HGM: A Novel Monte-Carlo Simulations based Model for Malware Detection

M Naveed, M Alrammal, A Bensefia
Higher Colleges of Technology, Computer Information System Department, Abu Dhabi, UAE
{malrammal, mnaveed, abensafia}@hct.ac.ae

Abstract. Malware detection is a challenging and non-trivial task due to ever increase in several attacks and their sophistication level. Detection of such attacks demands the exploration of new approaches to generalize the attack patterns. One such approach is the use of Monte-Carlo simulations to train a reinforcement learning model. In this paper, we propose a self-adaptive Monte-Carlo simulation-based reinforcement model called Heuristic-based Generative Model (HGM), which generalizes the attack patterns in such a way that the new unknown attacks can be detected and flagged in real-time. The results show that HGM can detect a variety of malware with high accuracy.

1. Introduction
The Internet plays a primary role in our life. Internet services like e-banking and e-commerce are used by a remarkable number of individuals and organizations. Such services encourage cyber-criminals to achieve their malicious intention by using Malware (malicious programs) [1]. Hundreds of millions of new samples of malicious software have been created. This indicates that approximately one million new threats/attacks are released daily [2].

Several anti-malware solutions have been proposed by security companies to protect organizations and individuals. These solutions are based on a repository where all the “known” types/definitions of malware are referenced; thus, they can be identified and blocked. However, due to the enormous increase in new malware samples, referencing all the malware has become a complicated task. This situation has created a need for intelligent approaches, flexible and adaptable, so that they can identify and block even “unknown” malware.

Machine learning approaches have shown real improvement and efficiency in analyzing and detecting malware. Most approaches are based on the selection of the best of features that describe best a malware, coupled to an appropriate machine learning algorithm [3-13]. In this paper, we extend our previous approach RCBM, based on reinforcement learning [14], to propose a new one called Heuristic-based Generative Model (HGM) that combines the best features of MOCART [15] and Random Forest [4-6].

The remaining of this paper is structured as follows: Section 2 discusses the different approaches used to detect and analyse malware. Section 3 explains the improvements made on our previous approach RBCM [14] to avoid converging to local-minima in the search spaces with a small range of values in an observation dataset. Sections 4 illustrates the experimentation setup and compares the performance of HGM with the state-of-work approaches. In sections 5, we present our conclusion and future work.
2. Related Work
Machines learning approaches, proposed in the literature, can be organized in three major dimensions, according to Ucci’s taxonomy [16]: malware’s objective, malware’s features, and the machine learning algorithm used. Since our study focuses on the Machine Learning relevance in the malware detection, we will focus on the third dimension mainly. Even, if the existing approaches have explored plenty of machine learning techniques, the most used ones are: Random Forest [4-6], Decision trees [7-8], Neuronal Network [9-11], Naïve Bayes [12-13], k-NN and SVM[13]; an exhaustive list is discussed in the following.

Several malware classification approaches have used the Random Forest clustering technique [4-6], due to it competitive performances compared to other algorithms. Karanja et al. [17] conducted a comparison between three clustering techniques, where he showed that the Random Forest overpasses the Naïve Bayes and the KNN algorithms; to this end he modelled the malwares as grayscale images. In another notable work presented in [6], the authors developed a malware clustering system: AMICO, evaluated thanks to a Random Forest technique, during nine months. This system was built around three main components: a download reconstruction module, which aims to reconstruct a real-time TCP traffic flow, where portable executable (PE) files were identified, and their reconstructed response copied by the download history module. Finally, a provenance classifier module analyses the behaviour of how the suspicious file was downloaded using the following set of features: the past file download, domain features, Server IP features, URL features, and the download request features. The training set of labelled malwares was monitored during two periods of time (one month and two months), and the experiments were conducted on the academic network for nine months and showed that AMICO was able to identify malware in 90% of the cases correctly.

