EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

Are Malware Detection Classifiers Adversarially Vulnerable to Actor-Critic based Evasion Attacks?

Hemant Rathore[1,*], Sujay C Sharma[1], Sanjay K. Sahay[1], Mohit Sewak[2]

[1]Department of CS & IS, Goa Campus, BITS Pilani, India
[2]Security & Compliance Research, Microsoft R & D, India

Abstract

Android devices like smartphones and tablets have become immensely popular and are an integral part of our daily lives. However, it has also attracted malware developers to design android malware which have grown aggressively in the last few years. Research shows that machine learning, ensemble, and deep learning models can successfully be used to detect android malware. However, the robustness of these models against well-crafted adversarial samples is not well investigated. Therefore, we first stepped into the adversaries' shoes and proposed the ACE attack that adds limited perturbations in malicious applications such that they are forcefully misclassified as benign and remain undetected by different malware detection models. The ACE agent is designed based on an actor-critic architecture that uses reinforcement learning to add perturbations (maximum ten) while maintaining the structural and functional integrity of the adversarial malicious applications. The proposed attack is validated against twenty-two different malware detection models based on two feature sets and eleven different classification algorithms. The ACE attack accomplished an average fooling rate (with maximum of ten perturbations) of 46.63% across eleven permission based malware detection models and 95.31% across eleven intent based detection models. The attack forced a massive number of misclassifications that led to an average accuracy drop of 18.07% and 36.62% in the above permission and intent based malware detection models. Later we also design a defense mechanism using the adversarial retraining strategy, which uses adversarial malware samples with correct class labels to retrain the models. The defense mechanism improves the average accuracy by 24.88% and 76.51% for the eleven permission and eleven intent based malware detection models. In conclusion, we found that malware detection models based on machine learning, ensemble, and deep learning perform poorly against adversarial samples. Thus malware detection models should be investigated for vulnerabilities and mitigated to enhance their overall forensic knowledge and adversarial robustness.

Received on 16 March 2022; accepted on 28 May 2022; published on 31 May 2022

Keywords: Adversarial Robustness, Android, Deep Neural Network, Malware Analysis and Detection, Machine Learning

doi:10.4108/eai.31-5-2022.174087

1. Introduction

Android popularity has grown exponentially in the last decade. There are currently more than 2.5 billion active android users, which is more than 35% of the world's population [1]. The android operating system holds 71.81% and 43.62% market share in smartphone and tablet market segments, respectively [2]. The wide acceptance of android is due to its openness and extensibility with millions of applications in the ecosystem. However, android devices (smartphones, tablets, etc.) hold a vast amount of personal user data, which is very attractive for malware developers. According to AV-TEST, there are more than twenty five-million malware and ten million potentially unwanted applications in the android ecosystem [3]. The principal protection against these malware are developed by anti-virus companies (like Bitdefender, Norton, McAfee, Trend Micro, etc.), forensic investigators and the anti-malware research community. The current anti-virus systems consist of signature, and behavior based malware detection engines [4]. However, the literature

*Corresponding author. Email: hemantr@goa.bits-pilani.ac.in

suggests that these engines are struggling to cope with the ever-increasing number and sophistication of malware attacks in the android ecosystem [5][6].

Recently anti-malware researchers have started examining machine learning, ensemble, and deep learning classifiers to construct new age malware detection engines, which will complement the existing anti-virus engines [4][7]. The development of these engines is a three-step process (1) Data Collection, (2) Feature engineering (3) Detection model construction using classification/clustering algorithms. Li et al. used the Drebin dataset containing the malicious applications and extracted android permissions. They performed three-level pruning on the feature vector using association rule mining and used a support vector machine to construct the android malware detection system [8]. Rathore et al. also used the Drebin dataset and extracted permissions using static analysis of android applications. They performed feature reduction and used many machine learning, ensemble, and deep learning classifiers to construct malware detection models [9]. These next-generation malware detection models have shown promising results in detecting new & old malware effectively and efficiently [6][7][10].

The popularity of machine learning, ensemble, and deep learning classifiers has led to their adoption in various domains like object recognition, NLP, etc. [11] [12]. However, the literature suggests that these classifiers might be vulnerable to adversarial samples [13][14][15][16]. Goodfellow et al. generated adversarial samples by intentionally adding a small number of perturbations to fool the image classification model [17]. Chen et al. added perturbations in images that are imperceptible to the human eye but could fool an image classification system in the white box & grey box scenarios [18]. He et al. found that deep convolutional neural networks used for biomedical image segmentation are also vulnerable to adversarial samples [19]. Similar adversarial samples can be generated using various forensic investigations to fool the malware detection models that will jeopardize the complete android security ecosystem.

In this work, we aim to investigate the adversarial robustness of different android malware detection models. Firstly we constructed twenty-two different malware detection models based on two feature sets and eleven distinct classifiers based on machine learning, ensemble, and deep learning. Then we stepped into the adversaries' shoes to design ***ACE attack – Actor-Critic based Evasion attack*** and perform forensic investigation to study the adversarial robustness of all the above malware detection models. The ACE attack aims to convert malicious applications into adversarial samples such that they are forcefully misclassified as benign and thus remain undetected by the malware detection engine. The perturbations are added in malicious applications while maintaining the structural and functional integrity of the application. The attack is also designed to add minimum perturbations in each malicious application and convert the maximum number of malicious applications into adversarial samples. The ACE attack is designed for real-world situations (grey-box scenario) where the adversaries have knowledge about the dataset and feature set but no information about classification algorithm(s) and its parameters. The ACE attack will exploit vulnerabilities in detection models to generate adversarial samples. Later we analyze these vulnerabilities and propose adversarial retraining as a defense to counter the attack. Finally, we made the following contributions to enhance the adversarial robustness of malware detection models.

- We proposed the ACE attack based on actor-critic architecture for grey box scenario against twenty-two different malware detection models. The models are developed using two different feature sets along with eleven classification algorithms derived from three distinct categories.
- The proposed ACE attack achieves an average fooling rate (with maximum of 10 perturbations) of 46.63% across eleven permission based malware detection models and 95.31% across eleven intent based detection models. The proposed attack leads to an average accuracy drop of 18.07% and 36.62% in the above permission and intent based malware detection models.
- We also designed an adversarial defense strategy to counter the attack and improved the average accuracy by 24.88% and 76.51% for the eleven permission and eleven intent based malware detection models. The defense also improves the average AUC by 82.39% and class probabilities by 14.29% for the eleven intent based detection models.

The rest of this paper is organized as follows: Section 2 will discuss the proposed framework to enhance malware detection models' robustness. Section 3 and Section 4 will discuss the experimental setup and results, respectively. Finally, Section 5 will conclude the paper.

2. Overview and Proposed Framework

This section first explains the overview of the proposed framework to enhance adversarial robustness in malware detection models, followed by a discussion on threat modeling. Later in the section, we will discuss the proposed ACE attack strategy followed by adversarial retraining as the defense strategy.

