Comparing Machine Learning Techniques for Malware Detection

Joanna Moubarak and Tony Feghali
Potech Labs, Beirut, Lebanon

Keywords: Malware Analysis, Data Science, Machine Learning, Detection, Malicious Behavior.

Abstract: Cyberattacks and the use of malware are more and more omnipresent nowadays. Targets are as varied as states or publicly traded companies. Malware analysis has become a very important activity in the management of computer security incidents. Organizations are often faced with suspicious files captured through their antiviral and security monitoring systems, or during forensics analysis. Most solutions funnel out suspicious files through multiple tactics correlating static and dynamic techniques in order to detect malware. However, these mechanisms have many practical limitations giving rise to a new research track. The aim of this paper is to tackle the use of machine learning algorithms to analyze malware and expose how data science is used to detect malware. Training systems to find attacks allows to develop better protection tools, capable of detecting unprecedented campaigns. This study reveals that many models can be employed to evaluate their detectability. Our demonstration results illustrates the possibility to analyze malware leveraging several machine learning (ML) algorithms comparing them.

1 INTRODUCTION

Malicious software has always been a threat to users. Nowadays, malware are intensely used to generate invalid links that drive the user to infected sites or launch Denial of Service Attacks (DDoS) to steal personal and confidential data using a variety of techniques and tactics such as 0-days exploits to allow faster replication.

Malware analysis (Afianian et al., 2018) is the discipline of studying a malware (Virus, Worm, Trojan Horse, Rootkit, Backdoor, APT ..), to determine the potential impact of an infection (Filiol, 2006). Malware analysis is divided into two parts: static analysis and dynamic analysis (Sikorski and Honig, 2012)(Ligh et al., 2010). In-depth analyzes are based on a mix of both.

Static analysis is to inspect the malicious binary file using various disassemblers and study its contents. Using static analysis techniques including image analysis and string analysis, the analyst observes the code, detects various routines it uses and concludes its features. In the other hand, dynamic analysis (Willems et al., 2007) consists of running the malware in a controlled environment. The idea is to observe the behavior of the malware and draw conclusions. While executed, a malicious software can modify the file system and some configuration files. It might change the windows registry and perform multiple network activities as well. Therefore, observing these behaviors allows to categorize multiple malware actions. Although it is easier and faster to perform a static analysis, it should be noted that some hidden features of the malware can be missed. Besides, malware authors are using anti-disassembly and obfuscation techniques and multiple strategies to mitigate static analysis. Furthermore, advanced malware nowadays implement several mechanisms to automatically change their behaviors once executed in a sandbox for dynamic analysis.

Malware have disrupted several industries and nations in recent years (Moubarak et al., 2017). New techniques are leveraged to enable more sophisticated behaviors and furtiveness (Saad et al., 2019)(Moubarak et al., 2018)(Moubarak et al., 2019). New breed of malicious software can leverage artificial intelligence (AI) to conceal payload and unleash the action when machine learning algorithms identify the target using patterns related to face and voice recognition combined with the geolocation (Stoecklin, 2018).

Furthermore, attackers can exploit machine learning tools to improve the recognition of their potential targets (Chebbi, 2018). Typically, these algorithms allow the collection of information, weaknesses' identification and key elements faster than traditional manual methods (Quinn, 2014). Besides, AI can be used for false data ingestion as well, generating fictitious

Moubarak, J. and Feghali, T.
Comparing Machine Learning Techniques for Malware Detection.
DOI: 10.5220/0009373708440851
In *Proceedings of the 6th International Conference on Information Systems Security and Privacy (ICISSP 2020)*, pages 844-851
ISBN: 978-989-758-399-5; ISSN: 2184-4356

data architectures (James et al., 2018). In the other hand, malware analysis can utilize these models as well.
The pledge of machine learning (ML) in detecting malware consists in apprehending the features of these malicious software to be able to differentiate between good and bad binaries. Different steps are needed for that purpose: malicious and benign binaries are collected and malware specific features are extracted (Saxe and Sanders, 2018) in order to develop appropriate inference.