Decision trees have also been considered in targeting malwares [7]. A Malware Targeting Recognition system (MaTR) was developed by combining decision tree algorithm and static heuristic features. In this study, the authors used over 100 heuristic features based on structural information (file attributes: name, path, attributes, size,…) and structural anomalies (section names, entry point, imports, exports,…). The system was trained with VX-Heaven dataset, and the decision tree classifier identified 99% of unknown malwares correctly, as announced by the authors.

Neuronal Network haven’t been considered widely in malware detection approaches as the other techniques stated above [9-11]. In Andrade et al. [9], the authors proposed a new public dataset, that was evaluated using a recurrent neuronal network (RNN) classifier. This RNN differs from other RNN by the formation of direct cycles between their neurons. The demand of greater memory of this classifier was resolved by the usage of Long Short-Term Memory (LTSM). The approach performed a good identification rate in 67% of the cases.

The state-of-the-art approaches demonstrated that Random Forests approach is promising. It outperforms other approaches like the Naïve Bayes and KNN. It is an ensemble model of decision tree which extracts efficiently the correlation-based rules. As for MOCART, it is a real-time Monte-Carlo simulation model that trains reinforcement learning structure using a generative model. The approach has been successfully applied in partially observable domains with high uncertainty in search space. The main goal of this paper is to extend our previous approach RBCM [14] in such a way that it will combine the best features among Random Forest and MOCART in the generative model of RBCM. The new approach will be compared with the state-of-the-art approaches to test its accuracy and performance.

3. Heuristic-based Generative Model
HGM is a Monte-Carlo simulation-based model which uses the best features of RBCM and Random Forest. The generative model uses a novel sampling approach which is guided by a heuristic to search for a quality solution under real-time constraints. HGM also self-tunes the parameters for heuristic search.

HGM addresses the random unguided sampling-based search in RBCM which can find solutions with low accuracy in domains with low branching factor as shown in [14]. The high-level description of HGM generative model is shown below (Procedure HGM) for a domain D, sample space $S$ and simulation length $t$, extension $n$. $E$ is the error function.
Procedure HGM(D, t, n, S, E):

1. Generate n samples at s
2. Evaluate Learning Structure on n samples
3. Measure $E(s)$ for n samples
4. If $E(s, d) < \theta$
5. Generate $n+t$ samples
6. Update Learning Structure
7. Evaluate Learning Structure on $n+t$ sample
8. $\theta = E(s, d)$

The generative model of HGM extends the simulation length by $d+n$ as shown above in procedure HGM. The update decision is made by using heuristics $E(s)$ boundary i.e. $\theta$, which depends on the current value of $Q$ function which is a learning structure used in [14]. The $\theta$ value also represents validation of $Q$ function on unseen data. HGM simulation model keeps the decision parameter dynamic and self-tuning to balance the trade-off between exploration and exploitation. Figure 1 explains the sampling strategy to guide the search along the path to quality solutions.

The search space at node $n1$ is simulated with three transitions at three nodes $n2$, $n3$ and $n4$. HGM simulates each sample strategy and evaluates the simulated paths so produce $E(s)$ for each solution quality. $n2$ brings in minimum $E(s)$ than of its siblings and provides a better heuristic for selection of direction to explore further. This also helps the learning structure to tune its weight values that reduce the error of the learner. The heuristic value is self-tuned on each extension of search depth, for example, $E(s)$ at $n3$ will be 0.002. If no siblings of $n1$ produces lower error values than $n1$ the look-ahead search will terminate at this state. This sample strategy is intuitive to take the learner out of local-minima as $E(s)$ will never be smaller than the best of local-minima and generative model will explore more solutions.

Due to self-tuning of the heuristic function $E(s)$ in generative model, the model can converge to biased strategies which always select exploration of the solution space instead of exploiting the best solution found yet. However, to counter this issue, the extensions to look ahead search remains limited to a fixed number. This will reduce the probability of jumping out of a promising region in the solution space. The adaptive use of $E(s)$ also adds to the benefit of avoiding visiting the same sample space for more than once. This feature optimizes the time of searching the best solution in space $S$.

