Detecção estática e dinâmica de *malwares* usando redes neurais sem peso

Luiz C. S. Ramos[1], **Leopoldo A. D. Lusquino Filho**[1], **Felipe M. G. França**[1], **Priscila M. V. Lima**[1]

[1]Programa de Engenharia de Sistemas e Computação (PESC/COPPE)
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

{sampaio,lusquino,felipe,priscilamvl}@cos.ufrj.br

Resumo. *A preocupação com a segurança e a integridade dos dados em sistemas de computação, incluindo áreas importantes como Internet das Coisas e Indústria 4.0, estão crescendo dramaticamente. Dessa forma, a existência de um malware pode ameaçar o bom funcionamento de sistemas inteiros, trazendo consequências irreversíveis. Este trabalho visa utilizar redes neurais sem peso para detecção estática e dinâmica de malwares. Na utilização de técnicas estáticas baseadas na imagem 2D do arquivo binário e de técnicas dinâmicas baseadas em API Calls, a rede WiSARD mostrou resultados próximos de técnicas do estado da arte utilizando redes neurais com peso, porém com tempos de treinamento e classificação uma ordem de grandeza menor.*

Abstract. *Concerns with security and integrity of data in computer systems, including important areas such as Internet of Things and Industry 4.0, are dramatically increasing. Therefore, the existence of malware can threaten the smooth functioning of whole systems, bringing irreversible consequences. This work aims to use weightless neural networks for static and dynamic detection of malwares. Using static techniques based on the 2D image of binary files and dynamic techniques based on API Calls, the WiSARD network showed results close to state-of-the-art techniques using convolutional neural networks, but shorter training and classification times one lower order of magnitude. The method presented shows the WiSARD as a faster alternative in detecting malware.*

1. Introdução

Malwares, ou *softwares* maliciosos, são programas que buscam perturbar as operações de um sistema computadorizado por meio de acesso não autorizado para conseguir informações sensíveis, ameaçando usuários [Aycock 2006]. Esses programas podem comprometer a integridade das informações nesse sistema, de forma que a detecção de *malwares*, ou seja, técnicas para identificar se um programa é malicioso ou não, é alvo de pesquisadores, já que novos *malwares* constantemente são lançados e as técnicas devem acompanhar tais mudanças [Bazrafshan et al. 2013].

Com a criação constante de novos programas maliciosos, além dos métodos tradicionais de detecção de *malwares*, como *signature-based methods* e *behavior-based methods*, métodos baseados em heurística, utilizando técnicas de aprendizado de máquinas, como Naïve Bayes [Schultz et al. 2001] começaram a ser explorados

para classificação como uma tentativa de contornar desvantagens nos dois primeiros métodos. *Signature-based methods* têm como principal limitação a impossibilidade de detectar programas novos e requerem bastante tempo para extrair a assinatura [Bazrafshan et al. 2013]. Por outro lado, *behavior-based methods* não apresentam taxas promissoras de falsos positivos e levam bastante tempo para serem executados [Elhadi et al. 2012].

Os métodos também podem ser classificados entre estáticos e dinâmicos. A análise estática de *malwares* busca descobrir padrões por meio dos *opcodes* e do arquivo binário do programa, revelando distribuição de frequências e gráficos de controle de fluxo de forma a criar assinaturas que possam identificar cada programa [Nataraj et al. 2011a]. Em contrapartida, na análise dinâmica, o programa é executado em um ambiente controlado, como uma máquina virtual, de forma a captar padrões de comportamento. Enquanto a análise estática mostra resultados mais rápidos, ela apresenta dificuldade em identificar *malwares* polimórficos e metamórficos. Por outro lado, a análise dinâmica, apesar de identificar essas características nos programas, é mais custosa em tempo e recursos [Bozkir et al. 2019]. Trabalhos recentes utilizando redes neurais tradicionais, por exemplo, mostram tempos médios na ordem de 8,4 milissegundos para classificação [Bozkir et al. 2019] [Rhode et al. 2018].

Este trabalho apresenta a utilização de redes neurais sem peso, adotando a rede neural baseada em memória RAM **WiSARD** (Wilkie, Stonham and Aleksander's Recognition Device) [Aleksander et al. 1984] como uma alternativa para a detecção de programas maliciosos. Com a WiSARD, explora-se mais uma técnica de aprendizado de máquinas nessa tarefa, tanto para análise estática como para análise dinâmica.

