DIGITAL FORENSIC RESEARCH CONFERENCE

Robust Malware Detection Models: Learning from Adversarial Attacks and Defenses

By:
Hemant Rathore (BITS Pilani), Adithya Samavedhi (BITS Pilani),
Sanjay K. Sahay (BITS Pilani), and Mohit Sewak (Microsoft)

From the proceedings of

The Digital Forensic Research Conference

DFRWS USA 2021

July 12-15, 2021

Forensic Science International: Digital Investigation 37 (2021) 301183

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

DFRWS 2021 USA - Proceedings of the Twenty First Annual DFRWS USA

Robust Malware Detection Models: Learning from Adversarial Attacks and Defenses

Hemant Rathore [a, *], Adithya Samavedhi [a], Sanjay K. Sahay [a], Mohit Sewak [b]

[a] Dept. of CS & IS, Goa Campus, BITS Pilani, India
[b] Security & Compliance Research, Microsoft, India

ARTICLE INFO

Article history:

Keywords:
Android
Adversarial learning
Deep neural network
Machine learning
Malware detection

ABSTRACT

The last decade witnessed an exponential growth of smartphones and their users, which has drawn massive attention from malware designers. The current malware detection engines are unable to cope with the volume, velocity, and variety of incoming malware. Thus the anti-malware community is investigating the use of machine learning and deep learning to develop malware detection models. However, research in other domains suggests that the machine learning/deep learning models are vulnerable to adversarial attacks. Therefore in this work, we proposed a framework to construct robust malware detection models against adversarial attacks. We first constructed twelve different malware detection models using a variety of classification algorithms. Then we acted as an adversary and proposed *Gradient-based Adversarial Attack Network* to perform adversarial attacks on the above detection models. The attack is designed to convert the maximum number of malware samples into adversarial samples with minimal modifications in each sample. The proposed attack achieves an average fooling rate of 98.68% against twelve permission-based malware detection models and 90.71% against twelve intent-based malware detection models. We also identified the list of vulnerable permissions/intents which an adversary can use to force misclassifications in detection models. Later we proposed three adversarial defense strategies to counter the attacks performed on detection models. The proposed *Hybrid Distillation* based defense strategy improved the average accuracy by 54.21% for twelve permission-based detection models and 59.14% for intent-based detection models. We also concluded that the adversarial-based study improves the performance and robustness of malware detection models and is essential before any real-world deployment.

1. Introduction

Today smartphones have become an integral part of our modern life and society. A recently published report by Simon Kemp (Hootsuite) suggests that the total number of unique smartphone users have reached 5.22 billion, which is 66% of the current world's population. Smartphones store many user information like profile, photographs, videos, etc., apart from the information stored by various Android applications installed on the device. This information has become a lucrative target of malware designers who can exploit it for commercial purposes. The McAfee Mobile Threat Report (Q1, 2020) suggests more than 35 million malware on mobile platforms, with the addition of 2 million new malware in the first two quarters of 2020. The Symantec's Internet Security Threat Report (2019) reveals that one in every thirty-six mobile devices had a high-risk Android application installed on it.

The primary defenses against malware attacks are designed and developed by the anti-malware community and antivirus companies. The current antiviruses rely extensively on the signature, heuristic, and behavior based detection engines to detect new and old malware. However, Buczak and Guven (2016), and Ye et al. (2017) suggest that these mechanisms are often slow, unscalable, primarily human-driven, and sometimes reactive in nature. Also, the exponential growth of mobile malware has led to tremendous pressure on existing detection engines. Therefore researchers and the anti-malware community are investigating to construct new-age state-of-the-art malware detection engines based on Machine Learning (ML) and Deep Learning (DL).

Ye et al. (2017) suggest that the construction of malware detection models using ML/DL is a two-stage process: (1) Feature

* Corresponding author. BITS Pilani, India.
E-mail address: hemantr@goa.bits-pilani.ac.in (H. Rathore).

https://doi.org/10.1016/j.fsidi.2021.301183

Extraction (2) Classification. Li et al. (2018) developed SigPID by extracting permissions from Android applications and then performing pruning of features to construct a malware detection model. Kim et al. (2018) extracted various features like permission, intent, etc., from applications and constructed a multimodal neural network for malware detection. However, research in other domains such as image classification, object detection, etc., clearly suggests that the classification models built using ML/DL can easily be fooled. Goodfellow et al. (2015) intentionally added a small amount of perturbation in test samples such that the image detection system is forced to misclassify them. Kurakin et al. (2017) forced misclassifications in a real-world Inception based classifier by feeding it perturbed images that human observers cannot detect. Similar adversarial attacks can also be designed and developed against malware detection models.

Pitropakis et al. (2019) suggest that the threat modeling of adversarial attack is defined based on the adversary's goal, knowledge, and capabilities for the target system. In this work, we first performed feature extraction from Android applications (malware/benign) and constructed many baseline malware detection models. Then we acted as an adversary with the goal to perform *evasion attack* on all the baseline detection models. We proposed a state-of-the-art Gradient Adversarial Attack Network (GAAN) to perform attacks for finding vulnerabilities and reduce the performance of the baseline detection models. The GAAN is designed to perform minimum modifications on malware samples to generate adversarial examples. We also ensured that the modifications are syntactically possible and do not break the application's functional and behavioral aspects. The attack is designed for *grey box scenario* where the adversary is assumed to have knowledge about training data and no information about the classification algorithm used to construct the detection model. We also assume that the adversary has the capabilities to modify test samples. All the above assumptions are very similar to any real-world malware detection scenario where malware samples are freely available on many platforms, and the adversary has the knowledge and capability to modify them. We conducted the GAAN based attack on two separate feature vectors (permission and intent) to validate its generalizability against different detection models. Later, we proposed three adversarial defenses, namely *Adversarial Retraining*, *GAN* and *Hybrid Distillation* to counter the attack on detection models. We compare the performance of proposed defense mechanisms, and the acquired knowledge is again fed back to the detection models. Finally, we validated the robustness of malware detection models and made the following contributions:

- We proposed the GAAN strategy to perform evasion attacks against twelve distinct malware detection models built using a variety of classification algorithms.
- The evasion attack with a maximum of 10 modifications achieved an average fooling rate of 98.68% against twelve permission-based malware detection models and 90.71% against intent based detection models.
- We developed vulnerability lists of permissions/intents, which adversaries can use to force misclassifications in malware detection models.
- We designed three adversarial defense strategies for malware detection models to counter evasion attacks. The proposed hybrid distillation defense strategy achieved an average accuracy improvement of 54.21% for twelve permission-based malware detection models and 59.14% for intent-based detection models.

