ROBUST: Deep Learning for Malware Detection under Changing Environments

Adel Abusitta[1], Talal Halabi[2] and Omar Abdel Wahab[3]

[1]McGill University, 845 Sherbrooke St W, Montreal, Quebec H3A 0G4, Canada
[2]The University of Winnipeg, 515 Portage Ave, Winnipeg, MB R3B 2E9, Canada
[3]Université du Québec en Outaouais, 283 Alexandre-Taché Blvd, Gatineau, Quebec J8X 3X7, Canada

Abstract

Malware is one of the most notorious security threats in cyberspace today, and Artificial Intelligence is playing a major role in pushing its design to the next level. Although several malware detection systems (MDS) based on machine learning have recently been designed to cope with the updated malware nature, most of these systems are still not mature enough to capture the modified malware using anti-analysis techniques. Moreover, the patterns extracted and adopted by a machine learning-based MDS are limited to the specific environment and infrastructure. Hence, they are not fully effective against anti-analysis techniques (e.g., obfuscation techniques) and/or when tested in different execution environments (e.g., different machines). Motivated by this fact, this paper presents a new framework for detecting malware in non-stationary environments by leveraging deep learning techniques to extract useful features that are robust against changing environments. More specifically, the framework is based on a special version of an Autoencoder, called a Denoising Autoencoder, which is adopted as a building block in an adaptable deep neural network. The experimental results using a real-world dataset show that the framework improves the detection accuracy compared to existing methods.

Keywords

Malware, malware detection, deep learning, changing environment

1. Introduction

For the last decade, Artificial Intelligence (AI) has been aggressively advancing many scientific, technological, and social fields and becoming gradually rooted in our day-to-day life. Nonetheless, it has also been extensively exploited by society's bad actors to create powerful cyberweapons and launch destructive cyberattacks. For instance, AI-enabled malware leverages machine and deep learning techniques to explore hidden and zero-day vulnerabilities in computer systems and networks, remain hidden from deployed security protocols and solutions, and improve social engineering practices. This emerging facet of malicious AI can easily trick conventional intrusion detection systems. Hence, without radically adapting these systems and substantially

AIofAI'21: 1st Workshop on Adverse Impacts and Collateral Effects of Artificial Intelligence Technologies, Montreal, CA

adel.abusitta@gmail.com (A. Abusitta); t.halabi@uwinnipeg.ca (T. Halabi); omar.abdulwahab@uqo.ca (O. A. Wahab)
https://ca.linkedin.com/in/adel-abusitta-4606968a (A. Abusitta); https://sites.google.com/view/talalhalabi/ (T. Halabi); http://w4.uqo.ca/dii/dyn/profs/omar.abdulwahab.php (O. A. Wahab)
0000-0003-0860-3220 (A. Abusitta); 0000-0002-1922-5803 (T. Halabi); 0000-0002-3991-4673 (O. A. Wahab)

increasing their intelligence [1], companies and organizations will not be able to face the future generation of stealthy cybercriminals, especially that crafting malware using AI techniques will become increasingly accessible and easy (e.g., using cloud-based Machine Learning-as-a-Service (MLaaS) platforms [2]).

Malware is a software programmed to harm organizations, computer systems, and users [3]. It can corrupt users' applications and operating systems, damage networks and internet connections, corrupt and steal sensitive data (e.g., user's password), and/or encrypt users and organizations' documents. Moreover, it can be used to launch large-scale attacks such as compromising hosts, servers and clouds, bringing down sensitive government servers, and damaging critical infrastructures that are vital to many sectors including Information Technology, health, transportation, power, and Education [4, 5, 6]. Nonetheless, the malware of the future will be significantly empowered by AI, which greatly adds to its fundamental characteristics as illustrated in Figure 1. It can adapt to its environment by changing its behavior accordingly to thwart detection and increase its impact. Moreover, it can perform deep targeting by stealthily learning the behavior of its target system and determining the optimal time of attack that reduces detection chances.

Malware detection systems (MDS) can be divided into two types: signature-based and machine learning-based approaches [3]. The former type compares suspicious malware against known malware signatures. On the other hand, the machine learning-based MDS enables the detection of unseen malware using machine learning techniques (e.g., SVM, NN). These techniques allow the system to raise security alarms when unexpected behaviors are observed. However, a large-enough dataset of malware and 'good' software is required in order to guide or train the machine learning model on how to distinguish malware among 'good' software. The learned model can then be used to detect whether a given activity or software component is malicious or not.

