Breaking the Anti-Malware: EvoAAttack based on Genetic Algorithm against Android Malware Detection Systems

Hemant Rathore[1], Praneeth B[1],
Sundaraja Sitharama Iyengar[2], and Sanjay K. Sahay[1]

[1] BITS Pilani, Department of CS & IS, Goa Campus, India
{hemantr, f20200096, ssahay}@goa.bits-pilani.ac.in
[2] Florida International University, USA
iyengar@cs.fiu.edu

Abstract. Today, android devices like smartphones, tablets, etc., have penetrated very deep into our modern society and have become an integral part of our daily lives. The widespread adoption of these devices has also garnered the immense attention of malware designers. Many recent reports suggest that existing malware detection systems cannot cope with current malware challenges and thus threaten the android ecosystem's stability and security. Therefore, researchers are now turning towards android malware detection systems based on machine and deep learning algorithms, which have shown promising results. Despite their superior performance, these systems are not immune to adversarial attacks, highlighting a research gap in this field. Therefore, we design and develop *EvoAAttack* based on a genetic algorithm to expose vulnerabilities in state-of-the-art malware detection systems. The EvoAAttack is a *targeted false-negative evasion attack* strategy for the *grey-box scenario*. The EvoAAttack aims to convert malicious android applications (by adding perturbations) into adversarial applications that can deceive detection systems. The EvoAAttack agent is designed to convert *maximum* malware into adversarial applications with *minimum* perturbations while maintaining syntactic, semantic, and behavioral integrity. We tested EvoAAttack against thirteen distinct malware detection systems based on machine and deep learning algorithms from four different categories. The EvoAAttack was able to convert an average of 97.48% of malware applications (with a maximum of five perturbations) into adversarial applications (malware variants). These adversarial applications force misclassifications and reduce the average accuracy of thirteen malware detection systems from 94.87% to 50.31%. Later we also designed a defense strategy (*defPCA*) to counter the adversarial attacks. The defPCA defense reduces the average forced misclassification rate from 97.48% to 59.98% against the same thirteen malware detection systems. Finally, we conclude that threat modeling improves both detection performance and adversarial robustness of malware detection systems.

Keywords: Adversarial Learning · Android Malware · Evasion Attack · Genetic Algorithm · Malware Detection.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_43

1 Introduction

The 21^{st} century has witnessed a massive adoption of computing and communication devices (like smartphones, tablets, etc.) in every aspect of our daily life. Smartphones are currently used by more than two-thirds of the world's population for their professional as well as personal needs [8]. All these computing devices need an operating system for their resource management. Here, android is a dominant player and holds a market share of 71% and 48% in the smartphone and tablet ecosystem [1]. These android devices store vast amounts of users' personal as well as business data. Therefore, they have become an attractive target for malware designers who want to steal data (like contacts, SMS, pictures, videos, etc.), disrupt devices, cripple services, etc. Several recent reports indicate that numerous malware and potentially unwanted applications exist in the android ecosystem [2][13]. The latest AV-ATLAS report suggests that the total number of malicious android applications and potentially unwanted applications have reached 60 million as of January 2023 [2]. It also notes that 77.8% of newly discovered malware file types were packaged as an APK, making it the most commonly used entry point for attackers. These malware attacks can execute security breaches and result in severe consequences for users' data and privacy in the android ecosystem.

Anti-virus and anti-malware software offer the first line of defense against malware attacks. However, many recent reports suggest that the existing malware detection techniques (like signature, heuristics, etc.) are ineffective against new, complex, sophisticated malware attacks [6][13]. Researchers are now turning towards android malware detection systems based on machine and deep learning algorithms, which have shown promising results [6]. Arora et al. (2019) proposed *PermPair* to identify android permission pairs for malware detection and achieved 95.44% accuracy [4]. Wozniak et al. (2020) designed a recurrent neural network model for malware threat detection and achieved more than 99% accuracy [20]. Wang et al. (2019) designed *LSCDroid* based on locally sensitive APIs [19]. They developed a random forest model that achieved more than 96% accuracy. As more and more machine and deep learning solutions are getting integrated into real-world malware detection systems, the potential for adversarial attacks on them has gained attention.

Malware designers can develop adversarial attacks to target malware detection systems based on machine and deep learning with the aim of decreasing their performance. An adversarial attack involves introducing small perturbations to a sample to force misclassification in a malware detection system. Many researchers have demonstrated vulnerabilities in various machine and deep learning models to such adversarial attacks [16][17]. Cara et al. (2020) proposed injecting system API calls in order to force misclassification in a multi-layer perceptron model [7]. Li et al. (2020) designed a generative adversarial network that forces misclassifications in detection models [12]. These studies highlight vulnerabilities of the machine and deep learning detection systems that malware designers or adversaries can potentially exploit. On the other hand, the anti-malware community can use this information to improve the adversarial robustness of malware

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_43

detection systems. The threat modeling of adversarial attacks on malware detection systems depends on the attacker's intel about the target system. It includes information about the training dataset, features, classification algorithms, and the architecture of the detection system. If the attacker possesses comprehensive knowledge about the target system, it is called a white-box scenario. Conversely, if the attacker has zero or partial knowledge about the target system, it is called a black or grey box scenario, respectively.

