Analysis of Label-Flip Poisoning Attack on Machine Learning Based Malware Detector

Kshitiz Aryal
Department of Computer Science
Tennessee Technological University
Cookeville, TN, USA
karyal42@tntech.edu

Maanak Gupta
Department of Computer Science
Tennessee Technological University
Cookeville, TN, USA
mgupta@tntech.edu

Mahmoud Abdelsalam
Department of Computer Science
North Carolina A&T State University
Greensboro, NC, USA
mabdelsalam1@ncat.edu

2022 IEEE International Conference on Big Data (Big Data) | 978-1-6654-8045-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/BigData55660.2022.10020528

Abstract—With the increase in machine learning (ML) applications in different domains, incentives for deceiving these models have reached more than ever. As data is the core backbone of ML algorithms, attackers shifted their interest towards polluting the training data itself. Data credibility is at even higher risk with the rise of state-of-art research topics like open design principles, federated learning, and crowd-sourcing. Since the machine learning model depends on different stakeholders for obtaining data, there are no existing reliable automated mechanisms to verify the veracity of data from each source.

Malware detection is arduous due to its malicious nature with the addition of metamorphic and polymorphic ability in the evolving samples. ML has proven to solve the zero-day malware detection problem, which is unresolved by traditional signature-based approaches. The poisoning of malware training data can allow the malware files to go undetected by the ML-based malware detectors, helping the attackers to fulfill their malicious goals. A feasibility analysis of the data poisoning threat in the malware detection domain is still lacking. Our work will focus on two major sections: training ML-based malware detectors and poisoning the training data using the label-poisoning approach. We will analyze the robustness of different machine learning models against data poisoning with varying volumes of poisoning data.

Index Terms—Cybersecurity, Poisoning Attacks, Machine Learning, Malware Detectors, Adversarial Malware Analysis

I. Introduction

Machine Learning (ML) techniques have been emerging rapidly, providing computational intelligence to various applications. The ability of machine learning to generalize to unseen data has paved its way from labs to the real world. It has already gained unprecedented success in many fields like image processing [1], [2], natural language processing [3], [4], recommendation systems used by Google, YouTube and Facebook, cybersecurity [5], [6], robotics [7], drug research [8], [9], and many other domains. ML-based systems are achieving unparalleled performance through modern deep neural networks bringing revolutions in AI-based services. Recent works have shown significant achievements in fields like self-driving cars and voice-controlled systems used by tech giants like autopilot in Tesla, Apple Siri, Amazon Alexa, and Microsoft Cortana. With machine learning being applied to such critical applications, continuous security threats are never a bombshell. In addition to traditional security threats like malware attack [10], phishing [11], man-in-the-middle attack [12], denial-of-service [13], SQL injection [14], adversaries are finding novel ways to sneak into ML models [15].

Data poisoning and evasion attacks [16]–[20] are the latest menaces against the security of machine learning models. Poisoning attacks enable attackers to control the model's behavior by manipulating a model's data, algorithms, or hyper-parameters during the model training phase. On the other hand, an evasion attack is carried out during the test time by manipulating the test sample. Adversaries can craft legitimate inputs imperceptible to humans but force models to make wrong predictions. Szegedy et al. [21] discovered the vulnerability of deep learning architecture against adversarial attacks, and ever since, there have been several major successful adversarial attacks against machine learning architectures [22], [23]. Sophisticated attackers are motivated by very high incentives to manipulate the result of the machine learning models. With the current data scale with which machine learning models are trained, it is impossible to verify each data point individually.

In most scenarios, it is unlikely that an attacker gets access to training data. However, with many systems adopting online learning [24], crowd-sourcing [25] for training data, open design principles, and federated learning, poisoning attacks already pose a serious threat to ML models [26]. There have been instances [27] when big companies have been compromised by a data poisoning attack. Malware public databases like VirusTotal[1], which rely on crowdsourced malware files for training its algorithm, can be poisoned by attackers while Google's mail spam filter can be thrown out of track by wrong reporting of spam emails.

Data poisoning relates to adding training data that either leaves a backdoor on the model or negatively impacts the model's performance. Figure 1 shows the architecture of the poisoning attack. In the given figure, the addition of poisoned data in the training bag forces the model to learn and predict so that attackers benefit from it. This type of poisoning is not limited to particular domains but has extended across all ML applications. Label flipping attack is carried out to flip the prediction of machine learning detectors. Among all the existing approaches, we chose one of the simplest poisoning techniques called label poisoning. We swap the

[1]https://www.virustotal.com/

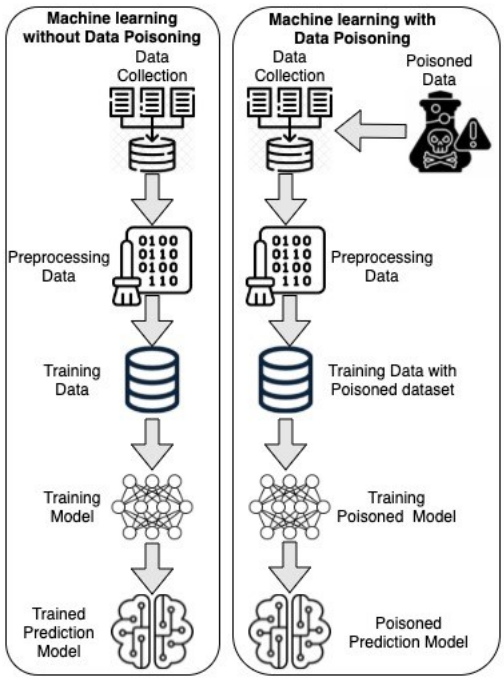

Fig. 1. General architecture for Poisoning Machine Learning Models

existing training data labels in label poisoning to check the ML models' robustness.

In this work, we perform a comparative analysis of different machine learning-based malware detectors' robustness against label-flipping data poisoning attacks. Unlike the existing approaches, we are demonstrating the impact of simple label-switching data poisoning in different malware detectors. We will first train eight different ML models widely used to detect malware, namely Stochastic Gradient Descent (SGD), Random Forest (RF), Logistic Regression (LR), K-Nearest Neighbor Classifier (KNN), Linear Support Vector Machine (SVM), Decision Tree (DT), Perceptron, and Multi-Layer Perceptron (MLP). This will be followed by poisoning 10% and 20% of training data by flipping the label of data samples. All of the models are retrained after data poisoning, and the performance of each model is evaluated. The major contributions of this paper are as follows.