Multiple studies related to this field have been undergone to analyze malware based on APIs (Fan et al., 2015), system calls (Nikolopoulos and Polenakis, 2017), network inspections (Boukhtouta et al., 2016) and to detect android malware (Wu et al., 2016). In this paper, several ML algorithms are tested and utilized to analyze input PE (Portable Executable) files to establish their malicious or harmless nature. The datasets were tested on several models including Random Forest, Logistic Regression, Naive Bayes, Support Vector Machines, K-nearest neighbors and Neural Networks. Finally, multiple tests are undergone on real data to test the accuracy of the models.

This paper is structured as follows: Section 2 overviews machine learning. Section 3 shows the aptness of several algorithms to analyse malware. Section 4 exposes the results of each detection algorithm. Finally, Section 5 concludes the paper and states our future work.

2 MACHINE LEARNING

Historically, the beginnings of AI date back to Alan Turing in the 1950s (Moor, 2003). In the common imaginary, artificial intelligence is a program that can perform human tasks, learning by itself. However, AI as defined in the industry is rather more or less evolved algorithms that imitate human actions. Subelements of AI include ML, NLP (Natural Language Processing), Planning, Vison and Robotics. The ML is a sub-part of artificial intelligence that focuses on creating machines that behave and operate intelligently or simulate that intelligence. ML is very effective in situations where insights must be discovered from large and diverse datasets. ML algorithms are grouped into five major classes, which correspond to different types of learning (Russell and Norvig, 2016):

1. Supervised learning: the algorithm is given a certain number of examples (inputs) to learn from, and these examples are labeled, that is, we associate them with a desired result (outputs). The algorithm then has for task to find the law which makes it possible to find the output according to the inputs. The aim is to estimate the best function f(x) able to connect the input (x) to the output (y). Through supervised learning, two major types of problems can be solved: classification problems and regression problems.
2. Unsupervised learning: no label is provided to the algorithm that discovers without human assistance the characteristic structure of the input. The algorithm will build its own representation and a human may have difficulty understanding it. Common patterns are identified in order to form homogeneous groups from the observations. Unsupervised learning also splits into two subcategories: clustering and associations. The idea behind clustering is to find similarities within the data in order to form clusters. Slightly different from clustering, association algorithms take care of finding rules in the data. These rules can take the form of "If conditions X and Y are met then event Z may occur".
3. Semi-supervised learning: it encompasses supervised learning and unsupervised learning leveraging labeled data and unlabeled data in order to improve the quality of learning (Zhu et al., 2003).
4. Reinforcement learning: it is an intermediary between the first two algorithms. This technique does not rely on the evaluation of labeled data, but operates according to an experience reward method. The process is evaluated and reinjected into the learning algorithm to improve decision rules and find a better way out of the problem. Oriented for decision-making, this learning is based on experience (failures and successes) (Littman, 1994).
5. Transfer learning: it is a learning that can come to optimize and improve a learning model already in place. The understanding is therefore quite conceptual. The idea is to be able to apply a set that is acquired on a task to a second relative set.

Several ML algorithms are incorporated depending on their relevance. The focus in this study includes the undermentioned algorithms:

- Random Forest: This algorithm belongs to the family of model aggregations, it is actually a special case of bagging (bootstrap aggregating). Moreover, random forests add randomness to the variable level. For each tree, a bootstrap sample is selected and at each stage, the construction of a node of the tree is done on a subset of variables randomly drawn.

