Article

A Novel Monte-Carlo Simulation-Based Model for Malware Detection (eRBCM)

Muath Alrammal *, Munir Naveed and Georgios Tsaramirsis

Faculty of Computer Information Science, Abu Dhabi Women College, Higher Colleges of Technology, Abu Dhabi 41012, United Arab Emirates; mnaveed@hct.ac.ae (M.N.); gtsaramirsis@hct.ac.ae (G.T.)

* Correspondence: malrammal@hct.ac.ae

Abstract: The use of innovative and sophisticated malware definitions poses a serious threat to computer-based information systems. Such malware is adaptive to the existing security solutions and often works without detection. Once malware completes its malicious activity, it self-destructs and leaves no obvious signature for detection and forensic purposes. The detection of such sophisticated malware is very challenging and a non-trivial task because of the malware's new patterns of exploiting vulnerabilities. Any security solutions require an equal level of sophistication to counter such attacks. In this paper, a novel reinforcement model based on Monte-Carlo simulation called eRBCM is explored to develop a security solution that can detect new and sophisticated network malware definitions. The new model is trained on several kinds of malware and can generalize the malware detection functionality. The model is evaluated using a benchmark set of malware. The results prove that eRBCM can identify a variety of malware with immense accuracy.

Keywords: malware detection; Monte-Carlo simulation; reinforcement learning

Citation: Alrammal, M.; Naveed, M.; Tsaramirsis, G. A Novel Monte-Carlo Simulation-Based Model for Malware Detection (eRBCM). *Electronics* **2021**, *10*, 2881. <https://doi.org/10.3390/electronics10222881>

Academic Editor: Suleiman Yerima

Received: 30 September 2021

Accepted: 16 November 2021

Published: 22 November 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

As the Internet has become essential in our life, the number of users who use internet services such as e-commerce and e-banking, has increased rapidly. Unfortunately, this increment is accompanied by an increased number of cyber-criminals who use malware (malicious programs) to achieve their malicious intentions [1].

Cyber-criminals launch new malware/attacks every year that are more sophisticated and harmful than previous years. Malware can adapt to the environment according to the security barriers set in an IT environment. Millions of new definitions are generated daily to exploit the vulnerabilities and compromise commercial information systems [2].

To overcome this severe threat, security companies such as Kaspersky and Symantec have introduced several anti-malware products to protect individuals and companies [2]. These products are for known malware definitions. While such solutions can detect known malware with high accuracy, they often lack the ability to detect unknown malware. Moreover, referencing all the different malware has become a complex task because of the enormous increase in the number of malware programs, making it difficult to find lasting solutions. These limitations have made it necessary to explore intelligent approaches that are flexible and adaptable in detecting unknown malware.

Most of the new intelligent approaches to malware detection are trained using the selective features of known malware that can represent malware in its best form. These representations are then used as training instances for a suitable machine-learning algorithm that generalizes or maps such features-based malware detection mechanisms [3–13]. This work extends a previously explored approach called RBCM, which is also based on reinforcement learning [3]. The RBCM extension is called eRBCM, and merges the most beneficial features of Monte-Carlo-based real-time learning (MOCART) [4] and random forest [5–7] to make it more scalable for higher-order training datasets.

The rest of this paper is organized as follows: Section 2 presents the various approaches adopted to detect and analyze network malware; Section 3 describes our motivations and contributions; Section 4 provides a short introduction to MOCART; Section 5 illustrates the enhancements to our previous approach (RBCM) [14], made to avoid converging to local minima in the search spaces with a narrow range of values in an observation dataset; Section 6 shows the experimental set-up and compares the performance of *e*RBCM with its state-of-the-art rivals; and Section 7 presents our conclusions and future work.

2. Related Work

According to the malware detection taxonomy outlined by [8], machine-learning approaches can be classified based on three major dimensions: malware targets, malware features, and the AI model used to generalize malware detection. This section focuses on the third dimension, the machine-learning algorithm, since our study evaluates algorithm performance in the malware detection task. Machine-learning algorithms are scalable to generalize non-linear problem spaces, which is the main motivation for exploring such approaches to optimize malware detection.

