A Sensitivity Analysis of Poisoning and Evasion Attacks in Network Intrusion Detection System Machine Learning Models

Kevin Talty
United States Army
Vilseck, Germany
kevin.f.talty.mil@mail.mil

John Stockdale
United States Army
Vilseck, Germany
john.m.stockdale2.mil@mail.mil

Nathaniel D. Bastian
Army Cyber Institute, U.S. Military Academy
West Point, New York
nathaniel.bastian@westpoint.edu

MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM) | U.S. Government work not protected by U.S. copyright | DOI: 10.1109/MILCOM52596.2021.9652959

***Abstract*—As the demand for data has increased, we have witnessed a surge in the use of machine learning to help aid industry and government in making sense of massive amounts of data and, subsequently, making predictions and decisions. For the military, this surge has manifested itself in the Internet of Battlefield Things. The pervasive nature of data on today's battlefield will allow machine learning models to increase soldier lethality and survivability. However, machine learning models are predicated upon the assumptions that the data upon which these machine learning models are being trained is truthful and the machine learning models are not compromised. These assumptions surrounding the quality of data and models cannot be the status-quo going forward as attackers establish novel methods to exploit machine learning models for their benefit. These novel attack methods can be described as adversarial machine learning (AML). These attacks allow an attacker to unsuspectingly alter a machine learning model before and after model training in order to degrade a model's ability to detect malicious activity. In this paper, we show how AML, by poisoning data sets and evading well trained models, affect machine learning models' ability to function as Network Intrusion Detection Systems (NIDS). Finally, we highlight why evasion attacks are especially effective in this setting and discuss some of the causes for this degradation of model effectiveness.**

1. Introduction

Network security is a critical consideration in military communications, serving as a vital component of having secure devices, sensors and systems within the broader Internet of Battlefield Things [1] [2]. Machine learning has grown in popularity, and organizations have turned to integrating machine learning (ML) models into their network security architecture, particularly in network intrusion detection systems (NIDS). In an effort to increase the accuracy and effectiveness of detecting network intrusions, NIDS have been supplemented with ML models to reduce the workload of the network analysts' and to offer insight and ability that may be beyond human capability [3] [4]. However, by incorporating ML models into NIDS, this has opened the door to novel adversarial machine learning attacks that take advantage of the design and implementation of ML models [5].

Adversarial machine learning (AML) is classified by the National Institute of Science and Technology (NIST) as being concerned with the design of ML algorithms that can resist security challenges, the study of the capabilities of attackers, and the understanding of attack consequences [6]. NIST expands this explanation to categorizing AML attacks as poisoning attacks and evasion attacks. NIST defines poisoning attacks as: attacks aiming to increase the number of misclassified samples at test time by injecting a small fraction of carefully designed adversarial samples into the training data. NIST, alternatively, defines evasion attacks as: the attacker manipulates input samples to evade (cause a misclassification) a trained classifier at test/deployment time [6] [7]. In other words, the poisoning attacks are conducted during the developmental phase of a ML model, and evasion attacks are executed during the deployment phase of a trained ML model. Both of these attack types can greatly degrade model performance, and we focus on both in regards to NIDS.

Poisoning attacks degrade the quality of NIDS ML models by leading to ineffective intrusion detection. These attacks can alter input sensors, the data pre- and post-processing, or the ML model itself. NIDS are especially susceptible to this form of attack because the developer of the NIDS model does not fully control data collection and manipulation [7]. NIDS model developers and implementers do not dictate the limits of the network traffic but instead monitor what is already occurring. Users on the network control what data is produced; thus, they dictate the training set for a NIDS on that network. This lack of control over data production gives adversaries the opportunity to distort the training data set to significantly decrease overall performance in the deployment phase.