The rest of the paper is organized as follows. Section-2 explains the proposed framework, while section-3 contains experimental setup details. Section-4 explains experimental results and discussion. Section-5 contains related work in this domain, and finally, section-6 concludes the paper.

2. Overview and framework

This section will first elaborate on problem definition and the proposed framework to construct robust malware detection model(s). Later in the section, we will explain the GAAN-based attack strategy followed by three adversarial defense mechanisms to counter the attack.

2.1. Problem definition

Given a dataset (D) containing sets of benign (B) and malicious (M) Android applications. The dataset (D) can be represented as:

$$D = \{\{(x_j, y_j)\}, \forall\, j \in (1, 2, \ldots n)\} \in (X, Y) \tag{1}$$

where X represents Android applications, and Y represents the application's class labels (malware/benign). The Android applications X can be represented with features like permission, intent, system call, etc. Assume X is a binary feature vector represented as $X \in \{0,1\}^d$ where d is the number of features. Here, x_j and y_j denotes the jth Android application and its corresponding class label. Also, $y_j \in \{0, 1\}$ where y_j is set to 0 for benign samples and y_j is set to 1 for malicious samples. Given the above information, many classification algorithms can be used to construct malware detection model(s). The performance of detection models can be evaluated with metrics like accuracy, precision, recall, AUC, etc.

The adversary aims to attack malware detection models and decrease their performance. In an evasion attack, the adversary's goal is to make intelligent perturbations in the test samples to force misclassifications in the detection models. Let us assume that the adversary considers samples from the malware set (M) to make intelligent perturbations such that malware samples are forcefully misclassified as benign by detection models.

$$x'_j = x_j + \delta \quad \text{s.t.} \quad f(x'_j) \neq f(x_j) \tag{2}$$

where $f(x)$ is the prediction function of the malware detection model. The adversary can generate an adversarial sample (x_j') by adding (δ) in the original sample (x_j) to force misclassification in the detection model. The adversary should ensure that perturbations are syntactically possible and should not break the application's functional or behavioral aspect. The adversary aims to perform minimum perturbations in each sample but wants to convert maximum malicious samples from M for force misclassification against the detection models.

The adversarial defense strategies aim to improve the robustness of malware detection models against adversarial attacks. It can be achieved by identifying and reducing vulnerability in the malware detection models. The defender can also use the adversarial samples x_j' generated during the attack to improve the model. The new knowledge acquired from adversarial defense strategy improves the overall robustness of detection models.

2.2. Proposed framework

Fig. 1 illustrates the proposed framework architecture to construct robust malware detection model(s). The first step consists of *Data Collection* which requires gathering malicious and benign Android applications. The second step involves *Feature Extraction* from downloaded applications to construct the feature

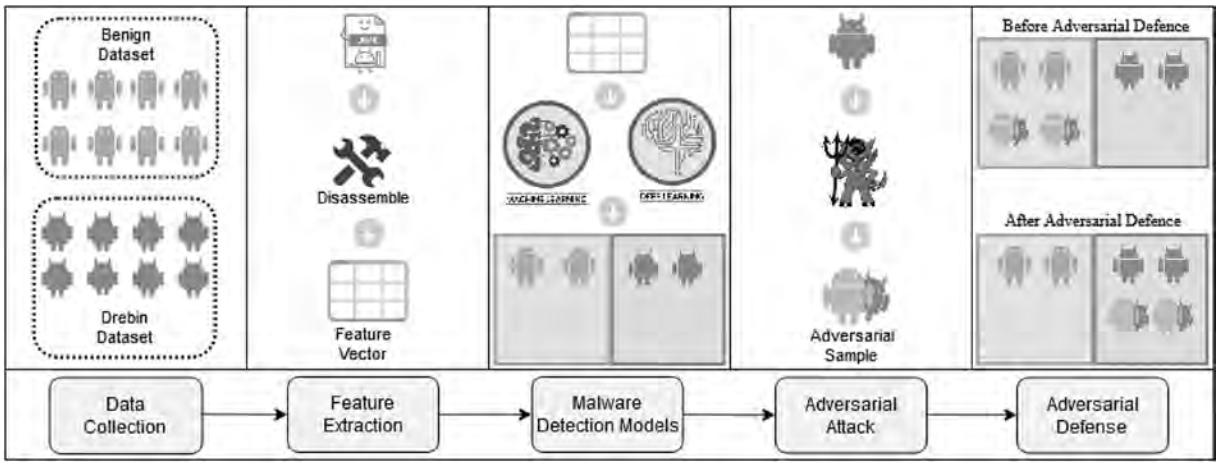

Fig. 1. Proposed framework for constructing robust malware detection model(s) using adversarial attack and defense strategies.

vector. We extracted permissions and intents to develop two separate feature vectors representing Android applications. The third step involves constructing baseline *Malware Detection Models* using features vectors developed in step-2. We construct twelve distinct detection models using different classification algorithms. The variety of detection models simulate the real-life scenario where the adversary is unaware of the target model, a.k.a *Grey Box Scenario*. The fourth step validates the performance of different baseline detection models against *Adversarial Attack*. We proposed the GAAN-based attack to find vulnerabilities by converting malicious samples into adversarial examples for force misclassifications in different detection models. The fifth and final step involves the use of *Defense Mechanism* to counter adversarial attack. We proposed and compared three defense mechanisms to finally construct robust malware detection models.

2.3. Attack strategy against malware detection model(s)

The adversarial attack strategy is designed for compromising and reducing the performance of malware detection models. We propose a GAAN based attack for the grey box scenario where the adversary is assumed to have information about the dataset and feature vector but no knowledge about the algorithm(s) used to construct the malware detection model(s). The adversary aims to intentionally add perturbations in malware samples such that they are forcefully misclassified as benign by detection models.

Given the dataset $D = \{X, Y\}$ as discussed in section 2.1. The proposed GAAN is a deep neural network architecture consisting of three hidden layers with 512, 256 and 64 neurons. Firstly, the GAAN based model is trained on D to minimize the binary cross-entropy/ log loss by changing weight w and bias b parameters.

$$\text{Log Loss} = -y \times log(p) - (1-y) \times log(1-p)) \tag{3}$$

where y is the sample's true label, whereas p is the label predicted by the detection model. Now the adversary's GAAN model calculates the gradients with respect to each input feature from the loss equation. Given a malware sample (m) to be converted into an adversarial sample (m'). The GAAN model derives the gradients (eq. (4)) with respect to each input feature (m^i) and then selects features for modification to force misclassification.

$$Grad_{m^i} = \frac{\partial Loss}{\partial m^i} \quad \text{where} \quad i \in (1 \ldots d) \tag{4}$$

These gradients act as an indicator of features that are likely to increase the loss. The adversary chooses to modify features with the highest gradients such that the loss is maximised, and the detection model is forced to misclassify the modified sample.