- Logistic Regression: It is a supervised classification algorithm where (Y) takes only two possible values (negative or positive). This algorithm measures the association between the occurrence of an event and the factors likely to influence it.
- Naive Bayes: The Bayesian naive classification method is a supervised machine learning algorithm that classifies a set of observations according to rules determined by the algorithm itself. This classification tool must first be trained on a set of learning data that shows the class expected according to the entries. This theorem is based on conditional probabilities. The descriptors (X_i) are two to two independent, conditionally values of the variable to predict (Y).
- Support Vector Machine (SVM): It is also a binary classification algorithm. A SVM algorithm looks for a hyperplane that separates the two categories from the problem. The optimal hyperplane must maximize the distance between the border of separation and the points of each class that are most close (Hastie et al., 2005).
- K-nearest neighbors (KNN): This algorithm is a supervised learning method that can be used for both regression and classification. To make a prediction, the KNN algorithm will be based on the entire dataset. For an observation, which is not part of the dataset, the algorithm will look for the K instances of the dataset closest to the observation. Then, for these K neighbors, the algorithm will be based on their output variables (y) to calculate the value of the variable (Y) of the observation that needs to be predicted.
- Neural Networks: A neural network is inspired by how the human brain works to learn. It is based on a large number of processors operating in parallel and organized in thirds. The first third receives raw information inputs, much like the human's optic nerves when dealing with visual cues. Subsequently, each third party receives the information output from the previous third party. The same process is found in humans when neurons receive signals from neurons close to the optic nerve. The last third, on the other hand, produces the results of the system. In general, neural networks are categorized by the number of thicknesses that separate the data input from the output of the result, based on the number of hidden nodes in the model, or the number of inputs and outputs of each node.

The next section abridges how these algorithms are applied to analyze and detect malicious software.

3 MALWARE ANALYSIS

This section depicts how ML algorithms are evoked to detect malware. The evaluation of the algorithms considered multiple malware features including PE headers, instructions, calls, strings, compression and the Import Address Table. The implementation was based on Python and sklearn (Saxe and Sanders, 2018).

3.1 Random Forest Classifier

A decision tree solves problems by automatically generating interrogation while training samples. For each node of the tree, a question is employed to decide whether the sample is a malware or a benignware. The random forest algorithm (Breiman, 2001) combine multiple decision tree where each tree is trained using different questions. Each tree was trained using a random chosen partial set of samples and the features of each sets are randomly selected.

$$hashed\_features = hasher.transform([string\_features])$$

The detection of a sample is performed on every tree and the algorithm decides on the maliciousness of the binary founded on the response of the majority of the trees.

3.2 Logistic Regression Classifier

The logistic regression (Harrington, 2012) classifier frames a line or a hyperplane that splits malware from benignware in the training dataset.
For a new binary, the algorithm will determine its maliciousness based on the limit between the two classes. The gradient descent algorithm (Ruder, 2016) is used to set the boundary of malware and benignware. The weighted sum of features is translated by the logistic regression classifier to a probability. Features with positive weights are considered malware.

3.3 Naive Bayes Classifier

The naive bayes classifier calculates the probability of a file belonging to a specific category depending on several metrics obtained from the training dataset. A file with a high probability to be clean is beginware, else if the probability is low, then it is a malware (Chebbi, 2018).

3.4 Support Vector Machines Classifier

The support vector machine classifier will also draw a hyperplane that splits malware from benignware in

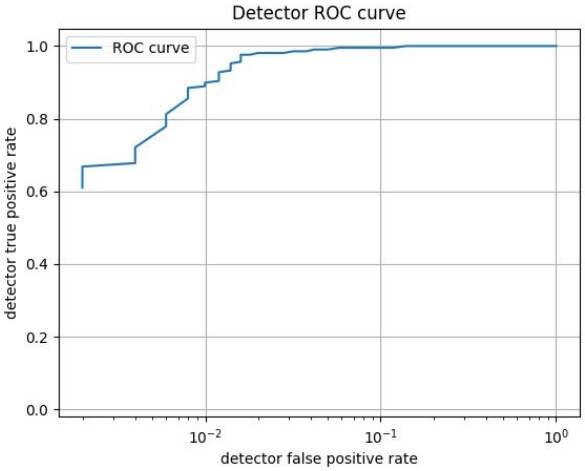

Figure 1: Random forest classifier performance graph.