The last few years have shown substantial progress in the design of malware detection systems that do not only detect malware but also deal with their evolution [3]. Nevertheless, the detection of malware is becoming increasingly challenging. One of the reasons is that malware designers have become experts at exploiting and building advanced anti-analysis techniques (e.g., obfuscation methods) that hide the malicious behavior of malware. Moreover, current and emerging computer systems are becoming highly distributed, heterogeneous, and dynamic, which creates many opportunities for the malware to take various forms depending on which machine it runs on. Although machine learning-based MDS are efficient in detecting new threats, they are still not able to detect malware under changing environments. The malware-related patterns extracted and adopted by the existing machine learning methods are limited to the specific environment and infrastructure and do not necessarily hold after applying complex obfuscating techniques on malware and/or run the malware on different machines. These shortcomings may reduce malware detection chances.

To address the above-mentioned challenges, we propose a robust MDS framework that integrates a deep learning approach to enhance the detection accuracy under changing environments. The proposed model learns a 'good' representation of inputs (Malware), which is robust against anti-analysis techniques and changing environments. The learned representation can then be used to train a classifier on how to detect malware. More particularly, our framework is based on Stacked Denoising Autoencoders (SDAE), where a Denoising Autoencoder (DA) is adopted as a building block to train the deep neural network [7] [8]. The DA allows the MDS to learn how to reconstruct original malware after being corrupted. This, in turn, enables the MDS to learn robust

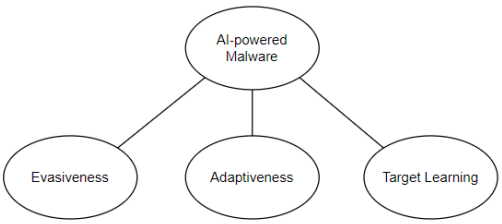


Figure 1: Main characteristics of AI-enabled malware.

and useful features that enhance its accuracy. Our contributions are summarized as follows:

- Proposing a robust malware detection system framework that enables the detection of complex malware under changing environments.
- Designing an efficient method to extract robust and useful features that lead to a better MDS accuracy.
- Studying the effectiveness of the proposed framework using a real-world dataset and comparing the proposed algorithm with the state-of-the-art methods.

The remainder of the paper is structured as follows. Section 2 discusses related work to intelligent malware detection. Section 3 describes the proposed framework. Section 4 describes the evaluation settings and analyses the obtained results. Finally, Section 5 concludes the paper and recommends some future works.

2. Related work

The section discusses machine learning-based MDS. These models can be classified as traditional machine learning models or deep learning models. Existing machine learning-based models can also be categorized according to the learning style used as follows: supervised, unsupervised or semi-supervised. In a supervised machine learning-based MDS [9, 10], the classification technique usually learns on a labeled dataset, which allows the classifier to calculate its accuracy on a training dataset. The unsupervised machine learning-based MDS [11, 12] provides unlabeled data which the machine learning algorithm tries to understand by extracting features without guidance. Finally, semi-supervised machine learning-based MDS [13, 14] use both types of data (labeled and unlabeled) [3].

The most popular traditional machine learning-based methods in the literature are Random Forest [15, 16], Hidden Markov Models [17, 18], Support Vector Machine [19, 20], Naive Bayes classifier [21, 20], decision tree [21, 11], rule-based classifier [20], K-nearest neighbors [10, 21], Bayesian Network [22], and Neural Network [23, 24]. On the other hand, deep learning methods for malware detection enables the abstraction and extraction of features automatically, which allows to build reliable malware detection systems. To this end, several layers of abstraction can be used to learn the 'good' representation of the malware [8].

Saxe and Berlin [24] designed a deep neural network for malware detection. The proposed method is based on a multi-layer perceptron (MLP), which is a class of feed-forward neural

networks. The activation functions that they used in the network are parametric rectified linear units (PReLU), which enable the deep neural network to improve the detection accuracy compared to other activation functions. The authors also adopted a Bayesian model to determine the probability that a given binary is malware. Similarly, Dahl et al. [23] used a deep learning approach to improve the malware detection accuracy. In particular, they used a restricted Boltzmann machine (RBM) as a building block for their deep neural network. The use of RBM allows to pre-train the model and obtain efficient initial weights to be used while applying the back-propagation training algorithm.

