Walling up Backdoors in Intrusion Detection Systems

Maximilian Bachl, Alexander Hartl, Joachim Fabini, Tanja Zseby
Technische Universität Wien
firstname.lastname@tuwien.ac.at

ABSTRACT

Interest in poisoning attacks and backdoors recently resurfaced for Deep Learning (DL) applications. Several successful defense mechanisms have been recently proposed for Convolutional Neural Networks (CNNs), for example in the context of autonomous driving. We show that visualization approaches can aid in identifying a backdoor independent of the used classifier. Surprisingly, we find that common defense mechanisms fail utterly to remove backdoors in DL for Intrusion Detection Systems (IDSs). Finally, we devise pruning-based approaches to remove backdoors for Decision Trees (DTs) and Random Forests (RFs) and demonstrate their effectiveness for two different network security datasets.

1 INTRODUCTION

Training a Machine Learning (ML) model for an IDS is a challenging task which involves massive datasets and significant amounts of computational power. In a practical deployment, it is therefore reasonable to assume that training of the model is done by a security company marketing either a complete Anomaly Detection (AD) system or just a pre-trained model that can be plugged into another AD software. If we have to question if such a security company can be trusted under all circumstances, the problem arises that the security company might have implemented backdoors which circumvent the AD system. This could be motivated by profitseeking or by government actors requiring ways to purposefully disable security measures in specific cases.

In addition to these problems, for the training of models, usually datasets are used which have been generated artificially in a controlled test environment. As a downside of this approach, it is unclear whether a ML model learns to classify based on characteristics that are inherent to the attacks which should be detected, or rather learns to classify based on patterns that were unintentionally created during dataset generation.

For a well-performing network AD technique it is therefore of utmost importance to study which features are useful and which patterns the technique looks at to distinguish attack traffic from normal traffic, and question if these explanations match with expert knowledge.

In this paper, we train models to detect network attacks similar to the approach of a recent paper [9], which bases on the UNSW-NB15 dataset [10] and evaluates the performance of several feature vectors and ML techniques for accurate AD in the context of IDSs. We then add a backdoor to the models and show that attack detection can efficiently be bypassed if the attacker had the ability to modify training data.

Then we discuss several techniques to detect or remove a backdoor from a trained model. In particular, we show how visualization techniques from explainable ML can be used to detect backdoors and problems emerging from the distribution of attack samples in the training dataset. We furthermore evaluate recently proposed techniques for CNNs for removing backdoors from image classifiers, which, however, surprisingly turn out to be ineffective for our Multilayer Perceptron (MLP) classifiers.

Finally, we put emphasis of our experiments on hardening RF classifiers, the probably most important ML method in the context of IDSs. We propose a new pruning technique specifically for removing backdoors from trained RF models.

For reproducibility, we make our code, data and figures publicly available at https://github.com/CN-TU/ids-backdoor.

2 RELATED WORK

Several recent publications aim at increasing robustness of decision trees against attacks. [2] proposes a defense mechanism against poisoning that uses bagging to try to minimize the influence of a backdoor that is introduced in the training dataset. The method is applicable to DTs. However, this approach cannot protect if a trained model is obtained from another (untrusted) party in which the other party might potentially have introduced a backdoor, which is the use case considered in this work. [3] develops a method to train DTs with a tunable parameter that trades off accuracy against robustness against evasion attacks. [13] makes SVMs robust against evasion attacks and outlines a possibility to also apply it to DTs and RFs.

Pruning for neural networks has been proposed as a method to simplify large neural networks [15]. Pruning as a defense for neural networks against poisoning has emerged recently [7]. In [7] the authors proposed pruning as a defense mechanism against backdoored CNNs and show that by removing infrequently used neurons from the last convolutional layer, potential backdoors can be removed. The rationale behind this is that some neurons specialize processing the regular samples while others focus on the backdoor. To our knowledge, these pruning defences have been applied to CNNs but not to MLPs, which are commonly used for IDSs [9]. Although various pruning techniques have been proposed for DTs in the last decades [5] with the aim of simplifying trees that overfit on the training data, pruning has not yet been investigated for its suitability for defending against backdoors for DTs and RFs.