Algorithm 1. Algorithm for GAAN based attack

Algorithm 1 Algorithm for GAAN based attack

Input:
GAAN: a deep neural network trained on original dataset
m: malware sample to be converted into adversarial sample
target_model: targeted malware detection model
max_mod: maximum number of modifications allowed in m
Function:
predict: prediction function of target_model
get_feature: returns list of features not used by m
gradient_sort: performs sorting on the list (descending order)
modify: add the feature in malware sample m
get_name: get the name of the feature
Output: $(m', modified\_list)$

1: $pred \leftarrow predict(target\_model, m)$
2: **if** $(pred == 1)$ **then**
3: $features \leftarrow get\_features(m)$
4: **for** i in $len(features)$ **do**
5: $feature\_gradient \leftarrow feature\_gradient \cup (\{\frac{\partial Loss}{\partial features[i]}\}, i)$
6: **end for**
7: $feature\_gradient \leftarrow gradient\_sort(feature\_gradient)$
8: **for** $i \leq max\_mod$ **do**
9: $m' \leftarrow modify(feature\_gradient[i][1], m)$
10: $modified\_list = modified\_list \cup get\_name(feature\_gradient[i][1])$
11: **if** $(predict(target\_model, m') == 0)$ **then**
12: break
13: **end if**
14: **end for**
15: **end if**
16: return $(m', modified\_list)$

The Algorithm 1 illustrates the GAAN strategy to generate the adversarial sample to force misclassification in the malware detection model. The algorithm's inputs are the GAAN model, a malware sample to be converted into an adversarial sample, a targeted malware detection model, and a maximum number of modifications allowed in the malware sample. The algorithm starts (line 1–3) with basic checking followed by lines 4–6 to calculate the gradient of features that are absent in the malware sample and store them in the form of a tuple (gradient_value, index of feature). Then the list is sorted in descending order based on the magnitude of the gradient. Finally, lines 8–14 in the GAAN strategy modifies the features one by one and checks for successful generation of an adversarial sample after every modification. The algorithm ends by

returning an adversarial sample and a list of features added to generate the adversarial sample.

2.4. Defense strategy

In this section, we will discuss the three proposed defense strategies to counter GAAN based attack.

The first defensive strategy against GAAN based attack is *Adversarial Retraining*. The strategy involves exposing the malware detection model to a set of adversarial samples (M') generated during the GAAN attack to force misclassification in the detection model. Literature suggests many researchers (Maiorca et al. (2019); Khoda et al. (2019), etc.) have explored retraining to improve the performance of classification models. Kurakin et al. (2017) used retraining on the Inception v3 model to significantly increase the detection model's robustness against single-step attacks. Adversarial retraining requires retraining the detection model on a newly created balanced dataset containing original samples and adversarial examples (with their correct labels). Exposing the adversarial samples during retraining helps the detection model to learn patterns and identify them in the future.

The second defensive strategy involves retraining the malware detection model using the Generative Adversarial Network (GAN). Usama et al. (2019) used GANs to produce adversarial samples to evade the detection model and later proposed a defensive mechanism by retraining the model using GAN samples. Our proposed GAN architecture consists of two neural networks, namely Generator *G* and Discriminator *D*. The generator and discriminator models try to optimize their actions by undermining each other and minimizing the overall loss. During training, the generator adds noise(s) to a malware sample to construct an adversarial sample that can fool the discriminator (detection model). However, there is no restriction to the number of features added during GAN generated adversarial samples. The set of adversarial samples (M') (with their correct labels) created by the generator are later added to the original dataset. Finally, the detection models are retrained on the newly created balanced dataset containing both original and adversarial samples.

The third proposed defensive strategy is *Hybrid Distillation*, which is an updated version of distillation. Hinton et al. (2015) introduced distillation as a compression method to transfer knowledge from a larger (pre-trained) model to a smaller model. Later Papernot et al. (2016) used distillation as a defense mechanism against adversarial attacks on DNN. The first step in this method is to train a detection model (*Model*) to learn the class probability distribution of the dataset *D*. Later during distillation, a second model (*Model'*) is trained on the same dataset *D* but mapped to the class probabilities of the first model's output (rather than just the class labels). The proposed *Hybrid Distillation* combines *Adversarial Retraining* and *Distillation* to construct a larger dataset containing adversarial samples (with correct label) generated during attack in addition to the original dataset. The rest of the steps in hybrid distillation are the same as the distillation.

3. Experimental setup

This section will first elaborate on the dataset and process of feature extraction followed by classification algorithms and performance metrics used to construct malware detection models.

3.1. Dataset (malware and benign applications)

Google Play store is the official marketplace to download Android applications. However, all the applications on the above platform are not always benign. Arp et al. (2014) downloaded applications from the Google Play store and found many to be malicious. They collected all the malicious applications and put them in *Drebin Dataset*. The dataset contains 5, 560 malicious applications from more than 179 malware families. We use the benchmarked Drebin dataset for representing malicious applications in all our experiments. Since the Drebin dataset did not contain any benign applications, we downloaded more than 8000 Android applications from the Google Play store. We used the services of VirusTotal to validate whether the downloaded applications are benign or not. VirusTotal is a subsidiary of Alphabet Inc., which aggregates more than 50 antivirus and scan engines. We upload all the downloaded applications on VirusTotal to identify and separate benign applications. We labeled an application as benign only if all the antivirus from VirusTotal label it as benign. Given the above criteria, we found only 5721 Android applications to be pure benign from the downloaded set, and the rest are discarded. So the final dataset contains 5553 malware and 5721 benign Android applications.

3.2. Feature extraction

Features are the backbone of any ML/DL based solution. We extracted two sets of features (permission and intent) from the Android applications to construct malware detection models. The Android permission system is designed to provide security to the system, user, and data, whereas the intent system is developed to handle communication between app components. The Android applications are first disassembled using *Apktool* which generates various files like *AndroidManifest.xml*, *resources.arc* etc. The AndroidManifest.xml file contains the declaration of app package name, components of the app, permission, hardware and software feature, etc. We also develop *Master Permission List* (refer Rathore et al. (2020b)) and *Master Intent List* (refer Sewak et al. (2020)), which contain all permissions and intents from the official Android documentation. Then another parser scans through each decompiled application to prepare a list of permission(s) and intent(s) used in that application. Finally, we combine all the above information to develop two feature vectors, namely *Android Permission* and *Android Intent* representing the permissions and intents used by applications in the dataset. The feature vector for Android permission and intent contains 195 permissions and 273 intents, respectively.