![Selection Process of Deeper Search](image_url)
4. Experimentation and Result

4.1. Experimental Setup
HGM and its rival techniques are simulated using Windows 10 Enterprise with 16GB RAM and dual Intel Core (TM) i7-4702MQ CPUs each of 2.20GHz speed. All experiments are conducted by using the benchmark malwares. The data malware forensic is analysed using different tools for deep visibility of malware behaviours. Wireshark and Network Miner are run on Security Onion to analyse malware files. These data files are then pre-processed to generate a training dataset for the models. The benchmark malware datasets which have been applied for model training are: Microsoft-Malware dataset [18], ARP attack-dataset [19], and ICMP attack-dataset [19]. HGM performance is compared with state-of-the-art supervise learning techniques which are: J48, Convolutional Neural Networks (CNN), Random Forest and Feedforward Neural Networks (FNN). Performance of these models is measured by using three parameters which are Correlation Coefficient (CC), Accuracy and Root Mean Square Error (RMSE). The results of each model are run ten time and then average of these runs is used for comparison purpose.

4.2. Result
HGM is compared with its rivals with respect to all three performance measurement parameters. Table 1 shows the best performance of each model.

| Model     | CC   | Accuracy | RMSE |
|-----------|------|----------|------|
| HGM       | 0.31 | 0.78     | 0.038|
| CNN       | 0.44 | 0.74     | 0.024|
| J48       | 0.04 | 0.49     | 0.493|
| Random Forest | 0.54 | 0.64 | 0.395|
| FNN       | 0.088 | 0.71    | 0.175|

The results show that HGM performs comparatively better than its rivals, particularly in accuracy; however, due to variations in the number of samples in each run, the error rates fluctuate tremendously in each episode of experimental runs. The use of a convolutional neural network in generalization attack behaviors shows a promising strategy. CNN requires extensive training episodes as compared to HGM. Random Forests performs better in terms of correlation coefficient thank other models; however, it does not produce consistent results in terms of accuracy as it does build ensemble models of decision tree and chooses the best one. For this reason, it performs better than J48. HGM is better than its predecessor and other models due to its adaptive approach to sample selection mechanism. The samples are selected based on the quality of the search for solutions during Monte-Carlo simulations.

5. Conclusion
A new approach for malware detection so-called HGM was presented. It is a Monte-Carlo simulation-based model that uses the best features of RBCM and Random Forest. Extensive experimentations were conducted using different malware, known as Microsoft malware, ARP attack, and ICMP attack. HGM demonstrated a better performance than its rival in detecting the malware of various kinds. It demonstrated high performance and self-adaptability to exploration and then balancing the trade-off between exploration and exploitation. Furthermore, as android is ubiquitous and has a very largest share of the mobile OS market with billions of application downloads from the official app market; therefore, we plan to explore HGM for application Malware.
Acknowledgments
This research was funded by the Research Funds of Higher Colleges of Technology (UAE) [HCT-RG-1394]. The authors would like to thank the HCT research office for their support, and also the HCT students who greatly assisted them in this research.