Como redes neurais sem peso apresentam, relativamente, baixos tempos de treinamento e classificação e são baseadas em memórias RAM, a simplicidade do modelo pode ser útil na utilização não só em grandes sistemas computadorizados, mas também em sistemas em IoT e em sistemas ciber-físicos. Tais sistemas muitas vezes necessitam de baixo custo energético e possuem limitação de memória, e redes neurais sem peso também contam com implementações em *hardware* [Aleksander et al. 1984]. Com isso, a proposta do trabalho mostra uma alternativa aos métodos de redes tradicionais e aprendizado de máquinas como uma forma de reduzir tempo e recursos utilizados sem perda significativa nas métricas convencionais.

O trabalho está organizado da seguinte forma. Na Seção 2, explora-se as principais técnicas utilizadas na detecção de softwares maliciosos. O método proposto está descrito na Seção 3 enquanto os resultados experimentais estão mostrados na Seção 4. Finalmente, a Seção 5 conclui o trabalho.

2. Trabalhos relacionados

Diversos métodos de detecção e classificação de *malwares* vêm sendo estudados, sejam estáticos ou dinâmicos, cada um com seus prós e contras. Nos últimos anos, percebe-se um foco maior em técnicas de aprendizado de máquinas como abordagem para o problema, já que percebeu-se a necessidade de uma rápida resposta aos novos *malwares* que surgem a cada dia. Como essa abordagem depende de *features* previamente escolhidas para treinamento, é possível que o agressor, conhecendo as *features* utilizadas, possa evadir da detecção [Vinayakumar et al. 2019]. Porém, a utilização de diversas técnicas com

diferentes *features* dificulta a ação do agressor.

Alguns métodos de análise estática consistem de utilizar técnicas de processamento de imagens, já que é comum a geração de novos *malwares* partindo-se de programas já existentes. Dessa forma, os arquivos binários dos programas podem ser tratados como imagens por conversão direta e analisado, como mostrado em [Nataraj 2015]. Como aplicação dessas técnicas, [Nataraj et al. 2013] mostra o SARVAM, um sistema de busca de *malwares* baseado na imagem gerada pelo arquivo e na técnica de *Nearest Neighbors* para estimar a similaridade de novos programas com *malwares* usados no treinamento.

Além disso, [Nataraj et al. 2011b] mostrou a análise de textura das imagens geradas como forma de classificação de *malwares*. Junto a técnicas de processamento de imagem, [Farrokhmanesh and Hamzeh 2016] utilizou a abordagem de processamento de áudio, representando os programas como sinais de áudio e aplicando técnicas de obtenção de informação de músicas para encontrar padrões. Depois disso, utilizou-se um modelo de aprendizado de máquinas com as *features* extraídas para classificar novos programas. Finalmente, [Garcia and Muga II 2016] mostra a utilização de *random forest* para classificação das imagens geradas.

As técnicas de análise dinâmica, com o estudo das chamadas feitas pelo programa ao sistema operacional, as *application programming interface calls* (API calls), mostram-se mais robustas a métodos de ofuscação do que a análise estática [Vinayakumar et al. 2019]. Com essa abordagem, encontra-se mais técnicas baseadas em aprendizado de máquinas; [Huang and Stokes 2016] propôs a utilização de redes neurais profundas utilizando API Calls como entrada. O trabalho mostra uma acurácia de mais de 99% utilizando uma rede com uma camada oculta, com tempo de treinamento de 6 horas e 58 minutos para 4,5 milhões de arquivos e com tempo de classificação de 1 hora e 34 minutos para 2 milhões de arquivos. Tal abordagem mostra tempos da ordem de 3 milissegundos para classificação de cada amostra.

Em [Pascanu et al. 2015] e [Kolosnjaji et al. 2016], explora-se a utilização de redes neurais convolucionais e redes neurais recorrentes junto com API Calls para classificação. Ambos utilizam na ordem de centenas ou milhares de neurônios na rede e não aprofundam o estudo com relação ao tempo de execução da classificação. Finalmente, [Rhode et al. 2018] propôs um método baseado em redes neurais recorrentes que apresentou uma acurácia de 91% em 4 segundos de execução, mostrando uma média da ordem de 8 milissegundos para a classificação de cada amostra.

3. Proposta

A proposta do trabalho consiste em utilizar a rede neural baseada em memória RAM **WiSARD** (Wilkie, Stonham and Aleksander's Recognition Device) [Aleksander et al. 1984] para detecção estática e dinâmica de programas maliciosos. Na seção 3.1, explicita-se a arquitetura da WiSARD, mostrando os detalhes do treinamento e classificação. Nas seções 3.2 e 3.3, mostra-se os métodos propostos para as análises estática e dinâmica, respectivamente.

3.1. WiSARD

Conhecidas originalmente como *classificadores de ênuplas* e desenvolvidas por [Bledsoe and Browning 1959], as redes neurais sem peso também são inspiradas no sis-

tema nervoso humano, porém com uma abordagem diferente das redes neurais tradicionais. Nelas, prioriza-se a emulação da topologia das conexões entre dendritos e axônios, ou seja, a árvore dendrítica [Grieco et al. 2009]. A informação aprendida é armazenada em memórias RAM, que funcionam como os neurônios artificiais.