Different malware detection approaches in the literature have adopted different machine-learning techniques, such as random forest (RF) [5–7], neural network [9–11], decision tree [12,13], naïve Bayes [14,15], KNN and SVM [15], ARIMA [16], and reinforcement learning [17,18].

The RF machine-learning technique has been applied in several malware classification problems in the literature [5–7] because of its competitive performance compared to other algorithms. In an original approach proposed in [19], where the malware features were modelled as grayscale images, a comparison between three machine learning techniques revealed that RF outperformed the naïve Bayes and KNN algorithms.

RF was explored in [6] to generalize the malware detection and classification. The authors presented a machine-learning technique called AMICO [6], which was trained using the network-traffic-based selection parameters. The main purpose was to evaluate the payload information in network traffic. Parameters such as IP address, source URL, target URL, file contents, etc. were analyzed to identify the malware patterns. In the sandbox environment, a download reconstruction module was used to generate the network traffic in real-time. The traffic was based on executable files and the malware detection technique was evaluated using real-time generated data. The training data constructed to generalize the AI model was based on both malware-based traffic data and normal traffic.

To distinguish between malware and benign files, Vadrevu et al. [7] opted for a supervised learning approach based on a RF algorithm, where the training set of labelled malicious instances was evaluated over a period of one to two months. The simulated data contained a fair distribution of both kinds of samples. The model trained on this data was tested using an academic network, and the test results showed that AMICO could detect 90% of the malicious content that travelled over the network during the testing phase.

A classifier based on a decision tree was used in [7] to detect malicious contents. The “Malware Target Recognition (MaTR)” model is a hybrid of a decision tree classifier and is optimized by using a sophisticated heuristic-based feature search to keep the rules exploration-focused towards the promising area of the search space. In their work, the heuristics are built using the structural information of malicious contents and structural anomalies. Examples of malicious content structure include file path, attributes, and size, while examples of structural anomalies include entry point and section names. The classifier was trained using a benchmark dataset called VX-Heaven. The heuristics were built in the pre-processing stages and remained part of the training instances to extract quality rules. The classifier was tested on malicious contents that were not used during the training phase. The test data showed an accuracy of 99% for the decision-tree-based classifier’s malware detection.

Neural-network-based approaches in malware detection were introduced in [9–11], while a recurrent neural network (RNN)-based model was explored by Andrade et al. [9].

The model was trained using a benchmark dataset that is publicly available for exploring new security solutions. Their neural network model creates new connections among the neurons based on cycles to increase the memory-based connectivity. The model also balances the trade-off between the long-term and short-term memory approaches. Short-term memory emphasizes the exploration of solution space, while long-term memory exploits the already known best regions of the solution space. The experimental results showed that RNN detected malicious content with 67% accuracy.

There are several approaches for app malware detection. Approaches including EspyDroid [20], AndroShield [21], Droidcat [22], and RevealDroid [23] are used as solutions for obfuscation camouflage techniques such as junk code insertion, package renaming, and altering control-flow [24–26].

Other approaches for app malware detection, such as API-Graph [27], DroidEvolver [28], and DroidSpan [29], are oriented towards solving the problem of sustainability (performance over time). However, it is unclear how these approaches address this problem in the case of a network attack or malware.

This paper focuses mainly on exploring a machine-learning model that can generalize the patterns of a variety of malware.

3. Motivations and Contributions

Our motivations are summarized below.

Because of the enormous increase in new malware samples, traditional approaches are not scalable to the sophistication of new attacks and lack the capability to detect and analyze these attacks. Intelligent, self-adaptive approaches for efficient network malware detection and analysis are required [2].

There is a need for an approach that can be easily scaled for large and high-dimensional malware datasets to avoid extensive training episodes. This is essential in order to generalize the characteristics of different kinds of malware. The security solution can be trained on different datasets without changes in the learning structures.

Our contributions are summarized as follows.

We improve RBCM [3] to avoid being trapped in local minima. The current version combines the best features of MOCART [4] and RF [5–7]. Monte-Carlo simulations are optimized to dynamically select the region and scale of samples used by the learning model. The dynamic sampling technique is used to enhance the performance of the RBCM learning model, which selects a sampling region of lower error and fixed size. This drawback decreases RBCM's performance in cases where the sample space is limited or there are large areas of low-quality samples that reduce the error with respect to the current surroundings, but the model does not learn new knowledge.