Besides pruning, a frequently used technique for removing backdoors from a trained DL model is fine-tuning. Fine-tuning was initially described as a transfer learning technique [19] and later proposed as part of an attack strategy against poisoning attacks [8]. For fine-tuning, training of the Model under Investigation (MuI) is continued with the validation set, hence reinforcing correct decisions and ideally causing the MuI to gradually forget backdoors. Moreover, the authors argue that since fine-tuning removes backdoors from neurons that are activated by the validation set, fine-tuning is the ideal complement for pruning, which removes backdoors from neurons that are not activated by the validation set. They thus propose fine-pruning as a combination of pruning and fine-tuning.

As with pruning, these methods have not been applied to classic MLPs so far.

While we in this paper aim at sanitizing possibly backdoored IDSs, [4] take a different approach: They create *GENESIDS*, a tool that enables extensive testing of an IDS. This approach can potentially also uncover backdoors if it finds a case, in which the IDS surprisingly misbehaves.

3 EXPERIMENTAL SETUP

We performed our experiments with an RF and an MLP model and intentionally added a backdoor to both. In particular, we used the following experimental setup:

3.1 Datasets

Several requirements have to be met for a dataset to allow realistic performance benchmarks. In this research, we use the UNSW-NB15 [10] and the CIC-IDS-2017 [14] datasets, which were developed by two independent institutions and are both freely available on the Internet.

The UNSW-NB15 dataset [10] was created by researchers of the University of New South Wales to overcome common problems due to outdated datasets. Network captures containing over 2 million flows of normal traffic and various types of attacks are provided together with a ground truth file. Attack traffic includes reconnaissance, DoS and analysis attacks, exploits, fuzzers, shellcode, backdoors and worms.

The CIC-IDS-2017 dataset [14] was created by the Canadian Institute of Cybersecurity to provide an alternative to existing datasets which are found to exhibit several shortcomings. The provided network captures contain more than 2.3 million flows, containing normal traffic and DoS, infiltration, brute force, web attacks and scanning attacks.

For processing the data, we base our analysis on the CAIA [18] feature vector as formulated in [9], which includes the used protocol, flow duration, packet count and the total number of transmitted bytes, the minimum, maximum, mean and standard deviation of packet length and inter-arrival time and the number of packets with specific TCP flags set.

All features except protocol and flow duration are evaluated for forward and backward direction separately. We also include the minimum, maximum and standard deviation of Time-to-Live (TTL) values in our feature vector as an attractive candidate for exploitation as a backdoor.

We used go-flows [16] for extracting features from the raw capture files and applied Z-score normalization to process the data. We used 3-fold cross validation to ensure that our results do not deviate significantly across folds.

3.2 ML Techniques

We used PyTorch [11] to build an MLP. For this we used 5 fully connected layers with 512 neurons each. We applied Rectified Linear Unit (ReLU) activation after each layer and dropout with a probability of 0.2 and used binary cross entropy as loss function for training.

In addition to the MLP, we trained an RF classifier to detect attacks in network traffic. We used the RandomForestClassifier implementation from scikit-learn [12] for this task and used 100 estimators for training the RF.

Table 1: Detection performance results.

	UNSW-NB15		CIC-IDS-2017	
	RF	DL	RF	DL
Accuracy	0.990	0.989	0.997	0.998
Precision	0.854	0.845	0.997	0.999
Recall	0.850	0.829	0.993	0.992
F1 score	0.852	0.837	0.995	0.995
Youden’s J	0.845	0.823	0.992	0.991
Backdoor acc.	1.000	0.998	1.000	1.000

3.3 The Backdoor

We added a backdoor to our MLP and RF models. Hence, we had to find a pattern usable as a backdoor in feature vectors. On one hand, this pattern has to be detectable reliably by the IDS system, on the other hand it must be possible to generate real network traffic which translates into feature vectors exhibiting this pattern.