3.3. Classification algorithms

We validate the proposed adversarial attack and defensive strategies by choosing twelve different classification algorithms to construct twelve distinct Android malware detection models. These classification algorithms are drawn from four categories:

- *Classical Machine Learning Algorithm*
  - — Decision Tree (DT)
  - — Support Vector Classifier (SVC)
- *Bagging based Algorithm*
  - — Random Forest (RF)
  - — Extra Trees Classifier (ET)
  - — Bagging based SVC (Bagged SVC)
- *Boosting based Algorithm*
  - — Gradient Boosting (GB)
  - — Adaptive Boosting (AB)
  - — eXtreme Gradient Boosting (XGB)
- *DNN based Architecture*
  - — DNN with 1 hidden layer (DNN-1L)
  - — DNN with 3 hidden layer (DNN-3L)
  - — DNN with 4 hidden layer (DNN-4L)

— DNN with 5 hidden layer (DNN-5L)

This work aims to explore all the twelve classification algorithms. Rathore et al. (2020b) contains a detailed description of all the developed malware detection models. Later we find adversarial vulnerabilities in detection models followed by defensive strategies to strengthen the malware detection models.

3.4. Performance metrics

The following metrics are used for performance measurement of malware detection models in our experiments:

- *True Positive* (TP) is when both the prediction outcome and the ground truth are true. It represents a correct malware prediction by the detection model.
- *False Positive (FP)* is when the ground truth is false, but the model prediction is true. It represents a prediction error when a benign application is misclassified as malware by the detection model.
- *True Negative (TN)* is when both the prediction outcome and ground truth are false. It represents a correct benign prediction by the detection model.
- *False Negative (FN)* is when the ground truth is true, but the model prediction is false. It represents a prediction error when a malicious application is misclassified as benign by the detection model.
- *Accuracy* represents the ratio of the sum of correct predictions across all classes over the total number of predictions. The higher accuracy of the detection model indicates its strong ability to predict the ground truth correctly.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

- *Precision* represents the ratio of correctly predicted malicious applications to the total number of applications predicted as malicious.

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

- *Recall* represents the ratio of malicious applications correctly predicted to the total number of malicious applications in the dataset.

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

- *Area under the ROC Curve (AUC)* measures the performance of the detection model across varying classification thresholds. It quantifies the trade-off between false positive and true positive rate.
- *Fooling Rate (FR)* is the percentage of the total number of malicious applications successfully modified to force misclassification in malware detection model.

$$FR = \frac{M' - FN}{M - FN} \times 100 \tag{8}$$

where M' represents the number of misclassified samples, M represents the number of malicious samples in the dataset, FN represents the number of false negatives on the initial malware detection model.

4. Experimental results and discussion

This section starts with a discussion on the performance of twelve baseline malware detection models. Later in the section, we will elaborate on the results achieved by adversarial attacks and defenses.

4.1. Baseline malware detection models

The proposed framework's (Fig. 1) next step is to build various Android malware detection models using a variety of classification algorithms. We develop two feature sets in the feature extraction phase, namely *Android permission* and *Android intent*. We perform exploratory data analysis on the feature sets to understand the data better. Then we construct twelve distinct detection models based on classical machine learning (DT, SVC), bagging based ensembles (RF, ET, Bagged SVC), boosting based ensembles (GB AB, XGB), and deep neural networks (DNN-1L, DNN-3L, DNN-4L, DNN-5L). The models are trained on 70% of the dataset, and the remaining 30% is used for testing. Separate detection models are created for permission and intent feature set to validate the generalizability of the proposed approach. The detection model's performance is evaluated based on accuracy, precision, recall, and AUC.

Table 1 shows the performance of the malware detection model using *Android permissions*. The highest malware detection accuracy is achieved by ET (95.27%) based model followed by RF (95.09%), whereas the lowest was attained by DNN-1L (89.74%). The twelve different detection models achieve an average accuracy of 93.35%. We observe a small variance between the accuracy of various models with ensemble-based models performing the best. A similar pattern is observed with AUC as well, where the highest AUC score was achieved by the ET-based model (96.34%), and the lowest was attained by DNN-1L (91.80%). The average AUC of all detection models is 94.19% with ensemble-based models outperforming machine learning and deep neural networks based models.

We conduct a second set of experiments (Table 1) by training the malware detection models using Android intents. Here, RF and DT based detection models achieved the highest accuracy of 81.97%, whereas the lowest was reported by DNN-1L (77.99%). Overall, intent-based models attain a lower average accuracy of 80.17% in contrast with permission-based models. However, intent-based models achieved a high average AUC of 90.23%, which signifies their ability to distinguish malware and benign classes. The ET based model achieves the highest AUC (92.18%) while the DNN-1L model obtained the lowest AUC (87.80%). In addition to these metrics, intent-based detection models show a small variance in AUC/accuracy similar to that observed with permission-based models.

4.2. Adversarial attack on malware detection models

The baseline malware detection models (permission and intent) are now investigated for robustness by performing adversarial attacks on them. We propose a GAAN based attack to generate adversarial samples by adding a small number of intelligent perturbations in malware samples. The gradients are generated using the GAAN model consisting of three hidden layers containing 512, 256, and 64 neurons. Each hidden layer uses the ReLU activation function and a dropout rate of 0.4 probability for better generalization. Two separate GAAN models (one each for Android permissions and intents) are trained on the dataset. The gradients produced by the GAAN model are then used to modify malicious

Table 1
Performance of various baseline malware detection models using Android permissions and intents.