Huang and Stokes [25] designed an artificial neural network to achieve multi-task training. The first task is used to find whether an unknown binary is benign or malicious. The second task is used to predict the malware family. Among other approaches, the authors adopted an efficient projection layer, which is used to reduce and normalize the deep network's dimension. They used four layers (hidden layers) and applied the RELU activation function.

Kolosnjaji et al. [26] proposed a convolutional neural network (CNN) integrated with Long Short-Term Memory (LSTM) networks to find the binary and malware families. To this end, they adopted the sequence of extracted system calls. The convolution layers in the proposed framework are used to capture the correlation between the malware's dynamic features. The CNN's output layer is then used to train the LSTM layer in order to model the dependencies of system calls.

Recently, Abusitta et al. [27] proposed a deep learning method for a proactive multi-cloud cooperative intrusion detection system (IDS). The proposed framework is based on a Denoising Autoencoder (DA), which is adopted as a building block to train the deep network. The DA enhances the ability of one IDS in learning how to reconstruct the feedback of other IDSs. As a result, the approach enables the IDSs to proactively decide about suspicious behaviour under incomplete information about the feedback. Similarly, Abusitta et al. [28] proposed a proactive cooperative MDS. They also used the DA to reconstruct all MDSs' feedback from incomplete feedback. This enabled them to enhance the detection accuracy in real-time [28].

The main limitation of the above-mentioned methods is that they assume that the security patterns extracted using machine learning methods are limited to the specific environment. This makes the detection method unable to detect malware when applying anti-analysis techniques such as obfuscation techniques. This paper proposes a new method for detecting malware under changing environments by adopting stacked denoising autoencoders to extract useful features that are robust against noisy inputs.

3. ROBUST: The Proposed Framework

This section first presents the basic concepts of vanilla autoencoders then describes our proposed framework.

3.1. Vanilla Autoencoders

An autoencoder is a method used to learn the representation of data using unsupervised learning [29]. It can be adopted as a pre-training process in a deep neural network in order to get better

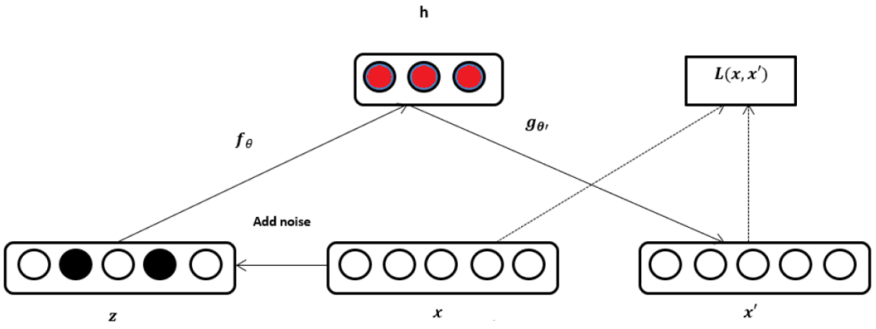


Figure 2: The proposed malware-related denoising autoencoder.

initial weights rather than random values [30]. Many works have shown that weight initialization using an autoencoder can improve the performance of deep neural networks [30].

In fact, an autoencoder can be considered as a building block for deep networks [30]. Specifically, it takes an input vector (malware data) $x \in [0,1]^v$, where v is the dimension of the vector, and maps it to a higher level representation (hidden representation) $h \in [0,1]^{v'}$ as follows:

Algorithm 1: ROBUST: Training malware-related Denoising Autoencoder

procedure TRAINDA(x, lr, ep, bt, θ)

- x= $[x_1, x_2, ...x_n]$: Input malware and benign data
- lr: learning rate
- ep: epoches
- bt: batches
- θ=$\{W, b, b_h\}$

for *i from 0 to ep* **do**

  for *j from 0 to bt* **do**

    - $z = add - noise(x, l)$: l is corruption level
    - $h = SigFun(z * W + b)$
    - $x' = SigFun(h * W^T + b_h)$
    - $Loss(x, x')$= $-\sum_{i=1}^{d}[x_i log x'_i + (1 - x_i) log(1 - x'_i)]$
    - $CostValue = mean(Loss(x, x'))$
    - g=determining the gradients with respect to θ

  for θ_i,$g_i \in$ (θ,g) **do**

    - θ_i = θ_i - $lr * g_i$

  end

  end

  end

end procedure

$$
\begin{aligned}
h &= f_\theta(x) \\
f_\theta(x) &= SigFun(W * x + b)
\end{aligned}
\tag{1}
$$

$SigFun$ is the sigmoid function, and $\theta = \{W, b\}$, where W is the neural network's weight and b is a bias [30].