In our case, we used the TTL value contained in the Internet Protocol (IP) header of Internet traffic as channel for signaling the backdoor. The TTL is used for mitigating problems due to routing loops in IP networks. It is reasonable to assume that the TTL field remains constant for all packets in one benign traffic flow. This assumption was perfectly reflected by the examined datasets, as only 965 flows exhibited a non-zero standard deviation for CIC-IDS-2017 and only 7 flows exhibited a non-zero standard deviation for UNSW-NB15.

Hence, we decided to establish a backdoor in our models by varying the TTL for the packets in flows of attack traffic. The models would thus learn to treat flows with a non-zero standard deviation of the TTL value as non-attack traffic.

In particular, we implemented the backdoor by incrementing the TTL of a flow’s first packet by one if its TTL is smaller than 128 and decrementing it by 1 if is larger. This results in a tiny standard deviation of the TTL as well as in changed maximum, minimum and mean.

3.4 Performance Results

Table 1 shows performance results for the MLP and RF, depicting both detection performance of normal samples and the efficacy of the backdoor. The models are thus able to detect the backdoor with high confidence while retaining high attack detection performance. Our results are consistent with previous work like, e.g., [9].

4 REMEDIES FOR POISONING ATTACKS

We now investigate several techniques which might be used to prevent security vulnerabilities that come with backdoored pretrained models. In this respect, the ability to detect a backdoor and the ability to remove a backdoor from the trained model can be considered as equally effective since one often has the option to fall back to a model obtained from a different source in case a model looks suspicious.

4.1 Explainability Plots

A number of methods have been proposed recently aiming to visualize and explain a non-interpretable ML model's decisions. Applied to the present problem, we can pick up ideas from Partial Dependence Plots (PDPs) and Accumulated Local Effects (ALE) plots, not only for identifying backdoors in the MuI, but also for finding wrong decisions it would take due to flawed training data.

4.1.1 Partial Dependence Plots. PDPs were proposed in [6] and visualize dependence of a model's predictions by plotting the MuI's prediction for a modified dataset for which the feature's value has been fixed to a certain value, averaging over the modified dataset.

If we denote by $\boldsymbol{X} \in \mathbb{R}^n$ a random vector drawn from the feature space and by $f(\boldsymbol{X}) \in [0,1]$ the prediction function, the PDP for the ith feature X_i can be expressed as

$$\mathrm{PDP}_i(w) = \mathbb{E}_{\boldsymbol{X}}\left(f(X_1,\ldots,X_{i-1},w,X_{i+1},\ldots X_n)\right). \tag{1}$$

Empirically, we can approximate the distribution of the feature space using the distribution of observed samples. Hence, at a given point w, the PDP for the ith feature can be found by setting the ith feature value in all samples in the dataset to w and averaging over the predictions of the resulting modified dataset.

4.1.2 Accumulated Local Effects. In real situations, datasets usually exhibit a non-negligible degree of feature dependence. Due to feature dependence, areas exist in the feature space which are unlikely to occur. Since a model is trained with real, observed data, the training set therefore might not include samples for these areas. As a consequence, the model's predictions become indeterminate for these areas, posing a problem when considering these predictions for computing PDPs.

In an attempt to overcome this problem, it is possible to only consider samples which are likely to occur for certain feature values, i.e. to consider the conditional distribution of remaining features, for computing explainability graphs.

ALE plots [1] make use of this idea. For the ith feature X_i, the ALE plot $\mathrm{ALE}_i(w)$ can be defined differentially as

$$\frac{d}{dw}\mathrm{ALE}_i(w) = \mathbb{E}_{\boldsymbol{X}\,|X_i}\left(\frac{\partial}{\partial X_i}f(\boldsymbol{X})\middle| X_i = w\right). \tag{2}$$

To combat ambiguity of this definition, we force $\mathrm{ALE}_i(w)$ to have zero mean on the domain of X_i. For empirical evaluation, we approximate the conditional distributions of $\boldsymbol{X}$ by averaging over samples for which $X_i \approx w$. In this paper, we used the 10 closest samples for estimating the distributions.