Classification Algorithm(s)	Android Permission				Android Intent			
	Accuracy	**Precision**	**Recall**	**AUC**	**Accuracy**	**Precision**	**Recall**	**AUC**
DT[a]	93.23%	92.96%	92.90%	93.25%	81.97%	91.06%	70.29%	92.18%
SVC[b]	93.67%	94.27%	92.41%	94.55%	78.36%	86.43%	66.50%	88.14%
RF[c]	95.09%	95.90%	93.77%	96.10%	81.96%	90.97%	70.36%	92.09%
ET[d]	95.27%	96.14%	93.89%	96.34%	81.97%	91.06%	70.29%	92.18%
Bagged SVC[e]	94.09%	95.22%	92.28%	95.48%	78.29%	86.56%	66.20%	88.29%
GB[f]	91.69%	92.35%	90.12%	92.74%	79.24%	87.54%	67.46%	89.11%
AB[g]	90.57%	91.62%	88.40%	92.09%	78.06%	85.12%	67.19%	86.86%
XGB[h]	94.41%	94.97%	93.27%	95.22%	81.07%	89.71%	69.55%	90.98%
DNN-1L[i]	89.74%	91.25%	86.91%	91.80%	77.99%	86.02%	66.05%	87.80%
DNN-3L[j]	93.97%	94.19%	93.15%	94.46%	80.88%	90.51%	68.34%	91.78%
DNN-4L[k]	94.15%	93.67%	94.14%	93.92%	81.04%	90.66%	68.56%	91.91%
DNN-5L[l]	94.27%	94.12%	93.89%	94.37%	81.24%	90.20%	69.46%	91.44%

[a] Decision Tree.
[b] Support Vector Classifier.
[c] Random Forest.
[d] Extra Trees Classifier.
[e] Bagging based Support Vector Classifier.
[f] Gradient Boosting.
[g] Adaptive Boosting.
[h] eXtreme Gradient Boosting.
[i] Deep Neural Network with 1-layer.
[j] Deep Neural Network with 3-layer.
[k] Deep Neural Network with 4-layer.
[l] Deep Neural Network with 5-layer.

samples by adding a constrained number of permissions/intents. The modified malware sample is then tested against the baseline malware detection models for forced misclassification. The modified application's functional behavior is ensured by restricting modification operation to addition only. It also signifies that we are only adding permission/intent (which are initially not used) to generate the adversarial samples. The adversarial attack's performance can be measured by fooling rate, accuracy, etc. The adversarial attack aims to reduce the baseline malware detection models' accuracy by adding intelligent perturbations in malware samples to force misclassification, thereby increasing the fooling rate.

4.2.1. Fooling rate against baseline detection models

During the construction of baseline malware detection models, few malicious samples are misclassified as benign due to the classifier's inability to fit the data correctly and are counted as false negatives. However, the proposed adversarial attack intentionally modifies malicious samples by adding perturbation to force misclassification in the detection models. The percentage of malicious samples successfully modified for misclassification is called fooling rate. The GAAN based adversarial attack strategy aims to minimize the number of modifications required in each malicious sample for misclassification while maximizing the total number of successfully modified malicious samples in the dataset. Therefore the proposed attack strategy starts with a maximum of one modification allowed in each malicious sample against the target detection model(s) and then gradually increases the threshold to ten modifications.

Fig. 2 shows the fooling rate achieved by the GAAN attack against twelve permission-based baseline malware detection models. The 1-bit adversarial attack achieved the highest fooling rate against the DT-based model (49.67%), whereas the lowest fooling rate is attained against the Bagged SVC model (8.15%). The 1-bit attack policy reaches an average fooling rate of 21.86% across all twelve permission-based detection models. The fooling rate due to adversarial attacks further improves as we allow more modifications in the malicious samples. The 2-bit and 3-bit attacks achieved an average fooling rate of 49.70% and 71.53%, respectively. The 5-bit attack achieves further improvement by reaching more than 85% fooling rates among all the permission-based detection models. The boosting and DNN based models appear to be more susceptible to the gradient-based attack. The 10-bit attack further achieves more than 99% fooling rate against nine detection models and an absolute 100% fooling rate against DNN-4L and DNN-5L. Thus we can conclude that with a *maximum of* 5.1% *permission-based perturbation* we can convert any malicious sample to force misclassification against any permission-based malware detection model.

In the second set of experiments (Fig. 3) we perform GAAN based adversarial attacks against intent-based malware detection models by modifying intents in the Android applications. The 1-bit and 2-bit adversarial attack achieved an average fooling rate of 32.82% and 53.91%, respectively, among twelve intent-based detection models. The 10-bit adversarial attack reached an average fooling rate of 90.71% for all detection models. The above results show that a maximum of 3.6% intent-based permutation in any malware sample during the adversarial attack is most likely to fool any intent-based malware detection model. In general, we observed that DNN based model(s) constructed using either permission or intent are more vulnerable to adversarial attacks than others. We also found that the first few permission/intent modifications play a critical role in adding benign nature to the malicious sample to be misclassified. These results solidify our approach in exploiting the statistical differences between benign and malicious samples to fool the malware detection models.

4.2.2. Vulnerability list (permission and intent)

The permissions extracted from Android applications can help the malware detection models to differentiate between benign and malware samples. However, the GAAN based adversarial attack policy modifies a particular set of permissions/intents to force misclassifications in the detection models. We have listed the set of permissions/intents that are frequently getting modified during the adversarial attack in the *vulnerability list*. We found a particular set of permissions with benign characteristics when added in malicious samples force misclassification in the detection models.

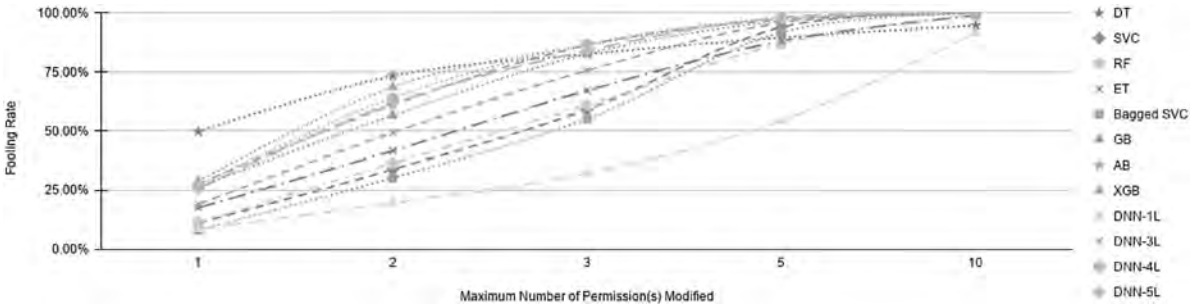

Fig. 2. Performance of permission-based malware detection models against GAAN attack concerning fooling rate.