Evasion attacks look to modify the input of an already trained ML model such that implemented models misclassify at a high rate. These attacks can be further broken down into targeted and untargeted attacks [7]. Targeted attacks direct which class a given sample is misclassified as, while untargeted increase the misclassification rate for all classes. Targeted attacks pose the largest threat to NIDS because an adversary can manipulate intrusions into a network system to appear as normal traffic. However, untargeted attacks can be used to overwhelm a network as security systems overreact to normal traffic. In both of these attacks, the adversary solves a constrained optimization problem to find a small input perturbation that causes a large change in the model's loss function [6]. The perturbations are limited to prevent human and non-human detection. In regards to NIDS, an adversary will limit distortions to network traffic to make the malicious acts appear normal and maintain feasibility.

2. Related Works

AML is a relatively new term, but the exploration of learning with malicious or incorrect data is nothing new. In 1993, a noteworthy finding, and perhaps a prescient forewarning, from Kearns and Li [8] is that a learning algorithm cannot learn beyond a rate of ρ if the malicious or incorrect noise exceeds $\rho/(1+\rho)$. This idea would later be explored by many researchers acknowledging that learning algorithms are bound to be expected to learn in the presence of bad data.

Many researchers have highlighted the brittleness of certain ML algorithms against certain data. Androutsopolis [3] show that naïve Bayesian anti-spam filtering is very brittle when faced with certain data. Nelson [9] also offers that virus detection systems which use support vector machines or Naïve Bayes can be evaded, as well as that unsupervised learning algorithms should be used on the training data before a model is trained on the data. Many other researchers recognize the faults of certain learning algorithms and offer solutions. Auer [10] offers a PACLearning algorithm which offers that pruning certain data may prevent against the negative impacts of noise in data. Dalvi et al. [4] offer a method of classifying adversarial information emplaced in datasets. Littlestone and Warmuth [11] propose using a weighted majority algorithm to held make models less vulnerable to poor data. Servedio [12] offers using ‘smooth boosting’ for the same purpose for models which employ a boosting algorithm for learning. Roberts [13] suggests that extreme value statistics could be used in the same effort to learn in the presence of noisy data and outlying data points.

The field of robust optimization attempts to address this concern as well [14]. Robust optimization, in short, uses a minimax function where it attempts to learn with uncertainty by maximizing error in the data and minimizing the loss or optimization function [15] [16]. Robust optimization also attempts to create better out-of-training-sample models by artificially inducing error into the data. However, it is of concern that the methods of robust optimization do not apply to the non-linear forms of optimization which many ML models employ.

Although the study of AML, or subjects related to it, is growing and many papers and ideas have been posed for decades now, it is not very clear that any research has been done exploring the effects of poisoning and evasion attacks on NIDS. In particular, no paper has explored which common ML algorithms are most brittle when it comes to being exposed with ’noisy’ or bad data while in the training phase–such as a poisoning attack would do–and no paper has conducted sensitivity analysis of novel AML evasion attack algorithms in respect to NIDS.

3. Methodology

To explore the threat of poisoning and evasion AML attacks in NIDS, we used an altered version of the KDD99 dataset. Specifically, the NSL-KDD dataset is an open-source dataset with 125,973 rows and 45 columns in the training set, along with 22,544 rows and 45 columns in the test set. Each row is independent network traffic that would be entered into a NIDS. Each feature vector is a different aspect of that traffic—these features range from bits per second to the type of protocol (TCP/UDP). The last column is the response variable that correctly categorizes the network traffic as normal or a type of network attack. To simplify our modeling process, we reduced the 39 types of attack to four main classes: Denial of Service (DoS), Probe, Remote to Local (R2L), and User to Root (U2R) [17] [18]. The quantity breakdown of the labels are given in Table 1.