References
[1] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, “A survey on malware detection using data mining techniques,” ACM Comput. Surv., vol. 50, no. 3, 2017, doi: 10.1145/3073559.
[2] K. O. Babaagba and S. O. Adesanya, “A study on the effect of feature selection on malware analysis using machine learning,” ACM Int. Conf. Proceeding Ser., vol. Part F1481, pp. 51–55, 2019, doi: 10.1145/3318396.3318448.
[3] M. Alrammal, M. Naveed, and S. Rihawi, “Using heuristic approach to build Anti-malware,” in ITT 2018 - Information Technology Trends: Emerging Technologies for Artificial Intelligence, 2019, doi: 10.1109/CTIT.2018.8649499.
[4] B. J. Kwon, J. Mondal, J. Jang, L. Bilge, and T. Dumitraundeined, “The Dropper Effect: Insights into Malware Distribution with Downloader Graph Analytics,” in Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015, pp. 1118–1129, doi: 10.1145/2810103.2813724.
[5] W. Mao, Z. Cai, D. Towsley, and X. Guan, “Probabilistic Inference on Integrity for Access Behavior Based Malware Detection BT - Research in Attacks, Intrusions, and Defenses,” 2015, pp. 155–176.
[6] P. Vadrevu, B. Rahbarinia, R. Perdisci, K. Li, and M. Antonakakis, “Measuring and Detecting Malware Downloads in Live Network Traffic BT - Computer Security – ESORICS 2013,” 2013, pp. 556–573.
[7] T. Dube, R. Raines, G. Peterson, K. Bauer, M. Grimalia, and S. Rogers, “Malware target recognition via static heuristics,” Comput. Secur., vol. 31, no. 1, pp. 137–147, 2012, doi: https://doi.org/10.1016/j.cose.2011.09.002.
[8] Q. K. Ali Mirza, I. Awan, and M. Younas, “CloudIntell: An intelligent malware detection system,” Futur. Gener. Comput. Syst., vol. 86, pp. 1042–1053, 2018, doi: https://doi.org/10.1016/j.future.2017.07.016.
[9] E. de O. Andrade, J. Viterbo, C. N. Vasconcelos, J. Guérin, and F. C. Bernardini, “A Model Based on LSTM Neural Networks to Identify Five Different Types of Malware,” Procedia Comput. Sci., vol. 159, pp. 182–191, 2019, doi: https://doi.org/10.1016/j.procs.2019.09.173.
[10] J. Saxe and K. Berlin, “Deep neural network based malware detection using two dimensional binary program features,” in 2015 10th International Conference on Malicious and Unwanted Software (MALWARE), 2015, pp. 11–20, doi: 10.1109/MALWARE.2015.7413680.
[11] I. Firdausi, C. lim, A. Erwin, and A. S. Nuroho, “Analysis of Machine learning Techniques Used in Behavior-Based Malware Detection,” in 2010 Second International Conference on Advances in Computing, Control, and Telecommunication Technologies, 2010, pp. 201–203, doi: 10.1109/ACT.2010.33.
[12] E. Menahem, A. Shabtai, L. Rokach, and Y. Elovici, “Improving malware detection by applying multi-inducer ensemble,” Comput. Stat. Data Anal., vol. 53, no. 4, pp. 1483–1494, 2009, doi: https://doi.org/10.1016/j.csda.2008.10.015.
[13] I. Santos, J. Devesa, F. Brezo, J. Nieves, and P. G. Bringas, “OPEM: A Static-Dynamic Approach for Machine-Learning-Based Malware Detection BT - International Joint Conference CISIS’12-ICEUTE’12-SOCO’12 Special Sessions,” 2013, pp. 271–280.
[14] M. Naveed and M. Alrammal, “Reinforcement learning model for classification of Youtube movie,” J. Eng. Appl. Sci., vol. 12, no. Specialissue9, 2017, doi: 10.3923/jeasci.2017.8746.8750.
[15] M. Naveed, A. Crampton, D. Kitchin, and L. McCluskey, “Real-Time Path Planning using a Simulation-Based Markov Decision Process,” in Research and Development in Intelligent
D. Ucci, L. Aniello, and R. Baldoni, “Survey of machine learning techniques for malware analysis,” *Comput. Secur.*, vol. 81, pp. 123–147, 2019, doi: https://doi.org/10.1016/j.cose.2018.11.001.

E. M. Karanja, S. Masupe, and M. G. Jeffrey, “Analysis of internet of things malware using image texture features and machine learning techniques,” *Internet of Things*, vol. 9, p. 100153, 2020, doi: https://doi.org/10.1016/j.iot.2019.100153.

R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, and M. Ahmadi, “Microsoft Malware Classification Challenge.” 2018, doi: 10.13140/RG.2.2.34695.91045.

M. Parkour, “Contagio Dataset.” http://contagiodump.blogspot.com/ (accessed Feb. 26, 2020).