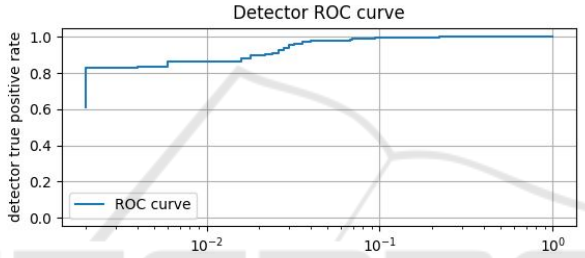

Figure 2: Logistic regression classifier performance graph.

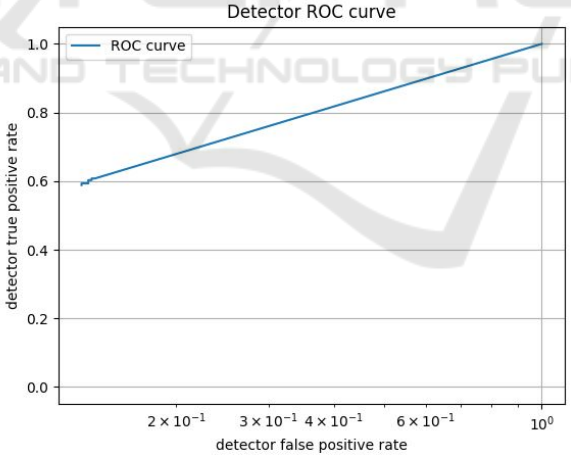

Figure 3: Naive base classifier performance graph.

the training dataset. The decision of being clean or suspicious depends on its location compared to the hyperplane (Chebbi, 2018).

3.5 K-nearest Neighbors Classifier

Having k representing the number of near neighbors, the idea behind the k-nearest neighbors algorithm is to consider that if mostly the characteristics of k-binaries are closest to the characteristics of a binary that is malicious then the binary is malicious. Else, if the majority of characteristics of k-binaries are closest to the characteristic of a benign binary, then this later is benign. For that purpose, the characteristics and features of the new binary are compared to the feature of the k samples. When the percentage of samples in the feature space exceeds a certain number, then the character (malicious or benign) is de-

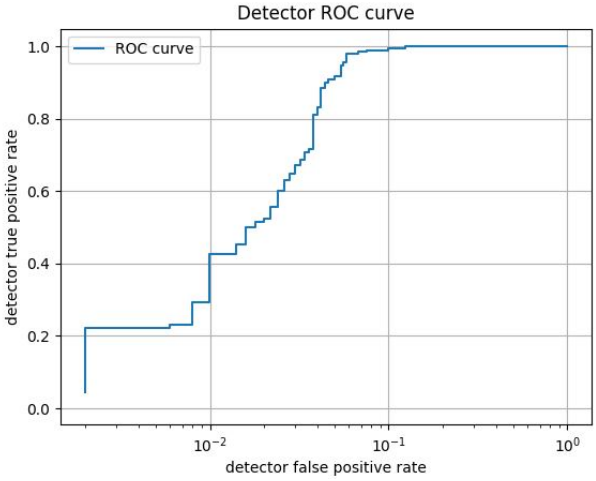

Figure 4: Support vector machines classifier performance graph.

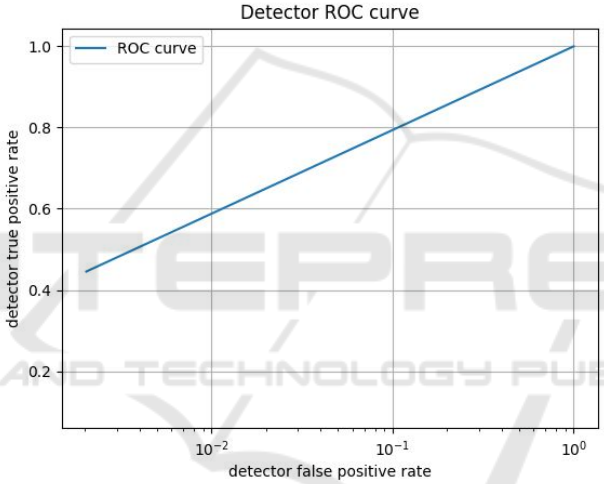

Figure 5: K-nearest neighbors classifier performance graph.