In this work, we propose a novel *targeted false-negative evasion attack* strategy *(EvoAAttack)* to generate malware variants that can force misclassifications in state-of-the-art malware detection systems. The EvoAAttack is designed based on a process of natural evolution (genetic algorithm) that aims to break the *Classification Robustness* of a machine and deep learning based target systems in the *grey-box scenario*. The EvoAAttack agent is designed to add minimum perturbations in malicious android applications and converts them into adversarial applications (malware variants) that can deceive detection systems. It also aims to convert maximum malware into adversarial applications by adding perturbations while ensuring their syntactic, semantic, and functional integrity. Therefore, the EvoAAttack ensures *Perturbation Measurability* to quantify the deviation of an application after perturbations, and the *Perturbation Invertibility* to enable the reconstruction of the original application after applying perturbations. The EvoAAttack investigated the adversarial robustness of thirteen distinct malware detection systems. These systems were constructed using thirteen machine and deep learning algorithms from four categories: machine learning, bagging, boosting, and neural networks. Later, we also designed and developed an adversarial defense strategy *(defPCA)* to counter the evasion attacks. We make the following contributions with this work:

1. We designed *EvoAAttack*, a novel *targeted false-negative evasion attack* strategy based on an optimized genetic algorithm to generate adversarial applications (malware variants) that can force misclassifications in next-generation malware detection systems.
2. The *EvoAAttack* exposes adversarial vulnerabilities in thirteen different malware detection systems based on four categories of classification algorithms (*machine learning*, *bagging*, *boosting*, and *neural networks*).
3. The *EvoAAttack* converted an average 97.48% of malware applications (with maximum of five perturbations) into adversarial applications (malware variants) that can force misclassifications in thirteen malware detection systems. The EvoAAttack reduced the average accuracy of thirteen malware detection systems from 94.87% to 50.31%.
4. We also designed *defPCA* defense strategy to counter the adversarial attack. The *defPCA* defense reduces the average forced misclassification rate from 97.48% to 59.98% against the same thirteen malware detection systems.

The paper is structured as follows. Section-2 explains the proposed framework for adversarial robustness, and Section-3 describes the experimental setup. Section-4 discusses the experimental results, and Section-5 explains the related work. Finally, Section-6 concludes the paper.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_43

2 Proposed Framework for Adversarial Robustness

In this section, we first describe the proposed framework that improves the adversarial robustness of android malware detection systems. It is followed by a detailed discussion on the *EvoAAttack* and *defPCA* defense strategy.

2.1 Framework Overview

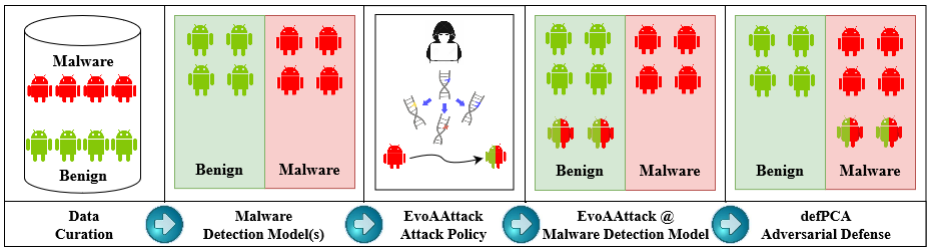

Fig. 1. Framework to improve adversarial robustness of malware detection systems.

Figure-1 illustrates the proposed five-stage framework pipeline to develop adversarially robust malware detection systems. *Stage-1* involves data curation of malware and benign android applications from authentic sources. *Stage-2* entails feature extraction from collected android applications and the construction of malware detection systems based on machine and deep learning algorithms. In *Stage-3*, we will put ourselves in the adversary's shoes and proactively design an adversarial attack strategy *(EvoAAttack)* against malware detection systems. *Stage-4* involves performing an actual *EvoAAttack* on malware detection systems developed in the second stage. The EvoAAttack aims to reduce the performance of detection systems. Then in *Stage-5*, we will develop the defense strategy *(defPCA)* to counter adversarial attacks. We employed evaluation metrics like accuracy, ROC, forced misclassification rate, etc., at various stages of the proposed pipeline. Finally, the proposed framework pipeline is expected to improve detection accuracy as well as the adversarial robustness of android malware detection systems.