- We taxonomize the existing data poisoning attacks on machine learning models in terms of domains, approaches, and targets.
- We provide threat modeling for adversarial poisoning attacks against malware detectors. The threat is modeled in terms of the attack surface, the attacker's knowledge, the attacker's capability, and adversarial goals.
- We train eight different machine learning-based malware detectors from malware data obtained from VirusTotal and VirusShare[2]. We compare the performance of these malware detectors with training and testing data in terms of accuracy, precision, and recall.
- Finally, we show a simple label-switching approach to poison the data without any knowledge of training models.

[2]https://virusshare.com/

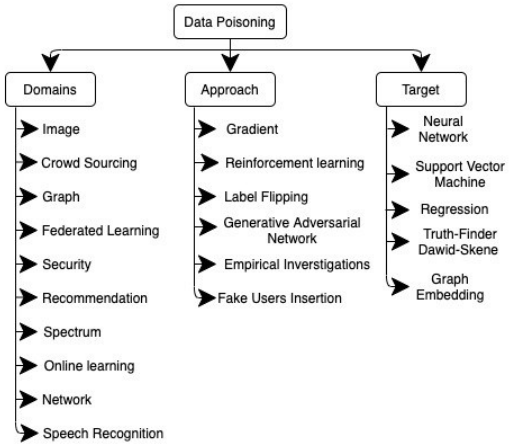

Fig. 2. Taxonomy of poisoning attack on attack domain, approach and target

The performance of malware detectors is analyzed while poisoning 10% and 20% of the total training data.

The rest of the paper is organized as follows. The existing literature for data poisoning attacks in different domains, including malware, is discussed in Section II. Section III provides the threat modeling for data poisoning attacks. An overview of ML algorithms that are used to train the malware detector in this paper is discussed in Section IV. Section V discusses experimental methodology elaborating on the algorithm and the testbed used for the experiment. The evaluation and discussion on the performed experiments are given in Section VI. Finally, Section VII concludes this work.

II. Literature Review

Data poisoning attacks have been used against the machine learning domain for a long time. The existing literature on data poisoning attacks can be taxonomized in terms of attack domains, approach, and the target (victim), as illustrated in Figure 2. The recently trending technologies like crowdsourcing and federated learning are always vulnerable as the veracity of individual data can never be verified. The recent victims of poisoning attacks have spread in security, network, and speech recognition domains. We also classified the major approaches that are taken to produce or optimize the poisoning attacks in Figure 2. The existing data poisoning approaches have targeted almost all the machine learning algorithms ranging from traditional algorithms like regression to modern deep neural network architectures.

Table I summarizes the existing literature on poisoning attacks. Biggio et al. [43] attacked a support vector machine using gradient ascent. To make poisoning attacks closer to the real world, Yang et al. [44] used a generative adversarial network with an autoencoder to poison deep neural nets. Gongalez et al. [45] extended poisoning from binary learning to multi-class problems. Shafahi et al. [28] proposed a targeted clean label poisoning attack on neural networks using an optimization-based crafting method. Shen et al. [31] performed an imperceptible poisoning attack on a deep neural network by clogging the back-propagation from gradient tensors during training while also minimizing the gradient norm. Jiang et

TABLE I
DATA POISONING ATTACKS

	Domains						Approach					Target				
Publications	Image	Crowd Sourcing	Graph	Federated Learning	Security	Online Learning	Gradient	Reinforcement Learning	Label Flipping	GAN	Others	Neural Network	SVM	Regression	Graph Embedding	Customized
Shafahi et al. [28]	√						√				√	√				
Liu et al. [29]	√		√				√									√
Cao et al. [30]				√					√			√				
Shen et al. [31]	√						√					√				
Zhang et al. [32]				√						√		√				
Jiang et al. [33]	√						√							√		
Kwon et al. [34]	√										√	√				
Zhang et al. [35]			√								√				√	
Bagdasaryal et al. [36]	√			√							√	√				
Li et al. [37]		√						√								√
Sasaki et al. [38]					√		√							√		
Zhang et al. [39]	√															
Lovisotto et al. [40]	√					√		√						√		
Li et al. [41]					√				√		√	√				
Kravchik et al. [42]					√		√					√				
This Work					√				√			√	√	√		

Domains:Poisoning domain for crafted attack, Approach: Approach to poison the training data, Target: Target of poisoning attack

al. [33] performed a flexible poisoning attack against linear and logistic regression. Kwon et al. [34] could selectively poison particular classes against deep neural networks. Cao et al. [30] proposed a distributed label-flipping poisoning approach to poison the DL model in federated architecture. Miao et al. [46] poisoned Dawid-Skene [47] model by exploiting the reliability degree of workers. Fang et al. [48] proposed a poisoning attack against a graph-based recommendation system by maximizing the hit ratio of target items using fake users.

In the given Table I, we can observe that only a handful of works have been carried out in the security domain. Sasaki et al. [38] proposed an attack framework for backdoor embedding, which prevented the detection of specific types of malware. They generated poisoning samples by solving an optimization problem and tested it against a logistic regression-based malware detector. To poison the Android malware detectors, Lie et al. [41] experimented backdoor poisoning attack against Drebin [49], DroidCat [50], MamaDroid [51] and DroidAPIMiner [52]. Kravchik et al. [42] attacked the cyber attack detectors deployed in the industrial control system. The back gradient optimization techniques used to pollute the training data successfully poison the neural network-based model. These works have focused their approach on some algorithm testing against some defense mechanism. However, none of the works compared the feebleness of multiple algorithms against data poisoning attacks. In this work, we demonstrate the effectiveness of label switch poisoning of the training data against eight machine learning algorithms widely used in malware detectors.

III. THREAT MODEL: KNOW THE ADVERSARY

All security threats are defined in terms of their goals and attack capabilities. Modeling the threat allows for identifying and better understanding the risk arriving with a threat. A poisoning attack is performed by manipulating the training data either at the initial learning or incremental learning

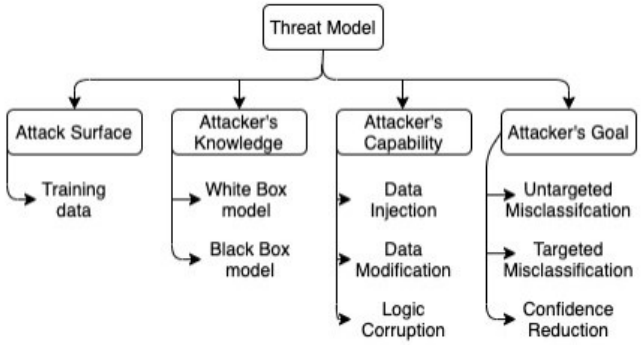

Fig. 3. Threat model for poisoning attack

period. The threat model of a poisoning attack reflects the attacker's knowledge, goal, capabilities, and attack surface, as shown in Figure 3.