fined. To measure the feature distance (Wang et al., 2007) between a new binary and the samples in the training set the *euclidean_distance* function is used to calculate the shortest distance between two points in the feature space (Saxe and Sanders, 2018). The K-Nearest neighbor algorithm is mostly adapted for malware family classification and connect similarities between binaries.

3.6 Neural Networks

The neural network was divided into layers composed from an input layer, a middle layer and the output layer that generates the final result. The middle layer is formed from 512 neurons that uses ReLU (Rectified Linear Unit) (Saxe and Sanders, 2018) as activation function. The last layer uses a sigmoid function (Gan et al., 2015) and comprise one neuron.

Table 1: Classifiers' Performance.

Classifier	**FPR**	**TPR**
Random forest classifier	10^{-2}	92%
Logistic regression classifier	10^{-2}	40%
Naive baise classifier	2.10^{-2}	65%
Support vector machines classifier	10^{-2}	40%
k-nearest neighbors classifier	10^{-2}	59%
Neural network classifier	10^{-2}	82%

middle = layers.Dense(units=512, activation='relu')(input) output = layers.Dense(units=1, activation='sigmoid')(middle)

Each neuron in the middle layer (dense layer) has access to all input values. The ReLU activation function, if positive, will output a positive value, else it is a zero. The extract_features function was applied to the input (that includes clean and malicious files) and

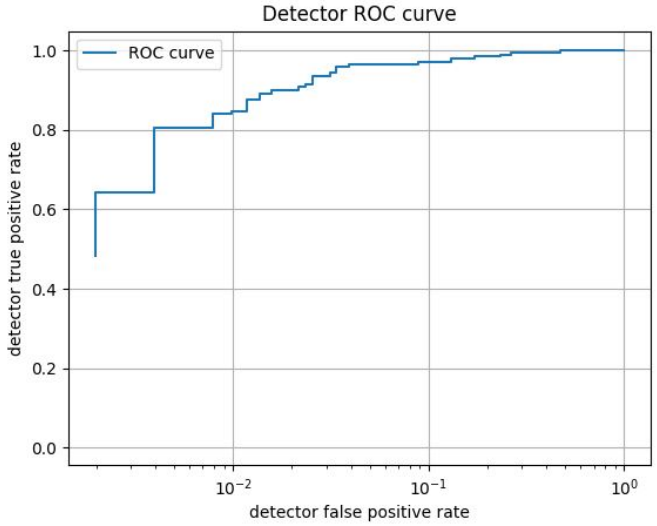

Figure 6: Neural networks classifier performance graph.

then the hash of each token is taken and distributed among this layer. Furthermore, the training was done in batches using a feature generator each time.
model.fit_generator (
generator=training_generator,
steps_per_epoch=num_obs_per_epoch / batch_size,
epochs=10, verbose=1)

4 TECHNICAL ANALYSIS

This section analyses malware leveraging Random Forest, Logistic Regression, Naive Bayes, Support Vector Machines, K-nearest neighbors and Neural Networks algorithms. These models were trained on datasets from (Saxe and Sanders, 2018). The evaluation between these algorithms includes the Receiver Operating Characteristic Curve, the detection time and the limitation of each algorithm.

