Scientific and Practical Cyber Security Journal (SPCSJ) 8(4): 123 – 131 ISSN 2587-4667
Scientific Cyber Security Association (SCSA)

SHORT NOTE REGARDING BLACKBOX ANDROID MALWARE DETECTION USING MACHINE LEARNING AND EVASION ATTACKS TECHNIQUES

Professor Dr. Razvan Bocu
Department of Mathematics and Computer Science, Transilvania University of Brasov, Romania

ABSTRACT: Over the past ten years, researchers have extensively explored the vulnerability of Android malware detectors to adversarial examples through the development of evasion attacks. Nevertheless, the feasibility of these attacks in real-world use case scenarios is debatable. Most of the existing published papers are based on the assumptions that the attackers know the details of the target classifiers used for malware detection. Nevertheless, in reality, malicious actors have limited access to the target classifiers. This proposed talk presents a problem-space adversarial attack designed to effectively evade blackbox Android malware detectors in real-world use case scenarios. The proposed approach constructs a collection of problem-space transformations derived from benign donors that share opcode-level similarity with malware applications through the consideration of an *n*-gram-based approach. These transformations are then used to present malware instances as legitimate entities through an iterative and incremental manipulation strategy. The proposed presentation will describe a manipulation model that is based on a query-efficient optimization algorithm, which can identify and implement the required sequences of transformations into the malware applications. The model has already been evaluated relative to more than 1,000 malware applications. This demonstrates the effectiveness of the reported approach relative to the generation of real-world adversarial examples in both software and hardware-related scenarios. The experiments that we conducted demonstrate that the proposed model may effectively trick various malware detectors into believing that malware entities are legitimate. More precisely, the proposed model generates evasion rates of 90%–95% relative to data sets like DREBIN, Sec-SVM, ADE-MA, MaMaDroid, and Opcode-SVM. The average number of required computational operations belongs to the range [1..7]. Additionally, it is relevant to note that the proposed adversarial attack preserves its stealthiness against the virus detection core of three popular commercial antivirus software applications. The obtained evasion rate is 87%, which further proves the proposed model's relevance for real-world use case scenarios.

KEYWORDS: *Blackbox Malware Detection, Android, Machine Learning, Evasion Attacks, Android APK Decompilation*

INTRODUCTION

Machine Learning (ML) determines a references conceptual system, which may be considered to address the challenges posed by continues to show promise in detecting complex and zero-day malicious programs. Thus, the suggested bibliographic references report interesting machinea learning-based contributions.

The first perspective relates to the feature representation of Android applications. Thus, the application of a slight change in the structure of the feature representation of a malware application may break its functionality, considering that malware features obtained from Android Application Packages (APK) are generally discrete (e.g., application permissions), as compared to continuous features, such as the pixel intensity in a grayscale image. A potential solution is represented by the manipulation of the features acquired from the Android Manifest file. Nevertheless, the feasibility of such structural changes relative to the generation of executable adversarial examples (AE) is debatable for a number of reasons. First, the modification of the features that are part of the Android Manifest, such as content providers, intents, is not meant to guarantee the proper function of the original applications that contain the malicious payload. Second, the addition of unused features to the Manifest file may be discarded through the application of preprocessing techniques. Additionally, the advanced Android malware

detectors especially consider the semantics of Android applications, which are represented by the Dalvik bytecode, in the detriment of the Manifest files.

Another difficulty is represented by the limitations of feature mapping techniques, which are considered to convert Android applications from the problem space, all the way to the feature space. These technical solutions are not reversible, which means that feature-space perturbations cannot be directly transformed into a malicious application. The proper approach to an inverse feature-mapping problem relates to a common approach to process real-world malware applications using problem-space transformations that relate to the features used as part of the respective ML models. Through the application of these feature-based transformations to the target Android applications the adversaries are able to create hazardous evasion attacks. Nevertheless, the determination of the proper transformations that address problem-space constraints is not determined by a simple process. First, certain modifications that mimic feature-space perturbations may not result in feasible adversarial examples, considering that they ignore feature dependencies generated from real-world objects. Moreover, certain transformations that meet problem-space constraints for manipulating real objects may introduce undesired or incompatible payloads into malware applications. These types of transformations not only might generate perturbations that are different from the reference that is expected by the attacker, but they may also lead to the functional crash of the adversarial malware applications.