Attack Surface: Attack surface denotes how the adversary attacks the model under analysis. Machine learning algorithms require data to pass through different stages in the pipeline, and each stage offers some kind of vulnerability. In this work, we are only concerned about poisoning attacks which make the training data an attack surface.

Attacker's Knowledge: The attacker's knowledge is the amount of information about the model under attack that an attacker has. Based on the amount of knowledge of the attacker, the poisoning approach is determined. Attacker's knowledge can be broadly classified into the two following categories:

- White box model: In the white box model, an attacker has complete information about the underlying target model, such as the algorithm used, training data, hyper-parameters, and gradient information. It's easier to carry out a white box attack due to the information available that helps the attacker to create a worst-case scenario for the target model.
- Black box model: In the black-box model, an attacker only

has information about the model's input and output. An attacker has no information about the internal structure of the model. Black-box models can also be divided further into complete black-box models and gray-box models. In the gray box model, the model's performance for each input the attacker provides can be known. As such, the gray box attack is considered to be relatively easier than the complete black box model.

In this paper, we perform a black box attack on different malware detection models. Our experiments will prove the vulnerability of these models to random label poisoning attacks without having any information about the models.

Attacker's Capability: The attacker's capability represents the ability of an adversary to manipulate the data and model in different stages of the ML pipeline. It defines the sections that can be manipulated, the mechanism used for manipulation, and constraints to the attacker. Poisoning can be carried out in a well-controlled environment if the attacker has complete information about the underlying model and training data. Attacker capabilities can be classified into the following categories:

- Data Injection: It is the ability to insert new data into the training dataset, leading machine learning models to learn on contaminated data.
- Data Modification: It is the ability to access and modify the training data as well as the data labels. Label flipping is a well-known approach carried out in poisoning attack domains.
- Logic Corruption: It is the ability to manipulate the logic of ML models. This ability is out of scope for data poisoning and is considered a model poisoning approach.

Adversarial Goals: The attacker's objective is to deceive the ML model by injecting poisoned data. However, poisoning training data might differ depending on the goals of an attacker. Attacker goals can be categorized as:

- Untargeted Misclassification: An attacker tries to change the model's output to a value different than the original prediction. Untargeted misclassification is a relatively easier goal for attackers.
- Targeted Misclassification: An attacker's goal is to add a certain backdoor in the models so that particular samples are classified to a chosen class.
- Confidence Reduction: An attacker can also poison training data to reduce the confidence of the machine learning model for a particular prediction. In this approach, changing the classification label is unnecessary, but reducing the confidence score is enough to meet the attacker's goal.

Our paper aims to cause the malware detector models to misclassify. However, since we are dealing with binary classification, it can be considered either targeted or untargeted misclassification.

IV. Overview of Machine Learning Algorithms

Almost all of the ML architectures have already been victimized by data poisoning attacks. In this section, we will brief some ML architectures in which we performed data poisoning attacks later in this paper.

Stochastic Gradient Descent: Stochastic gradient descent (SGD) is derived from the gradient descent algorithm, which is a popular ML optimization technique. A gradient gives the slope of the function and measures the degree of change of a variable in response to the changes of another variable. Starting from an initial value, gradient descent runs iteratively to find the optimal values of the parameters, which are the minimal possible value of the given cost function. In Stochastic Gradient Descent, a few samples are randomly selected in place of the whole dataset for each iteration. The term batch determines the number of samples to calculate each iteration's gradient. In normal gradient descent optimization, a batch is taken to be the whole dataset leading to the problem when the dataset gets big. Stochastic gradient descent considers a small batch in each iteration to lower the computing cost of the gradient descent approach while working with a large dataset.

Random Forest: A random forest is a supervised ML algorithm that is constructed from an ensemble of decision tree algorithms. Its ensemble nature helps to provide a solution to complex problems. The random forest is made up of a large number of decision trees that have been trained via bagging or bootstrap aggregation. The average mean of the output of constituent decision trees is the random forest's ultimate forecast. The precision of the output improves as the number of decision trees used grows. A random forest overcomes the decision tree algorithm's limitations by eliminating over-fitting and enhancing precision.

Logistic Regression: The probability for classification problems is modeled using logistic regression, which divides them into two possible outcomes. For classification, logistic regression is an extension of the linear regression model. For regression tasks, linear regression works well; however, it fails to replicate for classification. The linear model considers the class a number and finds the optimum hyperplane that minimizes the distances between the points and the hyperplane. As it interpolates between the points, it cannot be interpreted as probabilities. Because there is no relevant threshold for class separation, logistic regression is applied. It is a widely used classification algorithm due to its ease of implementation and strong performance in linearly separable classes.

K-Nearest Neighbors (KNN) Classifier: The KNN algorithm relies on the assumption that similar things exist in close proximity. It is a non-parametric and lazy learning algorithm. KNN does not carry any assumption for underlying data distribution. It does not require training data points for model generation, as all the training data are used in a testing phase. This results in faster training and a slower testing process. The costly testing phase will consume more time and memory. In KNN, K is the number of nearest neighbors and is generally considered odd. KNN, however, suffers from the curse of dimensionality. With increased feature dimension, it requires more data and becomes prone to overfitting.

Support Vector Machine (SVM): A support vector machine

Algorithm 1: Data Poisoning Algorithm

```
Input: Non-poisoned feature set
Output: Poisoned feature set
Data: Static features obtained from malware and
      benign training set
for all the samples do
    Train the machine learning models and measure
     the performance
    for 10% each of Malware and Benign data do
        if Training label is not flipped then
            label=Get training label of given data
            if label==0 then
                Flip the label to 1
            else if label==1 then
                Flip the label to 0
    Train all the models and measure the performance
    for 20% each of Malware and Benign data do
        if Training label is not flipped then
            label=Get training label of given data
            if label==0 then
                Flip the label to 1
            else if label==1 then
                Flip the label to 0
    Train all the models and measure the performance
```

is a popular supervised ML algorithm applied in both classification and regression tasks. SVM aims to find a hyperplane that classifies the data points. In SVM, there are several possible hyperplanes, and we need to determine the optimal hyperplane that maximizes the margin between the two classes. Hyperplanes are the decision boundary for SVM, where data points near to hyperplane are the support vectors. Due to its effectiveness in high dimensional spaces and memory-efficient properties, it is widely adopted in different domains.