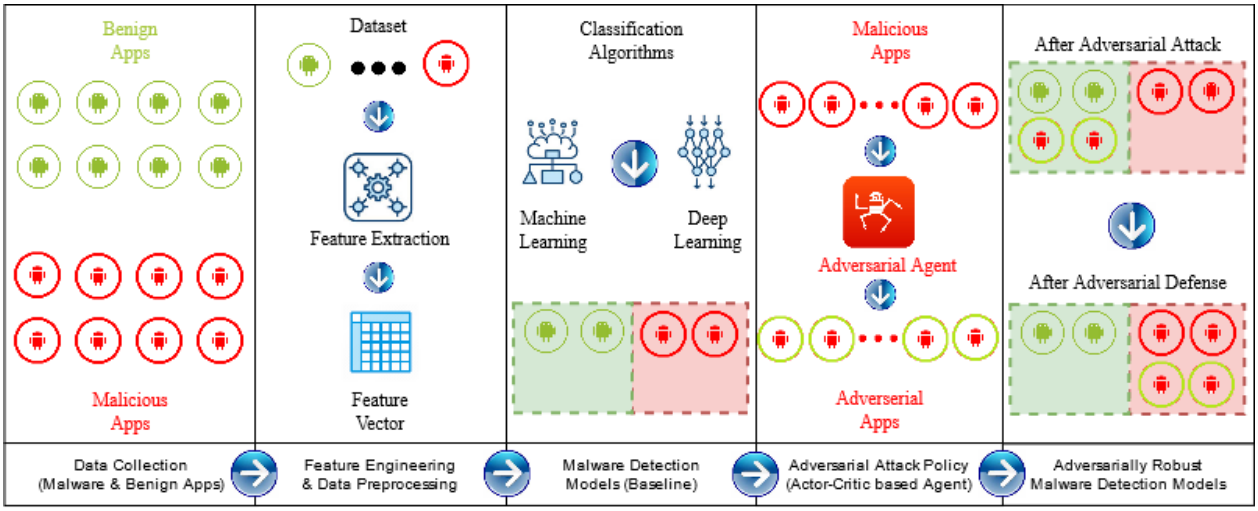


Figure 1. The proposed framework design for enhancing adversarial robustness of android malware detection model(s)

.

2.1. Overview of Proposed Framework Design

We propose the following framework design (refer to Figure 1) to construct adversarially robust android malware detection systems. The proposed design can be divided into the following five sequential steps:

Data Collection. The first step is gathering of android applications from various legitimate sources to develop the dataset. The dataset is expected to contain a roughly equal number of malicious and benign apps for the development of malware detection models designed as a classification problem.

Feature Engineering and Data Preprocessing. The second step is static or dynamic analysis of android applications to extract various features. We have extracted two different features, namely android permission and intent based on static analysis of android applications. These extracted features are then mapped to feature vector(s) for further analysis. Later exploratory data analysis of feature vector(s) is performed to understand its basic characteristics.

Malware Detection Models (Baseline). The third step is to construct various malware detection models using basic machine learning, ensemble, and deep neural network based classification algorithms. The performance of these malware detection models can be compared using accuracy, AUC score, class probabilities, etc. We have constructed twenty-two different android malware detection models based on two feature sets and eleven different classification algorithms.

Adversarial Attack Policy. The fourth step is to compare the adversarial robustness of different malware detection models constructed in Step-3. Here we stepped into the adversaries' shoes to propose the ACE attack that uses the actor-critic agent based on reinforcement learning against different malware detection models. The agent adds perturbations in malware applications such that they are forcefully misclassified as benign by detection models. The agent aims to add minimum perturbations in each malicious application and convert the maximum number of malicious applications into adversarial samples.

Adversarially Robust Malware Detection Models. The fifth step is to perform actual evasion attacks (ACE attack in our work) using adversarial samples on different malware detection models. The adversarial attack's performance can be measured using metrics like fooling rate, accuracy, etc. The next step is to explore and analyze the vulnerabilities exposed by the attack in the detection models. Lastly, adversarial defense strategies are designed to counter the attack and enhance the overall adversarial robustness of the malware detection models.

2.2. Threat Modeling @ Malware Detection Models

Threat modeling is a process in the field of security where potential threats are simulated, identified, and enumerated in a controlled environment [11][20]. It is followed by designing a thorough mitigation strategy to deal with the threat scenario. The ability to simulate real-world threat scenarios and develop mitigation strategies for the same makes threat modeling a vital exercise for any system. Threat modeling is a general practice applicable in many domains. However, the exercise done as a part of the proposed experiment will focus on the taxonomy and classification criteria introduced and widely adopted in the field of adversarial machine learning. The threat modeling focuses on evaluating three different aspects of the proposed attack strategy:

Attacker's Goal. The attacker's goal is defined based on the type of security violations induced in the target system and the attack's specificity. Security violations can disrupt the integrity, availability, and confidentiality of the target system. We propose the ACE attack strategy that disrupts the target system's integrity as malware applications are converted into adversarial samples and are misclassified as benign, thus remaining undetected by the malware detection engine. The system's availability and confidentiality can lead to denial of service or release of confidential information of the target system, which is currently an open research problem. The specificity of the attack can either be targeted or indiscriminate. The proposed ACE attack strategy is an indiscriminate attack since it targets a broad class of samples (all malware applications) rather than being restricted to a specific type of malware family.

Attacker's Knowledge. The attacker's knowledge is defined based on the information available to the attacker regarding the different components of the target system such as dataset, feature set, classification algorithm, classifier parameters, and classifier feedback. The adversarial attack can be broadly classified into one of the following three categories based on the level of information available to the attacker about the above components.

- ***Black Box Scenario*** assumes the attacker has no knowledge about any component of target system.
- ***Gray Box Scenario*** assumes the attacker has knowledge about only some selected components of the target system.
- ***White Box Scenario*** assumes the attacker has complete knowledge about all the components of the target system.

The proposed ACE attack strategy requires access to the dataset, feature set, and classifier feedback. However, it does not need any information regarding the classification algorithm or the parameters used to construct malware detection models, which qualifies the *ACE attack* as a gray box scenario attack.

Attacker's Capability. The attacker's capability is defined in terms of the phase of the attack and data manipulation constraints. The two popular adversarial attacks based on the attacker's capability are:

- ***Evasion Attack*** is designed with the goal to force misclassifications by the classifier. It can be achieved by performing intelligent feature alterations in the test data to convert them into adversarial samples.
- ***Poison Attack*** is designed to reduce the classifier capabilities by adding poisoned training data and indiscriminately trying to increase the classification error.

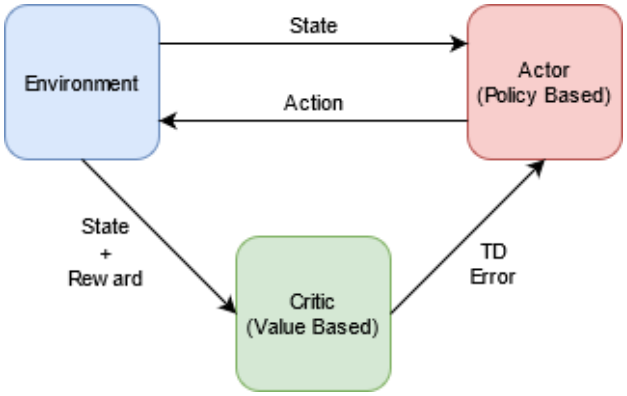


Figure 2. The Actor-Critic architecture

The proposed ACE attack performs evasion attacks on malware detection models with the goal to fool the classifier into misclassifying malware applications as benign by performing a small number of feature alterations.