The final problematic aspect is connected to the current methods, which generate adversarial examples based on the specific features of the target malware detectors, such as the ML algorithmic model, and the related features set. These solutions presume that attackers possess either Perfect Knowledge (PK), or Limited Knowledge (LK) regarding the target classifiers. Nevertheless, relative to real-world use case scenarios, adversaries usually possess Zero Knowledge (ZK) regarding the target malware detectors. This aligns in a closer manner with reality, considering that antivirus systems operate as blackbox engines that are interrogated. Thus, certain solutions have evaluated partial blackbox configurations to generate adversarial examples through leveraging the feedback from the target detectors. It is relevant to note that these approaches are inefficient relative to the evasion costs, which include the high number of queries that are necessary, and the extent of manipulation applied to the input sample. Thus, the efficiency of the involved interrogations is fundamental considering the associated costs, and the risk of detectors blocking suspicious queries. Moreover, the application of only the necessary manipulation operations is preferred, considering that, otherwise, the malicious functionality of the applications may be affected.

The research process, which this paper reports, determined the following contributions.

- We propose a comprehensive and generalized evasion attack, which can bypass blackbox Android malware classifiers through a two-step process: (i) *preparation* and (ii) *manipulation*. The first step involves implementing a donor selection technique to create an action set comprising a collection of problem-space transformations. This relates to code snippets known as *gadgets*.
- These gadgets are derived by conducting program slicing on benign apps, known as donors, which are publicly available. By injecting each gadget into a malware app, specific payloads from a benign donor can be incorporated into the malware application.
- The proposed technique utilizes an *n-gram-based similarity (**sequence of n adjacent symbols in a particular order**)* method to identify suitable donors, particularly benign apps that exhibit similarities to malware apps at the opcode level (**specifies operation to be executed**). Applying transformations derived from these donors to malware apps can enable them to appear legitimate(benign), or move them towards blind spots of ML classifiers.
- This approach aims to achieve the desired outcome of introducing transformations that not only ensure adherence to problem-space constraints (**preserved semantics, robustness to preprocessing, and plausibility**), but also possibly lead to malware classification errors.
- We propose a *blackbox evasion* attack that generates real-world Android Adversarial Attacks (AE-Adversarial Examples) that adhere to problem-space constraints. To the best of our knowledge, this is one of the few studies in the Android scope that successfully evades ML-based malware detectors by effectively manipulating malware samples without performing feature-space perturbations.

- We demonstrate this is a *query-efficient* attack capable of deceiving various blackbox ML-based malware detectors through minimal querying. Thus, our proposed problem-space adversarial attack achieves evasion rates of 92%, 88%, 89%, 98%, and 84% against DREBIN, Sec-SVM, ADE-MA, MaMaDroid, and Opcode-SVM, respectively.
- Our proposed attack can operate with either *soft labels* (confidence scores), or *hard labels* (classification labels) of malware apps, as specified by the target malware classifiers, to generate adversarial examples.
- We assess the practicality of the proposed evasion attack under real-world constraints by evaluating its performance in deceiving popular commercial antivirus products. Specifically, our findings indicate that proposed approach may significantly diminish the effectiveness of three popular commercial antivirus products, achieving an average evasion rate of approximately 86%.

PROPOSED ATTACK METHOD

The purpose of the adversarial goal is to manipulate Android malware samples in order to deceive static ML-based Android malware detectors. The proposed attack is an untargeted attack (Carlini et al., 2019) designed to mislead binary classifiers considered in Android malware detection, causing Android malware apps to be misclassified. More precisely, the objective is to trick malware classifiers into classifying malware samples as benign.

Concerning the adversarial knowledge, the proposed evasion attack has blackbox access to the target malware classifier. Therefore, it does not have knowledge of the training data D, the feature set X, or the classification model f, more precisely to the classification algorithm and its hyperparameters. The attacker can only obtain the classification results (hard labels or soft labels) by querying the target malware classifier.