We test *e*RBCM using the three datasets: Microsoft Malware [30], ARP attack, and ICMP attack [31]. Furthermore, we provide a comparison of *e*RBCM with four state-of-the-art, best-performing prediction algorithms.

4. Monte-Carlo-Based Real-Time Learning (MOCART)

MOCART [4] is a Monte-Carlo (MC)-simulation-based machine-learning algorithm that applies the Monte-Carlo tree search to obtain estimates from one node of the solution space to another to reach the goal node. The MC simulations explore the solutions using a sample space and build a learning structure. In MOCART, MC simulations build a value function that can predict the outcomes for each action in an uncertain or unknown environment. The simulations use a model of system which can predict outcomes for a deterministic or nondeterministic problem space. As a result of these characteristics, MOCART has been used in several domains, especially nondeterministic domains. Because of these capabilities, MOCART is particularly suitable for malware detection, as the behavior of a sophisticated piece of malware can be non-deterministic, and it might behave differently at the same state in a problem space. This is particularly true for a new set of malwares that are sensitive to sandboxing and Trojans. However, MOCART underperforms

in domains where the number of possible samples generated during simulations are limited or if the simulation model is biased towards more exploitation than exploration.

5. Reinforcement Learning Model RBCM

To generalize the pattern recognition of various malware attacks, an updated version of reinforcement learning called *e*RBCM is explored in this work. *e*RBCM combines the best features of MOCART and RF.

The sampling techniques are modified in RBCM to keep finding new samples until the error rate remains below a threshold θ .

RBCM suffers from local minima when space dimensions are of a small scale or data has fewer variations with respect to class labels [4]. *e*RBCM increases the number of samples in the simulation model if class labels are not equally distributed. The generative model of *e*RBCM is shown in Figure 1 for a sample S , simulation length d , and extension n .

Figure 1. Generative model for *e*RBCM.

The generative model of *e*RBCM extends the simulation length by n , as shown in step 6 of Figure 1. The update decision is made by using epsilon e , which depends on the current root mean square error (RMSE) of the learning structure (the learning structure is a Q function). This is a validation RMSE of the Q function on unseen data. The decision parameter is dynamic and keeps reducing itself depending on the RMSE, which makes the sample exploration self-adaptive and keeps the trade-off between exploration and exploitation in balance. Figure 2 explains the sampling process with respect to the depth of search for samples in the direction of solutions.

Figure 2. Selection process of deeper search.

The search space at $S1$ was simulated with three neighboring states and only $S3$ was extended as it met the criteria for decision making. The state $S3$ produced the smallest RMSE of its siblings and a more reliable and stronger heuristic to select the direction to explore deeper into. This also assisted the Q function to be updated with the weight values that reduce the error of the network. The epsilon value was updated on each extension of

the search process; for example, epsilon at S4 will be 0.002. If no neighbor of S4 produces a lower RMSE than S4, the search will stop at this state of the space. This process is also intuitive to bring the search out of local minima, as the RMSE will never be reduced below the best local minimum and the generative model will explore more spaces.

Due to dynamic changes in epsilon in the generative model, the model can learn biased strategies to explore the space rather than exploiting the best results. However, because of a fixed number of extensions, the search policy is kept balanced at the exploitation of the best and the search of new states in the search process. The adaptive use of epsilon also introduces the benefit of avoiding the visit for the same sample more than once. It reduces the time of searching for the best solution in the space and gives *e*RBCM the advantage of quicker convergence than CNN.

6. Reinforcement Learning Model Experimentation

6.1. Experimentation

All experiments were performed using Windows 10 Enterprise with 16 GB RAM and dual Intel Core (TM) i7-4702MQ CPUs, each of 2.20 GHz speed. The benchmark malware files were analyzed using different programs for deep visibility of attack data. The tools used were Wireshark and Network Miner. All tools were run in a special operating system called Security Onion. The attack files were further processed to generate a training dataset. The benchmark malware datasets analyzed were: Microsoft Malware dataset [30], ARP attack dataset [31], and ICMP attack dataset [31].