The proposed ACE attack based on threat modeling can be summarized as:

The malware detection models' robustness against the evasion attack in gray box scenarios can be validated by designing an integrity attack using an Actor-Critic agent based on reinforcement learning.

2.3. ACE Attack Strategy

The improvement in computational and storage capacities of modern computers has led to the increased adoption of reinforcement learning methods in various applications. The proposed ACE adversarial attack implements Actor-Critic architecture agent based on reinforcement learning to attack different android malware detection models. The actor-critic agent utilizes the power of two different Deep Neural Networks (DNN) to accomplish its goals [21]. The first DNN is *actor network* which decides the agent's action and behavior based on a policy. The second DNN is *critic network* that measures the quality of action taken by the actor-network towards a predefined goal.

Figure 2 depicts the basic framework of the proposed ACE attack strategy based on the actor-critic architecture. The environment consists of the training data and the anti-malware system, which provides the current state to the actor-network. The actor-network then computes the optimal action possible for the given state based on its current policy and returns it to the environment. The environment then performs the action on the current state and obtains the corresponding reward. The current state and the reward associated with the action are then fed to the critic-network. The critic-network is then used to approximate the value function, which measures the quality of the action taken and is used in the place of

an episode reward in updating the policy. This enables us to perform updates to the policy after every single action taken by the actor-network rather than waiting until the end of the episode. The computed correction (TD error) is also used to update the critic network. This entire cycle of events occurs after every single action taken during the training phase of the agent. Initially, there is a significant fluctuation in episodic rewards as the agent takes random actions to explore the environment and is still learning which actions are good and bad. The agent's training continues till there is a convergence in episodic rewards to a stable value.

After successfully training the adversarial attack policy, the ACE attack agent is now ready to perform evasion attacks on the malware detection models. Firstly, the environment maps the feature vector to the current state and passes it to the actor network. Here the best action is calculated based on the optimal policy and returned to the environment. The environment updates the current state based on the action and then feeds it again to the actor-network. This process continues until the malware detection model forcefully misclassifies the malicious sample as benign or we reach a predefined threshold to stop. At the end of the ACE attack, all the adversarial samples which have misclassified are collected and labeled as adversarial samples.

2.4. Adversarial Retraining Defense Strategy

The evasion attack (similar to the proposed ACE attack) exposes the vulnerabilities in malware detection models and raises major challenges in its technology adoption in real-world anti-virus systems. These detection models assume that the training and test data come from a similar distribution. Thus, they can effectively classify unknown data samples from a known distribution only. However, the evasion attack violates this assumption by changing the distribution of test samples. The adversarial samples are specifically created to evade the detection models as they do not belong to the same distribution as the training data. Therefore, an effective defense strategy for detection models is to make them capable of dealing with unknown data samples from an unknown distribution including the adversarial samples. The proposed defense strategy known as adversarial retraining is one such defense strategy that has proven to enhance the robustness of models in various applications [17][22][23][24].

The adversarial defense strategy focuses on injecting the adversarial samples with their correct labels into the original training dataset. Since our initial dataset roughly contained an equal number of malicious and benign applications, adding adversarial samples leads to an overabundance of malware samples and class balancing is essential to ensure the retrained models are unbiased. The addition of adversarial data samples with their correct labels makes the classifier more robust and capable of detecting similar adversarial data samples from the same/different evasion attacks in real-world scenarios. Exposing the classifier to data points outside the initial training dataset distribution will enhance its generalizability and robustness. Therefore, adversarial retraining essentially acts as a form of regularization that has even outperformed neural network techniques like dropout, etc. Furthermore, the literature indicates that the defense strategy makes classifiers more confident in their class predictions, making it harder to fool them by introducing minor changes to the data [13]. The enhanced robustness of the malware detection models due to the adoption of this defense strategy can be quantified by performance improvements across various metrics, as discussed in the following sections.

3. Experimental Setup

This section will first discuss the dataset, followed by feature engineering and data preprocessing techniques. Later we will discuss the different classifiers and performance metrics used in the experiments.

3.1. Data Collection (Malware & Benign Apps)

Google Play is the official application distribution platform developed by *Google* for users to browse and download android applications. Other third-party app stores like *Amazon App Store*, *AppBrain*, *Aptoide*, *APKPure*, etc., are also involved in distributing android apps. However, the literature suggests that these platforms contain many malicious apps along with benign samples [4]. Therefore for the construction of the *benign dataset*, we download more than 8000 android apps from Google Play. Then we used *Virus Total* to validate the benignness of each downloaded app. A downloaded app is labeled benign if and only if all the antiviruses from Virus Total label it as benign. Thus, the *benign dataset* contains 5721 android applications, and the rest of the downloaded applications which were malicious were discarded.

The *Drebin dataset* was proposed by Arp et al., and it contains malicious android apps downloaded from Google Play and other third-party app stores [25]. The Drebin dataset contains more than 1200 malicious apps from *Android Malware Genome Project* proposed by Zhou et al. Drebin dataset contains malicious apps from more than 150 malware families like *Plankton*, *Opfake*, *GingerMaster*, *FakeDoc*, etc. Literature suggests that the Drebin dataset is widely investigated and well accepted in the research community [4][8][9]. Therefore our final dataset consists of the benign dataset with 5721 android apps and the Drebin dataset (malware dataset) with 5553 malicious apps.

3.2. Feature Engineering and Data Preprocessing

The construction of android malware detection models requires feature engineering by using domain knowledge to extract raw features from android applications. The extraction of raw features can be performed using static and dynamic analysis of android apps. We used static analysis by disassembling each android app to extract two different features, namely permissions and intents. The applications were disassembled using *Apktool* which generated many original source files like *resources.arsc*, *classes.dex*, *AndroidManifest.xml* etc. The declaration of android permissions and intents should be added in the manifest file before its usage in the source code. We also referred to the original android documentation to construct the master permission and intent lists containing 195 permissions and 273 intents, respectively. Then we scanned through the manifest file of each android application and logged the permissions/intents used by that app to generate the final feature vector. We developed two separate feature vectors, namely *permission feature set* and *intent feature set*. We also performed exploratory data analysis on both the feature sets to understand their characteristics. Finally, the two feature sets were ready to be used for the construction of android malware detection models.

3.3. Classifiers for Malware Detection

We used a diverse set of classifiers to validate the applicability and generalizability of the proposed baseline models, ACE attack, and adversarial retraining defense strategy. We have used eleven classifiers and divided them into three categories as follows:

The *basic machine learning* category has four distinct traditional machine learning classifiers. DT is a classification technique where data is continuously split in order to form a tree-like structure where each leaf node corresponds to a class label (malware/benign). NB is an implementation of the naive bayes classification algorithm designed for data that follows multivariate Bernoulli distributions. SVM is a technique that focuses on mapping instances to points in space to maximise the width of the gap between the malware and benign classes. LR is a technique that uses a logistic function to model a binary dependent variable.