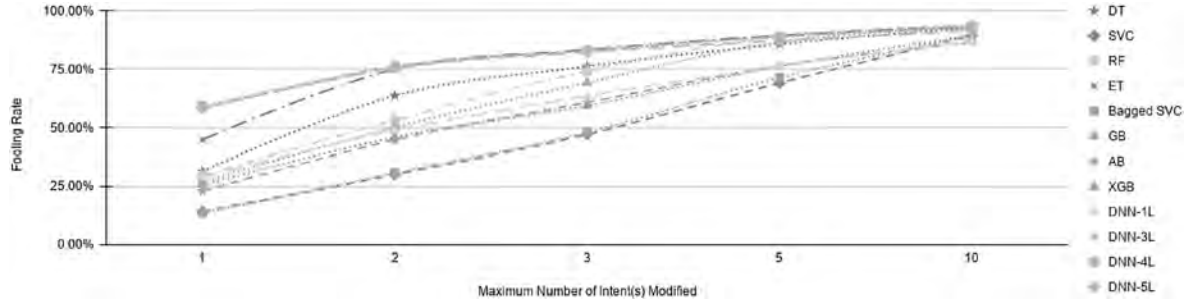

Fig. 3. Performance of intent-based malware detection models against GAAN attack concerning fooling rate.

Fig. 4(L) lists the ten most modified permissions during the 1-bit adversarial attack (which achieved an average fooling rate of 21.96%) on twelve detection models. We observe that five permissions account for most of the modifications performed for fooling in the malicious samples. These are *vulnerable permissions* when added, induce benign characteristics in a malicious sample, and force misclassification in detection models. Fig. 4(R) represents the ten most frequently modified permissions in the 10-bit attack (which achieved an average fooling rate of 98.68%) on twelve malware detection models. Firstly there is a significant rise in the average fooling rate from 1-bit to 10-bit attacks against the detection models. However, the pie chart shows a similar set of permissions, with the top five most frequently modified permissions accounting for most misclassifications. We also found that nine out of the ten most modified permissions in 1-bit and 10-bit are the same. It also signifies that their importance varies with the maximum number of modifications allowed for the attack, but the permission set remains the same. Hence we can infer that the *vulnerability list* containing Android permissions is most likely to help the adversary fool the target detection models.

In the second set of experiments, we have created the *vulnerable list* of intents during the GAAN based adversarial attack on malware detection models. The gradient-based attack strategy exploits the intents by adding them in the malicious samples to force misclassification in the detection model. Fig. 5(L) shows the ten most modified intents during the 1-bit adversarial attack on twelve intent-based detection models. Like permission, the maximum number of misclassified samples are created by adding just five intents. Fig. 5(R) shows the ten most modified intents during the 10-bit attacks on twelve intent-based detection models. The majority of intents in the 10-bit attack remain the same as those observed during the 1-bit attack, with only changes in the total percentage contribution. It signifies that the same set of intents of benign characteristics are added in a malicious sample during the 1-bit and 10-bit attack. There is also a similarity between 1-bit and 10-bit attack in terms of the type of intent getting modified. Nine out of ten most modified intent are of the class *android.intent.action*. Hence, from the above analysis, we derive the set of *vulnerable list* of intents that can aid the adversary for force misclassification in the detection models.

4.3. Adversarial defence for malware detection models

After successfully performing the GAAN based adversarial attack on malware detection models, we proposed three defensive strategies to improve the robustness and strength of detection models against similar attacks. We designed three defensive strategies based on adversarial retraining, GAN retraining, and hybrid distillation for detection models. The gradient-based attack has generated adversarial malware samples M' to force misclassification in detection models. The adversarial retraining strategy involves retraining of the detection models on a balanced dataset, including the M' samples with correct labels. The GAN based defense strategy requires training a GAN to generate adversarial examples, which are then added to the dataset with correct labels and further used for retraining detection models. The hybrid distillation combines distillation and adversarial retraining, where we perform distillation on the augmented dataset with adversarial samples rather than train on the original dataset. This combined version of hybrid distillation has the potential to perform better than both of the individual components. The distillation part helps the detection models to generalize well and curb overfitting issues. The class imbalance issue in the above defense strategies is solved by oversampling of the minority class.

We applied all three defense strategies on permission-based malware detection models. We observed a very slight variation (±2%) in AUC/accuracy of all the twelve permission-based detection models. In the machine learning models, we found a slight increase in model accuracy after using defensive techniques. It can be justified as more data is available while retraining with defensive methods compared to baseline models. A similar pattern is observed when defense strategies are applied to intent-based

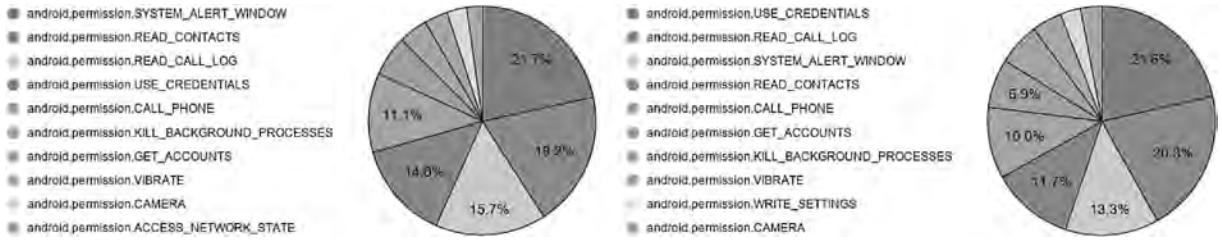

Fig. 4. Distribution of 10 most frequently modified permissions during 1-bit (L) and 10-bit (R) GAAN attack on twelve detection models.

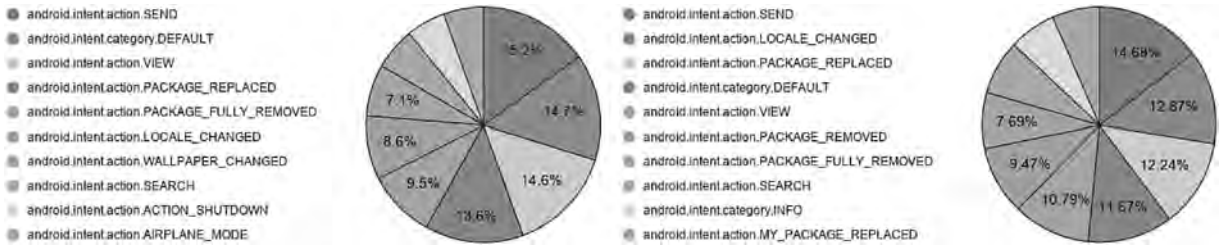

Fig. 5. Distribution of 10 most frequently modified intent during 1-bit (L) and 10-bit (R) GAAN attack on twelve detection models.

detection models. We found a slight variation (±3.5%) in the AUC/accuracy of all the twelve intent-based detection models. The increase in AUC/accuracy of retrained models compared to their baselines can again be attributed to increased data being fed to the model.