The resulting hidden layer h can be reconstructed to the layer x' using a decoding function as follows:

$$
\begin{aligned}
x' &= g_{\theta'}(h) \\
g_{\theta'}(h) &= SigFun(W' * h + b')
\end{aligned}
\tag{2}
$$

$\theta' = \{W', b'\}$, where W' and b' are the neural network's weight and a bias, respectively. The aim of the autoencoder model is to optimize the parameters of the model so that the reconstruction error between the input data and output data is minimized [31].

3.2. The Proposed Training Algorithm

In order to render the autoencoder robust against malware anti-analysis techniques and changing environments, the autoencoder should be trained to reconstruct malware data even in the presence of noise. The Vanilla autoencoder is not able to do that. The autoencoder that deals with noisy inputs is called denoising autoencoder [7]. This is achieved by applying noise to the initial input x before mapping it to the hidden layer. In other words, the objective is to reconstruct x, where x represents the original data points (malware or benign). Hence, a corrupted version $z = noise(x)$ will be obtained from x, where $noise$ is a corruption mechanism [7]. In the proposed framework, we use a Gaussian noise as the corruption method.

The autoencoder is then designed to map the noisy data z to the hidden layer and attempt to find the weights (learn) that can be used to reconstruct the original data x. This is achieved by enabling the input z to be mapped to a hidden representation as follows:

$$
\begin{aligned}
h &= f_\theta(z) \\
f_\theta(z) &= SigFun(W' * z + b')
\end{aligned}
\tag{3}
$$

As can be seen, we choose z as input instead of x. The hidden representation h is then used to reconstruct the original data x' as follows:

$$
\begin{aligned}
x' &= g_{\theta'}(h) \\
g_{\theta'}(h) &= SigFun(W * h + b)
\end{aligned}
\tag{4}
$$

The architecture of the proposed framework is shown in Figure 2. As shown in the traditional autoencoder, the model is trained to find the parameters (weights and bias) that lead to minimize the reconstruction error as follows:

$$
\begin{aligned}
\theta^*, \theta'^* &= arg \;\; minimize_{\theta,\theta'} \; \frac{\sum_{j=1}^{n} Loss(z^{(j)}, x'^{(j)})}{n} \\
&= \frac{\sum_{i=1}^{n} Loss(z^{(j)}, g_{\theta'}(f_\theta(z^{(j)})))}{n}
\end{aligned}
\tag{5}
$$

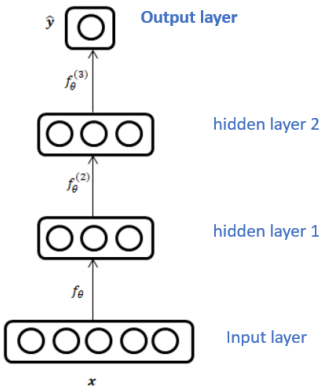


Figure 3: Deep Network.

The training algorithm of the proposed denoising autoencoder-based MDS is described in Algorithm 1. The algorithm first takes an input x from the dataset and applies noises on it to produce a corrupted version z. The corrupted version z is then mapped to the hidden layer, which is then used to reconstruct the output x'. The aim here is to optimize the parameters of the model so that the reconstruction error between x and x' can be minimized. The neural network's parameters are randomly initialized and then optimized using stochastic gradient descent algorithms. This algorithm is applied to each layer added within the proposed framework. To enable the model to classify malware from benign software, a logistic regression classifier is added to the last layer. Thereafter, the parameters of all the layers will be fine-tuned to minimize the error of predicting the target label using a back-propagation algorithm [32] [33] [30] [8][7]. Figure 3 shows the complete architecture of the proposed malware detection framework after adding the last layer.