4.1.3 Identifying Backdoors. Backdoors can be identified by computing PDP or ALE plots for the MuI and investigating if regions exist, for which the MuI behaves counter-intuitive. For our CIC-IDS-2017 MLP classifier, Figure 1 shows the PDP for the TTL value in forward direction, where the label 1 means classification as attack. We also provide plots for the corresponding models which were trained without backdoor. The plots are not available in a real situation, but we provide them here for comparison.

As shown in Figure 1, the PDPs for the MLP show a deep notch for certain low values of stdev(TTL). As discussed above, normal traffic is very unlikely to have deviating TTL values for different packets. In contrast to Figure 1, one would therefore expect this feature to have a negligible influence on the classification result. Hence, in our case, existence of a backdoor can be assumed since the PDP plummets to very low values for a specific value of stdev(TTL) for no apparent reason.

However, inconsistent behaviour of the MuI, detected using PDP or ALE plots, does not necessarily result from poisoning activity. For example, Figure 2 shows the mean(TTL) feature in forward direction. The models show a clear dependence of the mean TTL value of incoming packets, which is similarly counter-intuitive as for the feature discussed above. In our case, this behaviour results from the non-equal distribution of TTL values of attack and non-attack traffic in both the UNSW-NB15 and CIC-IDS-2017 datasets.

Independent of their origin, such patterns might be exploited for masquerading attacks and thus are clearly unwanted. PDPs and ALE plots therefore provide a convenient possibility for analyzing ML models for vulnerabilities.

4.2 DL Poisoning Defenses

4.2.1 Pruning. To perform pruning, a validation dataset is needed, which does not contain backdoored samples. We take a validation set that is $\frac{1}{4}$ of the training set. We use the validation set for pruning as described in the next sections and a test set that is also $\frac{1}{4}$ of the training set to verify whether the backdoor can be removed successfully and how much the accuracy on the original data suffers. Training, validation and test sets are pairwise disjoint.

We implemented three variants of the pruning defense [7]: Pruning neurons by their average activation in (1) all layers, (2) only in the last layer and (3) only in the first layer. Neurons with the smallest average activation (the least used ones) are pruned first. For this purpose, we look at the value of the activation function in the corresponding layer for all samples in the validation set and prune neurons by setting the weight and bias to 0 in the layer preceding the activation function.

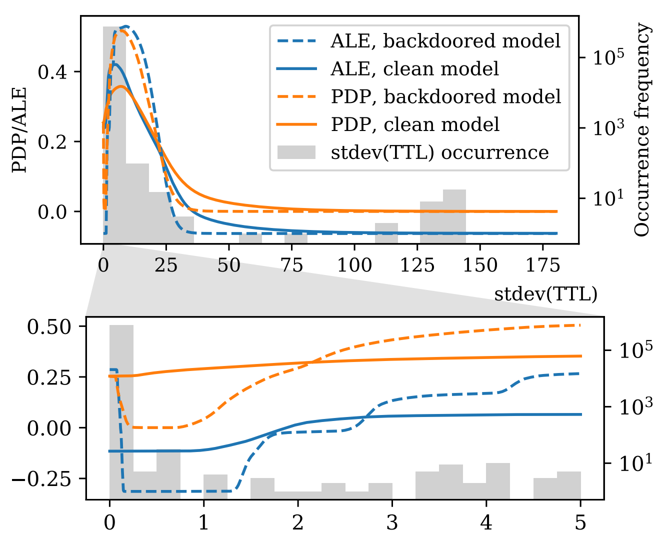

Figure 1: PDPs and ALE plots of the MLP for CIC-IDS-2017. Full range of stdev(TTL) values on top; stdev(TTL) values from 0 to 5 below.