TABLE 1. NSL-KDD Response Variable Breakdown

	KDDTrain+.txt	KDDTest+.txt
Normal	67343	9711
DoS	45927	7458
Probe	11656	2421
U2R	52	200
R2L	995	2654
Total	125973	22544

The NSL-KDD data was altered based on the corresponding attack. For poisoning attacks, a varying percentage of the training set was altered to create an improperly trained model. Then, the validation and training set were created from an randomly generated 80/20 split in the training data. The test set remained untouched and was withheld until the very end to see how poison attacks affect out-of-sample performance. For evasion attacks, the training set was untouched while the test set was altered to varying degrees to degrade model performance after it had been properly trained. When tampering with the datasets, only certain mutable variables were altered. Some features are immutable in the application of network traffic because their changes would make the network traffic infeasible [5]. All changes to features are restricted to values greater than or equal to zero. In addition, the binary feature of *Su Attempted* was constrained to zero and one. The 11 features altered for poisoning and evasion attacks are given in Table 2.

TABLE 2. FEATURES ALTERED BY ATTACKS

Feature Type	Feature Name	Feature Value Type
Basic	Src bytes	numeric
Basic	Duration	numeric
Content	Num file creations	numeric
Content	Num shells	numeric
Basic	Service	nominal
Content	Num failed logins	numeric
Basic	Wrong fragment	numeric
Content	Su attempted	binary
Content	Num root	numeric
Time	Serror rate	numeric
Time	Srv serror rate	numeric
Time	Rerror rate	numeric
Time	Srv rerror rate	numeric

3.1. Poisoning Attacks

To test the impact of poisoned data, multiple ML models were trained on varying degrees and forms of poisoned data. We accomplished this by randomly deleting a varying percentage of entries in the mutable feature columns and imputing the now missing values with common imputation algorithms. The algorithms used were K-Nearest Neighbors (KNN) imputation and matrix-completion using Singular-Vector Decomposition (commonly known as SoftImpute) [19]. We approached this as follows: The original data set had 10% of values in the mutable columns deleted at random and then had those missing values replaced by the two aforementioned imputation techniques. We performed this again with 20% deleted and once again with 30% deleted. Finally, these steps resulted in the following seven data sets for training the ML models: the original data set, the 10% imputed with K-Nearest Neighbors (KNN) data set, the 10% imputed with SoftImpute data set, the 20% imputed with KNN data set, the 20% imputed with SoftImpute data set, the 30% imputed with KNN data set, and, lastly, the 30% imputed with SoftImpute data set. This resulted in data sets with varying degrees and forms of altered training data upon which to train NIDS ML models.

The ML models trained to act as a NIDS on the poisoned data were the following: Multi-Layer Perceptron, Classification Tree, Random Forest, KNN Classifier, Multiple Logistic Regression, and Gradient Boosting Classifier [19]. A grid-search across all pertinent hyper-parameters was done for each ML model for each dataset. The hyper-parameters by model are available in Table 3. The purpose of this step was twofold: first, the grid-search helped ensure that the ML model was tuned as if it were to be trained for real-world employment; second, the grid-search produced different ML models based on the poisoned data they were trained on. After grid-searching for all the ML models on their corresponding poisoned datasets, the hyper parameters for each ML algorithm and data set combination were selected based on accuracy on the validation set. To test the impact of the altered training data, the untouched test data set was used to calculate out-of-sample accuracy, to simulate real-world employment. A baseline accuracy was calculated by tuning the given ML model on untouched training and validation data sets. We then calculated its out-of-sample accuracy on the untouched test data set. Thus, we were able to quantify the effect of a poisoned data set on a tuned ML model's out-of-sample accuracy.

TABLE 3. HYPER-PARAMETERS BY MODEL IN GRID SEARCH

Model	Hyper-Parameters
Multi-Layer Perceptron	'hidden_layer_sizes', 'activation', 'solver', 'alpha', and 'learning_rate'
Classification Tree	'criterion', 'max_depth', 'min_samples_leaf', and 'min_samples_split'
Random Forest	'bootstrap', 'max_depth', 'max_features', 'min_samples_leaf', 'min_samples_split', and 'n_estimators'
KNN Classifier	'n_neighbors', 'weights' and 'metric'
Logistic Regression	'C', 'penalty', 'fit_intercept', and 'solver'
Gradient Boosting Classifier	'loss', 'learning_rate', 'n_estimators', 'criterion' and 'max_depth'