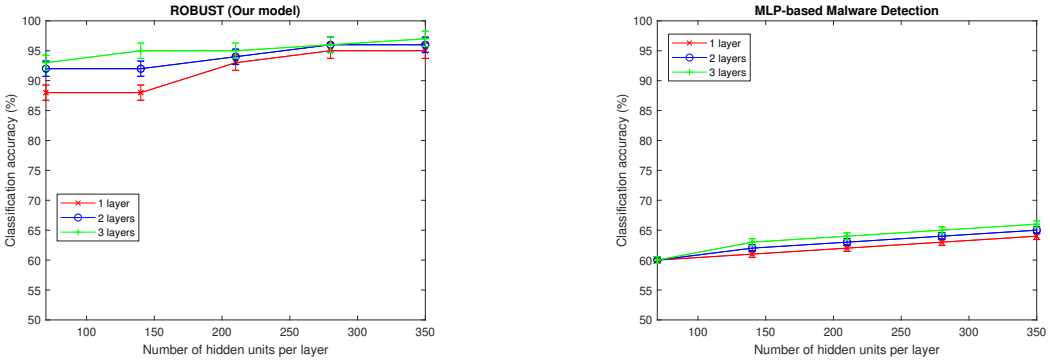
4. Evaluation Results

To experimentally evaluate the proposed framework, we use the CCCS-CIC-AndMal-2020 dataset [34], which is the most suitable dataset to represent unstable and changing environments. This is

Experimentation parameters

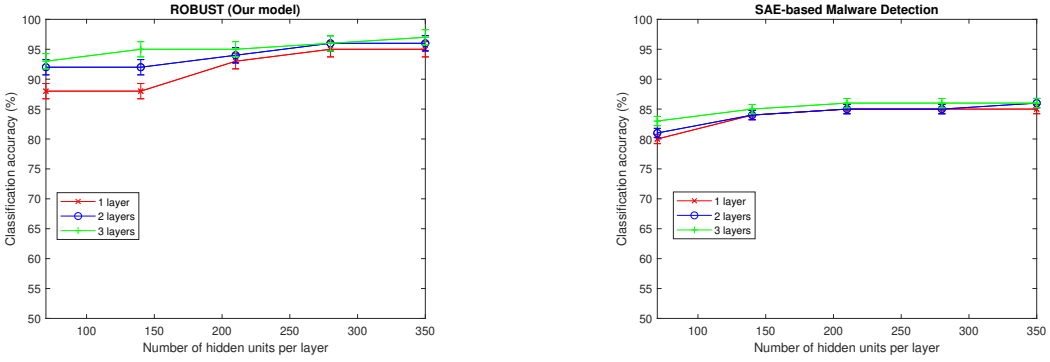
Parameter	Considered values
Number of hidden layers h	$\{1, 2, 3\}$
Corrupting noise level υ	25%
Number of epochs	200
Size of the batches	64
Learning rate	0.0005
Output layer	(multi-class) logistic regression

Figure 4: Experimentation parameters.



(a) (b)

Figure 5: Classification accuracy performance for ROBUST (a) and MLP-based MDS(b) (95% confidence intervals).

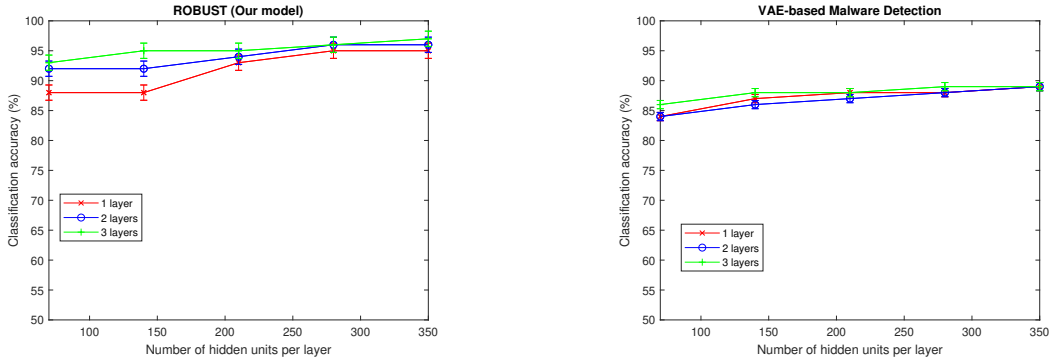


(a) (b)

Figure 6: Classification accuracy performance for ROBUST (a) and SAE-based MDS (b) (95% confidence intervals).

due to the fact that the dataset was collected in the presence of different obfuscations scenarios. We train our model on this dataset using the 10-fold cross-validation. We used the same parameters used in [3] [27] as shown in Figure 4. We compared our model with the state-of-the-art learning representation approaches: Multi-layers perception (MLP), Stacked Autoencoder (SAE) and Variational autoencoder (VAE).