Multi-Layer Perceptron: The term 'Perceptron' is derived from the ability to perceive, see, and recognize images in a human-like manner. A perceptron machine is based on the neuron, a basic unit of computation, with a cell receiving a series of pairs of inputs and weights. Although the perceptron was originally thought to represent any circuit and logic, non-linear data cannot be represented by a perceptron with only one neuron. Multi-Layer Perceptron was developed to overcome this limitation. In multi-layer perceptron, the mapping between input and output is non-linear. It has input and output layers and several hidden layers stacked with numerous neurons. Because the inputs are merged with the initial weights in a weighted sum and applied to the activation function, the multi-layer perceptron falls under the category of feedforward algorithms. Each linear combination is propagated to the following layer, unlike with a perceptron.

V. Experimental Methodology

In this paper, we are using the label-flipping approach to poison the training data. With source class C_S and a target class C_T from a set of classes C, the dataset D_I is poisoned. The detailed poisoning performed in the paper is shown in Algorithm 1. We perform a label poisoning attack of different volumes to training data without guiding the poisoning mechanism through machine learning architecture or the loss function. It is an efficient way to showcase the ability of random poisoning to hamper the model's performance. We are training all eight malware detector models three times in total. As illustrated in Algorithm 1, we begin the model training with clean data without adding any noise. After recording the model's performance on clean data, we proceed towards the first stage of poisoning our data. We take 10% of shuffled training data belonging to each malware and benign class, and we change their labels. We retrain all the models and again measure the performance of the models. We repeat the same operation with 20% of shuffled training data. The percentage of poisoned data is taken randomly for this experimental purpose, as the goal is to show the impact on the models. The algorithm we followed in carrying out this experiment is not a novel approach but a generic approach to poison the data.

A. Experimental Environment and Dataset

All the experiments are performed in Google-Colab using Google's GPU. All the implementation will be worked around using python libraries and Scikit-Learn. The training dataset [53] is obtained from the Kaggle repository, where data are collected from VirusTotal and VirusShare. The dataset comprises windows PE malware and benign files processed through static executable analysis. The dataset comprises 216,352 files (75,503 benign files and 140,849 malware files) with 54 features.

VI. Evaluation Results and Analysis

A. Data Pre-processing and Transformation

We begin our experiment by loading data from Kaggle dataset [53]. To clean the data, we followed two different approaches. First, we ignored rows that are missing more than 50% of data, whereas we replaced the null values with the arithmetic mean value of the column for rows with less than 50% missing values. Second, we normalized the data by scaling the values from 0 to 1. Afterward, 85% of data were used for training purposes while the remaining 15% were used for testing purposes. We trained selected eight machine learning models with standard hyper-parameters for each model. We didn't tweak many machine learning parameters to fine-tune the detection accuracy, resulting in significant overfitting in a few models.

B. Performance Indicators

We evaluated the malware detectors' performance using the following metrics:

TABLE II
MALWARE DETECTION TRAINING RESULT

Algorithm	Clean Data							
	Training Data				Testing Data			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
Stochastic Gradient Descent	93.41	92.49	88.29	90.34	72.98	58.6	78.77	67.20
Decision Tree	99.96	99.98	99.91	99.94	59.65	44.5	59.85	51.05
Random Forest	99.97	99.92	99.97	99.94	83.65	98.82	54.12	69.94
Logistic Regression	93.2	92.21	87.94	90.02	92.33	92.24	85.36	88.67
KNN Classifier	98.38	97.33	98.05	97.69	97.42	96.38	96.25	96.31
Support Vector Machine	93.15	92.44	87.51	89.91	92.03	90.89	85.94	88.34
Perceptron	90.93	88.6	84.91	86.72	75.39	60.28	87.86	71.50
Multi-Layer Perceptron	91.28	91.07	83.16	86.94	71.93	57.45	77.66	66.04

TABLE III
MALWARE DETECTION PERFORMANCE WITH 10% POISONING DATA

Algorithm	10% Poisoned Data							
	Training Data				Testing Data			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
Stochastic Gradient Descent	85.12	82.49	77.14	79.73	72.39	64.23	61.38	62.77
Decision Tree	96.77	99.44	92.01	95.58	51.92	38.33	43.98	40.96
Random Forest	96.77	98.92	92.51	95.61	80.13	82.68	60.22	69.68
Logistic Regression	84.51	82.29	75.39	78.69	83.26	81.06	72.91	76.77
KNN Classifier	89.49	85.47	87.1	86.28	86.59	83.1	81.15	82.11
Support Vector Machine	84.75	82.84	75.42	78.96	66.99	63.14	31.16	41.73
Perceptron	77.94	67.78	79.69	73.25	40.16	25.89	31	73.25
Multi-Layer Perceptron	83.85	82.72	72.58	77.32	83.33	82.81	70.74	76.30

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

$$\text{Precision} = \frac{TP}{TP + FP}, \text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1-score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$$

A positive outcome corresponds to a malware sample, while a negative result corresponds to a benign example. TP, TN, FP, and FN are true positives, true negatives, false positives, and false negatives, respectively. Accuracy is the percentage of correct predictions on the given data. Precision measures the ratio between true positives and all the positives. Recall provides the ability of our model to predict true positives correctly. The F1 score is the harmonic mean, the combination of a classifier's precision and recall.

C. Results and Discussion

Table II shows the accuracy, precision, and recall for training and testing data. Stochastic Gradient Descent, Decision Trees, Random Forest, and Perceptron looked overfitted to training data compared to other models. Since the data volume is a little bit high, decision tree-based classifiers are prone to overfitting problems. We used shallow layer neural networks leading perceptron to overfit in the data. However, classifiers like logistic regression, KNN classifier, and Support Vector Machine have shown the best performance in all three metrics. We have compared the performance of both the training and testing sets as we have only poisoned the training data while preserving the test data from attack.

We flipped the labels of 10% training data as a poisoning attack. On poisoning 10% of total data, the performance metric for each detector is displayed in Table III. The results show the robustness of decision trees and random forest-based malware detectors compared to other malware detectors. We further poisoned 20% of total training data to see the impact of increased poisoned data in each model, whose results are shown in Table IV. The left-most confusion matrix in each of the figures from Figure 4 to Figure 11 shows the number of TP, TN, FP, and FN for each classifier on clean data, whereas the middle and right one shows results with 10% and 20% poisoning, respectively. In the confusion matrix, label '0' is for malware, and label '1' is for benign samples. The top-left corner in the confusion matrix gives True Positive, the top-right corner gives False Positive, the bottom-left gives False Negative, and the bottom-right corner gives True Negative samples.