A WiSARD é conhecida como o modelo de rede neural sem peso mais representativo [Grieco et al. 2010]. Nela, separa-se uma entrada binária de tamanho Mxn em M tuplas de n bits, e cada uma dessas tuplas será associada a uma memória RAM. Com os dados de treinamento, cada tupla será utilizada para endereçar um endereço nessas memórias, e o treinamento é feito trocando o bit da posição de memória, com valor inicial 0, para 1, como na Figura 1. Com isso, cada memória é capaz de reconhecer um padrão previamente apresentado, mas possui dificuldade em generalizar com padrões semelhantes. Vale ressaltar que cada tupla só representa parte da entrada, de forma que cada padrão é dividido em M partes.

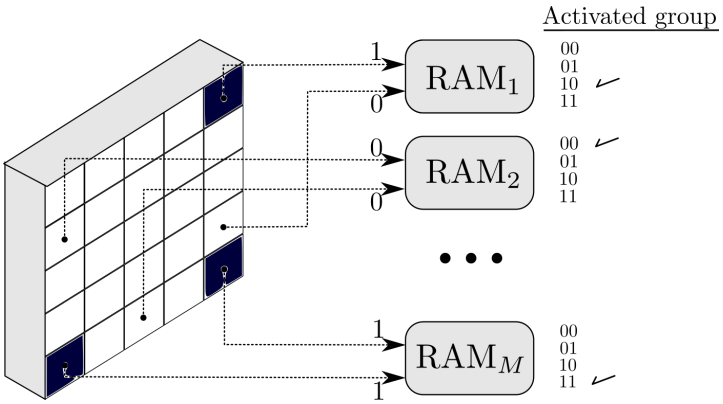

Figura 1. Ilustração do treinamento para M tuplas.

Dentro da rede, o aprendizado de cada classe é armazenado no que é chamado de **discriminador**, compostos por M memórias cada, que são responsáveis pela generalização e tolerância a ruídos [França et al. 2009]. Para a classificação, a entrada é utilizada para endereçar todas as memórias em todos os discriminadores, e o valor na posição de memória é lido. Cada discriminador retorna a soma das posições endereçadas, e o discriminador que obtiver maior pontuação é escolhido como a saída predita, como ilustrado na Figura 2.

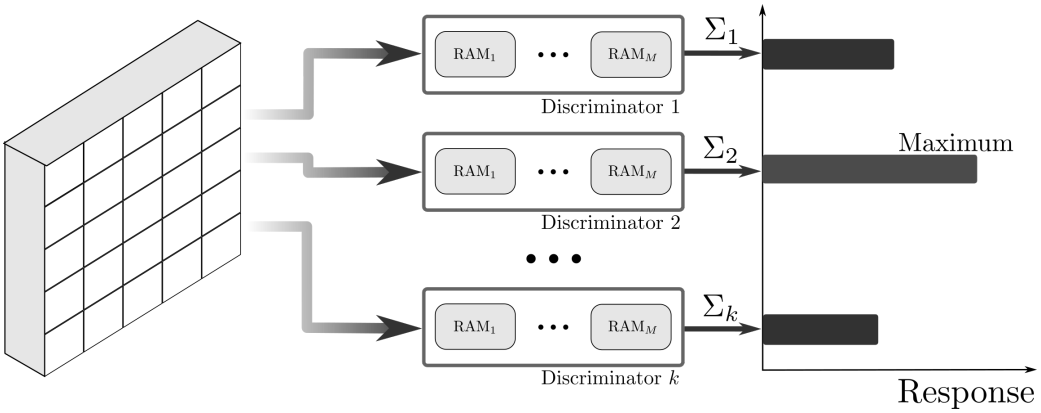

Figura 2. Escolha do discriminador com maior soma. Nesse exemplo, o discriminador 2 obteve maior soma.

Uma desvantagem é a possibilidade de que, para uma certa entrada durante a classificação, haja empate entre os discriminadores; nesse caso, a rede escolhe aleatoriamente uma classe entre aquelas representadas pelos discriminadores com maior pontuação. A técnica de *bleaching* foi elaborada de forma a minimizar esse problema, e consiste em armazenar nas posições de memória um contador de acessos. Dessa forma, durante a etapa de classificação, a pontuação de cada discriminador consistirá na quantidade de posições de memória acessadas, cujo contador possui valor igual ou superior a um limiar, que é inicializado com valor zero. Na medida que houverem empates entre discriminadores, este limiar é incrementado e novas pontuações são calculadas, até que haja um discriminador vencedor. Caso o limiar se torne superior a cardinalidade dos dados de treinamento ou ao maior contador pertencente as posições de memória acessadas, a rede irá sortear uma classe. Uma desvantagem dessa técnica é o aumento no tempo de classificação [de Souza 2011].