6.1.1. Microsoft Malware Data

The dataset in [30] is organized with respect to machines and has several input features (e.g., 'machineidentifier' and 'hasdetected') which are malware detected on the machine. This column is used as the actual output for training the machine-learning algorithms. The dataset contains the system details for each observation, including default browser, current OS version, firewall, processor, primary disk type, volume capacity, total physical RAM, casing details, and gaming systems. This dataset is used for training machine-learning algorithms to detect malware on end systems running Windows OS.

6.1.2. ARP Attack Dataset

The ARP dataset [31] is taken from the Contagion malware dataset. ARP attacks exploit the vulnerabilities related to Address Resolution Protocol. ARP vulnerabilities can lead to attacks such as ARP spoofing. These types of attacks require careful analysis of the network characteristics for detection. The dataset for this malware is given in pcap files, which contain the network characteristics of the malware attack. Wireshark is used to extract the pattern of the malware. The data in pcap files is exported to csv which is then used as a training dataset.

6.1.3. ICMP Attack Dataset

The ICMP malware dataset [31] is also in the form of pcap files or network data of ICMP-related attacks (ICMP smurf or ping of death, etc.). These malwares exploit the vulnerabilities of network traffic based on ICMP messages or echo messages. Such messages can penetrate a network without being flagged because most of the security solutions are used to filter TCP/UDP based messages. The pattern of such attacks is extracted using Wireshark and exported to a csv file which is then used to train machine-learning algorithms.

*e*RBCM was trained using the benchmark datasets. The model testing also included malware definitions not used in the training. The malware categories of ICMP and ARP include several patterns (definitions) of network malware that are part of the benchmark dataset [28]. These models were trained using 200 malwares in both categories. The testing of the *e*RBCM to measure its performance was conducted on 150 malwares that

were different from the 200 used in the training of *eRBCM*. The *eRBCM* performance was compared with the following state-of-the-art machine-learning techniques:

1. J48.
2. Convolutional neural network (CNN).
3. Feedforward neural network (FNN).
4. Random forest (RF).

The model performance was measured by applying the correlation coefficient (CC), RMSE, and accuracy. Higher correlation coefficient and accuracy values indicate a better performance, while a model with a lower RMSE is considered superior to those with a higher RMSE.

6.2. Results

The results of each model were averaged over ten runs, with the averages shown in Figure 3. The results show that RF established better correlation-based rules and had a superior performance than other models with respect to the CC. RF extracts the best possible rules as it is an ensemble model of a decision tree and identifies the best tree structure.

Figure 3. The correlation coefficient (CC) results of each model.

Figure 4 shows each model's accuracy. The accuracy profile indicates that *eRBCM*'s performance was better than its competitors. Because of variations in sample size in each run, the error-rate fluctuated greatly in each episode of testing. The application of convolutional neural network to extract the attack behaviors of the different malware was a promising strategy. The convolutional neural network took several training episodes to converge as compared to *eRBCM*.

Figure 4. The accuracy results of each model.

While random forest had a higher CC than other models, its performance lacked consistency in relation to accuracy due to the complex nature of malware patterns. When comparing RF and J48, RF performed better with respect to CC and accuracy because it is an ensemble model.

The performances of CNN and FNN were comparable in terms of accuracy, indicating the capability of neural network structures to generalize malware patterns. However, FNN identified fewer similar rules and produced low correlation-based outcomes.

The main success of *e*RBCM in terms of performance is its self-adaptability to explore and then balance the trade-off between exploration and exploitation. *e*RBCM can guide its search towards the promising area of a solution space due to epsilon. The generative model explores more on the lower sides of RMSE as compared to regions of higher RMSE.

Figure 5 displays the RMSE results of each machine-learning technique. The results show that *e*RBCM produced a lower RMSE than most of its rivals. *e*RBCM performed better than its predecessor, RBCM, and had a consistently better performance than other models because of its adaptive approach in simulations to keep the sample size and space suitable for model learning. The samples were selected based on the quality of the search for a solution during Monte-Carlo simulations.

Figure 5. The RMSE performance of each model.