The *ensemble* category has four distinct ensemble based classifiers. RF is a forest of 100 DTs whose average probabilistic prediction is the final prediction of the classifier. Stacked-SVM-DT is a combination of SVM and DT classifiers as an input to a final LR estimator whose prediction is the final prediction of the classifier. GB is a set of DT classifiers put through 100 stages of boosting, which is a technique to convert a group of weaker classifiers into a stronger one. XGB is a specially modified GB algorithm designed for speed, performance, and performing parallel tree boosting.

- ***Basic Machine Learning Classifier***
  - Decision Tree (DT)
  - Bernoulli Naive Bayes (NB)
  - Support Vector Machine (SVM)
  - Logistic Regression (LR)
- ***Ensemble based Classifier***
  - Random Forest (RF)
  - Stacked Classifier (Stacked-SVM-DT)
  - Gradient Boost (GB)
  - Extreme Gradient Boost (XGB)
- ***Deep Neural Network based Classifier***
  - Deep neural network with 1 hidden layer (DNN 1L)
  - Deep neural network with 3 hidden layers (DNN 3L)
  - Deep neural network with 5 hidden layers (DNN 5L)

The *Deep Neural Network* category has three distinct DNN classifiers of different depth. A DNN refers to the classifiers based on a feed-forward artificial neural network capable of distinguishing data that is not linearly separable. We designed three DNNs, namely DNN 1L, DNN 3L, DNN 5L with 1, 3, and 5 hidden layers. The *DNN 1L* classifier has an input layer, one hidden layer with 64 units, and an output layer. The *DNN 3L* classifier has an input layer, three hidden layers with 64, 32, 16 units, and an output layer in the end. The *DNN 5L* classifier has an input layer, five hidden layers with 128, 64, 32, 16, 8 units, and an output layer in the end. All the DNNs are trained using backpropagation with *Adam solver* and *Relu* activation function.

3.4. Performance Measures

The following performance metrics are used to evaluate the experimental results of the malware detection models, ACE attack, and adversarial retraining defense.

- ***True Positive (TP)*** is the number of malware applications which are correctly classified as malware by the classification model.
- ***False Positive (FP)*** is the number of benign applications which are falsely flagged as malware by the classification model.
- ***True Negative (TN)*** is the number of benign applications which are correctly classified as benign by the classification model.
- ***False Negative (FN)*** is the number of malware applications which are falsely flagged as benign by the classification model.
- ***Accuracy*** is the ratio of applications correctly classified to the total number of applications by the classification model.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

- ***Area Under Curve (AUC)*** refers to the area under the receiver operating characteristic curve which is a plot of the true positive rate against the false positive rate at various threshold settings.
- ***Class Probabilities (CP)*** is a prediction array returned by the classifier with probabilities that a given sample belongs to malware and benign class. This metric indicates the confidence of the prediction.
- ***Fooling Rate (FR)*** refers to the ratio of adversarial malware applications (M') forcefully misclassified as benign by the classification model to the total number of malware applications (M) during the evasion attack.

$$\text{Fooling Rate} = \frac{M' - FN}{M - FN} \times 100 \qquad (2)$$

4. Experimental Results

This section will discuss the experimental results of malware detection models (baseline), ACE attack on different classifiers, and adversarial retraining defense strategy for classifiers.

4.1. Malware Detection Models (Baseline)

The third step in the proposed architecture workflow (refer to Figure 1) is to design android malware detection models based on the eleven classifiers discussed in section 3.3. Each classifier is used for both the permission and intent feature vector extracted from android applications. The train-test ratio of 70:30 in the dataset is used for all the experiments in order to prevent overfitting and to enhance the detection model's generalizability. Finally, the malware detection model's performance is determined using evaluation metrics like accuracy, AUC, and class probabilities.

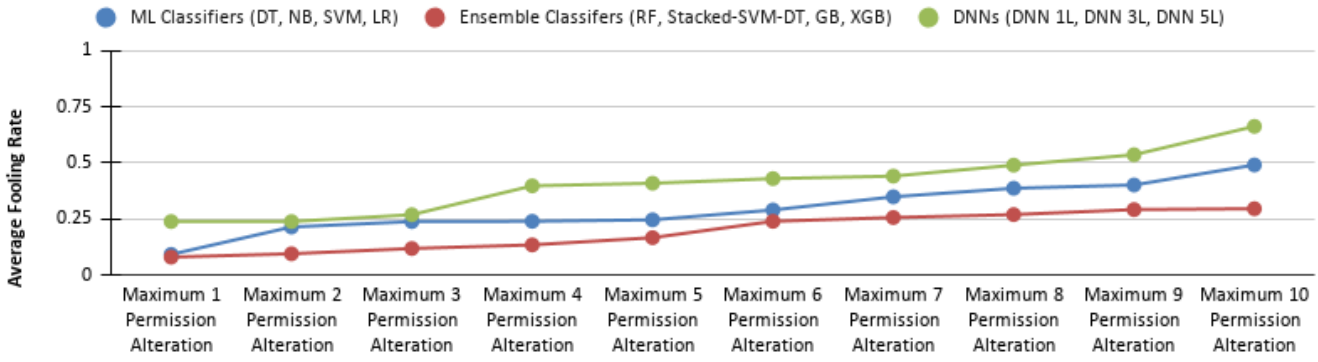
Table 1 and Fig 4a shows the accuracy of permission feature set based malware detection models (baseline) based on eleven classifiers. The accuracy for the permission based classifiers ranges from the highest 94.47% (RF classifier) to the lowest 83.77% (NB classifier). The average baseline accuracy across all eleven permission based classifiers is 92.03%. The models based on RF, DT, SVM, Stacked-SVM-DT, DNN 1L, DNN 3L, and DNN 5L classifiers achieve above average baseline accuracy, whereas the remaining classifiers attain below average baseline accuracy. Similarly, Table 1 and Fig 4b show the accuracy of intent feature set based malware detection models (baseline) based on eleven classifiers. The accuracy for the intent based classifiers ranges from the highest 78.04% (DNN 5L classifier) to the lowest 75.26% (NB classifier). The average baseline accuracy across all eleven intent based classifiers turned out to be 77.19%. The models based on RF, DT, SVM, GB, Stacked-SVM-DT, DNN 1L, DNN 3L, and DNN 5L classifiers achieve above average baseline accuracies, whereas the remaining classifiers attain below average baseline accuracies.