3.2. Análise estática

A abordagem utilizada para a análise estática está ilustrada na Figura 3. A análise consiste de utilizar técnicas de visão computacional na detecção de *malwares* após conversão direta da sequência de bytes do arquivo binário original para imagens em escala de cinza de tamanho 224x224 [Bozkir et al. 2019]. A viabilidade dessa técnica baseia-se no fato que programas maliciosos, em geral, apresentam pequenas diferenças em relação à família a qual pertencem (vírus, trojans, adwares etc) [Nataraj et al. 2011a]. As imagens são, então, redimensionadas para tamanho 128x128, vetorizadas para tamanho 1x16384, e é feita uma binarização; entradas do vetor com valores entre 0 e 127 recebem valor 0, e entradas do vetor com valores entre 128 e 255 recebem valor 1. O vetor, então, é utilizado como entrada na WiSARD para treinamento. A técnica escolhida baseia-se no fato de que redes neurais sem peso apresentam tempos de treinamento e classificação ordens de grandeza menores do que os de modelos convencionais, como SVM [Cardoso et al. 2016].

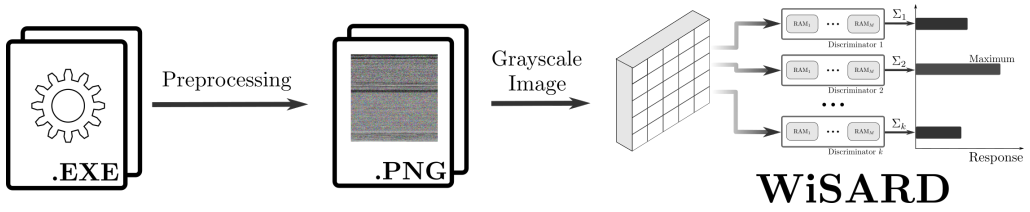

Figura 3. Pré-processamento e treinamento para análise estática.

3.3. Análise dinâmica

A abordagem utilizada para a análise dinâmica está ilustrada na Figura 4. A análise consiste de analisar as chamadas feitas pelo programa ao sistema operacional, chamadas de *application programming interface calls* (API calls) para tentar captar o comportamento de um programa como um *malware* [Bazrafshan et al. 2013]. As API calls podem ser obtidas executando o programa em um ambiente isolado, como Cuckoo Sandbox, um sistema open-source de análise automática de *malwares*, que monitora o comportamento dos processos [Stichting Cuckoo Foundation 2020]. Com os dados coletados, pode-se montar uma tabela das principais API calls. Assim, a entrada da rede neural corresponde às 1000 principais API Calls coletadas em programas benignos e maliciosos e cada entrada possui valor 1 se aquela chamada foi executada no processo daquele programa; a entrada recebe valor 0 caso contrário [Oliveira and Sassi 2019]. A técnica foi escolhida por conta da existência de um banco de dados livre para uso disponível em [Oliveira 2019].

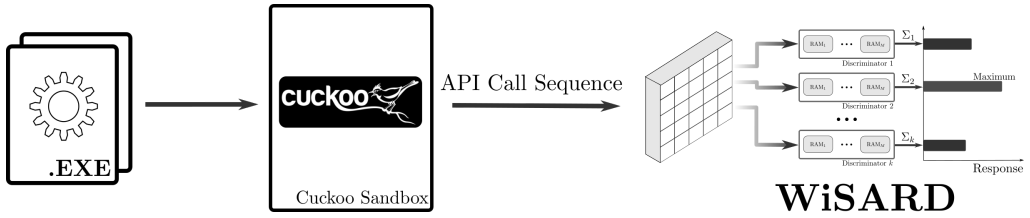

Figura 4. Coleta de dados e treinamento para análise dinâmica.

4. Experimentos

Nas seções 4.1 e 4.2, são expostas as condições dos experimentos e as métricas utilizadas nos resultados, respectivamente. A seção 4.3 detalha cada experimento feito, bem como detalhes da implementação. Finalmente, a seção 4.4 mostra os resultados obtidos.

4.1. Cenário de experimentos

Para a análise estática, os dados foram obtidos do conjunto de dados *Malevis* [Bozkir et al. 2019]. Esse conjunto de dados contém imagens com três canais (RGB) obtidas da conversão direta dos arquivos dos programas. Antes do treinamento, as imagens foram convertidas para escala de cinza e redimensionadas. Os programas estão classificados em 26 categorias (25 malignas e 1 benigna), porém para detecção só utilizou-se duas categorias (malígno e benígno). Finalmente, o conjunto de dados foi separado em 70% para treinamento e 30% para validação.