2.2 Evolutionary Adversarial Attack (EvoAAttack)

Threat modeling is used to investigate the adversarial robustness of systems. Therefore we perform threat modeling of malware detection systems by stepping into adversaries' shoes and designing adversarial attacks. We propose Evolutionary Adversarial Attack (*EvoAAttack*), a *targeted false-negative evasion attack* strategy for the *grey-box scenario* against malware detection systems. The EvoAAttack aims to convert malware android applications (by adding perturbations) into adversarial malware applications that can force misclassifications

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_43

in malware detection systems. The evasion attack is designed for the *grey-box scenario* that assumes the attacker possesses knowledge about the dataset and features but has no information about the classification algorithm and system architecture. The EvoAAttack agent aims to convert maximum malware applications into adversarial applications with minimum perturbations in each application. The proposed EvoAAttack agent performs perturbations while maintaining modified android applications' syntactic, semantic, and functional integrity.

The *EvoAAttack* strategy is an optimized variant of the *Genetic Algorithm (GA)*. The GA employs probability-based methods such as random selection and mutation to generate new solutions and examine the solution space. These algorithms are capable of solving problems represented as linear systems or graphs through the use of linear algebraic techniques. These necessitate the application of calculus, including gradients and partial derivatives, in the computation of the fitness function and identification of optimal candidates. The efficacy of GA is evaluated using statistical metrics like mean and standard deviation.

2.2.1 Optimization Function: The genetic algorithm optimizes the objectives using two approaches: *Pareto Optimization* and *Weighted Sum* of fitness function based optimization. The *Pareto set* contains *Pareto-optimal* solutions, representing the best trade-offs between objectives. We used *Weighted Sum* method that involves normalizing individual objectives for comparison and using a fitness function to calculate overall fitness as a weighted sum.

$$f_{raw} = \sum_{i=1}^{O} o_i * w_i \qquad \text{with} \qquad \sum_{i=1}^{O} w_i = 1 \tag{1}$$

where f_{raw} is the raw fitness value and o_i represents the objective functions and their corresponding weight w_i. Our objective is to maximize the probability of an adversarial application to be *forcefully misclassified* as *benign*. Therefore o, w_1 are set to 1 and $o_1 = 0.5 - q_c$ where q_c is the probability of the application being *malicious*. Here, q_c is 0 when the label is *benign* and 1 when the label is *malicious*.

2.2.2 Mutation and Crossover Function: A mutation event can be modeled as a *Bernoulli trial* with a random probability p of transferring a genetic marker from an exogenous genome b to a representative specimen in the genetic repository P. The objective is to imitate the benign profiles in b so that the deception of the malware detection system C is optimized. In the domain of genetic recombination (*crossing over*), two specimens in P, p_i and p_j, undergo a partial exchange of their genetic markers (*perturbations*), X_i and X_j where the exchange mechanism is defined by the operator: $E(X_i, X_j) = X_i \cup X_j$ and $X_i \cap X_j = \phi$. This exchange is subject to a fixed restriction, such that if $||X_s|| > \theta$, where $||X_s||$ represents the magnitude of genetic markers in a specimen, a process of genetic rejuvenation R must be initiated (also known as *hard restart*).

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_43

The strength of each application in the gene pool is evaluated using the *fitness function*. The fitness of each application is calculated after every generation, and the algorithm terminates upon reaching the first sample with *positive fitness*. The trace of the evading malware application is finally stored in the *trace file*. The application is marked as not evaded if the generation limit is reached before finding an evading/adversarial application. The EvoAAttack operates as follows:

1. The *EvoAAttack* commences by setting up the environment, where the attack parameters are registered, and benign dataset B along with adversarial perturbations from the past *Trace* are stored in memory. Given a malicious application m, the attack iterates through B to find the closest matching benign application b based on the *Euclidean* $L2$ norm. This sample is used as the external genome to transfer perturbations to the target application.

$$||x|| = \sqrt{x_1^2 + x_2^2 + ... + x_n^2} \tag{2}$$

   Here x is the difference between b and m and x_i represent the dimensions.
2. The next step involves populating the gene pool where a significant portion of the maximum pool capacity is filled with copies of the target sample m to which perturbations from the *trace file* have been added. Duplicates of the target sample take up the remaining slots. With every successive generation, EvoAAttack performs *crossover* and *mutation* operations on the gene pool.

Algorithm 1 shows the *EvoAAttack* policy to convert malware applications into adversarial applications. The algorithm's input is the set of benign B and malicious M applications and other hyperparameters. A target malware application m is selected from the set malware dataset M to convert it into an adversarial application. The population P is initialized with copies modified with respect to the data present in trace histories T or plain repetitions of m. Crossing over and mutations are executed with each evolving generation. The fitness value of each population member p is calculated, and the process stops with the first positive value after recording its perturbations in the trace T. The target is labeled as not evaded if the generation limit is reached before finding an adversarial application.

2.3 Adversarial Defense (defPCA)

We designed a novel defense strategy (*defPCA*) to improve the robustness of the malware detection systems. The approach involves reducing the dimensionality of the feature space using a technique known as *principal component analysis*. Feature reduction has been shown to help increase the generalization in various machine and deep learning systems.