Table 1. Permission and intent based malware detection models (existing literature and proposed baseline)

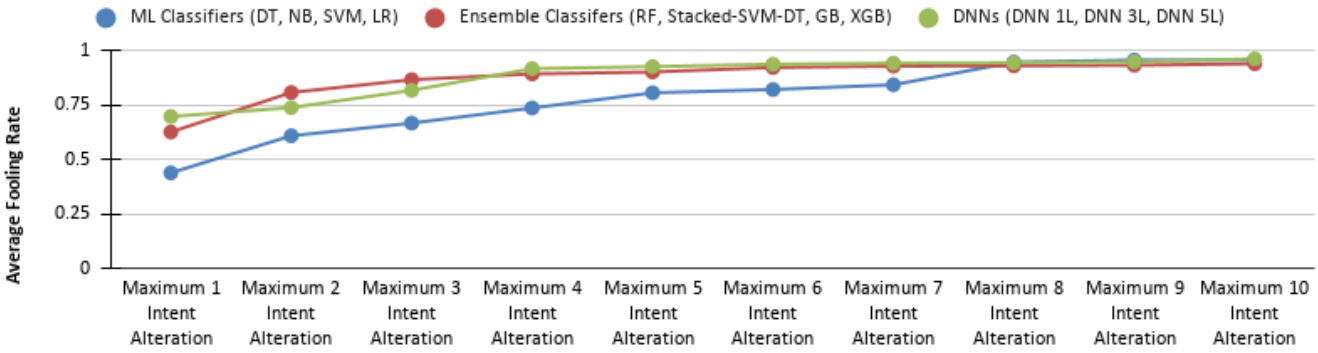
Classifier	Android Permission		Android Intent	
	Accuracy	AUC	Accuracy	AUC
Li et al. (2018) [8]	91.34%	-	-	-
Wang et al. (2019) [26]	94%	-	-	-
Arslan et al. (2019) [27]	91.95%	-	-	-
Arora et al. (2020) [28]	95.44%	-	-	-
Rathore et al. (2020) [9]	94.0%	-	-	-
Mat et al. (2021) [29]	91.1%	0.96	-	-
Sewak et al. (2020) [30]	-	-	77.2%	0.81
DT	92.82%	0.93	77.30%	0.77
NB	83.77%	0.84	75.26%	0.75
SVM	93.05%	0.93	77.51%	0.77
LR	90.60%	0.91	76.38%	0.76
RF	94.47%	0.94	77.71%	0.77
Stacked-SVM-DT	94.06%	0.94	77.33%	0.77
GB	90.30%	0.90	77.21%	0.77
XGB	90.60%	0.91	76.74%	0.76
DNN 1L	94.18%	0.94	77.74%	0.77
DNN 3L	94.24%	0.94	77.92%	0.77
DNN 5L	94.27%	0.94	78.04%	0.78

An analysis of the AUC scores of malware detection models (baseline) reveals that they follow a trend similar to accuracy. The average baseline AUC score across all eleven classifiers is 0.91 and 0.76 for the permission and intent feature sets. A look at the class probabilities of the eleven malware detection models (baseline) also yielded interesting results. The average malware and benign class probability for the permission feature set are 0.91 and 0.92, whereas same for intent feature set are 0.69 and 0.71 only. It indicates that permission feature set-based classifiers are more accurate in their predictions and more confident in predicting the class of an android application.

Table 1 also highlights recently published articles in reputed journals/conferences on android malware detection using permission and intent on same/ different dataset. Li et al. (2018) used android permission and performed feature pruning to achieve malware detection accuracy of 91.34% on the Drebin dataset [8]. Wang et al. (2019) also used permission and achieved 94% accuracy in android malware detection [26]. Similarly, [27][28][9][29] used android permission as the feature and achieved accuracies of 91.95%, 95.44%, 94.04%, and 91.1% respectively in malware detection. We also achieved malware detection accuracy in the range of 90% to 95% using android permission and different classifiers. A similar pattern is also seen for malware detection models based on android intents. The above tables shows that the performance of our proposed malware detection models are inline with the existing state-of-the-art detection models.



(a) Average fooling rate by classifier category achieved by the proposed ACE attack against permission-based malware detection models



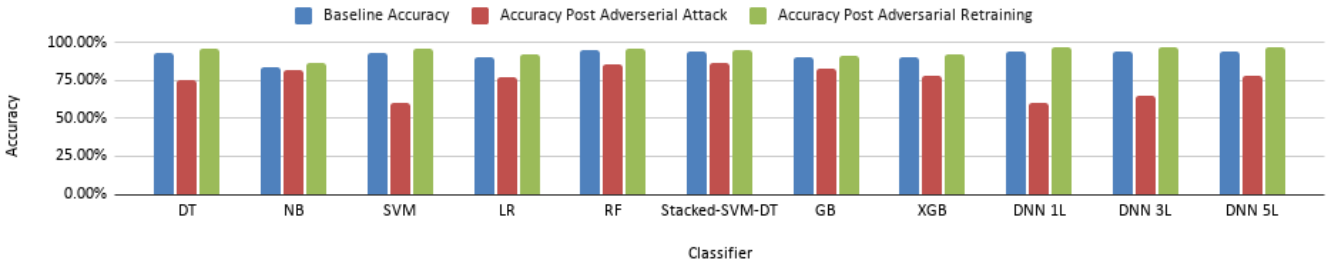
(b) Average fooling rate by classifier category achieved by the proposed ACE attack against intent-based malware detection models

Figure 3. Fooling Rate @ Malware Detection Model(s)

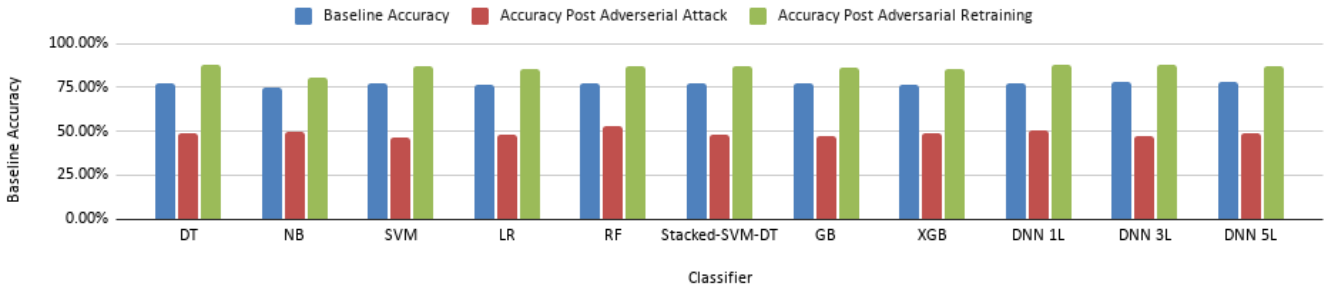
4.2. ACE Attack on Malware Detection Models

The fourth step in the proposed architecture workflow (refer to Figure 1) is to evaluate the robustness of different classifiers by performing the ACE attack on all malware detection models (baseline) constructed in section 4.1. The proposed actor-critic architecture based agent interacts with the environment and learns how to fool the classifier into misclassifying malicious samples as benign. It is achieved by performing a small number of feature alterations (a maximum of ten feature alterations is allowed) to the malicious applications. The feature alterations enforced by the agent are constrained such that it only adds features and thus does not break the functionality of the application. The agent is also optimized to minimize the alterations in each malicious sample while converting the maximum malware applications in the dataset into adversarial samples. The efficacy of any evasion attack can be determined using performance metrics such as fooling rate, classifier accuracy, etc.