6.3. Look-Ahead Search of *e*RBCM

Figure 6 provides a deep insight into the sample selection mechanism of *e*RBCM. The accuracy of each solution search in a simulation depends on a specific number of samples from the search space. The sample selection mechanism is non-linear and requires adaptation to each problem set given to the simulation model.

Figure 6. The dependence of RMSE on the number of samples per simulation.

e RBCM's selection mechanism depends on a threshold based on the error rate. It selects a threshold value that minimizes the RMSE. This is the main reason for the successful generalization of attack patterns by e RBCM. The self-adaptivity of epsilon enables e RBCM to explore the larger but focused area of search space compared to RBCM and CNN. e RBCM converges faster than its rival because of the self-tuning of epsilon.

The look-ahead search of the generative model also benefits e RBCM in terms of searching high-quality regions with a smaller number of iterations. The regions of lower RMSE are explored in more depth compared with the regions of higher RMSE. This can lead to local minima, but due to the dynamic value of epsilon, the generative model departs such regions in few iterations.

Figure 7 shows the results with respect to RMSE for look-ahead search self-adaptability. The results show that the extended search produced quality solutions with low RMSE. The enhanced performance in the look-ahead search during simulations is explained by the guided exploration of the generative model in the simulation. The higher the n -value of the simulation model (as given in Figure 1), the more e RBCM explores more promising states of the solution space.

Figure 7. The dependence of RMSE on the number of states extended in a simulation.

With shallow searches in the region of quality solutions, e RBCM remains biased towards the exploitation of the best solutions found and it converges to suboptimal solutions as shown in Figure 7. With extensions to look-ahead search, the deep search provides an optimal balance of exploration and exploitation of the current best-found solutions. It also explains the phenomena shown in Figure 2 relating to the look-ahead search of the generative model of e RBCM. At a deeper search, the e RBCM generative model mirrors the natural selection mechanism of evolutionary techniques. It provides a new solution as a mutation of the existing best solution, as shown in Figure 2. At state S3, for example, the generative model generates a new state S4 which is a mutation of S3.

7. Conclusions

In this paper, we presented a new approach called e RBCM to detect malware. The new model was designed using the reinforcement learning approach, which utilizes the strength of Monte-Carlo simulations and builds a strong machine-learning model to detect complex malware patterns. It combines the most beneficial elements of MOCART's reinforcement learning and RF's exploration capabilities. A large number of experiments were conducted using different malware benchmarks, including ARP attack, ICMP attack, and Microsoft Malware. e RBCM was consistently better than its competitors in terms of learning the new malware patterns and detecting unknown malware. This was mainly explained

by *eRBCM*'s self-adaptability to exploration and intelligent tuning of the balance for the trade-off between exploration and exploitation.

For future work, we plan to test our approach with various attacks to measure its scalability and accuracy. Furthermore, *eRBCM* will be explored for mobile malware using benchmark datasets. The mobile malware will be analyzed using sophisticated forensics tools, identifying key patterns via an innovative pre-processing stage. The malware will be scanned and categorized based on its malicious agenda. In each category, the common parameters will be explored using clustering, with these clusters used to generate a training dataset.

Author Contributions: Conceptualization, M.A.; methodology, M.N.; software, G.T.; validation, G.T.; formal analysis, M.A., M.N. and G.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Higher Colleges of Technology, grant number 1394.