4.3.1. Adversarial attack (10-bit) on defense strategies

The robustness of the proposed framework can be validated by again performing the GAAN-based attack on malware detection models trained with three adversarial defense strategies. Fig. 6 shows the accuracy after the 10-bit attack on permission-based detection models built using three different defense strategies. We achieved an average fooling rate of 98.67% with the 10-bit attack on baseline detection models leading to an average 45−50% decrease in different models' accuracy. However, the performance of all the detection models is drastically improved after using adversarial defense. Even after the 10-bit attack, the highest accuracy is achieved by the RF-based model (95.80%) followed by an ET-based model (95.16%) built using the adversarial retraining defense strategy. The highest average accuracy improvement is shown by adversarial retraining strategy (55.86%) followed by hybrid distillation (54.21%) and GAN retraining (36.87%) for twelve detection models.

The second set of experiments is conducted on intent-based malware detection models (Fig. 7) trained using three defense strategies. The 10-bit GAAN-based attack on baseline models achieved a fooling rate of 90.71%, which resulted in an average accuracy of 49.83% in twelve malware detection models. All three defense strategies show a considerable improvement even after the 10-bit gradient-based adversarial attack. The adversarial retraining

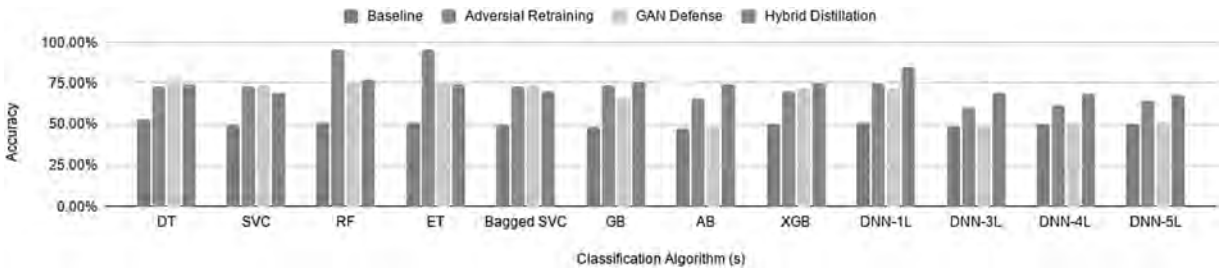

Fig. 6. Performance of permission-based malware detection models (baseline and defense strategies) against GAAN attack.

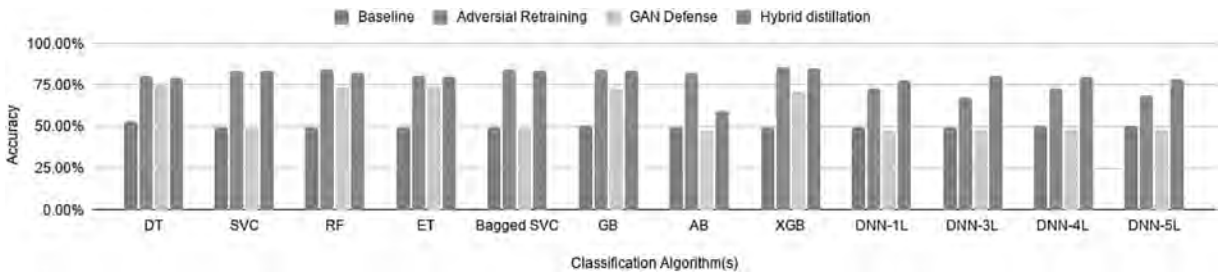

Fig. 7. Performance of intent-based malware detection models (baseline and defense strategies) against GAAN attack.

strategy for detection models achieved an average increase of 58.18% whereas GAN based defense and hybrid distillation showed improvement of 17.62% and 59.14% respectively. We observe that the mixture of distillation improves model generalization, and exposure to adversarial samples through adversarial retraining aids in more robustness. Overall adversarial retraining and hybrid distillation based defense strategy are most effective to make the malware detection models more robust, while GAN based retraining has been the least effective.

5. Related work

Malware detection is an endless battle between malware designers and the anti-malware community. The current antivirus systems are not able to cope up with ever-increasing malware attacks. Thus the anti-malware research community is investigating to build state-of-the-art malware detection models based on ML/DL.

Ye et al. (2017) suggests that the process of development of malware detection model is usually divided into two stages: *feature extraction* and *classification/clustering*. Arp et al. (2014) in *Drebin* extracted features like permission, intent, API calls, etc. and used SVC to construct a malware detection model that attain 94% accuracy. However, the model used 545000 features and thus suffered from the curse of dimensionality. Karbab et al. (2018) proposed *MalDozer* by extracting API call methods and used deep learning models to perform malware and its family detection. MalDozer achieved an F1-Score of 96%–99% but used millions of parameters, and thus again suffers from the curse of dimensionality. Researchers in the literature (Yan and Yan (2018); Qiu et al. (2020), etc.) have suggested many malware detection models showing promising results. Pitropakis et al. (2019) reveals that classification models built using ML & DL are vulnerable to adversarial attacks. However, minimal literature is available on adversarial attacks and defense for malware detection models.

Goodfellow et al. (2015) suggests that the vulnerability of the model against adversarial attacks is due to its linear nature and not because of overfitting or other issues. Ji et al. (2019) assumed white-box scenario and proposed DEEPARMOUR, which performs voting between three classifications models to make the detection resilient against adversarial attacks. They also proposed adversarial retraining as a defensive mechanism to counter the attack. Taheri et al. (2020) also assumed white-box scenario and proposed five attack strategies to perform adversarial attacks on the detection models. However, their attack results showed an increase in false-positive rate, which ideally should not change due to adversarial attack on detection models. They also proposed adversarial retraining and GAN-based defense for detection models, which showed limited improvements. Reinforcement learning can also be used to craft adversarial examples to force misclassifications in the detection model. Rathore et al. (2020a) used Q-Leaning to develop two attack strategies: single-policy for white box and multi-policy for grey box scenarios. The adversarial attack strategy attains a limited fooling rate of 44.21%. They also used adversarial retraining as the defense mechanism. Wu et al. (2019) used a generative adversarial network to automatically add perturbations for generating adversarial traffic flow to deceive the detection model. Most of the studies in this domain assume white-box scenarios to design adversarial attacks, which is not entirely correct for real-world malware detection systems. The cost of generating adversarial examples concerning the number of modifications required is rarely discussed. Also, limited literature is available regarding adversarial defense strategies to counter the attacks on malware detection models.