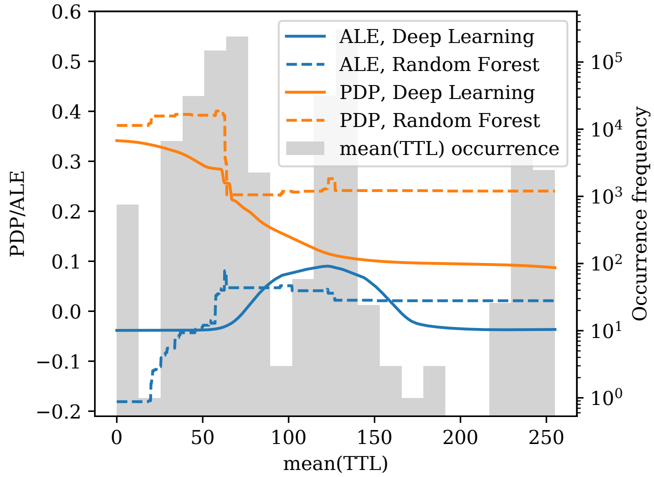

Figure 2: PDPs and ALE plots for mean(TTL) for the CIC-IDS-2017 RF and MLP classifiers.

Our experiments revealed that pruning does not remove the backdoor at all, while decreasing the accuracy for normal data if too many neurons are pruned. To check whether other reasons are responsible for the technique's failure to remove the backdoor, we also conducted the following experiments: (1) We did not take the average activation but the average of the binary activations of each neuron, which means that we did not consider the quantity of the activation but only if the activation is larger than 0 and then averaged the activations for all data samples. (2) We checked whether the dropout regularization might be the reason. (3) We hypothesized that the issue might be that there are a lot fewer malicious samples in the dataset than benign ones. Thus we reversed the backdoor and made the backdoor so that it would falsely classify benign samples as malicious ones.

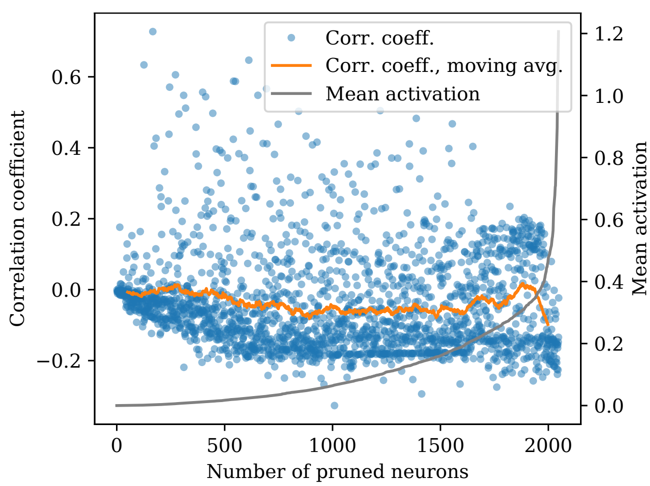

Figure 3: Correlation coefficient of neuron activation with backdoor usage throughout the pruning process for CIC-IDS-2017.

However, none of the experiments could remove the backdoor. To investigate further, we computed the correlation of the activation of each neuron with the presence of the backdoor in the data. Neurons which are responsible for the backdoor should have a high correlation because they only become active when a sample is backdoored. We plotted the correlation of each neuron at the time step it is pruned, which is depicted in Figure 3. If the pruning method worked, we would expect that neurons pruned in the beginning have a high correlation while later ones have a low correlation.

Figure 3 shows that this is not the case. It also shows that neurons are not completely separated in backdoor neurons and regular neurons: If this were the case, the correlation would either be 1 or 0, but we observe many values in between, which indicates that most neurons are both responsible for the backdoor as well as for regular data.

4.2.2 Fine-Tuning. In addition to pruning, we tried to use fine-tuning to remove the backdoor from the MLP. Fine-tuning exclusively makes sense if both the computational effort and the required training set size for fine-tuning are substantially lower than for the original training procedure.