The *defPCA* strategy begins by centering the data, where the mean of each android permission (refer Section 3.1) is subtracted from each android application. Let D be mean-centered dataset matrix. The covariance matrix of centered data is calculated to measure the linear relationship between each pair of permissions $VarCov = \frac{D.D^T}{n-1}$, where n is the number of instances in the dataset.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_43

Algorithm 1 Evolutionary Adversarial Attack (EvoAAttack) Strategy.

Input:
B: set of benign applications
M: set of target malware applications
C: target android malware detection model
MAX_POP_SIZE: maximum size of population
MAX_ITER: maximum number of generations
MUT_PROB: probability of mutation
Output:
M′: set of adversarial malware applications and perturbations

1 : $T = []$ *//Traces to store perturbations*
2 : Select a sample m from M without repetition, if empty then GOTO Step 7
3 : $P = []$ *//Population*
4 : Initialize POPULATION
4.1 : If T is not empty
4.1.1 : Generate new sample t for each trace and add t to P
4.1.2 : While $len(P) < MAX\_POP\_SIZE$
4.1.2.1 : Add m to P
5 : while $iteration < MAX\_ITER$
5.1 : MUTATION
5.1.1 : Each sample p is selected with probability MUT_PROB
5.1.2 : The closest benign sample b (from B) to p is selected
5.1.3 : A random feature is transplanted to b from p
5.2 : CROSSING-OVER
5.2.1 : Select a pair of samples p_1 and p_2 from P
5.2.2 : Exchange a random set of features between them
5.3 : FITNESS-TEST
5.3.1 : For each sample p in P
5.3.1.1 : Query label from C and calculate malicious probability q_c
5.3.1.2 : If $0.5 - q_c > 0$
5.3.1.2 : Add p to M'
5.3.1.2 : Store trace of p in T
5.3.1.2 : GOTO step 2
6 : Mark m as NOT_EVADED and GOTO Step 2
7 : return M' and T

Let A be a square matrix of size $m \times n$. The eigenvector v of A is a non-zero vector that satisfies the following equation: $Av = \lambda v$, where λ is the eigenvalue corresponding to the eigenvector v. The characteristic equation of the matrix A is given by $det(A - \lambda I) = 0$, where I is the identity matrix and det is the determinant operator. The eigenvalue decomposition of the matrix A is then given by: $A = Q\Lambda Q^{-1}$, where Λ is a diagonal matrix containing the eigenvalues of A and Q is the eigenvector matrix of A. Eigen-decomposition is performed on $VarCov$ to diagonalize it as $VarCov = V(\lambda)V^T$, where V is the orthonormal matrix containing the eigen-permissions of $VarCov$ in decreasing order of importance.

A lower dimensional representation of the data can be obtained by retaining only the top k eigen-permissions. The optimal value for k was set as 8 based on experiments. It was determined by evaluating the average accuracy of three malware detection systems ($RF + LR + HGB$) before and after subjecting them to the *EvoAAttack*. Finally, the centered data is transformed to the lower dimensional space using the principal component matrix, resulting in a reduced representation of the data: $D' = D \cdot V'$, where V' is the eigen-permission matrix with k leading columns and T is the transpose of the matrix. The malware detection systems are then trained on the reduced feature space.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_43

3 Experimental Setup

This section presents the experimental setup details, including data curation (malware and benign applications), the feature extraction process, classification algorithms, and evaluation metrics used in this work.

3.1 Data Curation and Feature Extraction

We conducted all the experiments using a large dataset collected from authentic sources. The malware applications were collected from benchmarked *Drebin Dataset* proposed by Arp et al. [5], and benign applications were downloaded from *Google Play Store* [3]. The dataset contains 11,281 android applications of which 5,560 were malware, and 5,721 were benign. The malware dataset contains malicious applications from over twenty families, including *DroidKungFu*, *Opfake*, *BaseBridge*, etc. On the other hand, all the benign applications were validated using *VirusTotal*. Later, we disassemble each android application using *Apktool* to perform static analysis on them. We developed a parser that scans through each disassembled android application to extract its *android permission* usage and develop the feature vector. The *rows* in the *feature vector* represent *android applications* and the *columns* represent their *permission usage*.

3.2 Classification Algorithms

We examine the adversarial robustness of thirteen distinct malware detection systems based on various classification algorithms from four different categories. Table-1 lists thirteen distinct classification algorithms and their corresponding category. The detailed design and implementation of these malware detection systems are well explained in another paper [15].

Table 1. Classification algorithms and their categories for constructing malware detection models.