6. Conclusion

Today, smartphones have been an indispensable part of our daily lives, which has drawn massive attention from malware designers. Traditional malware detection techniques like the signature, heuristic, behavior etc., are unable to cope with new-age polymorphic and metamorphic malware. Thus researchers have proposed machine learning/deep learning models for malware detection. However, these models are vulnerable to adversarial attacks that will jeopardize their real-world deployment.

We proposed a framework to validate and construct robust malware detection models against adversarial attacks. Firstly we develop various malware detection models using twelve classifiers (machine learning, bagging, boosting, and deep neural network). We validated the performance of each detection model by performing GAAN-based adversarial attacks on them. We achieved an average fooling rate of 98.98% against twelve permission-based and 90.71% against another twelve intent-based detection models. The above attack executes a maximum of 10 modifications in each malware sample to convert it into an adversarial example. We also develop a vulnerability list of permissions/intents that increases the probability of force misclassifications in detection models. The above results conclude that malware detection models are very vulnerable to adversarial attacks. An adversary can exploit the above scenario to force misclassifications in the detection models.

We also proposed three adversarial defense strategies to counter attacks on malware detection models. Firstly we again developed malware detection models using defense strategies. We achieved almost the same accuracy for all the detection models for both permission and intent features. Then we again performed the GAAN attack on all the detection models and compared the performance. The Hybrid Distillation based defense improved the average accuracy by 54.21% for permission-based and 59.14% for intent-based detection models. The highest accuracy was achieved by the Random Forest model (95.80%) followed by the Extra Tree model (95.16%) with adversarial retraining based defense strategy. Overall the hybrid distillation performed best, followed by adversarial retraining and GAN based defense. The above results conclude that the adversarial defense does improve the robustness of malware detection models and should be validated before any real-world deployment of the detection models.

In this work, we assumed a grey box scenario with the adversary has the capacity to modify the test samples only. However, in the future, we want to explore adversarial attacks and defenses for black-box scenarios where an adversary is assumed to have no knowledge about the system. We would also like to explore poisoning attacks on malware detection models and their defenses.

References

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

Buczak, A.L., Guven, E., 2016. A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications Surveys & Tutorials 18, 1153–1176.

Goodfellow, I.J., Shlens, J., Szegedy, C., 2015. Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (ICLR).

Hinton, G., Vinyals, O., Dean, J., 2015. Distilling the knowledge in a neural network. In: 28^{th} Conference on Neural Information Processing Systems (NIPS).

Ji, Y., Bowman, B., Huang, H.H., 2019. Securing malware cognitive systems against adversarial attacks. In: 2019 IEEE International Conference on Cognitive Computing (ICCC). IEEE, pp. 1–9.

Karbab, E.B., Debbabi, M., Derhab, A., Mouheb, D., 2018. Maldozer: automatic framework for android malware detection using deep learning. Digit. Invest. 24, S48–S59.

Kemp Hootsuite, Simon, 2021. Global digital report. Available: https://datareportal.com/reports/digital-2021-global-overview-report. Last Accessed: Jan 2021.

Khoda, M., Imam, T., Kamruzzaman, J., Gondal, I., Rahman, A., 2019. Selective

adversarial learning for mobile malware. In: 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE). IEEE, pp. 272–279.

Kim, T., Kang, B., Rho, M., Sezer, S., Im, E.G., 2018. A multimodal deep learning method for android malware detection using various features. IEEE Trans. Inf. Forensics Secur. 14, 773–788.

Kurakin, A., Goodfellow, I., Bengio, S., 2017. Adversarial examples in the physical world. In: International Conference on Learning Representations (ICLR).

Li, J., Sun, L., Yan, Q., Li, Z., Srisa-An, W., Ye, H., 2018. Significant permission identification for machine-learning-based android malware detection. IEEE Transactions on Industrial Informatics 14, 3216–3225.

Maiorca, D., Biggio, B., Giacinto, G., 2019. Towards adversarial malware detection: lessons learned from pdf-based attacks. ACM Comput. Surv. 52, 1–36.

McAfee Mobile Threat Report, Q1, 2020. Available: https://www.mcafee.com/content/dam/consumer/en-us/docs/2020-Mobile-Threat-Report.pdf. Last Accessed: Jan 2021.

Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A., 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In: IEEE Symposium on Security and Privacy (S & P). IEEE, pp. 582–597.

Pitropakis, N., Panaousis, E., Giannetsos, T., Anastasiadis, E., Loukas, G., 2019. A taxonomy and survey of attacks against machine learning. Computer Science Review 34, 100199.

Qiu, J., Zhang, J., Luo, W., Pan, L., Nepal, S., Xiang, Y., 2020. A survey of android malware detection with deep neural models. ACM Comput. Surv. 53, 1–36.

Rathore, H., Sahay, S.K., Nikam, P., Sewak, M., 2020a. Robust android malware detection system against adversarial attacks using q-learning. Inf. Syst. Front 1–16.

Rathore, H., Sahay, S.K., Rajvanshi, R., Sewak, M., 2020b. Identification of significant permissions for efficient android malware detection. In: International Conference on Broadband Communications, Networks and Systems. Springer, pp. 33–52.

Sewak, M., Sahay, S.K., 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. IEEE, pp. 1–6.

Symantec's Internet Security Threat Report, 2019. Available: https://www-west.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.pdf. Last Accessed: Jan 2021.

Taheri, R., Javidan, R., Shojafar, M., Vinod, P., Conti, M., 2020. Can machine learning model with static features be fooled: an adversarial machine learning approach. Cluster Comput. 1–21.

Usama, M., Asim, M., Latif, S., Qadir, J., et al., 2019. Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems. In: 15th International Wireless Communications & Mobile Computing Conference (IWCMC). IEEE, pp. 78–83.

Wu, D., Fang, B., Wang, J., Liu, Q., Cui, X., 2019. Evading machine learning botnet detection models via deep reinforcement learning. In: IEEE International Conference on Communications (ICC). IEEE, pp. 1–6.

Yan, P., Yan, Z., 2018. A survey on dynamic mobile malware detection. Software Qual. J. 26, 891–919.

Ye, Y., Li, T., Adjeroh, D., Iyengar, S.S., 2017. A survey on malware detection using data mining techniques. ACM Comput. Surv. 50, 41.