It is also relevant to discuss about the adversarial capabilities. Thus, the designed attack model is conceived to deceive blackbox Android malware classifiers during their prediction phase. Our attack manipulates an Android malware app by applying a set of safe transformations, known as Android gadgets (slices of the benign apps' bytecode), which are optimized through interactions with the blackbox target classifier. Furthermore, in order to avoid major disruptions to apps, the manipulation process of a malware app is conducted gradually, making it resemble benign apps. This is achieved by injecting a minimal number of gadgets extracted from benign apps into the malware app, and the process continues until the malware app is misclassified or reaches the predefined evasion cost. In addition to the problem-space constraints discussed in previous research contributions, our model must also adhere to two additional constraints highlighting the significance of minimizing evasion costs:

- **Number of queries.** Proposed approach is a decision-based adversarial attack that aims to generate AEs while minimizing the number of queries, thus reducing the associated costs.
- **Size of adversarial payloads.** In order to generate executable and visually inconspicuous AEs, such as those with minimal file size, proposed approach aims to minimize the size of injected adversarial payloads.

It is relevant to note that that each gadget consists of an organ, which represents a slice of program functionality, an entry point to the organ, and a vein, which represents an execution path that leads to the entry point. Proposed model extracts gadgets from benign apps by identifying entry points, which are typically API calls, through string analysis. The proposed attack assumes that the benign apps used for gadget extraction are not obfuscated, particularly in terms of their API calls. This is because the proposed model relies on string analysis to identify entry points, which limits its ability to extract gadgets from obfuscated apps. The gadget injection is **considered successful** when both the classification loss value of the manipulated app increases, and the injected adversarial payload conforms to the predefined size of the adversarial payload. Additionally, the injected gadgets are placed within the block of an obfuscated condition statement that is always evaluated as False during runtime, and cannot be analyzed during early preprocessing stages.

It is relevant to note that, concerning the defender's capabilities, it is assumed that the target ML models do not employ adaptive defenses that are aware of the operations performed by proposed model due to

disclosing detectors' vulnerability to the detection core of our proposed model. Specifically, these target models are unable to enhance their resilience by incorporating AEs generated by proposed model during adversarial training. Furthermore, they lack the capability to detect and block queries from proposed model, if they become suspicious of its origin. Additionally, our analysis suggests that proposed model can still be effective, even if we relax the second assumption regarding the defender's capabilities. This is supported by empirical evidence demonstrating that our attack often requires only a minimal number of queries to generate adversarial examples.

METHODOLOGY

The primary goal of proposed model is to transform a malware app into an adversarial app insuch a way that it retains its malicious behavior, but is no longer classified as malware by ML-based malware detectors. This is achieved through an iterative and incremental algorithm used in the proposed attack, which aims to disguise malware APKs as benign ones. The attack algorithm generates real-world adversarial examples from malware apps using *problem-space transformations* that satisfy problem space constraints. These transformations are extracted from benign apps in the wild, which are similar to malware apps using an *n-gram*-based similarity model.
Considering this approach, a random search (RS) algorithm is used to optimize the manipulations of applications. Each malware app undergoes incremental tweaking during the optimization process, where a sequence of transformations is applied over different iterations. These transformations are extracted from benign apps in the wild, which are similar to malware apps using an *n-gram*-based similarity model.
The attack pipeline is structured according to the diagram, which is described in Figure 1.