Para a análise dinâmica, os dados foram obtidos do conjunto de dados *Top-1000 PE Imports* [Oliveira 2019]. Esse conjunto de dados contém um arquivo *.csv* com as

API Calls de 47580 programas executados dentro da Cuckoo Sandbox. Como o conjunto de dados é desbalanceado (há mais *malwares* que programas benignos), além da análise feita nos 47500 programas, também selecionou-se aleatoriamente programas benignos de forma a se ter a mesma quantidade das duas classes; o conjunto foi chamado de balanceado. Os dados foram separados em 70% para treinamento e 30% para validação.

4.2. Métricas

Como forma de comparação com outros métodos, utilizou-se métricas baseadas nas seguintes grandezas: número de positivos verdadeiros (TP), que são programas maliciosos classificados corretamente como maliciosos; número de negativos verdadeiros (TN), que são programas benignos classificados corretamente como benignos; número de falsos positivos (FP), que são programas benignos classificados erroneamente como maliciosos; e número de falsos negativos (FN), que são programas malignos classificados erroneamente como begnignos. As métricas usadas foram:

- **Precisão**: $\frac{TP}{TP+FP}$
- **Revocação**: $\frac{TP}{TP+FN}$
- **F1-Score**: média harmônica entre *precisão* e *revocação*.
- **Acurácia**: $\frac{TP+TN}{TP+TN+FP+FN}$
- **Tempo de treinamento**: tempo necessário para treinar a rede.
- **Tempo de classificação**: tempo necessário para classificar novas entradas com a rede treinada.

4.3. Experimentos

Em cada experimento, após separar o conjunto de treinamento e validação, treinou-se as redes utilizando um endereçamento de $n = 20$ bits. O número foi escolhido após busca de $n = 2$ bits a $n = 64$ bits de forma a maximizar a acurácia. A técnica de *bleaching* foi ativada de forma a minimizar o número de empates entre respostas dos discriminadores. Cada experimento foi repetido 10 vezes mudando-se os conjuntos de validação e treinamento. Com isso, foram obtidos os valores para as métricas da seção 4.2. Os experimentos foram feitos utilizando linguagem *Python*, e, para implementação da WiSARD, foi utilizada a biblioteca *wisardpkg* [Filho et al. 2020]. Os experimentos foram feitos em um computador Intel Core i5-4210U, 4 cores, 8 threads, 1,70 GHz, 8 GB RAM e 1 TB HDD.

4.4. Resultados

4.4.1. Análise estática

A tabela 1 mostra os resultados (média e desvio padrão) da rede utilizada na detecção estática, enquanto a tabela 2 mostra os tempos de treinamento e teste. Observa-se que, pelas métricas escolhidas, os valores foram bem próximos da unidade, com desvios padrões menores que 0,1%. Utilizando um endereçamento com 20 bits e considerando que a entrada é uma imagem de dimensões 128x128, cada discriminador contará com aproximadamente 820 RAMs. Embora este enorme tamanho de espaço de memória seja definido, estruturas simples e disponíveis de acesso associativo, como dicionários baseados em tabelas tipo Hash lidam com agilidade a escrita e leitura dos discriminadores das duas classes (software benigno ou malígno).

Tabela 1. Resultados para detecção estática de malwares.

	WiSARD - $n = 20$ bits	
Métrica	Média	Desvio Padrão
F1-Score	0,9818	0,0005
Precisão	0,9658	0,0001
Revocação	0,9984	0,0010
Acurácia	0,9644	0,0010

Além disso, o tempo médio de treinamento e classificação foram menores que os encontrados em [Bozkir et al. 2019]. Vale ressaltar, porém, que a WiSARD proposta executou a tarefa de detecção, enquanto a DenseNet executou a tarefa de classificação.

Tabela 2. Tempos (por amostra), em milissegundos, de processamento para detecção estática de malwares. Em [Bozkir et al. 2019], os tempos são de classificação.

	WiSARD - $n = 20$ bits		Malevis DenseNet121	Malevis DenseNet169
Tempo (ms)	Média	Desvio Padrão	Média	Média
Treinamento	2	0,17	19	26
Classificação	3	0,25	5,8	8,4

Para comparação com [Bozkir et al. 2019], também testou-se a WiSARD para classificação de 26 classes de programas. A Figura 5 mostra a matriz de confusão para a classificação com $n = 20$ bits e a Tabela 3 mostra os resultados para as métricas propostas. Observa-se que, para a tarefa de classificação, a WiSARD obteve desempenho menor que as *DenseNet* propostas em [Bozkir et al. 2019].