Figure 5 shows the backdoor efficacy when continuing training of our trained model for CIC-IDS-2017 with the validation set, hence without backdoored samples. It indicates clearly that, unfortunately, with reasonable computational effort, fine-tuning is uneffective for cleaning the MuI from backdoors in our case. In fact, training a model from scratch takes less computational effort than fine-tuning. We observed a similar behaviour for UNSW-NB15.

In addition to fine-tuning, we also tried fine-pruning [8] by first applying a large number of different pruning strategies and then using fine-tuning. From all pruning strategies, only pruning a certain fraction of only the MLP's first layer resulted in a discernible drop of backdoor efficacy after one epoch of fine-tuning, primarily for CIC-IDS-2017.

The significance of the MLP's first layer for the backdoor presumably results from the simplicity of our backdoor pattern. Hence, applicability of this pruning technique for general situations is questionable. We conclude that also fine-pruning cannot be considered a reliable method for backdoor removal in the context of IDSs.

4.3 RF Pruning

We developed a defense approach specifically for RF classifiers. The concept behind our pruning defense is that leaves that are never used by samples in the validation set might be used by backdoored inputs. If these "useless" leaves are removed, performance of the classifier on the validation set should not decrease significantly, while decisions that are used for the backdoor are likely to be removed. We developed several variants of our pruning defense with an increasing level of sophistication:

(1) Pruning leaves based on their usage: This means that the least used leaf is pruned first and the most used one last.
(2) Like (1) but considering only "benign" leaves: We assume that attackers want malicious samples appear benign.
(3) Like (2) but additionally using depth to decide when leaves are used equally often: The rationale is that we assume that handcrafted backdoors require fewer rules to classify than regular samples and thus have lower depth.

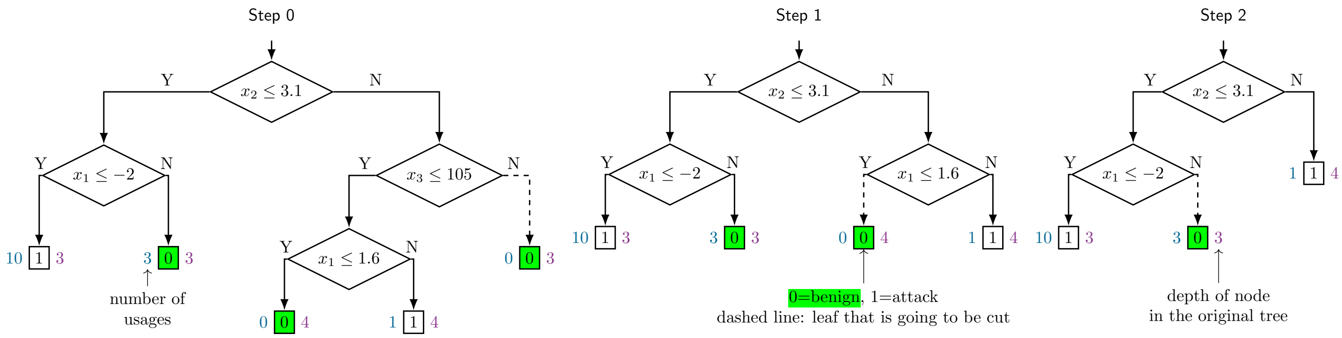

Figure 4: Toy example of a DT being pruned. The decision trees of the random forests we use usually have many thousands of leaves and decisions and are thus not trivial to visualize.

Figure 4 shows an example of variant (3) of pruning applied to a tree and Figure 6 shows the resulting accuracy for pruning according to variant (3) for the UNSW-NB15 dataset. With sufficient leaves being pruned, the accuracy of the backdoor approaches zero. The accuracy for the regular data does not decrease significantly. It might even increase by reducing overfitting. With 90% of leaves pruned, accuracy is still well above 99%. Even with a very small validation set of just 1% of its original size, the pruning still works as expected.