Figure 5 shows the detection accuracy of the proposed framework (ROBUST). We compared our framework with multi-layers perception (MLP)-based MDS. MLP-based MDS is a traditional deep neural network which does not include a pre-training process during the training. Instead, random numbers are used to initialize the weight of the deep neural network. Our study was done with a different number of layers. As for the first layer (1-layer), Figure 5a shows the accuracy (average) reported by the proposed framework at different numbers of hidden units. The results reported are better than the results reported using one layer in MLP-based MDS (5a). Our framework also improved the detection accuracy compared to MLP-based MDS when we



(a) (b)

Figure 7: Classification accuracy performance for ROBUST (a) and VAE-based MDS (b) (95% confidence intervals).

applied two and three layers. This is justified by the fact that the proposed framework adopts a pre-training process. This, in turn, allows the deep network to have better initialization of parameters, which helps during the training of the deep neural network.

Figure 6 shows our results compared to Stacked Auto Encoder (SAE)-based MDS (SAE-based MDS). SAE-based MDS applies traditional autoencoders (as building blocks) when training the deep neural networks. We consider different numbers of layers and hidden nodes in the study. As can be seen in Figure 6, our model enhanced the accuracy compared to SAE-based MDS. In particular, Figure 6a shows the accuracy reported when applying the proposed framework (ROBUST). These results are better than the results reported using SAE-based MDS (6a).

The above results are justified by the fact that our framework uses a denoising autoencoder instead of a vanilla autoencoder as a building block when training the deep neural networks. The DA enables us to extract robust and useful features that increase the chances of obtaining good results despite the noisy and complex malware given as inputs to the deep neural network [7] [8]. More specifically, the denoising autoencoder allows us to learn how to reconstruct the original malware from obfuscated inputs. In contrast, the vanilla autoencoder is not able to produce robust features against noisy inputs.

We also compared the proposed framework against the variational autoencoder-based MDS (VAE-based MDS). In VAE-based MDS, a variational autoencoder is used as a building block during the training [8]. Different numbers of hidden units and layers have been considered in the experiments. As can be seen in Figure 7, our model improved the accuracy compared to VAE-based MDS. This is justified by the fact that the proposed framework adopts a denoising criterion during the training process. This, in turn, allows our mode to extract useful features that enhance the detection under the changing environment.

We also compared the proposed framework with two other frameworks, namely training with noisy inputs: TWNI(1) and TWNI(2) (Figures 8 and 9). TWNI(1) is a stacked autoencoder-based deep neural network, where noisy data are only used for the pre-training. However, TWNI(2) uses noisy data for training in both pre-training and fine-tuning. Both TWNI(1) and TWNI(2) use 3-hidden layers. As can be seen in Figures 8 and 9, the proposed framework is much more robust

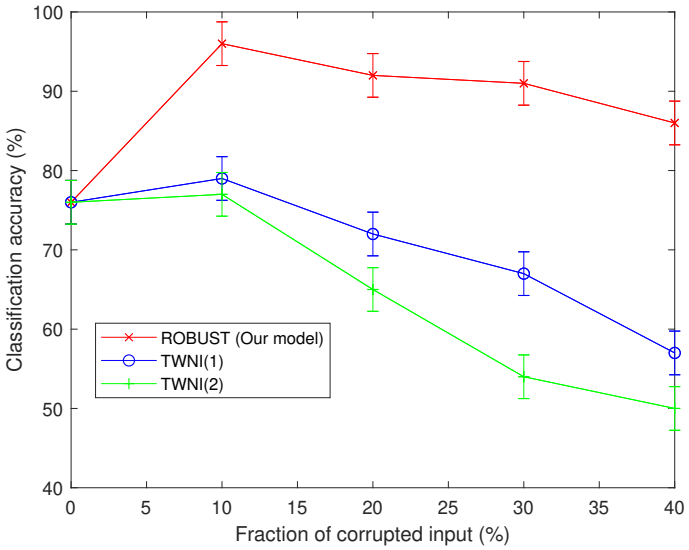


Figure 8: ROBUST (Our model) vs. TWNI(2).

