Neural Networks in a multi-kernel pproach," *In: 2014 IEEE International Conference on Systems, Man and Cybernetics SMC, San Diego*, 2014. DOI: https://doi.org/10.1109/SMC.2014.6974041.

[40] G. B. HUANG, "Extreme Learning Machine for Regression and MultiClass Classification," *IEEE Transactions on Systems, Man, and Cybernetics*, v. 42(2),

pp. 513–519, 2012. DOI: https://doi.org/10.1109/TSMCB.2011.2168604.

[41] ——, "Classification ability of single hidden layer feedforward neural networks," *The IEEE Transactions on Neural Networks and Learning Systems*, v. 11(3), pp. 799–801, 2000. DOI: https://doi.org/10.1109/72.846750.

[42] W. W. AZEVEDO, "Fuzzy Morphological Extreme Learning Machines to detect and classify masses in mammograms," *In: 2015 IEEE International Conference on Fuzzy Systems (FUZZIEEE), Istanbul.*, 2015. DOI: https://doi.org/10.1109/FUZZ-IEEE.2015.7337975.

[43] ——, "Morphological extreme learning machines applied to detect and classify masses in mammograms," *In: 2015 International Joint Conference on Neural Networks (IJCNN), Killarney.*, 2015. DOI: https://doi.org/10.1109/IJCNN.2015.7280774.

[44] ——, "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 edical Image Analysis. 1ed.Londres: Elsevier Science.*, pp. 1–30, 2020. DOI: https://doi.org/10.1016/B978-0-12-819295-5.00003-2.

[45] S. M. L. LIMA, SILVA-FILHO e W. P. SANTOS, *Morphological Decomposition to Detect and Classify Lesions in Mammograms.In: Wellington Pinheiro dos Santos; Maíra Araújo de Santana; ashington Wagner Azevedo da Silva. (Org.). Understanding a Cancer Diagnosis.* https://novapublishers.com/shop/understanding-a-cancer-diagnosis/, 2020, pp. 27–64.

[46] S. LIMA, A. G. SILVA-FILHO e W. P. SANTOS, "Detection and classification of masses in mammographic images in a multi-kernel approach," *Computer Methods and Programs in Biomedicine*, v. 134, pp. 11–29, 2016. DOI: https://doi.org/10.1016/j.cmpb.2016.04.029.

[47] J. M. S. PEREIRA, *Method for Classification of Breast Lesions in Thermographic Images Using ELM Classifiers. In: Wellington Pinheiro dos Santos; Maíra raújo de Santana; Washington Wagner Azevedo da Silva. (Org.). Understanding a Cancer Diagnosis.* https://novapublishers.com/shop/understanding-a-cancer-diagnosis/, 2020, pp. 117–132.

[48] W. P. SANTOS, *Mathematical Morphology In Digital Document Analysis and Processing.* New York: Nova Science, 2011, vol. 8, pp. 159–192.

[49] S. I. TAY, T. LEE, N. HAMID e A. N. A. AHMAD, "An overview of industry 4.0: Definition, components, and government initiatives," *Journal of Advanced Research in Dynamical and Control Systems*, v. 10, n. 14, pp. 1379–1387, 2018.

[50] W. ZHOU e S. PIRAMUTHU, "Security/privacy of wearable fitness tracking IoT devices," em *2014 9th Iberian Conference on Information Systems and Technologies (CISTI)*, IEEE, 2014, pp. 1–5.

[51] D. PATTERSON e A. WATERMAN, *The RISC-V Reader: an open architecture Atlas.* Strawberry Canyon, 2017.