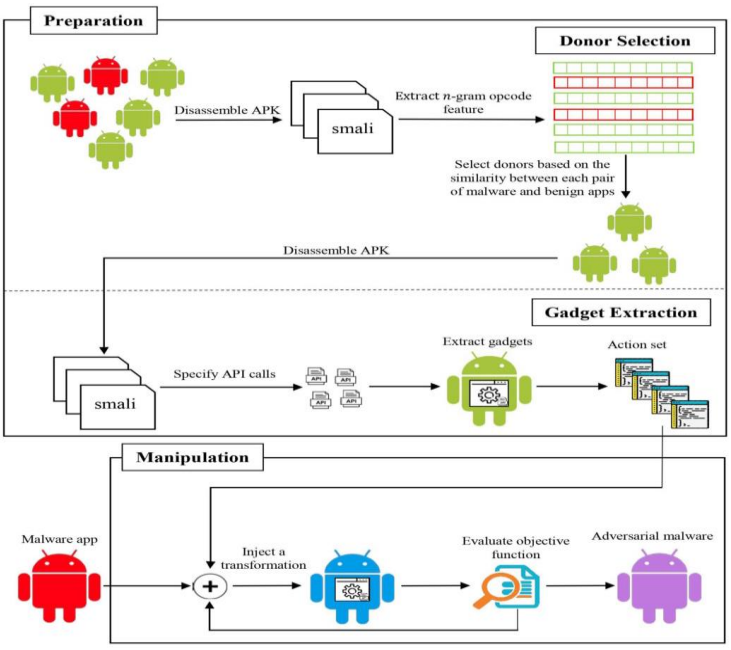

Fig.1. *Logical flow of the attack pipeline*

EXPERIMENTAL PROCESS

The experimental dataset consists of $\approx 170K$ samples, each represented using the DREBIN feature set (Arp et al., 2014). The samples are feature representations of Android applications collected from AndroZoo (AndroZoo, 2024). These collected applications were published between January 2017 and December 2018. Thus, an APK is considered malicious or clean if it has been detected by any 4+ or 0 VirusTotal (VT) (VirusTotal, 2024) engines, respectively. It is important to note that the threshold-

based labeling approach does not rely on specific engines, but instead considers the number of engines involved. Therefore, the engines used for labeling may vary from sample to sample.
Relative to the performance metrics, we consider the True Positive Rate (TPR), and False Positive Rate (FPR) as performance metrics for evaluating the effectiveness of malware classifiers in detecting Android malware. Additionally, we consider the Evasion Rate (ER) and Evasion Time (ET) as proposed model's performance assessment metrics in deceiving malware classifiers. Thus, ER is calculated as the ratio of correctly detected malware samples that are able to evade the target classifiers after manipulation, relative to the total number of correctly classified malware samples. Additionally, ET represents the average time, expressed in seconds, required by proposed model to generate an AE, encompassing both the optimization and query times. The optimization time primarily consists of the execution times of random search, injecting problem-space transformations, and performing feature extraction to represent manipulated applications within the feature space. Following, six research questions were considered.

RQ1. The evasion rates of proposed approach in fooling various malware detectors under different adversarial payload sizes and query numbers

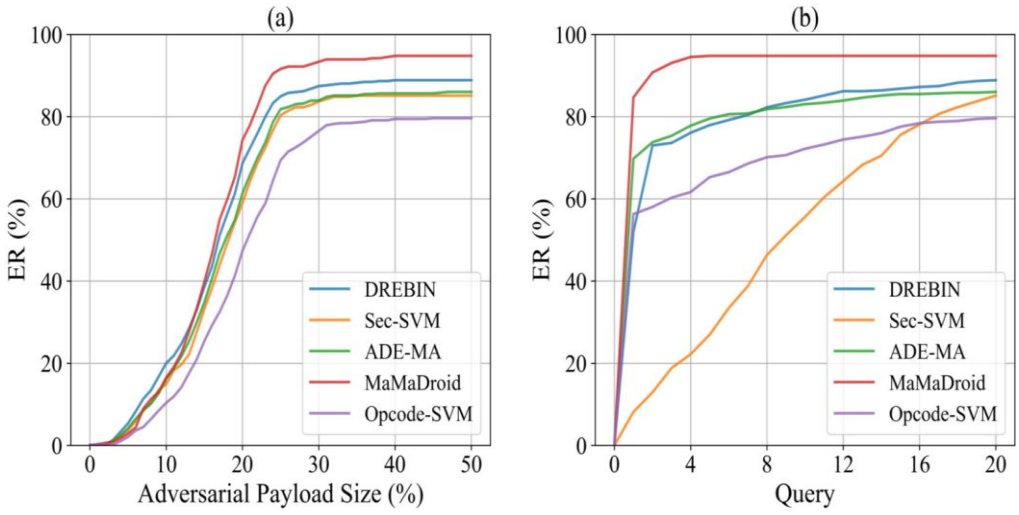

***Fig.2**. Experimental results related to RQ1*

RQ2. Effectiveness of proposed model in misleading different malware detectors. NoQ, NoT, and AS denote Avg. No. of Queries, Avg. No. of Transformations, and Avg. Adversarial Payload Size, respectively.

