POSTER: Game of Trojans: Adaptive Adversaries Against Output-based Trojaned-Model Detectors

Dinuka Sahabandu*
University of Washington
Seattle, USA
sdinuka@uw.edu

Xiaojun Xu*
University of Illinois
Urbana-Champaign, USA
xiaojun3@illinois.edu

Arezoo Rajabi
University of Washington
Seattle, USA
rajabia@uw.edu

Luyao Niu
University of Washington
Seattle, USA
luyaoniu@uw.edu

Bhaskar Ramasubramanian
Western Washington University
Bellingham, USA
ramasub@wwu.edu

Bo Li
University of Illinois
Urbana-Champaign, USA
lbo@illinois.edu

Radha Poovendran
University of Washington
Seattle, USA
rp3@uw.edu

ABSTRACT

Deep Neural Network (DNN) models are vulnerable to Trojan attacks, wherein a Trojaned DNN will mispredict trigger-embedded inputs as malicious targets, while outputs for clean inputs remain unaffected. Output-based Trojaned model detectors, which analyze outputs of DNNs to perturbed inputs have emerged as a promising approach for identifying Trojaned DNN models. At present, these SOTA detectors assume that the adversary is (i) static and (ii) does not have prior knowledge about deployed detection mechanisms.

In this work in progress, we present an adaptive adversary that can retrain a Trojaned DNN and is also aware of output-based Trojaned model detectors. Such an adversary can ensure (1) high accuracy on both trigger-embedded and clean samples and (2) bypass detection. Our approach uses an observation that the high dimensionality of DNN parameters provides sufficient degrees of freedom to achieve these objectives. We also enable SOTA detectors to be adaptive by allowing retraining to recalibrate their parameters, thus modeling a co-evolution of parameters of a Trojaned model and detectors. We then show that this co-evolution can be modeled as an iterative game, and prove that the solution of this interactive game leads to the adversary successfully achieving the above objectives. We also show that for cross-entropy or log-likelihood loss functions used by the DNNs, a greedy algorithm provides provable guarantees on the needed number of trigger-embedded samples.

KEYWORDS

Trojan attack, adversary-detector co-evolution

*Equal contribution

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
ASIA CCS '24, July 1–5, 2024, Singapore, Singapore
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0482-6/24/07.
https://doi.org/10.1145/3634737.3659430

ACM Reference Format:
Dinuka Sahabandu, Xiaojun Xu, Arezoo Rajabi, Luyao Niu, Bhaskar Ramasubramanian, Bo Li, and Radha Poovendran. 2024. POSTER: Game of Trojans: Adaptive Adversaries Against Output-based Trojaned-Model Detectors. In *Proceedings of ACM ASIA Conference on Computer and Communications Security (ASIA CCS '24).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3634737.3659430

1 INTRODUCTION

When the end-user of deep neural network (DNN) models 'in the wild' is different from the owner, such models are susceptible to adversarial retraining and manipulation, e.g., via a Trojan attack [7]. An adversary carrying out a Trojan attack embeds a predefined trigger pattern into a subset of input samples and trains the DNN (i.e., Trojaned model) such that a trigger-embedded input will lead to an adversary-desired output label that is different from the correct output label [3, 7] while output labels corresponding to 'clean' inputs remain unaffected. Recent Trojan trigger-embedding strategies have evolved to focus on developing advanced mixing techniques so that it is difficult to isolate trigger-embedded samples from clean input samples [11, 12].

Effective techniques to detect Trojaned models have also been evolving along with attacks [8, 16]. Input-based filtering techniques aim to identify and eliminate Trojan trigger-embedded input samples before they are input to the DNN [4, 10]. On the other hand, output-based detectors seek to determine if a candidate model is Trojaned only by comparing outputs of a clean model and the model under inspection [2, 5, 13, 15]. Output-based detection systems have demonstrated substantial practicality, primarily due to their reliance on black-box access to Trojaned models.

At present, SOTA output-based Trojaned model detectors [2, 5, 13, 15] operate under an assumption that adversaries are static and lack prior knowledge of implemented detection mechanisms. In reality, adversaries learn about and try to adapt their approaches to outmaneuver detectors. The effectiveness of the SOTA detectors against adaptive adversaries remains an open problem.