Category	Classification Algorithm(s)	Abbreviation
Machine Learning	Decision Trees	DT
	Support Vector Machine	SVM
	Logistic Regression	LR
Bagging based Learning	Random Forest	RF
	Extra Trees	ET
	Bagged Decision Trees	BagDT
Boosting based Learning	Adaptive Boosting	AB
	Gradient Boosting	GB
	Histogram-based Gradient Boosting	HGB
Neural Network (NN) based Learning	NN with 1 hidden Layer	NN1L
	NN with 3 hidden Layers	NN3L
	NN with 5 hidden Layers	NN5L
	Long Short Term Memory	LSTM

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_43

3.3 Platform

All the experiments were conducted in the *Google Colaboratory* environment. The runtime had a storage limit of 70 GB and a RAM limit of 12 GB. The CPU was an Intel Xeon with 1 core and 2 threads running at 2.2 GHz. *Python* was used as the main programming language. The principal libraries used include *scikit-learn*, *TensorFlow*, *Keras*, and *NumPy*. *Pandas* was used to handle CSV files, and all graphs were generated using *Matplotlib* and *Google Sheets*.

3.4 Other Parameters

The *population size* during *EvoAAttack* is the number of samples actively participating in the genetic evolution and was set to 32. The *generation cap* was determined to be 100, based on empirical verification that the chances of finding an evasive sample (adversarial application) were slim after many generations. The *trace size* or the number of trace histories stored in memory was set to 24. The *mutation probability* was set to 0.8. The *crossover rate* was set to 0.2. The *maximum number of perturbations* allowed for each sample in the gene pool was set to 5.

3.5 Evaluation Metrics

The following evaluation metrics were used to measure the detection performance and adversarial robustness of malware detection systems in our work:

- **True Positive (TP)** is the number of malware applications correctly classified as malicious by the malware detection system.
- **True Negative (TN)** is the number of benign applications correctly classified as benign/good by the malware detection system.
- **False Positive (FP)** is the number of benign applications incorrectly or wrongly classified as malicious by the malware detection system.
- **False Negative (FN)** is the number of malware applications wrongly classified as benign by the malware detection system.
- **Accuracy** is the percentage ratio of correctly classified applications to the total number of classifications performed by the malware detection system.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \times 100 \tag{3}$$

- **Receiver Operating Characteristic (ROC)** is the degree of separability between malware and benign classes predicted by malware detection system.
- **Forced Misclassification Rate (FMR)** is the percentage of malware applications successfully converted into adversarial malware applications by the adversarial attack agent.

$$\text{Forced Misclassification Rate} = \frac{\#\text{ of Adversarial Apps} - FN_{baseline}}{\#\text{ of Malware Apps} - FN_{baseline}} \times 100 \tag{4}$$

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_43

4 Experimental Results

This section explains the detailed experimental results achieved by *EvoAAttack* and *defPCA* for thirteen distinct malware detection systems.

4.1 Baseline Android Malware Detection Systems

The ***Stage-1*** of the proposed framework is data curation and feature extraction as discussed in Section 3.1. The ***Stage-2*** focuses on the construction of thirteen distinct android malware detection systems based on data gathered in *Stage-1*. These systems were based on thirteen distinct classification algorithms from four different categories (machine learning, bagging, boosting, and neural network) (refer Table-1). The idea is to investigate the adversarial robustness of malware detection systems based on different classification algorithms from various categories. The performance evaluation of these detection systems was evaluated using accuracy, ROC, F1 score, precision, recall, etc.

Table-2 shows the performance of thirteen baseline android malware detection systems. The highest accuracy was achieved by the RF model based malware detection system (97.48%) followed by the ET model (97.47%). On the other hand, the lowest accuracy was attained by the AB model (90.70%). The thirteen detection systems achieved an average accuracy of 94.87%. Similarly, the highest and lowest ROC was achieved by the RF model (0.977) and AB model (0.91). The thirteen systems achieved an average ROC, precision and recall of 0.95, 0.97 and 0.93, respectively. The bagging based learning models performed best with an average accuracy of 97.41%, while machine learning models performed the worst with an average accuracy of 91.91%.

Table 2. Performance of thirteen distinct android malware detection systems based on various classifications algorithms from four different categories.

Classification Algorithm(s)	Accuracy (%)	ROC Score	F1 Score	Precision	Recall
DT	97.46%	0.97	0.97	0.99	0.96
SVM	92.03%	0.92	0.92	0.95	0.89
LR	91.79%	0.92	0.91	0.94	0.89
RF	97.48%	0.97	0.97	0.99	0.96
ET	97.47%	0.97	0.97	0.99	0.96
BagDT	97.29%	0.97	0.97	0.99	0.96
AB	90.70%	0.91	0.90	0.93	0.88
GB	91.92%	0.92	0.92	0.94	0.89
HGB	95.83%	0.96	0.96	0.98	0.94
NN1L	92.03%	0.92	0.92	0.94	0.90
NN3L	96.14%	0.96	0.96	0.98	0.94
NN5L	96.04%	0.96	0.96	0.98	0.94
LSTM	97.14%	0.97	0.97	0.99	0.95