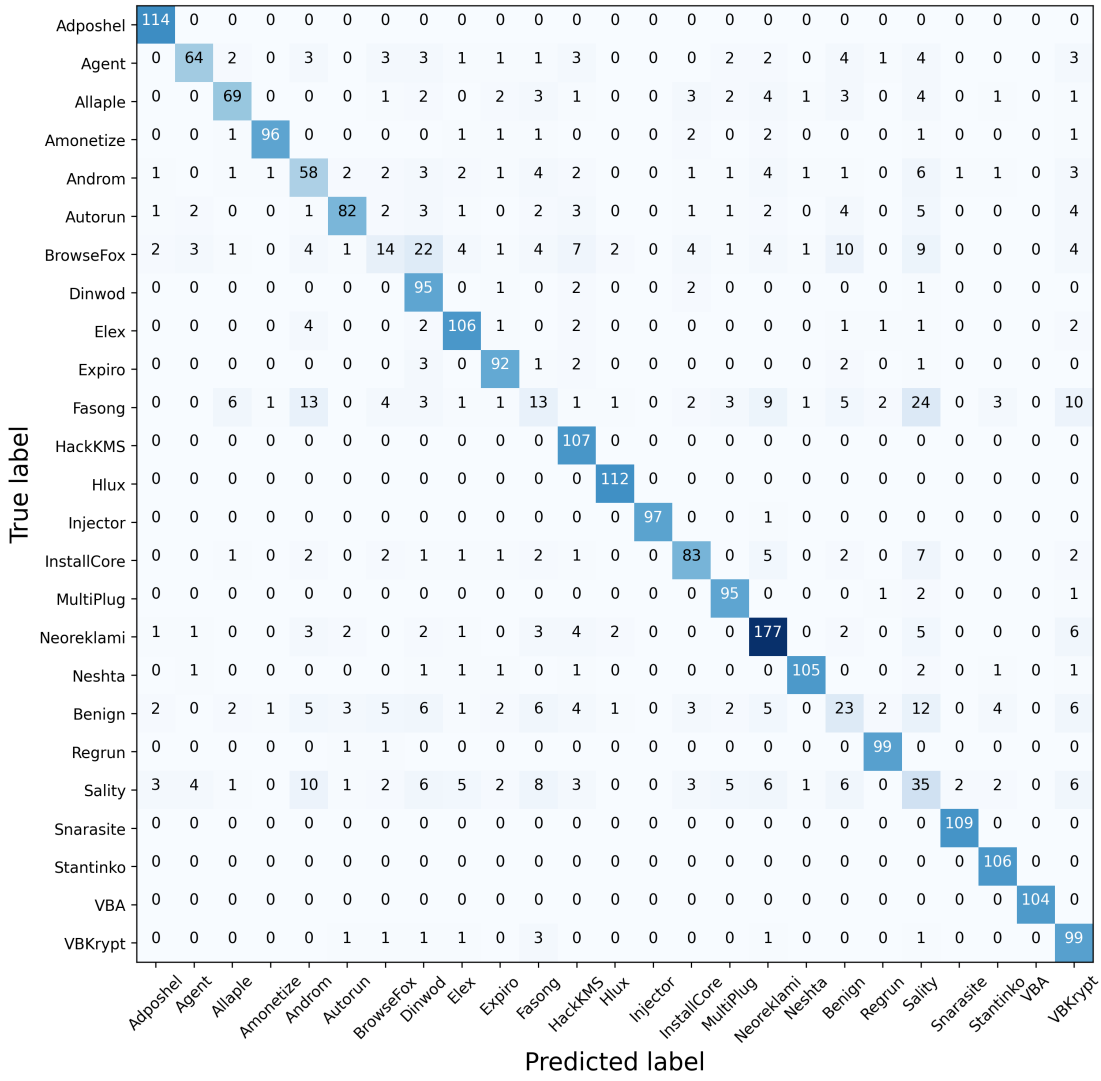

Figura 5. Matriz de confusão para classificação estática de malwares.

Tabela 3. Resultado para classificação estática de malwares [Bozkir et al. 2019].