Acknowledgments: The authors would like to thank, Higher Colleges of Technology for their support to this project.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ye, Y.; Li, T.; Adjeroh, D.; Iyengar, S.S. A Survey on Malware Detection Using Data Mining Techniques. *ACM Comput. Surv.* **2017**, *50*, 1–40. [\[CrossRef\]](#)
2. Babaagba, K.O.; Adesanya, S.O. A Study on the Effect of Feature Selection on Malware Analysis using Machine Learning. In Proceedings of the 2019 8th International Conference on Educational and Information Technology, Cambridge, UK, 2–4 March 2019; pp. 51–55. [\[CrossRef\]](#)
3. Naveed, M.; Alrammal, M. Reinforcement learning model for classification of Youtube movie. *J. Eng. Appl. Sci.* **2017**, *12*, 8746–8750. [\[CrossRef\]](#)
4. Naveed, M.; Crampton, A.; Kitchin, D.; McCluskey, L. Real-Time Path Planning using a Simulation-Based Markov Decision Process. In *Research and Development in Intelligent Systems XXVIII*; Springer: London, UK, 2011; pp. 35–48.
5. Kwon, B.J.; Mondal, J.; Jang, J.; Bilge, L.; Dumitras, T. The Dropper Effect. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015; pp. 1118–1129. [\[CrossRef\]](#)
6. Mao, W.; Cai, Z.; Towsley, D.; Guan, X. Probabilistic Inference on Integrity for Access Behavior Based Malware Detection. In Proceedings of the 18th International Symposium on Research in Attacks, Intrusions, and Defenses—RAID 2015, Kyoto, Japan, 2 November 2015; pp. 155–176.
7. Vadrevu, P.; Rahbarinia, B.; Perdisci, R.; Li, K.; Antonakakis, M. Measuring and Detecting Malware Downloads in Live Network Traffic. In Proceedings of the European Symposium on Research in Computer Security—ESORICS 2013, Egham, UK, 9–13 September 2013; pp. 556–573.
8. Ucci, D.; Aniello, L.; Baldoni, R. Survey of machine learning techniques for malware analysis. *Comput. Secur.* **2018**, *81*, 123–147. [\[CrossRef\]](#)
9. Andrade, E.D.O.; Viterbo, J.; Vasconcelos, C.N.; Guérin, J.; Bernardini, F.C. A Model Based on LSTM Neural Networks to Identify Five Different Types of Malware. *Procedia Comput. Sci.* **2019**, *159*, 182–191. [\[CrossRef\]](#)
10. Saxe, J.; Berlin, K. Deep neural network based malware detection using two dimensional binary program features. In Proceedings of the 10th International Conference on Malicious and Unwanted Software (MALWARE), Fajardo, PR, USA, 20–22 October 2015; pp. 11–20. [\[CrossRef\]](#)
11. Firdausi, I.; Lim, C.; Erwin, A.; Nugroho, A.S. Analysis of Machine learning Techniques Used in Behavior-Based Malware Detection. In Proceedings of the 2010 Second International Conference on Advances in Computing, Control, and Telecommunication Technologies, Jakarta, Indonesia, 2–3 December 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 201–203. [\[CrossRef\]](#)
12. Dube, T.; Raines, R.; Peterson, G.; Bauer, K.; Grimaila, M.; Rogers, S. Malware target recognition via static heuristics. *Comput. Secur.* **2012**, *31*, 137–147. [\[CrossRef\]](#)
13. Mirza, Q.K.A.; Awan, I.; Younas, M. CloudIntell: An intelligent malware detection system. *Future Gener. Comput. Syst.* **2018**, *86*, 1042–1053. [\[CrossRef\]](#)
14. Menahem, E.; Shabtai, A.; Rokach, L.; Elovici, Y. Improving malware detection by applying multi-inducer ensemble. *Comput. Stat. Data Anal.* **2009**, *53*, 1483–1494. [\[CrossRef\]](#)
15. Santos, I.; Devesa, J.; Brezo, F.; Nieves, J.; Bringas, P.G. OPEM: A Static-Dynamic Approach for Machine-Learning-Based Malware Detection. In Proceedings of the International Joint Conference CISIS'12-ICEUTE'12-SOCO'12 Special Sessions 2013, Ostrava, Czech Republic, 24 August 2012; pp. 271–280.