4.1 Receiver Operating Characteristic Curve

The Receiver Operating Characteristic Curve (ROC Curve) permits to predict the correctness of machine learning algorithms (Bradley, 1997). It consists of a plot visualizing the algorithm true positive rate (TPR) versus its false positive rate (FPR).

$$TPR = \frac{True\ Positives}{True\ Positives\ +\ False\ Negatives}$$

$$FPR = \frac{False\ Positives}{False\ Positives\ +\ True\ Negatives}$$

Predictive rates are divided in four classes:

- **True Positive (TP):** The binary is correctly predicted as being malicious.
- **True Negative (TN):** The binary is correctly predicted as being benign.
- **False Positive (FP):** The binary is incorrectly predicted as being malicious.
- **False Negative (FN):** The binary is incorrectly predicted as being benign.

A good classifier will try therefore to maximize the true positive rate and to minimize the false positive rate. Comparing the several ROC Curves (Table 1), the performance of the random forest classifier is the best among the other algorithms. Besides, comparing the plots of the detectors, the random forest classifier performs well, noting that the execution can be enhanced when scaling the training dataset to a bigger amount of test data adding millions of samples. Others parameters can be added as well.

4.2 Detection Time

To highlight the performance of ML detectors, the same binaries were tested using the different classifiers. Figure 7 summarizes the detection time of each classifier. For a same new binary to test, the neural network and logistic regression classifier achieved the fastest detection rate (4.6 secondes) and the random forest classifier the slowest average (16.5 secondes).

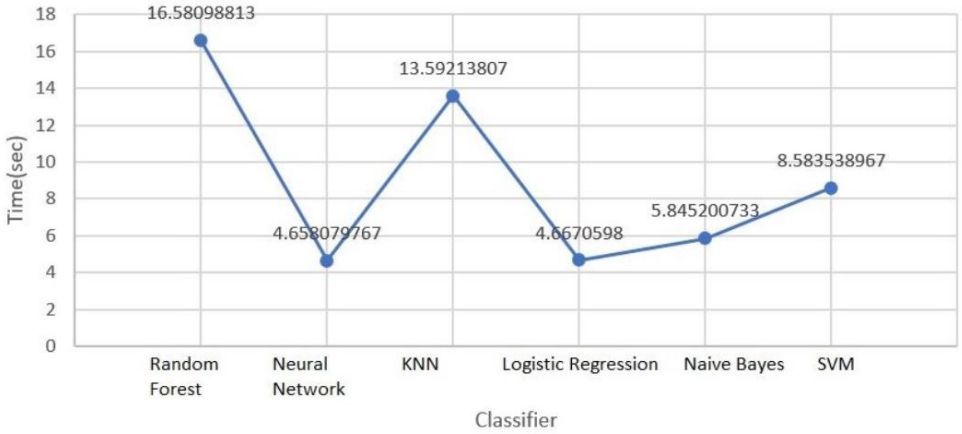

Figure 7: Average Detection Time.

5 CONCLUSION

The information age has recently discovered the value of big data and information that can hide in disparate, large data sources. The current interest in data has also spread across multiple applications to detect and prevent attacks. New technologies permit nowadays an advanced analytics approach leveraging big data. In cybersecurity, machine learning algorithms can be used to detect external intrusions, for example by identifying patterns in the behavior of attackers performing reconnaissance, but also to detect internal risks. The analysis simply aims to provide visualization so that human interaction can be applied to infer ideas. By combining data from system log files, historical data on IP addresses, honeypots, system and user behaviors, etc. a more comprehensive overview of a normal situation is conceived. The wit is to analyze multiple sources and patterns to signal unwanted behavior. Furthermore, machine learning is used for attack detection and attribution. Besides, several use cases of machine learning are employed for penetration testing. The work done in this paper proves that different approaches can be leveraged to detect malware using machine learning. Several algorithms have been implemented, trained and tested. For each algorithm, the methodology of detecting malware have been abridged in details. Moreover, the ROC Curve of each classifier has been illustrated showing that some algorithms perform better than others. This study and classifiers' evaluation show that random forest operates satisfactorily comparing to other algorithms even that the average detection time is not the lowest.

Our future plans consist in studying and enhancing the detection of malware using hybrid training model and ensemble learning. These algorithms can be built also leveraging other parameters and training data. In addition, in a next step we envisage to associate multiple analysis techniques to detect malware. For a complete detection mechanism, we plan to combine static, dynamic and machine learning techniques to analyse malware.