	WiSARD - $n = 20$ bits		Malevis DenseNet121
Métrica	Média	Desvio Padrão	Média
Acurácia	0,7901	0,0027	0,9748

4.4.2. Análise dinâmica

A tabela 4 mostra os resultados (média e desvio padrão) da rede utilizada na detecção dinâmica, bem como os resultados encontrados para o mesmo conjunto de dados em [Oliveira and Sassi 2019]. Além disso, a tabela 6 mostra os tempos de treinamento e teste. Observa-se que, pelas métricas escolhidas, os valores foram competitivos com os

obtidos por outros métodos, com desvios padrões menores que 0,1%. Utilizando um endereçamento com 20 bits e considerando que a entrada é um vetor contendo 1000 posições binárias, de acordo com a presença ou não das API Calls, cada discriminador contará com aproximadamente 50 RAMs. Embora este enorme tamanho de espaço de memória seja definido, estruturas simples e disponíveis de acesso associativo, como dicionários baseados em tabelas tipo Hash lidam com agilidade a escrita e leitura dos discriminadores (software benigno ou maligno).

Tabela 4. Resultados para o conjunto de dados balanceado [Oliveira and Sassi 2019].

	WiSARD - $n = 20$ bits		2-layer DGCNN	LSTM
Métrica	Média	Desvio Padrão	Média	Média
F1-Score	0,9095	0,0041	0,9201	0,8424
Precisão	0,9348	0,0057	0,9216	0,8317
Revocação	0,8857	0,0074	0,9186	0,8534
Acurácia	0,9136	0,0038	0,9244	0,8488

Também como forma de comparação, a tabela 5 mostra os resultados obtidos para o conjunto desbalanceado.

Tabela 5. Resultados para o conjunto de dados desbalanceado [Oliveira and Sassi 2019].

	WiSARD - $n = 20$ bits		2-layer DGCNN	LSTM
Métrica	Média	Desvio Padrão	Média	Média
F1-Score	0,9737	0,0005	0,9942	0,9953

Tabela 6. Tempos (por amostra), em milissegundos, de processamento para o conjunto de dados balanceado [Rhode et al. 2018].

	WiSARD - $n = 20$ bits		RNN
Tempo (ms)	Média	Desvio Padrão	Média
Treinamento	0,269	0,0048	38,0
Classificação	0,312	0,0110	8,0

5. Conclusão

O trabalho estudou a viabilidade da utilização de redes neurais sem peso (em particular, a WiSARD) na detecção estática e dinâmica de softwares maliciosos. Para tal, utilizou-se conjuntos de dados de imagens convertidas dos programas e de API Calls para análises estática e dinâmica, respectivamente. Experimentos mostraram uma acurácia de 96% na análise estática e 97% na análise dinâmica. Além disso, o tempo de classificação de cada

amostra, na média, foi menor que os tempos encontrados por outros métodos. Com isso, a WiSARD mostrou-se como uma alternativa na detecção de *malwares* possuindo arquitetura mais simples, já que utiliza um parâmetro (tamanho da tupla), enquanto métodos consagrados utilizando redes neurais com peso podem utilizar uma quantidade de parâmetros na ordem de milhares ou milhões. Na análise dinâmica, por exemplo, conseguiu-se tempos de treinamento e classificação da ordem de 0,3 milissegundos, uma ordem de grandeza menor que outros métodos.

Trabalhos futuros incluem a aplicação da WiSARD na classificação de *malwares*, que inclui não somente detectá-los, mas também em inferir suas categorias de acordo com o propósito e características (vírus, *trojans*, *adwares*, etc.) e explorar como modificações na arquitetura influenciam no resultado, como, por exemplo, a utilização da ClusWiSARD [Cardoso et al. 2016], uma expansão da WiSARD que utiliza mais de um discriminador para cada classe, criando dinamicamente novos discriminadores quando necessita treinar um exemplo que não seja suficientemente semelhante aos já aprendidos pelos discriminadores existentes.

Referências

Aleksander, I., Thomas, W., and Bowden, P. (1984). Wisard·a radical step forward in image recognition. Sensor Review, 4(3):120–124.

Aycock, J. (2006). Computer Viruses and Malware. Advances in Information Security. Springer US.

Bazrafshan, Z., Hashemi, H., Fard, S. M. H., and Hamzeh, A. (2013). A survey on heuristic malware detection techniques. In The 5th Conference on Information and Knowledge Technology, page 113–120.

Bledsoe, W. W. and Browning, I. (1959). Pattern recognition and reading by machine. In IRE-AIEE-ACM Computer Conference, IRE-AIEE-ACM '59 (Eastern), page 225–232. Association for Computing Machinery.

Bozkir, A. S., Cankaya, A. O., and Aydos, M. (2019). Utilization and comparision of convolutional neural networks in malware recognition. In 27th Signal Processing and Communications Applications Conference (SIU), page 1–4.

Cardoso, D. O., Carvalho, D. S., Alves, D. S. F., Souza, D. F. P., Carneiro, H. C. C., Pedreira, C. E., Lima, P. M. V., and França, F. M. G. (2016). Financial credit analysis via a clustering weightless neural classifier. Neurocomputing, 183:70–78.

de Souza, C. R. (2011). Redes Neurais Sem-Peso Aplicadas na Categorização de Subtipos do HIV-1. Master's thesis, Dissertação de Mestrado - COPPE/UFRJ, Brasil.