3.2. Evasion Attacks

The impact of evasion attacks was calculated in a similar method but the testing set was perturbed in varying degrees and forms. By attacking the test set, we simulated real-world employment as evasion attacks look to avoid detection of a well-trained ML-based NIDS. Our evasion attacks utilized the IBM Adversarial Robustness Toolbox (ART) to produce altered test sets which required a ML model, an IBM ART estimator object, and an IBM ART attack object for each evasion attack [7]. The first step was to initialize a ML model. We used four different models: Multi-Layer Perceptron, Stochastic Gradient Descent, Support Vector Machine, and Random Forest. Then, we initialized and trained an IBM ART estimator object on the training set to create a trained IBM ART model. For the purpose of this paper, only the IBM ART SklearnClassifier estimator was utilized since all of our ML models were initialized in sklearn [19]. Then, we utilized the IBM ART evasion attack object, which took the trained estimator, original test set, and a target vector as input to create an altered test set. The target vector was the class for each row of data we wanted the ML model to predict after the attack. Since our models were attempting to predict malicious network intrusions, the target vector was all class *normal*. Thus, the attack simulated real-world employment because an attacker would want a network intrusion system to classify their attacks as normal traffic.

We utilized the IBM ART evasion attacks of Fast Gradient Sign Method (FGSM) [20], Projected Gradient Descent (PGD) [21], and Boundary Attack [22] [23]. Not every combination of estimator and attack method were feasible. FGSM and PGD required the estimator to have a gradient loss function for the attack to be successful. Thus, the FGSM and PGD attack methods were only applied to the Stochastic Gradient Descent and Support Vector Machine trained estimators. These three attack methods had the input of ‘eps’ which regulates the maximum perturbation allowed to the test set. This value was varied to quantify the amount by which the perturbation affected the model’s out-of-sample performance. To calculate the impact of the perturbed data sets, the estimator was used to calculate out-of-sample accuracy for the unaltered test set and the perturbed test sets.

4. Results and Discussion

4.1. Poisoning Attacks

The effect of the poisoned data attacks on overall model performance was minimal. Out-of-sample accuracy can be seen in Figure 1, which shows that accuracy generally decreased for almost all NIDS ML models. The model that was not only impervious to poisoned data but actually improved in out-of-sample accuracy was the classification tree. The classification tree had small accuracy improvements for both the KNN and SoftImpute imputation methods. On the other hand, the Multi-Layer Perceptron model performance was the most degraded as the percent of poisoned data grew. As the level of poisoning grew for each data set, the in-sample accuracy for both models remained high but the out-of-sample accuracy began to fall for MLP. The hyper-parameters chosen for MLP that maximized validation set accuracy, led to an over-trained model lacking robustness. The other models were not over-trained and experienced little effect on out-of-sample performance as poisoning attacks increased.

Out-of-sample accuracy was unimpressive across ML models for all levels of poisoning. However, the individual performance of the low sample response variables, ‘U2R’ and ‘R2L’ was incredibly degraded. This degradation occurred even though these two response variables were not targeted for the attack. All models had almost no ability to detect those two response variables in the out-of-sample phase once the data was poisoned at 20%. It is alarming that no model would sufficiently detect these response variables where it would be useful to deploy in a real-world setting– especially due to the fact that these two network intrusions would be the most essential for the NIDS to correctly classify.

The likely explanation for these observations is that the simple linear classification model, Classification Tree, was not prone to over-fitting and noise in the dataset did not break the optimizer. However, the complex model, such as Multi-Layer Perceptron, was highly responsive to the noise and erroneous data allowed the optimizer to lose control quickly.