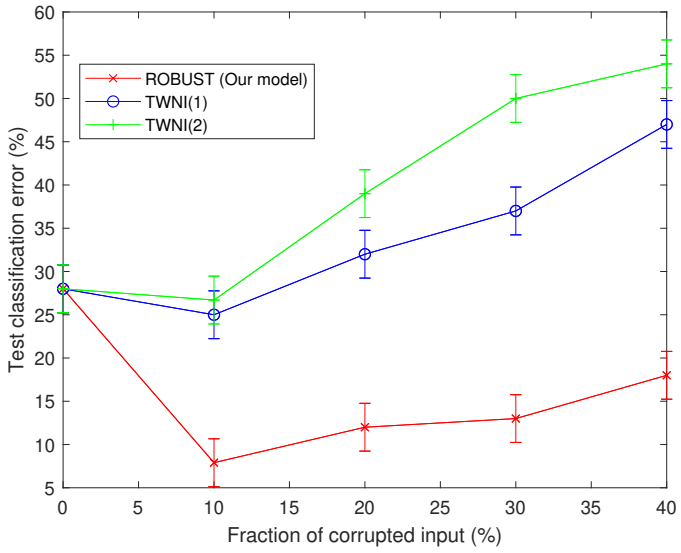


Figure 9: ROBUST (Our model) vs. TWNI(2).

against noisy inputs compared to TWNI(1) and TWNI(2). All models (TWNI(1), TWNI(2) and ROBUST) reported the same error rate and classification accuracy when setting the percentage of corrupted data (input) at zero. This is because when no noise is applied, the three models will become the same as a vanilla stacked autoencoder.

5. Conclusion

In this paper, we proposed a new framework for detecting malware under changing environments. The proposed framework is based on a denoising autoencoder, which allows us to extract robust and useful features to enhance the detection accuracy under changing environments. In particular, the denoising autoencoder is used as a building block during the training of deep neural networks. The proposed model is used to learn how to reconstruct malware after applying noise on it. This is useful to extract features that are robust against anti-analysis techniques and unstable environments. Our model was implemented using a real-world dataset. The results demonstrate the effectiveness of the proposed framework compared to the state-of-the-art malware detection methods.

In the future, we would like to study the prediction of future attacks for new generation malware detection and analysis. Although promising results have already been achieved through layers of abstraction, the existing solutions are still largely vulnerable to zero-day attacks. In fact, compared to malware analysts, malware designers have the advantage of knowing current anti-malware measures and thus novel variants can be designed accordingly. A novel trend in malware analysis is investigating the feasibility to fill that gap by predicting how future malware will look like, so as to allow analysts to proactively update anti-malware measures and remain one step ahead in this arms race.

References

[1] S. M. Devine, N. D. Bastian, Intelligent systems design for malware classification under adversarial conditions, arXiv preprint arXiv:1907.03149 (2019).

[2] M. Ribeiro, K. Grolinger, M. A. Capretz, Mlaas: Machine learning as a service, in: 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA), IEEE, 2015, pp. 896–902.

[3] A. Abusitta, M. Q. Li, B. C. Fung, Malware classification and composition analysis: A survey of recent developments, Journal of Information Security and Applications 59 (2021) 102828.

[4] Z. Xu, H. Wang, Z. Xu, X. Wang, Power attack: An increasing threat to data centers., in: NDSS, 2014.

[5] K. Kimani, V. Oduol, K. Langat, Cyber security challenges for iot-based smart grid networks, International Journal of Critical Infrastructure Protection 25 (2019) 36–49.

[6] M. Jakobsson, Z. Ramzan, Crimeware: understanding new attacks and defenses, Addison-Wesley Professional, 2008.

[7] P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, Extracting and composing robust features with denoising autoencoders, in: Proceedings of the 25th international conference on Machine learning, ACM, 2008, pp. 1096–1103.

[8] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion, Journal of machine learning research 11 (2010) 3371–3408.

[9] S. S. W. Piyanuntcharatsr, S. Adulkasem, C. Chantrapornchai, On the comparison of

malware detection methods using data mining with two feature sets, International Journal of Security and Its Applications 9 (2015) 293–318.

[10] E. Raff, C. Nicholas, An alternative to ncd for large sequences, lempel-ziv jaccard distance, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2017, pp. 1007–1015.