Type of Threat	Target Model	ER (%)	ET (s)	NoQ	NoT	AS (%)
Soft Label	DREBIN	88.9	210.3	3	2	15.5
	Sec-SVM	85.1	495.4	9	4	16.4
	ADE-MA	86.0	126.2	2	1	16.3
	MaMaDroid	94.8	131.4	1	1	15.9
	Opcode-SVM	79.6	114.1	3	2	18.3
Optimal Hard Label	DREBIN	84.5	240.6	4	2	16.2
	Sec-SVM	82.6	613.1	9	6	16.5
	ADE-MA	84.4	121.2	2	1	16.3
	MaMaDroid	94.8	133.7	1	1	15.9
	Opcode-SVM	74.1	101.2	2	1	18.2
Non-optimal Hard Label	DREBIN	79.7	357.2	4	4	16.9
	Sec-SVM	78.2	782.8	9	9	17.3
	ADE-MA	82.7	157.3	2	2	16.4
	MaMaDroid	94.8	132.6	1	1	15.9
	Opcode-SVM	66.6	76.2	1	1	18.3

***Fig.2**. Experimental results related to RQ2*

Data in Figure 2 demonstrate a 15% improvement in the evasion rate(ER) of proposed model when considering Opcode-SVM in the soft-label setting, compared to the non-optimal hard-label setting. Furthermore, when operating in the soft-label setting, proposed model requires notably fewer transformations to bypass DREBIN and Sec-SVM, as compared to the non-optimal hard-label setting, which confirms the effectiveness of described approach in solving the optimization problem.

The table further illustrates that our optimization leads to a substantial reduction in evasion time (ET) compared to the non-optimal hard-label setting. Specifically, for DREBIN and Sec-SVM, this leads to a time reduction of $\approx 41\%$ and $\approx 37\%$, respectively. This significant enhancement can be attributed to the reduction in the number of transformations, achieved through the utilization of our proposed optimization model.

Thus, the experimental results demonstrate that the proposed adversarial attack is a versatile blackbox attack that does not make assumptions about target detectors, including the ML algorithms or the features used for malware detection. As a consequence, it can operate effectively in various attack settings.

RQ3. Proposed model relative to other attacks

We conduct an empirical analysis to assess how the proposed attack pattern performs in comparison to other similar attacks. We consider four baseline attacks: PiAttack, Sparse-RS, ShadowDroid, and GenDroid operating in whitebox, graybox, semi-blackbox, and blackbox settings, respectively. These attacks serve as suitable benchmarks, allowing us to assess the performance of described approach from different perspectives, such as evasion rate and the number of queries. Similar to our model, Sparse-RS, ShadowDroid, and GenDroid generate adversarial examples by querying the target detectors.

Additionally, PiAttack is a problem-space adversarial attack that employs a similar type of transformation to generate AEs. Although PiAttack is a whitebox evasion attack, it establishes a benchmark for optimal evasion performance, facilitating the evaluation of the comparative effectiveness of other attacks with limited or zero knowledge about the targeted detectors.

We selected DREBIN, Sec-SVM, and ADE-MA as the target detectors be-cause they align with the threat models of PiAttack, Sparse-RS, and ShadowDroid. Although our model has zero knowledge about DREBIN, Sec-SVM, and ADE-MA, its evasion rates for bypassing these detectors are comparable to PiAttack, where the adversary has full knowledge of the target detectors.

Our empirical analysis shows that the reported model requires adding more features to evade DREBIN, Sec-SVM, and ADE-MA. Concretely, on average, proposed model makes 54–90 new features appear in the feature representations of the malware apps when it applies transformations to the apps for evading DREBIN, Sec-SVM, and ADE-MA, while the transformations used by PiAttack, on average, trigger 11–68 features. It is important to note that the reference attack's ability to add a smaller number of features is attributed to its complete knowledge of the details of DREBIN, Sec-SVM, and ADE-MA, while our model lacks this specific information.