4.2. Evasion Attacks

The effect of evasion attacks on model performance was much more significant. The effects of altering test data by a maximum amount in order to have the NIDS ML model predict the network traffic as normal (correct classification would cause 43% of the test data to be classified as normal) can be seen in Figure 2. While all model performance was underwhelming on unaltered test sets, small perturbations to the test set led to a large decrease in performance. The IBM ART Boundary Attack created an almost linear decrease in performance across all the models as percent of altered data increase. On the other hand, the IBM ART attacks of FGSM and PGD caused an exponential decrease in ML model performance. A maximum perturbation of 10% led to nearly a 20% drop in SGD model performance after the FGSM attack. Both of these gradient-based attacks were able to classify nearly all network traffic as normal at 30% perturbation. The Support Vector Machine for both gradient attack methods did not require 30% perturbation to classify all traffic as normal due to low starting model accuracy. The FGSM method had a faster decrease in model performance due to the fact it is not constrained by attempting to make the data alterations indistinguishable like PGD.

The explanation for the Boundary Attack being outperformed by FGSM and PGD is that the Boundary Attack’s step-sizes were limited. The Boundary Attack is a black box method and only changes a predetermined amount of observations that are likely to be misclassified based on its own confidence level. FGSM and PGD can estimate the loss function of the ML model it is targeting and change features, rather than just observations, that are most unstable and heavily weighted when classifying data points.

5. Conclusions

In this paper, we demonstrated that Classification Tree, Random Forest, KNN Classification, and Logistic Regression operating as a NIDS were very stable when

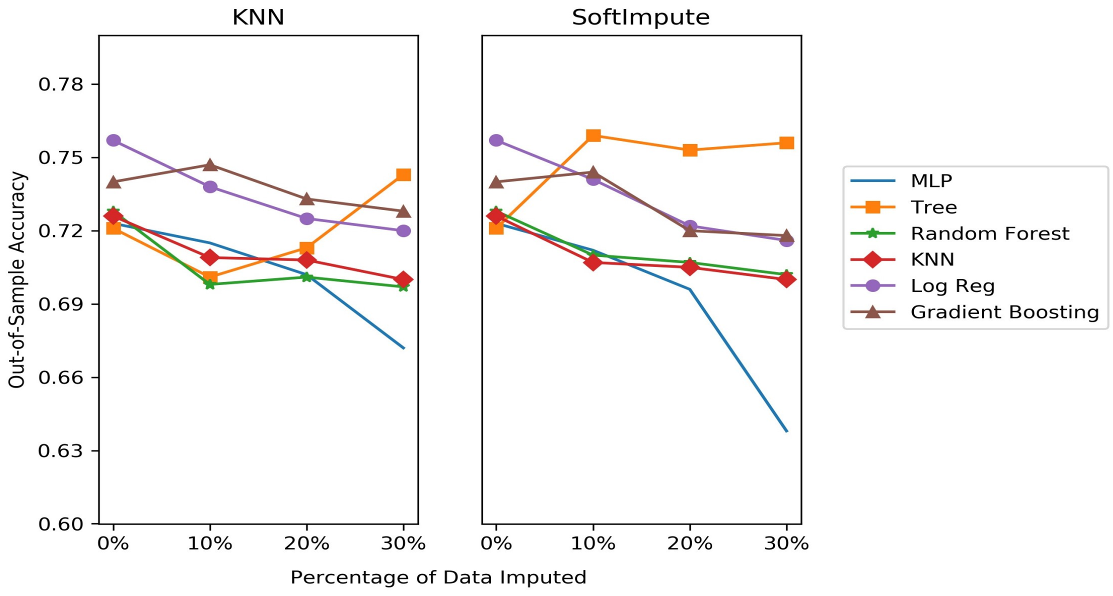

Figure 1. Poisoning Attack Effect on ML Model Performance on Out-of-Sample Accuracy