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_43

4.2 EvoAAttack against Malware Detection Systems

The ***Stage-3*** in the proposed framework is to investigate the adversarial robustness of all the above thirteen baseline malware detection systems. We designed *EvoAAttack*, a genetic algorithm based *targeted false-negative evasion attack* for the *grey-box scenario*. The fundamental idea is to emulate the process of natural evolution into the creation of new malware variants that are much deadlier and can easily deceive state-of-the-art malware detection systems. The EvoAAttack agent aims to convert maximum malware applications (by adding perturbations) into adversarial applications that force massive misclassifications in detection systems. The performance of adversarial attacks against malware detection systems was measured using forced misclassification rate, accuracy drop, etc.

4.2.1 Forced Misclassification Rate @EvoAAttack: The ***Stage-4*** of the proposed framework is to perform actual *EvoAAttack* against thirteen baseline malware detection systems constructed in Section 4.1. Figure-2 shows the performance of EvoAAttack, where the *y-axis* represents the forced misclassification rate, and the *x-axis* represents the maximum number of perturbations performed by EvoAAttack. The EvoAAttack agent initially adds only one perturbation and gradually increases it to a maximum of five. The EvoAAttack with just one perturbation achieved an average forced misclassification rate of 66.81% against thirteen baseline android malware detection systems. The highest forced misclassification rate (with a maximum of one perturbation) was achieved against the AB model based malware detection system (99.94%), while the lowest was attained against the RF model (16.02%). The EvoAAttack with a maximum of five perturbations achieved an average forced misclassification rate of 97.48% against thirteen detection systems with a 100% forced misclassification rate against six malware detection systems. Furthermore, RF and ET models exhibit the highest resistance to the evasion attack with FMR of 82.92% and 86.46%, respectively.

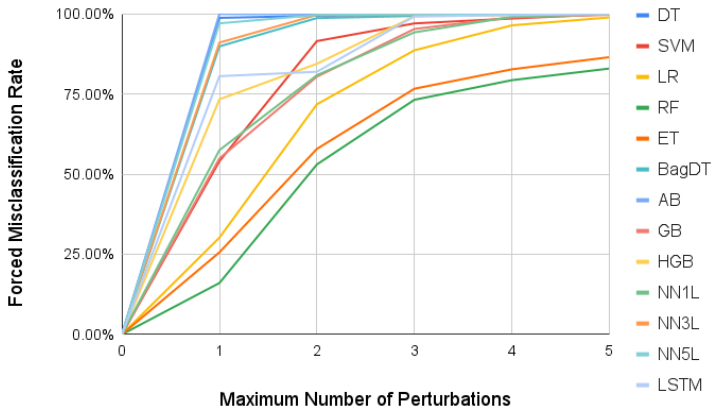

Fig. 2. EvoAAttack against different android malware detection systems.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_43

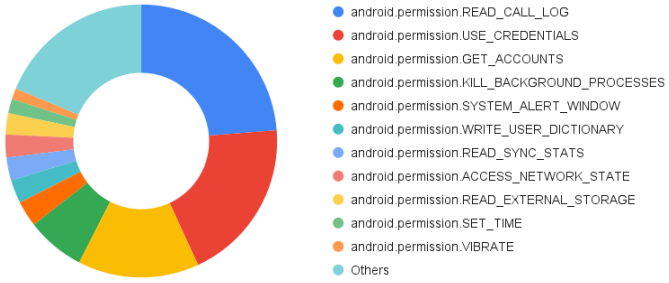

Fig. 3. Top 10 most frequently perturbed android permissions by EvoAAttack agent.

4.2.2 Vulnerable Android Permissions: Figure-3 presents data distribution of the most frequently perturbed android permissions by the EvoAAttack agent against thirteen malware detection systems. The pie chart clearly shows that few android permissions were perturbed majority of the time by EvoAAttack agent to force misclassification in malware detection systems. The top three perturbed android permissions were android.permission.READ_CALL_LOG, android.permission.USE_CREDENTIALS, and android.permission.GET_ACCOUNTS, which together accounted for over 55% of all perturbations. This finding suggests that malicious actors could also potentially use these android permissions to evade state-of-the-art malware detection systems.

4.3 defPCA Defense Strategy

The ***Stage-5*** of the proposed framework is to develop a defense strategy for malware detection systems to counter adversarial attacks. We propose *defPCA* based on principal component analysis that improves the generalizability and robustness of machine and deep learning systems.

Figure-4 shows the overall performance of various malware detection systems at different stages of the proposed framework. The *blue bars* represent the accuracy of thirteen different baseline malware detection systems, whereas the *red bars* represent the accuracy of detection systems after *EvoAAttack*. The *yellow bars* represent the accuracy of detection systems after implementing *defPCA* defense strategy, and the *green bars* represent *EvoAAttack on defPCA* based malware detection systems. The thirteen baseline malware detection systems (*blue bars*) achieved accuracy between 90.70% to 97.46%. Then, we performed threat modeling and designed EvoAAttack against detection systems. The EvoAAttack forced a massive number of misclassifications that drastically reduced accuracy (*red bars*) in all thirteen malware detection systems.