The evasion rate of Sparse-RS for DREBIN and Sec-SVM demonstrates that random alterations in malware features do not necessarily result in the successful generation of AEs, even when adversaries have access to the target models' training set. Although proposed model operates solely in a blackbox setting, this attack outperforms Sparse-RS by a considerable margin for both DREBIN and Sec-SVM, i.e., 74.8% and 89.8% improvement, respectively. Moreover, GenDroid exhibits superior evasion rates compared to our model when targeting DREBIN and ADE-MA; nevertheless, its efficacy is substantially nullified when facing Sec-SVM, a resilient malware detector. Our empirical analysis also highlights the remarkable efficiency of our model in terms of the number of queries compared to other query-based attacks. Specifically, on average, our model requires only 1–7 queries to bypass DREBIN, Sec-SVM, and ADE-MA, while Sparse-RS, ShadowDroid, and GenDroid demand 2–195, 29–64, and 81–336 queries, respectively.

RQ4. Real-world effectiveness

We selected three popular antivirus engines in the Android ecosystem: Total AV, AVG, Avast. Thus, our proposed attack pattern can effectively evade all antivirus products with a few queries. Thus, the effectiveness of our model can be primarily attributed to the transformations rather than the optimization technique. This is evident from the fact that in most cases, only one query is required to generate adversarial examples.

It is relevant to mention that our model is capable to effectively deceive VirusTotal (VT) engines with an average of 73.97%. It is worth noting that the findings in this experiment validate the results observed in previous studies, such as (Ceschin et al., 2020).

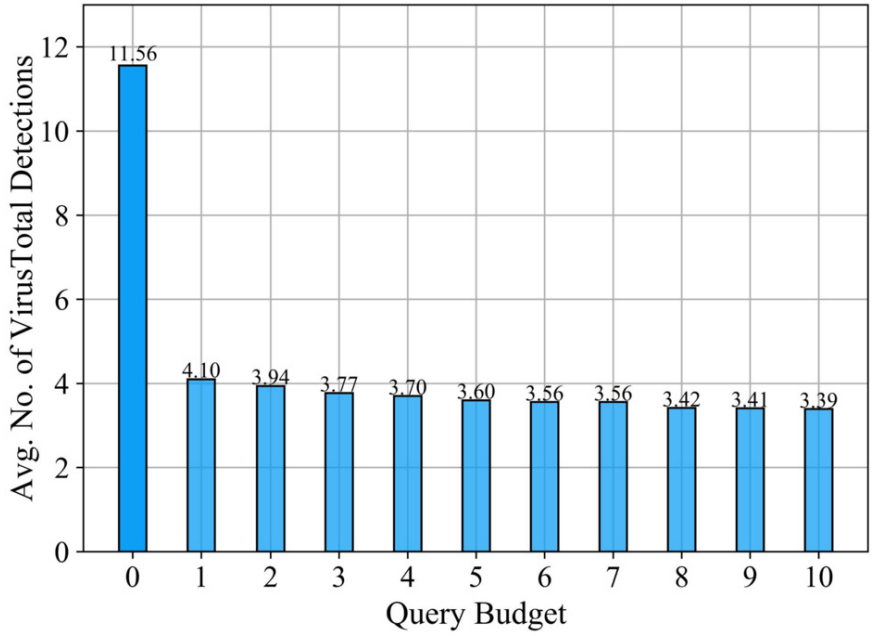

***Fig.3**. Experimental results related to RQ4*

RQ5. Real-world relevance of advanced comparative performance analyses

We evaluated the evasion rates of adversarial examples generated on a model (e.g., Sec-SVM), which works as a surrogate model, in misleading other target models (e.g., DREBIN). This is a stricter threat model that evaluates the performance of our model in cases where adversaries are not capable of querying the target detectors. Thus, as soon as our model considers a stronger surrogate model, such as Sec-SVM, the adversarial examples exhibit higher transferability.

RQ6. Real-world effectiveness of proposed model when utilizing an alternative search strategy for manipulation.

We performed an empirical analysis to evaluate the performance of our model when utilizing an alternative search strategy for manipulation. We introduced a baseline manipulation method based on genetic algorithms (GA) for use relative to our model, where the fitness function of the baseline is the same as the RS-based method.