REFERENCES

Afianian, A., Niksefat, S., Sadeghiyan, B., and Baptiste, D. (2018). Malware dynamic analysis evasion techniques: A survey. *arXiv preprint arXiv:1811.01190*.

Boukhtouta, A., Mokhov, S. A., Lakhdari, N.-E., Debbabi, M., and Paquet, J. (2016). Network malware classification comparison using dpi and flow packet headers. *Journal of Computer Virology and Hacking Techniques*, 12(2):69–100.

Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Chebbi, C. (2018). *Mastering Machine Learning for Penetration Testing*. Packt Publishing.

Fan, C.-I., Hsiao, H.-W., Chou, C.-H., and Tseng, Y.-F. (2015). Malware detection systems based on api log data mining. In *2015 IEEE 39th annual computer software and applications conference*, volume 3, pages 255–260. IEEE.

Filiol, E. (2006). *Computer viruses: from theory to applications*. Springer Science & Business Media.

Gan, Z., Henao, R., Carlson, D., and Carin, L. (2015). Learning deep sigmoid belief networks with data augmentation. In *Artificial Intelligence and Statistics*, pages 268–276.

Harrington, P. (2012). *Machine learning in action*. Manning Publications Co.

Hastie, T., Tibshirani, R., Friedman, J., and Franklin, J. (2005). The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85.

James, J., Hou, Y., and Li, V. O. (2018). Online false data injection attack detection with wavelet transform and deep neural networks. *IEEE Transactions on Industrial Informatics*, 14(7):3271–3280.

Ligh, M., Adair, S., Hartstein, B., and Richard, M. (2010). *Malware analyst's cookbook and DVD: tools and techniques for fighting malicious code*. Wiley Publishing.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier.

Moor, J. (2003). *The Turing test: the elusive standard of artificial intelligence*, volume 30. Springer Science & Business Media.

Moubarak, J., Chamoun, M., and Filiol, E. (2017). Comparative study of recent mea malware phylogeny. In *2017 2nd International Conference on Computer and Communication Systems (ICCCS)*, pages 16–20. IEEE.

Moubarak, J., Chamoun, M., and Filiol, E. (2018). Developing a k-ary malware using blockchain. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–4. IEEE.

Moubarak, J., Chamoun, M., and Filiol, E. (2019). Hiding malware on distributed storage. In *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, pages 720–725. IEEE.

Nikolopoulos, S. D. and Polenakis, I. (2017). A graph-based model for malware detection and classification using system-call groups. *Journal of Computer Virology and Hacking Techniques*, 13(1):29–46.

Quinn, M. J. (2014). *Ethics for the information age*. Pearson Boston, MA.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

Saad, S., Briguglio, W., and Elmiligi, H. (2019). The curious case of machine learning in malware detection. *arXiv preprint arXiv:1905.07573*.

Saxe, J. and Sanders, H. (2018). *Malware Data Science*. No Startch Press.

Sikorski, M. and Honig, A. (2012). *Practical malware analysis: the hands-on guide to dissecting malicious software*. no starch press.

Stoecklin, M. P. (2018). Deeplocker: How ai can power a stealthy new breed of malware. *Security Intelligence*, 8.

Wang, J., Neskovic, P., and Cooper, L. N. (2007). Improving nearest neighbor rule with a simple adaptive distance measure. *Pattern Recognition Letters*, 28(2):207–213.

Willems, C., Holz, T., and Freiling, F. (2007). Toward automated dynamic malware analysis using cwsandbox. *IEEE Security & Privacy*, 5(2):32–39.

Wu, S., Wang, P., Li, X., and Zhang, Y. (2016). Effective detection of android malware based on the usage of data flow apis and machine learning. *Information and Software Technology*, 75:17–25.

Zhu, X., Ghahramani, Z., and Lafferty, J. D. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919.