4.3.1 Detection Performance @ defPCA Systems: The *yellow bars* in Figure-4 represent the accuracy of different *defPCA* based malware detection systems. The thirteen defPCA systems achieved an average accuracy of 92.31%. Here, the RF model based detection system accomplished the highest accuracy of 97.46%, whereas the LR model attained the lowest accuracy of 87.05%. On the other hand, the thirteen detection systems achieved an average ROC of 0.92.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_43

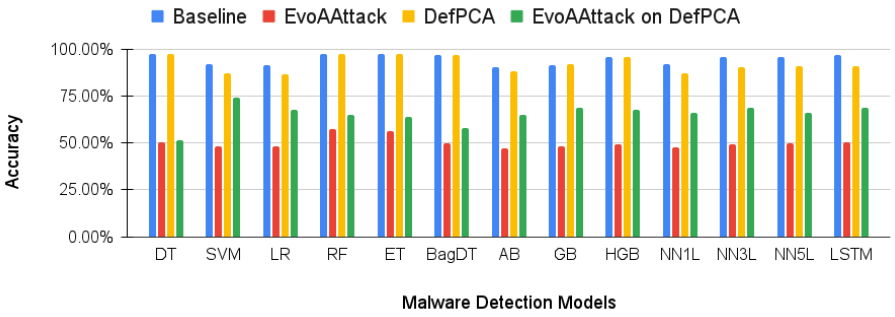

Fig. 4. Overall detection performance and robustness of malware detection systems.

4.3.2 Adversarial Robustness @ defPCA Systems: Now, we again performed EvoAAttack on the defPCA malware detection systems to investigate their adversarial robustness. The *green bars* in Figure-4 show the accuracy of defPCA systems after EvoAAttack. The thirteen defPCA systems after EvoAAttack achieved an average accuracy of 65.65%, which is a 15.34% improvement without any defense strategy. The LR and SVM models with defPCA showed the biggest improvement at 26.09% and 19.74%, respectively. On the other hand, DT models with defPCA showed the least improvement of 1.25%.

5 Related Work

Table-3 lists the existing literature on adversarial attacks on malware detection models and compares them with our proposed approach. Grosse et al. (2016) used the computation of forward derivatives to create adversarial samples against neural network models and attained a misclassification rate of 85% [10]. Though it provides a good stepping stone, the lack of generalization is quite a vulnerability in the work. Hu and Tan (2017) developed a GAN-based attack model with a substitute detector to mimic the target classifier, making it a white box attack [11]. Taheri et al. (2020)[18], and Rathore et al. (2021)[14] also proposed white box strategies to fool different machine and deep learning models. However, this falls short when it comes to real-world applications, as the working mechanisms of most malware detection systems are hidden from the adversary. Fang et al. (2019) proposed grey-box approaches based on deep reinforcement learning that use limited knowledge about the target system, obtaining a success rate of 19.13% and 75%, with a maximum of 100 perturbations [9]. However, it failed to achieve a significant drop in accuracy after the attack. Taheri et al. (2020) performed random perturbations on the ranked feature space of the dataset to achieve an accuracy drop of just 12% [18]. In contrast, Rathore et al. (2021) attained a forced misclassification rate of 44.28% using a single policy attack based on q-learning [14]. They failed to perform in-depth feature vulnerabilities analysis or suggest suitable countermeasures to enforce robustness.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_43

Table 3. Comparison of our proposed work with existing literature.

Paper	Attack Scenario	Max # of Perturbations	Forced Misclassification Rate	# of Models Analyzed	Vulnerability Analysis	Robustness Analysis
Grosse et al. (2016) [10]	White Box	20	40.97%-84.05%	Only DNNs	No	No
Hu and Tan (2017)[11]	White Box	All	99%	6	No	No
Fang et al. (2019) [9]	Grey Box	80	46.56%	1	No	No
Cara et al. (2020) [7]	Grey Box	100	80%	1	No	No
Li et al. (2020) [12]	Grey Box	100	85%	9	No	No
Rathore et al. (2021) [14]	White Box	5	44.28%	8	No	Yes
Proposed (EvoAAttack & defPCA)	**Grey Box**	**5**	**97.49%**	**13**	**Yes**	**Yes**

There are many limitations in the existing literature that we have addressed in this work. Authors have assumed a white box scenario for threat modeling, which is impractical since most details about the target system are generally unknown. Also, authors have achieved very low forced misclassification rates even with a very high number of perturbations. The perturbations should be minimized to decrease the overall cost of generating adversarial samples. Detailed analysis of perturbations is also not discussed in the existing literature. The defense strategy is incomplete without investigating its adversarial robustness, and it is hardly discussed in the literature.