The inclusion of RS into the algorithmic model indicates that this strategy outperforms GA. Specifically, RS not only leads to a 36.5% enhancement in (Evasion Rate)ER but also accelerates our model by ≈ 3×. These improvements are achieved with only 3 queries compared to the GA's 24 queries.

CONCLUSION

This paper reports a novel Android evasion attack in the problem space, designed to generate real-world adversarial Android malware, which is capable of evading ML-based Android malware detectors relative to a blackbox setting.
It is important to note that, unlike previous approaches, this directly operates in the problem space without initially focusing on finding feature-space perturbations. Experimental results demonstrate the effectiveness of this approach in deceiving various academic and commercial malware detectors. Therefore, the results of the experimental process suggest that this integrated approach may be used in order to further tweak the effectiveness of existing and future malware detectors.

ACKNOWLEDGMENT

This work was supported by Shota Rustaveli National Science Foundation of Georgia (SRNSFG) - CG-24-220

REFERENCES

Yousra Aafer, Wenliang Du, and Heng Yin, "DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android," in *Springer eBooks*, 2013, 86–103, https://doi.org/10.1007/978-3-319-04283-1_6.

"Can Machine/Deep Learning Classifiers Detect Zero-Day Malware With High Accuracy?," IEEE Conference Publication | IEEE Xplore, December 1, 2019, https://ieeexplore.ieee.org/document/9006514.

Cara, Fabrizio, Michele Scalas, Giorgio Giacinto, and Davide Maiorca. 2020. "On the Feasibility of Adversarial Sample Creation Using the Android System API" *Information* 11, no. 9: 433. https://doi.org/10.3390/info11090433

"DroidEye: Fortifying Security of Learning-Based Classifier Against Adversarial Android Malware Attacks," IEEE Conference Publication | IEEE Xplore, August 1, 2018, https://ieeexplore.ieee.org/document/8508284.

Francesco Croce et al., "Sparse-RS: A Versatile Framework for Query-Efficient Sparse Black-Box Adversarial Attacks," *Proceedings of the AAAI Conference on Artificial Intelligence* 36, no. 6 (June 28, 2022): 6437–45, https://doi.org/10.1609/aaai.v36i6.20595.

"Yes, Machine Learning Can Be More Secure! A Case Study on Android Malware Detection," IEEE Journals & Magazine | IEEE Xplore, August 1, 2019, https://ieeexplore.ieee.org/abstract/document/7917369.

"Malware Detection and Classification Based on N-Grams Attribute Similarity," IEEE Conference Publication | IEEE Xplore, July 1, 2017, https://ieeexplore.ieee.org/document/8005908.

"Program Slicing," IEEE Journals & Magazine | IEEE Xplore, July 1, 1984, https://ieeexplore.ieee.org/abstract/document/5010248.

"ShadowDroid: Practical Black-box Attack Against ML-based Android Malware Detection," IEEE Conference Publication | IEEE Xplore, December 1, 2021, https://ieeexplore.ieee.org/document/9763777.

Jinrong Bai, Junfeng Wang, and Guozhong Zou, "A Malware Detection Scheme Based on Mining Format Information," *The Scientific World JOURNAL* 2014 (January 1, 2014): 1–11, https://doi.org/10.1155/2014/260905.

Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, „DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket“, 2014, https://media.telefonicatech.com/telefonicatech/uploads/2021/1/4915_2014-ndss.pdf .

AndroZoo. Available online: https://androzoo.uni.lu/ , (Accessed: 7 November 2024).

VirusTotal. Available online: https://www.virustotal.com/gui/home/upload , (Accessed: 7 November 2024).

Fabrício Ceschin, Marcus Botacin, Heitor Murilo Gomes, Luiz S. Oliveira, and André Grégio. 2020. Shallow Security: on the Creation of Adversarial Variants to Evade Machine Learning-Based Malware Detectors. In Proceedings of the 3rd Reversing and Offensive-oriented Trends Symposium (ROOTS'19). Association for Computing Machinery, New York, NY, USA, Article 4, 1–9. https://doi.org/10.1145/3375894.3375898 .