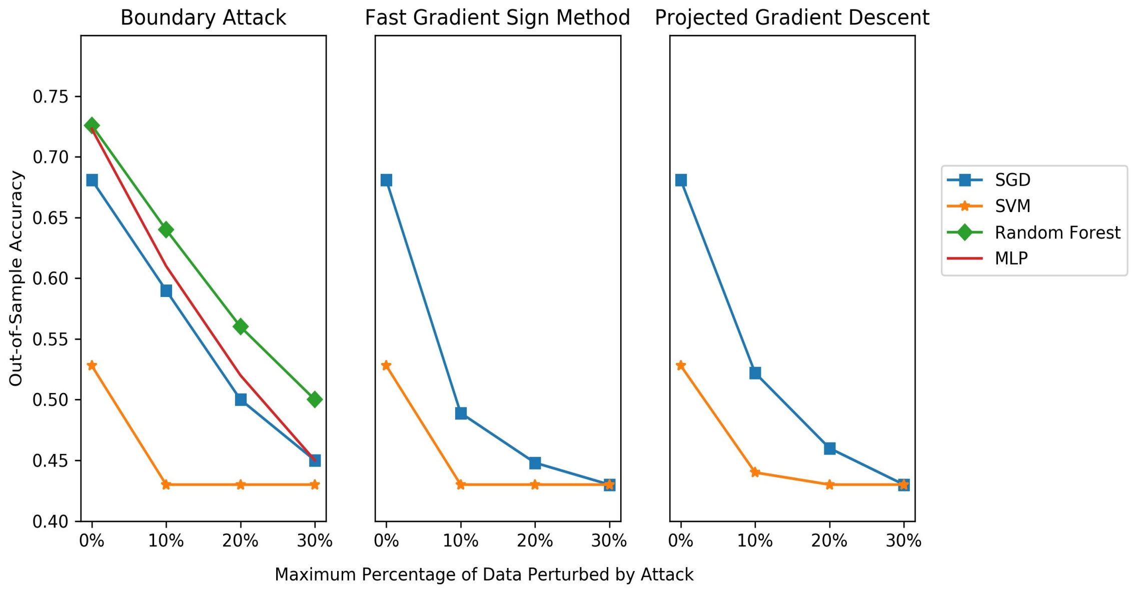

Figure 2. Evasion Attack Effect on ML Model Performance on Out-of-Sample Accuracy

exposed to poisoning attacks. Multi-Layer Perceptron was the only model that experience significant degradation in performance. On the other hand, evasion attacks had a much larger impact. The IBM ART evasion attacks were all able to cause the ML models to incorrectly report nearly all network traffic as normal with 30% change to the test data. This effect was achieved when the attack method knew the ML model's loss function and when it was hidden. This difference of effectiveness between the two main types of attack is likely due to the network traffic data that the NIDS model is attempting to correctly classify. For example, a user to root attack (U2R) requires the attacker to attempt to switch to the super user ('su_attempted' was the binary variable that represented this) or it would not be an attack. A poisoning attack that does not completely remove this characteristic from the training data set will have minimal impact on the final trained model. However, an evasion attack can easily hide the attack if it changes that feature value in the test set. This inherent weakness of NIDS models allows evasion attacks to have a much larger effect.

To mitigate the effects of these evasion attacks, we suggest NIDS ML models utilize feature engineering to create multiple features for key features to network traffic classification. Instead of solely including whether or not super user was attempted, adding the amount of attempts, when it was attempted, and what user attempted it would force an attacker to change more than one feature to successfully perform an evasion attack. However, the developer looking to employ a NIDS ML model with feature engineering needs to ensure that the final set of features are not overly correlated with one another to avoid unstable coefficients within the ML model.

In future work, we hope to create a generalizable system that establishes the characteristics of features that make them susceptible to a type of AML attack. In addition, we hope to employ methods such as Sample Minority Over-Sampling Technique (SMOTE) or selective re-sampling in the training phase, such that the model better accounts for the under-sampled variables. This over-sampling will allow us to see if poisoning attack effectiveness on user to root ('U2R') and remote to local ('R2L') was due to the poisoning attack or from the lack of these types of attack in the training set.