Elhadi, A. A. E., Maarof, M. A., and Osman, A. H. (2012). Malware detection based on hybrid signature behaviour application programming interface call graph. American Journal of Applied Sciences, 9(33):283–288.

Farrokhmanesh, M. and Hamzeh, A. (2016). A novel method for malware detection using audio signal processing techniques. In 2016 Artificial Intelligence and Robotics (IRANOPEN), page 85–91.

Filho, A. S. L., Guarisa, G. P., Filho, L. A. D. L., Oliveira, L. F. R., Franca, F. M. G., and Lima, P. M. V. (2020). wisardpkg – a library for wisard-based models.

França, H. L., SILVA, J., Lengerke, O., França, F. M., and Dutra, M. S. (2009). Um sistema de visão artificial para o controle de perseguição de movimento por uma plataforma Stewart. In 2º Congreso Internacional de Ingeniería Mecatrónica, UNAB, Bucaramanga, Colômbia.

Garcia, F. C. C. and Muga II, F. P. (2016). Random forest for malware classification. arXiv:1609.07770 [cs]. arXiv: 1609.07770.

Grieco, B. P., Lima, P. M., De Gregorio, M., and França, F. M. (2009). Extracting fuzzy rules from "mental" images generated by modified wisard perceptrons. In 17th European Symposium on Artificial Neural Networks, pages 313–318.

Grieco, B. P. A., Lima, P. M. V., De Gregorio, M., and França, F. M. G. (2010). Producing pattern examples from "mental" images. Neurocomputing, 73(7):1057–1064.

Huang, W. and Stokes, J. W. (2016). Mtnet: A multi-task neural network for dynamic malware classification. In Caballero, J., Zurutuza, U., and Rodríguez, R. J., editors, Detection of Intrusions and Malware, and Vulnerability Assessment, Lecture Notes in Computer Science, page 399–418. Springer International Publishing.

Kolosnjaji, B., Zarras, A., Webster, G., and Eckert, C. (2016). Deep learning for classification of malware system call sequences. In Kang, B. H. and Bai, Q., editors, AI 2016: Advances in Artificial Intelligence, Lecture Notes in Computer Science, page 137–149. Springer International Publishing.

Nataraj, L. (2015). A signal processing approach to malware analysis. University of California, Santa Barbara.

Nataraj, L., Karthikeyan, S., Jacob, G., and Manjunath, B. S. (2011a). Malware images: visualization and automatic classification. In Proceedings of the 8th International Symposium on Visualization for Cyber Security, VizSec '11, page 1–7. Association for Computing Machinery.

Nataraj, L., Kirat, D., Manjunath, B., and Vigna, G. (2013). Sarvam: Search and retrieval of malware. In Proceedings of the Annual Computer Security Conference (ACSAC) Worshop on Next Generation Malware Attacks and Defense (NGMAD).

Nataraj, L., Yegneswaran, V., Porras, P., and Zhang, J. (2011b). A comparative assessment of malware classification using binary texture analysis and dynamic analysis. In Proceedings of the 4th ACM workshop on Security and artificial intelligence, AISec '11, page 21–30. Association for Computing Machinery.

Oliveira, A. (2019). Malware analysis datasets: Top-1000 PE Imports. IEEE-DataPort: https://ieee-dataport.org/open-access/malware-analysis-datasets-top-1000-pe-imports, Acesso em Julho de 2020.

Oliveira, A. and Sassi, R. J. (2019). Behavioral malware detection using deep graph convolutional neural networks. TechRxiv. https://www.techrxiv.org/articles/preprint/Behavioral_Malware_Detection_Using_Deep_Graph_Convolutional_Neural_Networks/10043099.

Pascanu, R., Stokes, J. W., Sanossian, H., Marinescu, M., and Thomas, A. (2015). Malware classification with recurrent networks. In 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), page 1916–1920.

Rhode, M., Burnap, P., and Jones, K. (2018). Early-stage malware prediction using recurrent neural networks. Computers & Security, 77:578–594.

Schultz, M., Eskin, E., Zadok, F., and Stolfo, S. (2001). Data mining methods for detection of new malicious executables. In Proceedings 2001 IEEE Symposium on Security and Privacy. S P 2001, page 38–49.

Stichting Cuckoo Foundation (accessed July, 2020). Cuckoo Sandbox (Automated Malware Analysis). `https://cuckoosandbox.org/`.

Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., and Venkatraman, S. (2019). Robust intelligent malware detection using deep learning. IEEE Access, 7:46717–46738.