D. Analysis and Observations

The goal of this work is to show the vulnerability of popular machine-learning models that are used for malware detection. Results in Tables II, III and IV reflect the limitations of all the experimented machine learning models even with the basic label poisoning attack. Figure 12 shows the ROC curve, comparing the models' performance on the clean data, 10% and 20% poisoned data. In the ROC curve, the blue curve corresponds to the performance of clean data, the orange curve corresponds to 10% poisoned data, and the green curve corresponds to the 20% poisoned data. The curve closest to the top-left corner is the one performing best. We can infer from the graph that logistic regression, K-Nearest Neighbors,

TABLE IV
MALWARE DETECTION PERFORMANCE WITH 20% POISONING DATA

Algorithm	20% Poisoned data							
	Training Data				Testing Data			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
Stochastic Gradient Descent	78.56	75.65	70.21	72.83	62.69	54.86	50.72	52.71
Decision Tree	96.54	93.54	98.34	95.88	40.26	34.25	49.67	40.54
Random Forest	96.54	93.04	98.94	95.90	72.8	68.77	61.66	65.02
Logistic Regression	78.38	74.3	72.13	73.20	77.58	75.1	76.78	75.93
KNN Classifier	87.41	82.48	87.94	85.12	82.15	76.16	82.2	79.06
Support Vector Machine	78.58	74.45	72.6	73.51	75.39	74.74	60.37	66.79
Perceptron	75.16	68.58	72.57	72.57	49.37	38.28	38.28	38.28
Multi Layer Perceptron	77.6	75.45	67.1	71.03	76.85	74.81	65.66	69.94

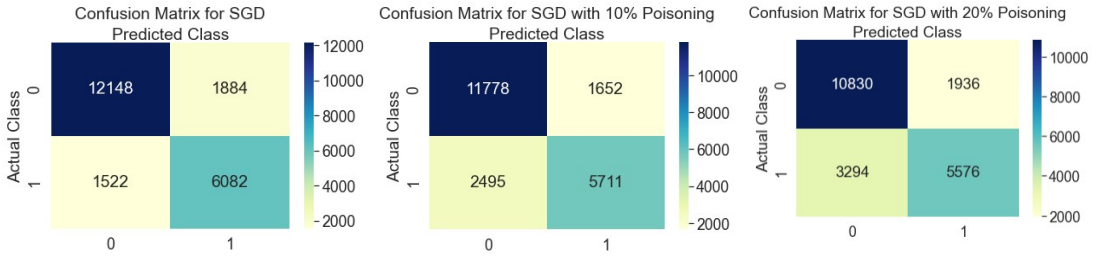

Fig. 4. Confusion Matrix for Stochastic Gradient Descent Based Malware Detector

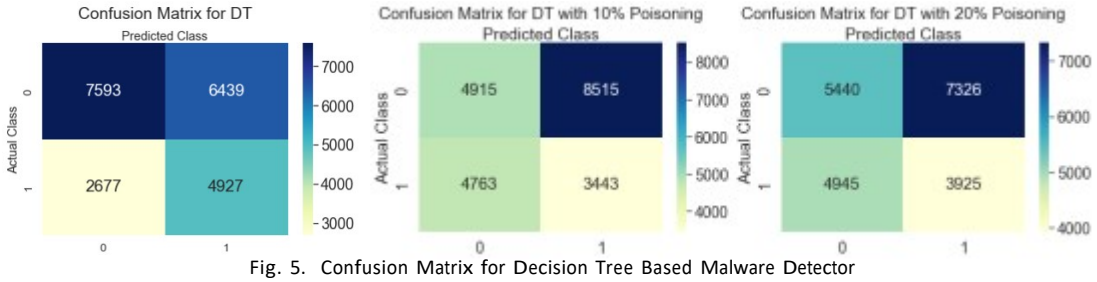

Fig. 5. Confusion Matrix for Decision Tree Based Malware Detector

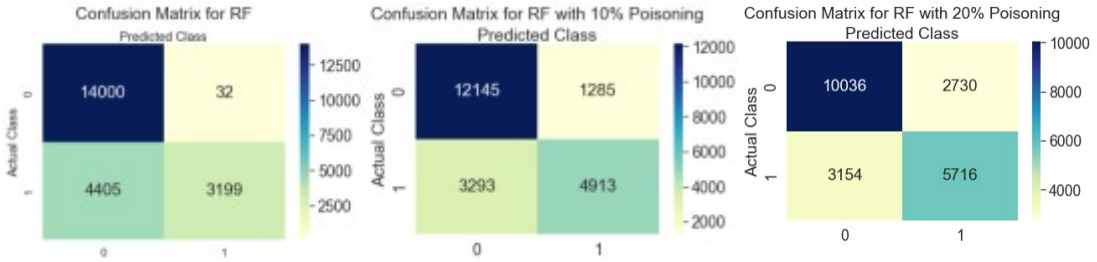

Fig. 6. Confusion Matrix for Random Forest-Based Malware Detector

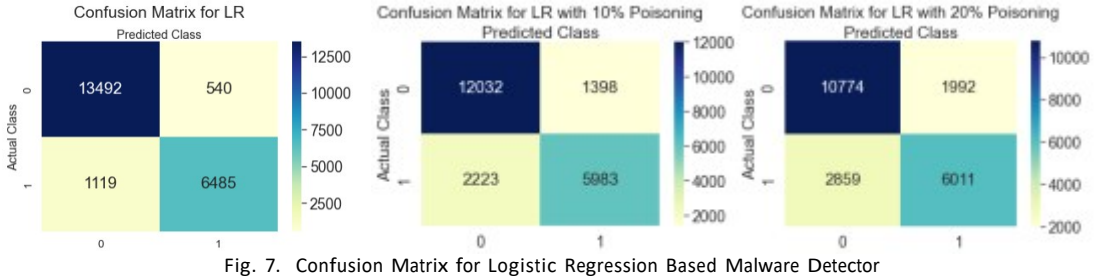

Fig. 7. Confusion Matrix for Logistic Regression Based Malware Detector

Support Vector Machine, and Multi-Layer Perceptron are the best models on the clean data. However, the distance between the three curves represents the robustness of the model toward the poisoning attack. If the separation between the curves of clean data and poisoning data is low, it infers that the poisoning attack has a minimal impact on the model's performance. In the ROC graph, we can observe that Random Forest, Logistic Regression, K-Nearest Neighbors, and Multi-Layer Perceptron