Acknowledgments

This work was supported by the U.S. Army Combat Capabilities Development Command (DEVCOM) Army Research Laboratory under Support Agreement No. USMA21050 and the U.S. Army DEVCOM C5ISR Center under Support Agreement No. USMA21056. The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United States Military Academy, the United States Army, the United States Department of Defense, or the United States Government.

References

[1] N. Suri, M. Tortonesi, J. Michaelis, P. Budulas, G. Benincasa, S. Russell, C. Stefanelli, and R. Winkler, "Analyzing the applicability of internet of things to the battlefield environment," in *2016 International Conference on Military Communications and Information Systems (ICMCIS)*, 2016, pp. 1–8.

[2] A. Azmoodeh, A. Dehghantanha, and K.-K. R. Choo, "Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning," *IEEE Transactions on Sustainable Computing*, vol. 4, no. 1, pp. 88–95, 2019.

[3] I. Androutsopoulos, J. Koutsias, K. Chandrinos, G. Paliouras, and C. Spyropoulos, "An evaluation of native bayesian antispam filtering," *Machine Learning in the New Information Age*, vol. 11th, pp. 9–17, 2000.

[4] N. Dalvi, P. Domingos, S. Sanghai, and D. Verma, "Adversarial classification," *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 99–108, 2004.

[5] E. Alhajjar, P. Maxwell, and N. Bastian, "Adversarial machine learning in network intrustion detection systems," *Expert Systems with Applications*, vol. 186, 2021.

[6] E. Tabassi, K. J. Burns, M. Hadjimichael, A. D. Molina-Markham, and J. T. Sexton, "A taxonomy and terminology of adversarial machine learning, national institute of standard and technology," *arXiv*, 2019.

[7] M. Nicolae, M. Sinn, T. N. Minh, A. Rawat, M. Wistuba, V. Zantedeschi, I. M. Molloy, and B. Edwards, "Adversarial robustness toolbox v0.2.2," *CoRR*, vol. abs/1807.01069, 2018. [Online]. Available: http://arxiv.org/abs/1807.01069

[8] M. Kearns and M. Li, "Learning in the presence of malicious errors," *SIAM Journal on Computing*, vol. 22, no. 4, pp. 807–837, 1993. [Online]. Available: https://doi.org/10.1137/0222052

[9] B. Nelson, "Designing, implementing, and analyzing a system for virus detection," *Technical Report No. UCB/EECS-2006-27.*, 2006.

[10] P. Auer, "Learning nested differences in the presence of malicious noise," *Theorectical Computer Science*, vol. 185, pp. 159–175, 1997.

[11] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Information and computation*, vol. 108, no. 2, pp. 212–261, 1994.

[12] R. Servedio, "Smooth boosting and learning with malicious noise." *Journal of Machine Learning Research*, vol. 4, pp. 633–648, 01 2003.

[13] S. Roberts, "Novelty detection using extreme value statistics," *Vision, Image and Signal Processing, IEE Proceedings -*, vol. 146, pp. 124 – 129, 07 1999.

[14] E. G. L. N. A. Ben-Tal, A., *Robust Optimization*, National Science Foundation Grant No. 0619977.

[15] G. Strang, *Linear Algebra and Learning from Data*. Wellesley, Massachusetts: Wellesley-Cambridge Press, 2019.

[16] S. Raschka and V. Mirjalili, *Python Machine Learning 3rd Edition*. Birmingham, United Kingdom: Packt Publishing, 2019.

[17] L. Dhanabal and S. Shantharajah, "A study on nsl-kdd dataset for intrusion detection system based on classification algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 6, pp. 446–452, 2015.

[18] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications*, ser. CISDA'09. IEEE Press, 2009, p. 53–58.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[20] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, ser. AISec '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 43–58. [Online]. Available: https://doi.org/10.1145/2046684.2046692

[21] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2019.

[22] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," 2018.

[23] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," 2017.