Fooling Rate by ACE Attack. We first perform the proposed ACE attack on eleven permission based classifiers. The ACE attack achieves the fooling rate in the range from the highest 82.51% (DNN 1L classifier) to the lowest 25.66% (Stacked-SVM-DT classifier). The average fooling rate achieved against all eleven permission feature based classifiers is 46.63%. Later, we grouped the permission models based on classifier category (ref section 3.3) and averaged the fooling rates, which leads to interesting insights as shown in Figure 3a. The average fooling rate achieved against three deep neural network classifiers is 66.23%, four basic machine learning classifiers is 49.04%, and four ensemble classifiers is 29.53%. Later we perform the ACE attack on intent based classifiers. We observed that the ACE attack achieved the fooling rate in the range from the highest 100.00% (SVM classifier) to the lowest 87.32% (RF classifier). The average fooling rate achieved against all eleven intent based classifiers is 95.31%. We did similar grouping in intent feature set models based on classifier category and averaged the fooling rates as shown in Figure 3b. The average fooling rate achieved against three deep neural network classifiers is 96.35%, four basic machine learning classifiers is 95.77%, and four ensemble classification methods is 94.08%. These results indicate that neural network classifiers are the most vulnerable against evasion attack while ensemble classifiers are the least vulnerable. As we increase the number of perturbations to generate adversarial samples, the increment in fooling rate start to diminish



(a) Accuracy of eleven different permission-based malware detection models (Baseline, After ACE Attack, After Adversarial Retraining)



(b) Accuracy of eleven different intent-based malware detection models (Baseline, After ACE Attack, After Adversarial Retraining)

Figure 4. Accuracy @ Malware Detection Model(s)

and beyond 10 perturbations it become stable. Also, the goal is to make as-few-perturbations-as-possible while converting as-many-malicious-samples-as-possible.

Classifier Accuracy after ACE Attack. Another critical performance metric to compare the classifier's performance against the proposed ACE attack is model accuracy. Figure 4a shows the accuracy (red bar) of eleven permission based malware detection models after the ACE attack. The lowest accuracy is attained by DNN 1L classifier (59.71%) followed by SVM classifier (60.12%) and DNN 3L classifier (65.21%). The average classifier accuracy after the ACE attack is reduced to 75.40% for eleven permission based malware detection models, which is a drop of 18.07% from baseline models. Then, we perform the proposed ACE attack against eleven intent based malware detection models. Figure 4b shows the accuracy (red bar) of eleven intent based malware detection models after the ACE attack ranges from the lowest 46.67% (SVM classifier) to the highest 53.21% (RF classifier). The average classifier accuracy after the ACE attack across all eleven classifiers is 48.92%, which is a drop of 36.62% from baseline models. These results align with the expected behavior that there will be a significant drop in the classifier's accuracy after the ACE attack compared to the baseline classifier. As expected, the classifiers against which the ACE attack achieved the most success ended up with the lowest classifier accuracy (post evasion attack) and vice-versa.

Discussion about ACE Attack. We notice that the proposed ACE attack performed better against the intent based models as compared to permission based models. We think that this is due to the combination of many factors. Firstly, it must be noted that the average baseline accuracy for the eleven permission based models (92.03%) is significantly higher than the average baseline accuracy for the eleven intent based models (77.19%). Secondly, the average malware class probability for the 5553 malicious android applications is 0.91 for the permission feature set and 0.69 for the intent feature set. These two factors make the permission feature set classifiers much harder to fool. They are also the primary reason for the steep increase in the fooling rate during the initial steps of the ACE attack for intent based models vis-à-vis permission based models.

4.3. Adversarial Retraining Defense Strategy for Malware Detection Models

After the ACE attack's success, the final step of the proposed architecture workflow (refer to Figure 1) is to design the defense strategy for all the classifiers (permission and intent feature set). The defense strategy focuses on marking the adversarial samples with their correct label as malicious since they were originally malware applications and their functionality is still intact. These samples are now added to the original dataset, followed by class balancing. Now, the twenty-two classifiers are retrained using this newly

Table 2. Comparison of threat modeling and adversarial robustness against different malware detection models

Schemes	Attack Scenario	Number of Malware Detection Models Attacked	Fooling Rate or Forced Misclassification Rate	Accuracy Drop (Baseline vs. After Adversarial Attack)	Maximum Perturbation Allowed	Adversarial Robustness @Reattack
Chen et al. (2017) [31]	White box	1	-	15.86%	50	No
Grosse et al. (2017) [32]	White box	3	-	10.75%	20	No
Fang et al. (2019) [33]	Grey box	1	46.56%	-	80	No
Fang et al. (2020) [34]	Grey box	1	19.13%	-	100	Yes
Taheri et al. (Trival) (2020) [35]	White box	3	-	10.93%	20%	No
Taheri et al. (Distribution) (2020) [35]	White box	3	-	11.43%	20%	No
Taheri et al. (kNN) (2020) [35]	White box	3	-	11.61%	20%	No
Taheri et al. (LR) (2020) [35]	White box	3	-	11.70%	20%	No
Taheri et al. (ACO) (2020) [35]	White box	3	-	6.26%	20%	No
Proposed (Permission)	Grey box	11	46.63%*	18.07%	10	Yes
Proposed (Intent)	Grey box	11	95.31%*	36.62%	10	Yes

*average across eleven distinct permission / intent based malware detection models

created dataset. The defense strategy's performance can be evaluated using performance measures like classifier accuracy and class probabilities.

Figure 4a shows the accuracy (green bar) of eleven permission feature set based adversarially retrained classifiers. The retrained classifier accuracy ranges from the highest 97.15% (DNN 1L classifier) to the lowest 86.44% (NB classifier). The average accuracy across all eleven adversarially retrained classifiers is 94.16%, which is 24.88% improvement over post ACE attack classifiers and 2.31% improvement over the baseline classifiers. Figure 4b shows the accuracy (green bar) of eleven intent feature set based adversarially retrained classifiers. The retrained classifier accuracy ranges from the highest 87.69% (DT classifier) to the lowest 80.45% (NB classifier). The average accuracy across all eleven adversarially retrained classifiers is 86.35%, which is 76.51% improvement over post ACE attack classifiers and 11.87% improvement over the baseline classifiers. The AUC scores of the adversarially retrained classifier followed a similar trend to that observed with retrained classifier accuracies. The average retrained AUC score across all eleven classifiers is 0.94 and 0.86 for the permission and intent feature sets, respectively. This is a 26.21% and 82.39% improvement over the post ACE attack scores and a 2.38% and 12.54% improvement over the respective baseline AUC scores for the permission and intent feature sets respectively. The malware and benign class probabilities also showed a significant improvement across all classifiers. The average malware and benign class probabilities of adversarially retrained classifiers for the permission feature set are 0.93 and 0.93, whereas same for intent feature set are 0.79 and 0.81. The adversarially retrained classifier shows an improvement of 1.64% and 14.29% in class probabilities compared to the corresponding baseline values for the permission and the intent feature set. Finally, the ACE attack was performed once more on the retrained models to validate the proposed defense strategy. The results obtained in that case were promising. In some cases, the fooling rates were drastically reduced compared to the initial fooling rates observable for both the permission and intent feature set. Therefore we can conclude that adversarial retraining enhances the adversarial robustness of malware detection models.