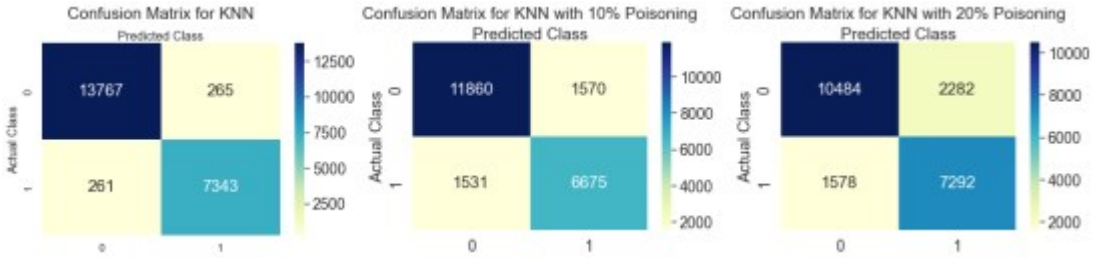

Fig. 8. Confusion Matrix for KNN Based Malware Detector

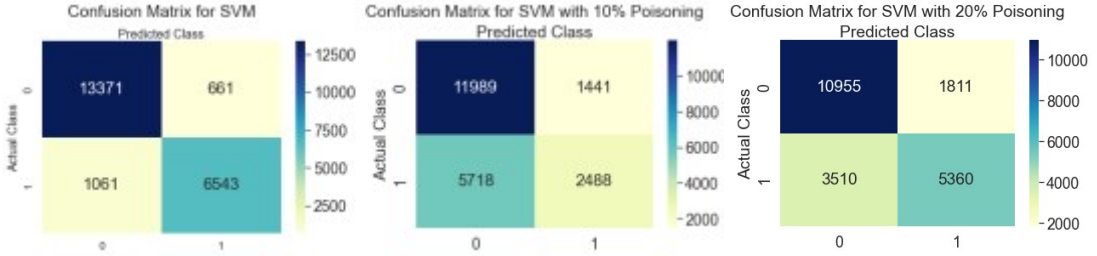

Fig. 9. Confusion Matrix for Support Vector Machine-Based Malware Detector

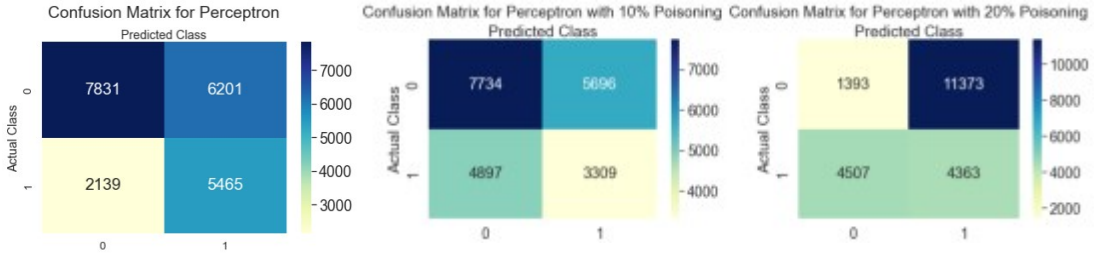

Fig. 10. Confusion Matrix for Perceptron Based Malware Detector

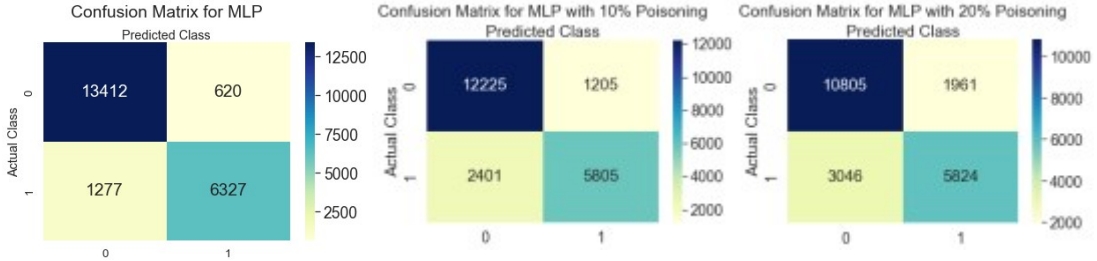

Fig. 11. Confusion Matrix for Multi-Layer Perceptron Based Malware Detector

have their graphs close to each other, proving their robustness against poisoned data. Random Forest's robustness can be attributed to its ensemble nature which helps it to capture better insights about the data. The robustness of logistic regression and K-Nearest Neighbors can be due to the low dimensionality of our training data. Further, we can observe the performance of models, like SVM and perceptron, doing better with the 20% poisoned data than with 10% poisoned data. The gain in the performance of these models is due to unrestricted data poisoning. Since we are not guiding our poisoning approach according to the models, further adding poisoning data after some threshold point slightly improves the models' performance. In the end, even the least sophisticated attacks, like label poisoning, are causing the performance decay of the models to a large extent. This further alerts us toward the catastrophic consequences of more sophisticated attacks like gradients and reinforcement learning.

VII. Conclusion

In this work, we perform a feasibility analysis of label-flip poisoning attacks on ML-based malware detectors. We evaluated eight different ML models that are widely used in malware detection. Spotting the lack of poisoning attacks work in the malware domain, this paper analyses the robustness of ML-based malware detectors against different volumes of poisoned data. We observed the decay in performance of all the models while poisoning 10% and 20% of total training data. The significant decrease in the performance of the models shows the severe vulnerability of malware detectors to guided poisoning approaches. We also observed differences in the effect of poisoning attacks across the different models. Our work is carried out within the limited scope of one generic poisoning algorithm and a single malware dataset. There are few future research directions that are clearly visible. The malware detectors can be tested against many advanced poisoning approaches using numerous datasets from the industry.