16. Almshari, M.; Tsaramirsis, G.; Khadidos, A.O.; Buhari, S.M.; Khan, F.Q.; Khadidos, A.O. Detection of Potentially Compromised Computer Nodes and Clusters Connected on a Smart Grid, Using Power Consumption Data. *Sensors* **2020**, *20*, 5075. [[CrossRef](#)] [[PubMed](#)]
17. Alrammal, M.; Naveed, M. Monte-Carlo Based Reinforcement Learning (MCRL). *Int. J. Mach. Learn. Comput.* **2020**, *10*, 227–232. [[CrossRef](#)]
18. Naveed, M.; Alrammal, M.; Bensefia, A. HGM: A Novel Monte-Carlo Simulations based Model for Malware Detection. *IOP Conf. Ser. Mater. Sci. Eng.* **2020**, *946*, 012003. [[CrossRef](#)]
19. Karanja, E.; Masupe, S.; Jeffrey, M.G. Analysis of internet of things malware using image texture features and machine learning techniques. *Internet Things* **2019**, *9*, 100153. [[CrossRef](#)]
20. Gajrani, J.; Agarwal, U.; Laxmi, V.; Bezawada, B.; Gaur, M.S.; Tripathi, M.; Zemmari, A. EspyDroid⁺: Precise reflection analysis of android apps. *Comput. Secur.* **2019**, *90*, 101688. [[CrossRef](#)]
21. Amin, A.; Eldessouki, A.; Magdy, M.T.; Abdeen, N.; Hindy, H.; Hegazy, I. AndroShield: Automated Android Applications Vulnerability Detection, a Hybrid Static and Dynamic Analysis Approach. *Information* **2019**, *10*, 326. [[CrossRef](#)]
22. Cai, H.; Meng, N.; Ryder, B.; Yao, D. DroidCat: Effective Android Malware Detection and Categorization via App-Level Profiling. *IEEE Trans. Inf. Forensics Secur.* **2018**, *14*, 1455–1470. [[CrossRef](#)]
23. Garcia, J.; Hammad, M.; Malek, S. Lightweight, Obfuscation-Resilient Detection and Family Identification of Android Malware. *ACM Trans. Softw. Eng. Methodol.* **2018**, *26*, 1–29. [[CrossRef](#)]
24. Zheng, M.; Lee, P.P.C.; Lui, J.C.S. ADAM: An Automatic and Extensible Platform to Stress Test Android Anti-Virus Systems. In *Detection of Intrusions and Malware, and Vulnerability Assessment, Proceedings of the 9th International Conference, DIMVA 2012, Heraklion, Crete, Greece, 26–27 July 2012*; Lecture Notes in Computer Science; Flegel, U., Markatos, E., Robertson, W., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 82–101. [[CrossRef](#)]
25. Schrittwieser, S.; Katzenbeisser, S.; Kieseberg, P.; Huber, M.; Leithner, M.; Mulazzani, M.; Weippl, E. Covert Computation—Hiding code in code through compile-time obfuscation. *Comput. Secur.* **2014**, *42*, 13–26. [[CrossRef](#)]
26. Wei, F.; Li, Y.; Roy, S.; Ou, X.; Zhou, W. Deep Ground Truth Analysis of Current Android Malware. In *Detection of Intrusions and Malware, and Vulnerability Assessment*; Lecture Notes in Computer Science Book Series; Springer: Cham, Switzerland, 2017; Volume 10327, pp. 252–276. [[CrossRef](#)]
27. Zhang, X.; Zhang, Y.; Zhong, M.; Ding, D.; Cao, Y.; Zhang, Y.; Zhang, M.; Yang, M. Enhancing State-of-the-art Classifiers with API Semantics to Detect Evolved Android Malware. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, 9–13 November 2020*; pp. 757–770. [[CrossRef](#)]
28. Xu, K.; Li, Y.; Deng, R.; Chen, K.; Xu, J. DroidEvolver: Self-Evolving Android Malware Detection System. In *Proceedings of the 2019 IEEE European Symposium on Security and Privacy (EuroS P), Stockholm, Sweden, 17–19 June 2019*; pp. 47–62. [[CrossRef](#)]
29. Cai, H. Assessing and Improving Malware Detection Sustainability through App Evolution Studies. *ACM Trans. Softw. Eng. Methodol.* **2020**, *29*, 1–28. [[CrossRef](#)]
30. Ronen, R.; Radu, M.; Feuerstein, C.; Yom-Tov, E.; Ahmadi, M. Microsoft Malware Classification Challenge. 2018. Available online: <https://www.researchgate.net/publication/323470001> (accessed on 22 November 2021).
31. Parkour, M. Contagio Malware Dump. Available online: <https://contagiodump.blogspot.com/> (accessed on 26 February 2020).