[11] A. Mohaisen, O. Alrawi, M. Mohaisen, Amal: High-fidelity, behavior-based automated malware analysis and classification, computers & security 52 (2015) 251–266.

[12] M. Polino, A. Scorti, F. Maggi, S. Zanero, Jackdaw: Towards automatic reverse engineering of large datasets of binaries, in: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, 2015, pp. 121–143.

[13] A. Tamersoy, K. Roundy, D. H. Chau, Guilt by association: large scale malware detection by mining file-relation graphs, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2014, pp. 1524–1533.

[14] L. Chen, T. Li, M. Abdulhayoglu, Y. Ye, Intelligent malware detection based on file relation graphs, in: Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015), IEEE, 2015, pp. 85–92.

[15] W. Mao, Z. Cai, D. Towsley, X. Guan, Probabilistic inference on integrity for access behavior based malware detection, in: International Symposium on Recent Advances in Intrusion Detection, Springer, 2015, pp. 155–176.

[16] T. Wüchner, M. Ochoa, A. Pretschner, Robust and effective malware detection through quantitative data flow graph metrics, in: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, 2015, pp. 98–118.

[17] A. Kalbhor, T. H. Austin, E. Filiol, S. Josse, M. Stamp, Dueling hidden markov models for virus analysis, Journal of Computer Virology and Hacking Techniques 11 (2015) 103–118.

[18] A. Raghavan, F. Di Troia, M. Stamp, Hidden markov models with random restarts versus boosting for malware detection, Journal of Computer Virology and Hacking Techniques 15 (2019) 97–107.

[19] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, G. Giacinto, Novel feature extraction, selection and fusion for effective malware family classification, in: Proceedings of the sixth ACM conference on data and application security and privacy, ACM, 2016, pp. 183–194.

[20] J. Sexton, C. Storlie, B. Anderson, Subroutine based detection of apt malware, Journal of Computer Virology and Hacking Techniques 12 (2016) 225–233.

[21] N. Kawaguchi, K. Omote, Malware function classification using apis in initial behavior, in: 2015 10th Asia Joint Conference on Information Security, IEEE, 2015, pp. 138–144.

[22] I. Santos, F. Brezo, X. Ugarte-Pedrero, P. G. Bringas, Opcode sequences as representation of executables for data-mining-based unknown malware detection, Information Sciences 231 (2013) 64–82.

[23] G. E. Dahl, J. W. Stokes, L. Deng, D. Yu, Large-scale malware classification using random projections and neural networks, in: Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, IEEE, 2013, pp. 3422–3426.

[24] J. Saxe, K. Berlin, Deep neural network based malware detection using two dimensional binary program features, in: Malicious and Unwanted Software (MALWARE), 2015 10th International Conference on, IEEE, 2015, pp. 11–20.

[25] W. Huang, J. W. Stokes, Mtnet: a multi-task neural network for dynamic malware classifica-

tion, in: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, 2016, pp. 399–418.
[26] B. Kolosnjaji, A. Zarras, G. Webster, C. Eckert, Deep learning for classification of malware system call sequences, in: Australasian Joint Conference on Artificial Intelligence, Springer, 2016, pp. 137–149.
[27] A. Abusitta, M. Bellaiche, M. Dagenais, T. Halabi, A deep learning approach for proactive multi-cloud cooperative intrusion detection system, Future Generation Computer Systems 98 (2019) 308–318.
[28] A. Abusitta, O. A. Wahab, T. Halabi, Deep learning for proactive cooperative malware detection system, in: Edge Intelligence Workshop, volume 711, 2020, p. 7.
[29] C.-Y. Liou, W.-C. Cheng, J.-W. Liou, D.-R. Liou, Autoencoder for words, Neurocomputing 139 (2014) 84–96.
[30] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, in: Advances in neural information processing systems, 2007, pp. 153–160.
[31] I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, Deep learning, volume 1, MIT press Cambridge, 2016.
[32] G. E. Hinton, R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, science 313 (2006) 504–507.
[33] G. E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural computation 18 (2006) 1527–1554.
[34] S. Mahdavifar, A. F. A. Kadir, R. Fatemi, D. Alhadidi, A. A. Ghorbani, Dynamic android malware category classification using semi-supervised deep learning, in: 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), IEEE, 2020, pp. 515–522.