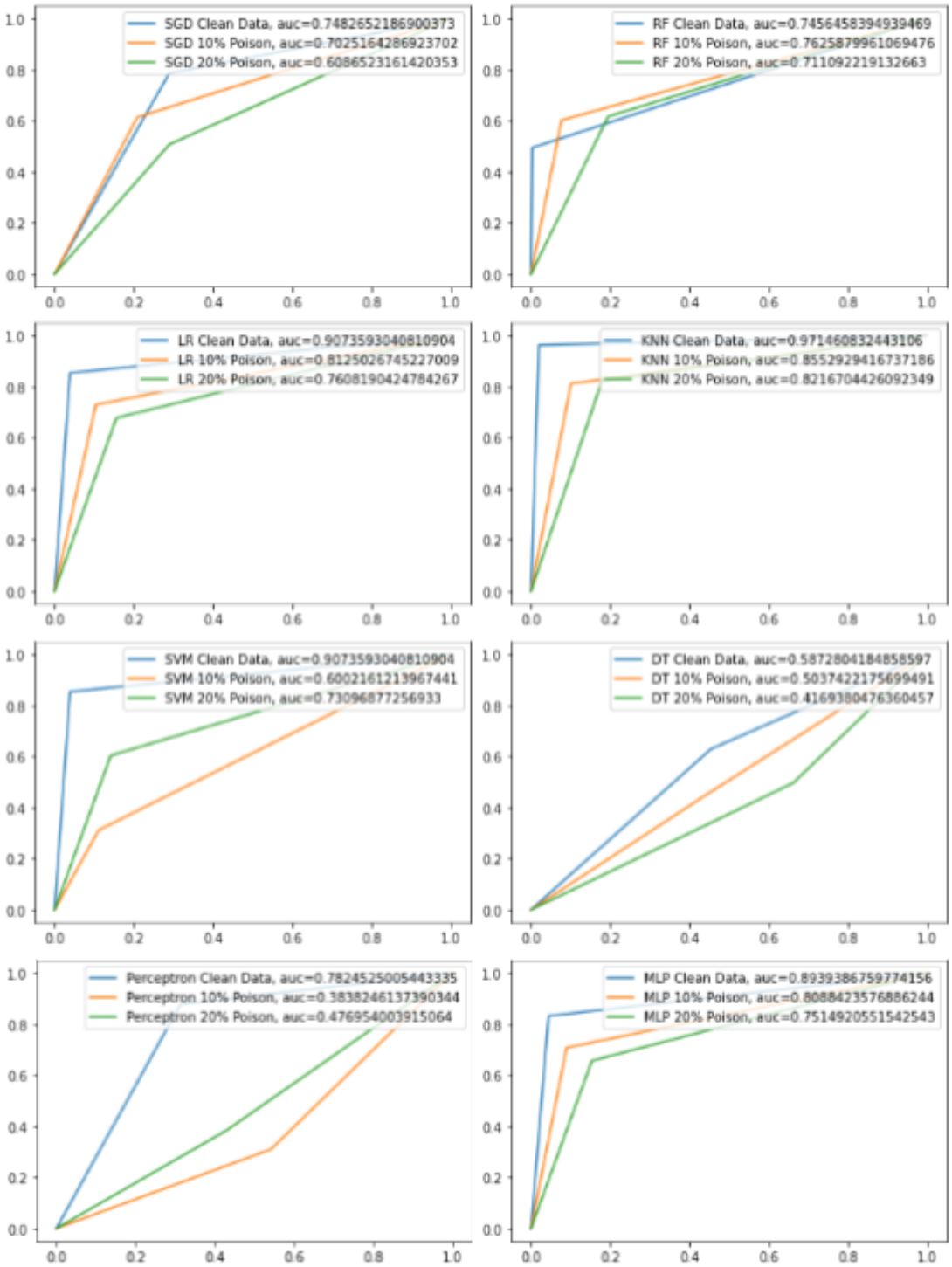

Fig. 12. ROC Curve for Malware detectors under Poisoning Environments

The poisoning can be tested in a more real environment by poisoning the executable files. The research community still lacks exhaustive studies on the vulnerabilities of malware detectors and how to make detectors more robust against these poisoning attacks.

REFERENCES

[1] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 3642–3649.

[2] J. Schmidhuber, U. Meier, and D. Ciresan, "Multi-column deep neural networks for image classification," in 2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE Computer Society, 2012.

[3] K. Chowdhary, "Natural Language Processing," Fundamentals of Artificial Intelligence, pp. 603–649, 2020.

[4] J. Hirschberg and C. D. Manning, "Advances in Natural Language Processing," Science, vol. 349, no. 6245, pp. 261–266, 2015.

[5] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: A review," Expert Systems with Applications, vol. 36, no. 10, pp. 11 994–12 000, 2009.