For CIC-IDS-2017 we get similar results, but pruning does not remove the backdoor completely (~30% backdoor accuracy remains). We attribute this to the fact that in this dataset there are also regular flows which have stdev(TTL) > 0. Thus, backdoored flows cannot always be sharply distinguished from regular flows. We find that pruning only benign leaves is very beneficial for UNSW-NB15 but not for CIC-IDS-2017. However, considering depth when making the decision which leaf to prune first, leads to the backdoor being removed considerably earlier in the pruning process.

5 DISCUSSION

From our experiments, we can make three main recommendations for the deployment of ML models which have been obtained from a third-party.

To ensure that no backdoor is contained in the model, it has to be analyzed carefully for questionable decisions and potentially unnecessary features. For this purpose, PDPs and ALE plots are an effective tool. In fact, already throughout the training process explainability plots constitute a useful tool to ensure that the model is not unintentionally trained to artifacts the dataset yields. On the other hand, the implementation of a backdoor as conducted in this research is only possible when using several features involving the TTL value. Even though it might seem tempting to provide all possible features to a DL or RF classifier and let it learn the most important ones, this strategy should be avoided.

For RF classifiers, which can be considered one of the most important classifiers for IDSs, the pruning technique we proposed is able to reduce backdoor efficacy significantly. At the same time, the classifier's detection performance is not substantially reduced.

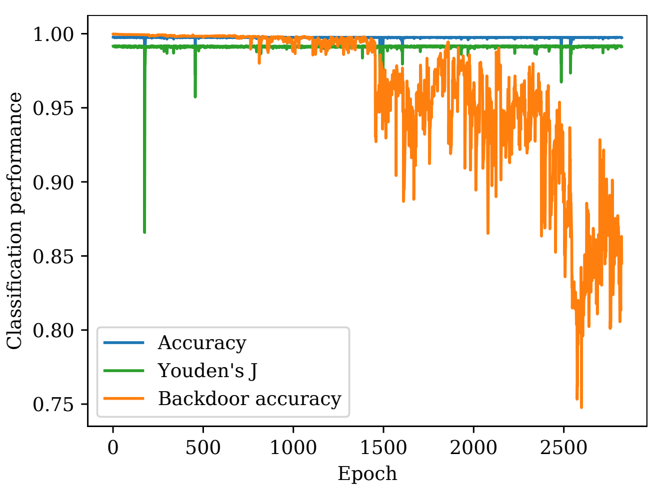

Figure 5: Fine-tuning of the MLP for UNSW-NB15: While fine-tuning eventually lowers backdoor accuracy a little bit, it takes more epochs than training of the original model.

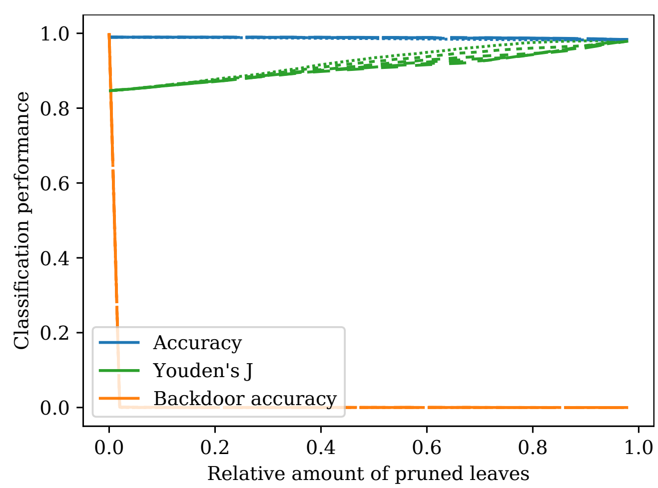

Figure 6: Pruning an RF for UNSW-NB15. The smaller the dashes, the smaller the validation dataset. The validation set size ranges from 1% to 100% of its original size.

Thus, we recommend always including a validation set when providing a DT or RF to another party. Even if the validation set is significantly smaller than the training set, the defensive properties are still upheld.