6 Conclusion

Android malware have exploded in the last few years and have become a real threat to smartphones and tablets. Researchers propose next-gen malware detection systems based on machine and deep learning. These systems have demonstrated encouraging results but might be at risk against adversarial attacks.

In this work, we performed threat modeling of malware detection systems to explore their vulnerabilities. We designed and developed *EvoAAttack*, a novel *targeted false-negative evasion attack* against state-of-the-art malware detection systems for *grey-box scenario*. The EvoAAttack agent converted an average of 97.49% of malware applications into adversarial applications (malware variants) that forced massive misclassifications in thirteen malware detection systems. The EvoAAttack also reduced the average accuracy of thirteen malware detection systems from 94.87% to 50.31%. We also published the list of ten android permissions that adversaries might use to generate more malware variants. Later, we develop a defense strategy (*defPCA*) to counter the attack on malware detection systems. The defPCA reduced the average forced misclassification rate from 97.48% to 59.98% against the same thirteen malware detection systems. Finally, we conclude that threat modeling improves both detection performance and adversarial robustness of malware detection systems.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_43

References

1. Android - Statistics & Facts. Available: https://www.statista.com/topics/876/android/ (2023)
2. AV-TEST. Available: https://portal.av-atlas.org/malware/statistics (2023)
3. Google Play Store. Available: https://play.google.com/store/ (2023)
4. Arora, A., Peddoju, S.K., Conti, M.: Permpair: Android malware detection using permission pairs. IEEE Transactions on Information Forensics and Security **15**, 1968–1982 (2019)
5. Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K.: Drebin: Effective and explainable detection of android malware in your pocket. In: Network and Distributed System Security Symposium (NDSS). vol. 14, pp. 23–26 (2014)
6. Bhat, P., Dutta, K.: A survey on various threats and current state of security in android platform. ACM Computing Surveys (CSUR) **52**(1), 1–35 (2019)
7. Cara, F., Scalas, M., Giacinto, G., Maiorca, D.: On the feasibility of adversarial sample creation using the android system API. Information **11**(9), 433 (2020)
8. Digital 2023: Global Overview Report: Simon Kemp (Hootsuite). Available: https://datareportal.com/reports/digital-2023-global-overview-report (2023)
9. Fang, Z., Wang, J., Li, B., Wu, S., Zhou, Y., Huang, H.: Evading anti-malware engines with deep reinforcement learning. IEEE Access **7**, 48867–48879 (2019)
10. Grosse, K., Papernot, N., Manoharan, P., Backes, M., McDaniel, P.: Adversarial perturbations against deep neural networks for malware classification. arXiv preprint arXiv:1606.04435 (2016)
11. Hu, W., Tan, Y.: Generating adversarial malware examples for black-box attacks based on GAN. In: DMBDA. pp. 409–423. Springer (2023)
12. Li, H., Zhou, S., Yuan, W., Li, J., Leung, H.: Adversarial-example attacks toward android malware detection system. IEEE Systems Journal **14**(1), 653–656 (2019)
13. McAfee 2023 Consumer Mobile Threat Report: Available: https://www.mcafee.com/blogs/internet-security/mcafee-2023-consumer-mobile-threat-report/ (2023)
14. Rathore, H., Sahay, S.K., Nikam, P., Sewak, M.: Robust android malware detection system against adversarial attacks using q-learning. Information Systems Frontiers **23**, 867–882 (2021)
15. Rathore, H., Sahay, S.K., Rajvanshi, R., Sewak, M.: Identification of significant permissions for efficient android malware detection. In: 11th EAI BROADNETS. pp. 33–52. Springer (2021)
16. Rathore, H., Samavedhi, A., Sahay, S.K., Sewak, M.: Robust malware detection models: learning from adversarial attacks and defenses. Forensic Science International: Digital Investigation **37**, 301183 (2021)
17. Sewak, M., Sahay, S.K., Rathore, H.: DRLDO: A Novel DRL based De-obfuscation System for Defence Against Metamorphic Malware. Defence Science Journal **71**(1), 55–65 (2021)
18. Taheri, R., Javidan, R., Shojafar, M., Vinod, P., Conti, M.: Can machine learning model with static features be fooled: an adversarial machine learning approach. Cluster Computing **23**, 3233–3253 (2020)
19. Wang, W., Wei, J., Zhang, S., Luo, X.: LSCDroid: Malware detection based on local sensitive API invocation sequences. IEEE Transactions on Reliability **69**(1), 174–187 (2019)
20. Woźniak, M., Siłka, J., Wieczorek, M., Alrashoud, M.: Recurrent neural network model for iot and networking malware threat detection. IEEE Transactions on Industrial Informatics **17**(8), 5583–5594 (2020)

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_43