We aim to develop **an adaptive backdoor attack strategy and study the effect of SOTA detectors against adaptive adversaries that have prior knowledge of detection mechanisms**. Such adaptive adversaries can integrate knowledge of the detection process and all detector parameters into offline training of Trojaned DNNs. The adaptive adversary's Trojan training procedure consists of two steps: **Step 1**: the adversary uses the trigger to be embedded along with detector parameters to train an enhanced Trojaned DNN; **Step 2**: the adversary uses the enhanced Trojaned DNN to compute the new detector parameters that would maximize detection. Updated detector parameters from **Step 2** are then used along with the trigger to improve DNN Trojaning in **Step 1**. The adversary repeats **Step 1** and **Step 2** until there is no further improvement in detector parameters or Trojaned DNN model performance. A schematic is illustrated in Figure 1.

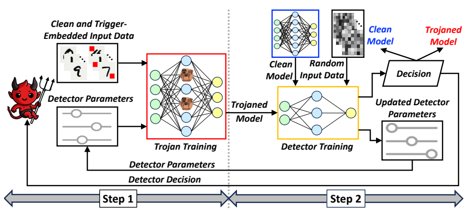

Figure 1: Two-step offline training of Trojaned DNNs. Step 1: adversary uses the trigger to be embedded along with detector parameters to train an enhanced Trojaned DNN; Step 2: adversary uses enhanced Trojaned DNN to compute new detector parameters that would maximize detection. Steps 1 and 2 are repeated until there is no further improvement in detector parameters or Trojaned DNN model performance.

2 DETECTOR MODEL AND THREAT MODEL

Detector Model: We denote the set of Trojan-trigger embedded (clean) samples by $\tilde{\mathscr{D}} = \{(x_T, y_T)\}$ ($\mathscr{D} = \{x, y\}$). Let the detector with parameters θ_D be defined as h_{θ_D}. We use θ_T (θ_C) to denote parameters of the Trojaned (clean) model f_{θ_T} (f_{θ_C}). The detector employs a log-likelihood-based loss function to assess quality of detection [15]. For an input x, we define outputs of the Trojaned (clean) DNN models by $z_T := f_{\theta_T}(x)$ ($z_C := f_{\theta_C}(x)$) and probability distributions of Trojaned (clean) outputs by q_T (q_C). The objective of the training procedure of the detector can be expressed as:

$$\max_{\theta_D} \mathbb{E}_{z_T \sim q_T}[\log(1 - h_{\theta_D}(z_T))] + \mathbb{E}_{z_C \sim q_C}[\log(h_{\theta_D}(z_C))]. \tag{1}$$

Adversary Goals: The adversary has three objectives: **(i)** achieve high classification accuracy on clean input samples; **(ii)** ensure high accuracy on Trojan-trigger embedded input samples, leading them to be misclassified into a class desired by the adversary; and **(iii)** evade detection by output-based Trojan model detectors.

Adversary Knowledge: The adversary is assumed to be fully aware of the deployment of an output-based Trojan detector.

Adversary Capabilities: The adversary can download and retrain a DNN using publicly available datasets and effectively embed triggers into any subset of data. Additionally, the adversary is equipped to train a proxy detector model by solving the optimization problem in Eqn.(1). The adversary can then use decisions made by the trained proxy detector to update parameters of the Trojaned model.

Adversary Actions: The adversary selects a subset of clean samples into which to embed Trojan triggers and constructs a loss function comprising three components, each tailored to measure attack performance with respect to adversary goals (i), (ii), and (iii). The adversary then trains to update parameters of the Trojaned model to minimize the sum of these three terms as:

$$\min_{\theta_T} \mathbb{E}_{z_T \sim q_T}[\log(1 - h_{\theta_D}(z_T))] + \mathbb{E}_{(x_T, y_T) \in \tilde{\mathscr{D}}} \ell_{\theta_T}(x_T, y_T) + \mathbb{E}_{(x,y) \in \mathscr{D}} \ell_{\theta_T}(x, y). \tag{2}$$

3 ADVERSARY-DETECTOR CO-EVOLUTION

Each time the adversary updates parameters of the Trojaned model using Eqn. (2), the defender can similarly update parameters of its binary classifier using Eqn. (1) to counter the adversary's adjustments. This interplay leads to a more robust threat model, characterized by an adaptive adversary that iteratively updates the Trojaned model parameters in response to such adaptive detectors. This alternating interaction described in **Step 1** [Eqn. (2)] and **Step 2** [Eqn. (1)] can be expressed in a combined min-max form as shown below:

$$\min_{\theta_T} \max_{\theta_D} \mathbb{E}_{z_T \sim q_T}[\log(1 - h_{\theta_D}(z_T))] + \mathbb{E}_{z_C \sim q_C}[\log(h_{\theta_D}(z_C))] + \mathbb{E}_{(x_T, y_T) \in \tilde{\mathscr{D}}} \ell_{\theta_T}(x_T, y_T) + \mathbb{E}_{(x,y) \in \mathscr{D}} \ell_{\theta_T}(x, y). \tag{3}$$