4.4. Comparison with Existing Work

Adversarial robustness is an active area of research in the domain of malware analysis and detection. The adversarial attacks against any malware detection models can be classified into black box/grey box/white box scenario based on the attacker's knowledge about different components of the target ecosystem [11]. [31, 32, 35] have shown that evasion attack against malware detection models is possible in the white box scenario. However, this is seldom the case in the real world that an attacker has complete knowledge of the target system. We have considered a grey box scenario (which is much similar to real-world setting) in the proposed ACE attack. Furthermore, we have validated our evasion attack strategy more comprehensively by evaluating it against 22 distinct malware detection models built using three different categories of classification algorithms. On the other hand, most existing literature has only tested their adversarial attack against 1-3 detection models. The constraints on the number of perturbations during the evasion attack vary from 20 in [32, 35] to over 50 in [31, 33, 34]. More perturbations can lead to successful attacks but do not consider the costs associated with each perturbation. Therefore, we try to limit the

modifications to malware samples and keep them similar to their original counterparts by only adding a maximum of 10 perturbations. Despite a very limted number of perturbations, the proposed ACE attack managed to obtain a high fooling rate of 46.56% for the permission-based models and a much higher drop in accuracy compared to the other attacks. Adversarial retraining is a popular defense strategy to counter the adversarial attack and is suggested by many authors [34, 35]. Taheri et al. compare the improvement in accuracy of the detection models between after the evasion attack and after adversarial retraining [35]. While this successfully justifies the effectiveness of adversarial retraining, we go a step further and perform a re-attack against the retrained models to validate the adversarial robustness of malware detection models.

4.5. Limitations

The proposed experiment primarily focuses on providing proof of concept on the adversarial robustness of malware detection models. However, certain limitations and enhancements can further improve the system. Firstly, techniques like prioritized experience replay can be used during the agent's training phase to improve the fooling rate obtained for the permission feature set. Secondly, the proposed ACE attack is designed for the gray box attack scenario. It requires access to the dataset and feature set used, which may not always be possible in a real-world scenario. The system security should be designed assuming the white box scenarios from the anti-malware designer perspective. Thirdly, as the classifier becomes highly complex, the training and attack time taken by the proposed ACE attack against the classifier increases. Finally, the proposed adversarial retraining defense strategy requires the use of adversarial samples. Therefore, the proposed defense strategy is dependent on the success of the evasion attack, and it would not function well when used independently or with an unsuccessful evasion attack.

5. Related Work

The last decade has seen massive adoption of machine learning and deep learning classifiers in various domains like malware detection, object detection, spam detection, etc. However, recently adversarial robustness of these systems has become an active field of research due to many vulnerabilities in the existing systems. Taheri et al. proposed five different white-box evasion attack strategies against three distinct android malware detection models based on machine learning classifiers [35]. They allowed a maximum of twenty perturbations against permission / intent / API call based malware detection models and achieved a drop of around 10% during the attack. They also proposed adversarial retraining and GAN-based defense strategies. However, they did not explore the improvement in adversarial robustness of the proposed defense mechanisms against reattack on malware detection models. Grosse et al. proposed a Jacobian matrix-based algorithm called *JSMA* to perform a white box evasion attack on android malware detection models built using Drebin dataset [32]. An adversarial attack against Windows-based malware detection models for grey box scenario is proposed by Fang et al. [33]. Chen et al. in SecureDroid performed a white box evasion attack against android malware detection models built using permission, intent, API call, and New-Instances. They achieved a baseline accuracy of 96.34% and a drop of 15.86% with a maximum 50 perturbations. Kouliasridis et al. performed static and dynamic analysis of coronavirus contact tracing android applications. They performed manual analysis of applications and found many weaknesses and vulnerabilities in them which might risk users' security and privacy in the android ecosystem [36]. Tsiatsikas et al. added perturbations to achieve denial of service (DoS) in the session initiation protocol (SIP). They also proposed a defense that induces negligible overhead but can counter the evasion attack [37]. Zhang et al. showed that ensemble models might be more vulnerable to evasion attacks than single classification models in white and grey box scenarios. However, they did not discuss the defense strategy to counter the evasion attack [38]. Sewak et al. proposed DRL-based deobfuscation to counter the evasion attacks in windows malware detection models [39]. Rathore et al. proposed gradient based evasion attacks against android malware detection models. They also proposed three defense strategies to counter the evasion attack [40]. Most of the existing literature assumes a white box scenario for threat modeling, which is a very optimistic assumption for adversaries in real-world applications. Also, most of the authors have constructed only 1 to 3 unique malware detection models and performed adversarial attacks on them. Further, threat modeling in adversarial scenarios has a constraint to reduce the number of perturbations to generate adversarial samples, which authors rarely incorporate.

6. Conclusion

The machine learning, ensemble and deep learning classifier based next-generation android malware detection models have shown promising results. However, these classifiers are vulnerable to adversarial samples. In this work, we stepped into the adversaries' shoes to propose the ACE attack policy based on actor-critic architecture for grey-box scenario. The ACE attack (with maximum ten perturbations) achieved an average fooling rate of 46.63% against eleven permission based malware detection models and 95.31% against eleven

intent based detection models. Later we proposed adversarial retraining as the counter and achieved an average accuracy improvement of 24.88% and 76.51% against the attacked eleven permission and eleven intent based detection models, respectively. Finally, we can conclude that android malware detection models' vulnerability and forensic analysis in adversarial scenarios is essential and should be performed before their real-world deployment.

References

[1] (2021), Android - Statistics & Facts, Available: https://www.statista.com/topics/876/android/. Last Accessed: Aug 2021.

[2] (2021), Statcounter, Available: https://gs.statcounter.com/os-market-share/mobile/worldwide. Last Accessed: Aug 2021.

[3] (2021), AVTEST, Available: https://portal.av-atlas.org/malware/statistics. Last Accessed: Aug 2021.

[4] Ye, Y., Li, T., Adjeroh, D. and Iyengar, S.S. (2017) A survey on malware detection using data mining techniques. *ACM Computing Surveys (CSUR)* **50**(3): 1–40.

[5] Liu, K., Xu, S., Xu, G., Zhang, M., Sun, D. and Liu, H. (2020) A review of android malware detection approaches based on machine learning. *IEEE Access* **8**: 124579–124607.

[6] Razgallah, A., Khoury, R., Hallé, S. and Khanmohammadi, K. (2021) A survey of malware detection in android apps: Recommendations and perspectives for future research. *Computer Science Review* **39**: 100358.

[7] Qiu, J., Zhang, J., Luo, W., Pan, L., Nepal, S. and Xiang, Y. (2020) A survey of android malware detection with deep neural models. *ACM Computing Surveys (CSUR)* **53**(6): 1–36.