[6] N. Peiravian and X. Zhu, "Machine learning for android malware detec-

tion using permission and api calls," in 2013 IEEE 25th International Conference on Tools with Artificial Intelligence, 2013, pp. 300–305.
[7] J. Kober and J. Peters, "Learning motor primitives for robotics," in 2009 IEEE International Conference on Robotics and Automation, 2009.
[8] R. Manicavasaga, P. B. Lamichhane, P. Kandel, and D. A. Talbert, "Drug repurposing for rare orphan diseases using machine learning techniques," in The International FLAIRS Conference Proceedings, vol. 35, 2022.
[9] A. Dhakal, C. McKay, J. J. Tanner, and J. Cheng, "Artificial intelligence in the prediction of protein–ligand interactions: recent advances and future directions," Briefings in Bioinformatics, vol. 23, no. 1, p. bbab476, 2022.
[10] M. H. R. Khouzani, S. Sarkar, and E. Altman, "Maximum Damage Malware Attack in Mobile Wireless Networks," IEEE/ACM Transactions on Networking, vol. 20, no. 5, pp. 1347–1360, 2012.
[11] S. Gupta, A. Singhal, and A. Kapoor, "A literature survey on social engineering attacks: Phishing attack," in 2016 International Conference on Computing, Communication and Automation (ICCCA), 2016.
[12] F. Callegati, W. Cerroni, and M. Ramilli, "Man-in-the-Middle Attack to the HTTPS Protocol," IEEE Security Privacy, vol. 7, no. 1, 2009.
[13] C. Schuba, I. Krsul, M. Kuhn, E. Spafford, A. Sundaram, and D. Zamboni, "Analysis of a denial of service attack on TCP," in Proceedings. 1997 IEEE Symposium on Security and Privacy (Cat. No.97CB36097).
[14] W. G. Halfond, J. Viegas, A. Orso et al., "A classification of sql-injection attacks and countermeasures," in Proceedings of the IEEE International Symposium on Secure Software Engineering, vol. 1. IEEE, 2006.
[15] I. Yilmaz and R. Masum, "Expansion of cyber attack data from unbalanced datasets using generative techniques," arXiv preprint arXiv:1912.04549, 2019.
[16] B. Kolosnjaji, A. Demontis, B. Biggio, D. Maiorca, G. Giacinto, C. Eckert, and F. Roli, "Adversarial malware binaries: Evading deep learning for malware detection in executables," in IEEE European Signal Processing Conference, 2018, pp. 533–537.
[17] F. Kreuk, A. Barak, S. Aviv-Reuven, M. Baruch, B. Pinkas, and J. Keshet, "Adversarial examples on discrete sequences for beating whole-binary malware detection," arXiv preprint :1802.04528, 2018.
[18] L. Demetrio, B. Biggio, G. Lagorio, F. Roli, and A. Armando, "Explaining Vulnerabilities of Deep Learning to Adversarial Malware Binaries," arXiv preprint arXiv:1901.03583, 2019.
[19] O. Suciu, S. E. Coull, and J. Johns, "Exploring adversarial examples in malware detection," in 2019 IEEE Security and Privacy Workshops.
[20] K. Aryal, M. Gupta, and M. Abdelsalam, "A Survey on Adversarial Attacks for Malware Analysis," arXiv preprint arXiv:2111.08223, 2021.
[21] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," arXiv preprint arXiv:1312.6199, 2013.
[22] S. M. P. Dinakarrao, S. Amberkar, S. Bhat, A. Dhavlle, H. Sayadi, A. Sasan, H. Homayoun, and S. Rafatirad, "Adversarial attack on microarchitectural events based malware detectors," in Proceedings of the 56th Annual Design Automation Conference 2019, 2019, pp. 1–6.
[23] W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on GAN," arXiv preprint arXiv:1702.05983, 2017.
[24] S. Shalev-Shwartz et al., "Online learning and online convex optimization," Foundations and Trends® in Machine Learning, vol. 4, 2012.
[25] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: Zero-effort Crowdsourcing for Indoor Localization," in Proceedings of the 18th Annual International Conference on Mobile Computing and Networking, 2012, pp. 293–304.
[26] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecnỳ, S. Mazzocchi, B. McMahan et al., "Towards federated learning at scale: System design," Proceedings of Machine Learning and Systems, vol. 1, pp. 374–388, 2019.
[27] "Tay, Microsoft's AI chatbot, gets a crash course in racism from Twitter," https://www.theguardian.com/technology/2016/mar/24/tay-microsofts-ai-chatbot-gets-a-crash-course-in-racism-from-twitter, 2016.
[28] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks," Advances in Neural Information Processing Systems, vol. 31, 2018.
[29] X. Liu, S. Si, X. Zhu, Y. Li, and C.-J. Hsieh, "A unified framework for data poisoning attack to graph-based semi-supervised learning," arXiv preprint arXiv:1910.14147, 2019.
[30] D. Cao, S. Chang, Z. Lin, G. Liu, and D. Sun, "Understanding distributed poisoning attack in federated learning," in 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS). IEEE, 2019.
[31] J. Shen, X. Zhu, and D. Ma, "TensorClog: An imperceptible poisoning attack on deep neural network applications," IEEE Access, vol. 7, 2019.
[32] J. Zhang, J. Chen, D. Wu, B. Chen, and S. Yu, "Poisoning Attack in Federated Learning using Generative Adversarial Nets," in 2019 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.
[33] W. Jiang, H. Li, S. Liu, Y. Ren, and M. He, "A Flexible Poisoning Attack Against Machine Learning," in ICC 2019-2019 IEEE International Conference on Communications (ICC). IEEE, 2019, pp. 1–6.
[34] H. Kwon, H. Yoon, and K.-W. Park, "Selective poisoning attack on deep neural network to induce fine-grained recognition error," in IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering, 2019, pp. 136–139.
[35] H. Zhang, T. Zheng, J. Gao, C. Miao, L. Su, Y. Li, and K. Ren, "Data poisoning attack against knowledge graph embedding," in Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 4853–4859. [Online]. Available: https://doi.org/10.24963/ijcai.2019/674
[36] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in International Conference on Artificial Intelligence and Statistics. PMLR, 2020, pp. 2938–2948.
[37] M. Li, Y. Sun, H. Lu, S. Maharjan, and Z. Tian, "Deep reinforcement learning for partially observable data poisoning attack in crowdsensing systems," IEEE Internet of Things Journal, vol. 7, 2020.
[38] S. Sasaki, S. Hidano, T. Uchibayashi, T. Suganuma, M. Hiji, and S. Kiyomoto, "On embedding backdoor in malware detectors using machine learning," in IEEE International Conference on Privacy, Security and Trust, 2019, pp. 1–5.
[39] X. Zhang, X. Zhu, and L. Lessard, "Online Data Poisoning Attack," in Learning for Dynamics and Control. PMLR, 2020, pp. 201–210.
[40] G. Lovisotto, S. Eberz, and I. Martinovic, "Biometric backdoors: A poisoning attack against unsupervised template updating," in 2020 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2020.
[41] C. Li, X. Chen, D. Wang, S. Wen, M. E. Ahmed, S. Camtepe, and Y. Xiang, "Backdoor attack on machine learning based android malware detectors," IEEE Trans. on Dependable and Secure Computing, 2021.
[42] M. Kravchik, B. Biggio, and A. Shabtai, "Poisoning attacks on cyber attack detectors for industrial control systems," in Proceedings of the 36th Annual ACM Symposium on Applied Computing, 2021.
[43] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," arXiv preprint arXiv:1206.6389, 2012.
[44] C. Yang, Q. Wu, H. Li, and Y. Chen, "Generative Poisoning Attack Method Against Neural Networks," preprint arXiv:1703.01340, 2017.
[45] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization," in Proceedings of the 10th ACM workshop on Artificial Intelligence and Security, 2017.
[46] C. Miao, Q. Li, L. Su, M. Huai, W. Jiang, and J. Gao, "Attack under Disguise: An Intelligent Data Poisoning Attack Mechanism in Crowdsourcing," in Proceedings of the 2018 World Wide Web Conference.
[47] A. P. Dawid and A. M. Skene, "Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm," Journal of the Royal Statistical Society: Series C (Applied Statistics), vol. 28, no. 1, 1979.
[48] M. Fang, G. Yang, N. Z. Gong, and J. Liu, "Poisoning attacks to graph-based recommender systems," in Proceedings of the 34th Annual Computer Security Applications Conference, 2018, pp. 381–392.
[49] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket." in NDSS, vol. 14, 2014, pp. 23–26.
[50] H. Cai, N. Meng, B. Ryder, and D. Yao, "Droidcat: Effective android malware detection and categorization via app-level profiling," IEEE Transactions on Information Forensics and Security, vol. 14, no. 6, 2018.
[51] E. Mariconti, L. Onwuzurike, P. Andriotis, E. De Cristofaro, G. Ross, and G. Stringhini, "Mamadroid: Detecting android malware by building markov chains of behavioral models," preprint arXiv:1612.04433, 2016.
[52] Y. Aafer, W. Du, and H. Yin, "Droidapiminer: Mining api-level features for robust malware detection in android," in International Conference on Security and Privacy in Communication Systems. Springer, 2013.
[53] "Malware detection," https://www.kaggle.com/competitions/malware-detection/data.