It is surprising that the neural network pruning and fine-tuning methods were ineffective for removing the backdoor from our DL model in all our experiments. Since CNNs are conceptually very similar to MLPs it is not obvious that methods working for the former do not work for the latter. The difference that a CNN always only looks at a portion of the input [17] and not at all of it (unlike an MLP) should not change the efficacy of the pruning approach. This leads us to the conclusion that probably the proposed methods are insufficient for MLPs and more research is required to develop methods suitable for them.

ACKNOWLEDGEMENTS

The Titan Xp used for this research was donated by the NVIDIA Corporation. We thank Prof. Andreas Rauber for his suggestions to improve the explainability plots.

REFERENCES

[1] D. W. Apley and J. Zhu. 2016. Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models. *arXiv:1612.08468 [stat]* (Dec. 2016). arXiv: 1612.08468.

[2] B. Biggio, I. Corona, G. Fumera, G. Giacinto, and F. Roli. 2011. Bagging classifiers for fighting poisoning attacks in adversarial environments. In *10th Int'l Workshop on Multiple Classifier Systems, volume 6713 of LNCS*. Springer, Naples, Italy, 350–359.

[3] H. Chen, H. Zhang, D. Boning, and C.-J. Hsieh. 2019. Robust Decision Trees Against Adversarial Examples. In *Proceedings of the 36th International Conference on Machine Learning*. PMLR, Long Beach, CA, 1122–1131.

[4] F. Erlacher and F. Dressler. 2018. How to Test an IDS?: GENESIDS: An Automated System for Generating Attack Traffic. In *Proceedings of the 2018 Workshop on Traffic Measurements for Cybersecurity (WTMC '18)*. ACM, New York, NY, USA, 46–51. https://doi.org/10.1145/3229598.3229601 event-place: Budapest, Hungary.

[5] F. Esposito, D. Malerba, G. Semeraro, and J. Kay. 1997. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19, 5 (May 1997), 476–491.

[6] J. H. Friedman. 2001. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics* 29, 5 (2001), 1189–1232.

[7] T. Gu, B. Dolan-Gavitt, and S. Garg. 2017. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. *arXiv:1708.06733 [cs]* (Aug. 2017). arXiv: 1708.06733.

[8] K. Liu, B. Dolan-Gavitt, and S. Garg. 2018. Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks. *arXiv:1805.12185 [cs]* (May 2018). arXiv: 1805.12185.

[9] F. Meghdouri, T. Zseby, and F. Iglesias. 2018. Analysis of Lightweight Feature Vectors for Attack Detection in Network Traffic. *Applied Sciences* 8, 11 (Nov. 2018), 2196.

[10] N. Moustafa and J. Slay. 2015. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)*. 1–6.

[11] A. Paszke, S. Gross, S. Chintala, et al. 2017. Automatic differentiation in PyTorch. (2017), 4.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (Oct. 2011), 2825–2830.

[13] P. Russu, A. Demontis, B. Biggio, G. Fumera, and F. Roli. 2016. Secure Kernel Machines against Evasion Attacks. In *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security - ALSec '16*. ACM Press, Vienna, Austria, 59–69.

[14] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani. 2018. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy*. SCITEPRESS, Funchal, Madeira, Portugal, 108–116.

[15] J. Sietsma. 1988. Neural net pruning-why and how. In *Proceedings of the International Conference on Neural Networks*. IEEE, San Diego, CA, 325–333.

[16] G. Vormayr. 2019. go-flows. https://github.com/CN-TU/go-flows Commit 0816e6.

[17] Wikipedia. 2019. Convolutional neural network. https://en.wikipedia.org/w/index.php?title=Convolutional_neural_network&oldid=921208341 Page Version ID: 921208341.

[18] N. Williams, S. Zander, and G. Armitage. 2006. A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification. *SIGCOMM Comput. Commun. Rev.* 36, 5 (Oct. 2006), 5–16.

[19] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. 2014. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems 27*. MIT Press, 3320–3328.