[8] Li, J., Sun, L., Yan, Q., Li, Z., Srisa-An, W. and Ye, H. (2018) Significant permission identification for machine-learning-based android malware detection. *IEEE Transactions on Industrial Informatics* **14**(7): 3216–3225.

[9] Rathore, H., Sahay, S.K., Rajvanshi, R. and Sewak, M. (2020) Identification of significant permissions for efficient android malware detection. In *International Conference on Broadband Communications, Networks and Systems (BROADNETS)* (Springer): 33–52.

[10] Rathore, H., Sahay, S.K., Thukral, S. and Sewak, M. (2020) Detection of malicious android applications: Classical machine learning vs. deep neural network integrated with clustering. In *International Conference on Broadband Communications, Networks and Systems (BROADNETS)* (Springer): 109–128.

[11] Pitropakis, N., Panaousis, E., Giannetsos, T., Anastasiadis, E. and Loukas, G. (2019) A taxonomy and survey of attacks against machine learning. *Computer Science Review* **34**: 100199.

[12] Otter, D.W., Medina, J.R. and Kalita, J.K. (2020) A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems* **32**(2): 604–624.

[13] Zhang, W.E., Sheng, Q.Z., Alhazmi, A. and Li, C. (2020) Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)* **11**(3): 1–41.

[14] Akhtar, N. and Mian, A. (2018) Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access* **6**: 14410–14430.

[15] Apruzzese, G., Andreolini, M., Marchetti, M., Venturi, A. and Colajanni, M. (2020) Deep reinforcement adversarial learning against botnet evasion attacks. *IEEE Transactions on Network and Service Management (IEEE TNSM)* **17**(4): 1975–1987.

[16] Tong, L., Li, B., Hajaj, C., Xiao, C., Zhang, N. and Vorobeychik, Y. (2019) Improving robustness of {ML} classifiers against realizable evasion attacks using conserved features. In *28th {USENIX} Security Symposium ({USENIX})*: 285–302.

[17] Goodfellow, I.J., Shlens, J. and Szegedy, C. (2014) Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*.

[18] Chen, C., Zhao, X. and Stamm, M.C. (2019) Generative adversarial attacks against deep-learning-based camera model identification. *IEEE Transactions on Information Forensics and Security* .

[19] He, X., Yang, S., Li, G., Li, H., Chang, H. and Yu, Y. (2019) Non-local context encoder: Robust biomedical image segmentation against adversarial attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-2019)*, **33**: 8417–8424.

[20] Joshi, C., Aliaga, J.R. and Insua, D.R. (2020) Insider threat modeling: An adversarial risk analysis approach. *IEEE Transactions on Information Forensics and Security* **16**: 1131–1142.

[21] Grondman, I., Busoniu, L., Lopes, G.A. and Babuska, R. (2012) A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **42**(6): 1291–1307.

[22] Rathore, H., Sahay, S.K., Nikam, P. and Sewak, M. (2020) Robust android malware detection system against adversarial attacks using q-learning. *Information Systems Frontiers* : 1–16.

[23] Khoda, M.E., Imam, T., Kamruzzaman, J., Gondal, I. and Rahman, A. (2019) Robust malware defense in industrial iot applications using machine learning with selective adversarial samples. *IEEE Transactions on Industry Applications* **56**(4): 4415–4424.

[24] Liu, K., Yang, H., Ma, Y., Tan, B., Yu, B., Young, E.F., Karri, R. *et al.* (2020) Adversarial perturbation attacks on ml-based cad: A case study on cnn-based lithographic hotspot detection. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* **25**(5): 1–31.

[25] Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H. and Rieck, K. (2014) Drebin: Effective and explainable detection of android malware in your pocket. In *Network and Distributed System Security Symposium (NDSS)*, **14**: 23–26.

[26] Wang, C., Xu, Q., Lin, X. and Liu, S. (2019) Research on data mining of permissions mode for android malware

detection. *Cluster Computing* **22**(6): 13337–13350.

[27] Arslan, R.S., Doğru, İ.A. and Barişçi, N. (2019) Permission-based malware detection system for android using machine learning techniques. *International Journal of Software Engineering and Knowledge Engineering* **29**(01): 43–61.

[28] Arora, A., Peddoju, S.K. and Conti, M. (2020) Permpair: Android malware detection using permission pairs. *IEEE Transactions on Information Forensics and Security* **15**: 1968–1982.

[29] Mat, S.R.T., Ab Razak, M.F., Kahar, M.N.M., Arif, J.M. and Firdaus, A. (2021) A bayesian probability model for android malware detection. *ICT Express* .

[30] Sewak, M., Sahay, S.K. and Rathore, H. (2020) Deepintent: implicitintent based android ids with e2e deep learning architecture. In *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)* (IEEE): 1–6.

[31] Chen, L., Hou, S. and Ye, Y. (2017) Securedroid: Enhancing security of machine learning-based detection against adversarial android malware attacks. In *Proceedings of the 33rd Annual Computer Security Applications Conference (ACSAC)*: 362–372.

[32] Grosse, K., Papernot, N., Manoharan, P., Backes, M. and McDaniel, P. (2017) Adversarial examples for malware detection. In *European Symposium on Research in Computer Security (ESORICS)* (Springer): 62–79.

[33] Fang, Z., Wang, J., Li, B., Wu, S., Zhou, Y. and Huang, H. (2019) Evading anti-malware engines with deep reinforcement learning. *IEEE Access* **7**: 48867–48879.

[34] Fang, Y., Zeng, Y., Li, B., Liu, L. and Zhang, L. (2020) DeepDetectNet vs RLAttackNet: An adversarial method to improve deep learning-based static malware detection model. *Plos one* **15**(4): e0231626.

[35] Taheri, R., Javidan, R., Shojafar, M., Vinod, P. and Conti, M. (2020) Can machine learning model with static features be fooled: an adversarial machine learning approach. *Cluster Computing* **23**(4): 3233–3253.

[36] Kouliaridis, V., Kambourakis, G., Chatzoglou, E., Geneiatakis, D. and Wang, H. (2021) Dissecting contact tracing apps in the android platform. *Plos one* **16**(5): e0251867.

[37] Tsiatsikas, Z., Kambourakis, G., Geneiatakis, D. and Wang, H. (2018) The devil is in the detail: Sdp-driven malformed message attacks and mitigation in sip ecosystems. *IEEE Access* **7**: 2401–2417.

[38] Zhang, F., Wang, Y., Liu, S. and Wang, H. (2020) Decision-based evasion attacks on tree ensemble classifiers. *World Wide Web* **23**(5): 2957–2977.

[39] Sewak, M., Sahay, S.K. and Rathore, H. (2021) Drldo: A novel drl based de-obfuscation system for defence against metamorphic malware. *Defence Science Journal* **71**(1).

[40] Rathore, H., Samavedhi, A., Sahay, S.K. and Sewak, M. (2021) Robust malware detection models: Learning from adversarial attacks and defenses. *Forensic Science International: Digital Investigation* **37**: 301183.