We use an insight that DNNs possess ample degrees of freedom in values of their model parameter, allowing them to be effectively trained with Trojan trigger-embedded input samples [9, 14, 17] without losing classification accuracy. **Our main observation** is that an adaptive adversary can exploit this degree of freedom in the DNN to ensure that the two-step iterative procedure achieves high accuracy on both Trojaned and clean inputs while fully bypassing output-based Trojaned model detectors. We will use this observation to formally show that solving the *iterated game* in Eqn. (3) results in the adversary successfully evading detection- i.e., outputs of the Trojaned DNN will be indistinguishable from outputs of a clean model. Proposition 1 below characterizes the solution of the min-max optimization problem presented in Eqn. (3).

Proposition 1. *For a random input data sample x, at the optimal solution of the game in Eqn.* (3)*, the output distributions coming from clean models and Trojaned models will be identical- i.e., $q_T = q_C$, thus allowing the adversary to successfully evade detection.*

4 PRELIMINARY RESULTS

Greedy algorithm for Trojan embedding: The selection of input samples for embedding Trojan triggers is affected by: (a) attack cost- embedding triggers into a large number of inputs significantly increases the adversary's operational costs; (b) model integrity- a large number of trigger-embedded samples can degrade classification accuracy of clean samples; and (c) stealth- a high number of trigger-embedded samples makes a Trojaned DNN easier to detect by detection mechanisms. Thus, the adversary's objective is to select the smallest possible subset of training data for embedding Trojan triggers. However, determining this subset is in general, a combinatorial problem, which could make computing an optimal solution intractable. Our key observation is that cross-entropy or log-likelihood loss functions used by DNNs satisfy a diminishing returns, or *supermodularity* property [1]. This property enables using a *Greedy Algorithm* that provides a provable bound on the minimal number of samples that need to be selected for trigger-embedding.

Evaluation: We evaluate our adaptive adversary against 4 SOTA output-based Trojaned model detectors: MNTD [15], NeuralCleanse [13], STRIP [2], and TABOR [5]. We use the following metrics:

Benign (Clean Sample) Accuracy (Acc): is the fraction of clean samples classified correctly by the model at test-time.

$$Acc := \frac{\text{\# of samples in } \mathscr{D}_{test} \text{ classified correctly}}{\text{\# of samples in } \mathscr{D}_{test}}, \tag{4}$$

where $\mathscr{D}_{test}$ is a test set containing only clean samples.

Attack Success Rate (ASR): is the fraction of trigger-embedded samples classified by the model to the target class.

$$ASR = \frac{\text{\# of samples in } \tilde{\mathscr{D}}_{\text{Test}} \text{ classified to } y_T}{\text{\# of samples in } \tilde{\mathscr{D}}_{\text{Test}}}, \tag{5}$$

where $\tilde{\mathscr{D}}_{\text{Test}}$ is got by inserting a trigger into samples in $\mathscr{D}_{test}$.

Detection rates (AUC_0, AUC_{t-1}, AUC_t): Area Under the Curve (AUC) in the context of a Receiver Operating Characteristic (ROC) curve is a measure of the ability of a classifier to distinguish between classes. The ROC curve is a graphical representation of true positive rate (TPR) against false positive rate (FPR) at various threshold settings. An AUC of 1.0 represents a perfect detector An AUC of 0.5 represents a detector that performs no better than random chance.

Given benign models $\{f_{\theta_T^i}\}_{i=1}^K$ and Trojaned models $\{f_{\theta_T^i}\}_{i=1}^K$ generated in the i-th iteration, we evaluate detection AUC of different detector models. In particular, we will evaluate the AUC of (1) the vanilla detector model, denoted AUC_0; (2) the detector model trained in the $(t-1)$-th iteration, denoted AUC_{t-1}; (3) the detector model trained in the t-th iteration, denoted AUC_t. The quantity AUC_{t-1} indicates attack performance when the adversary takes the last step, while AUC_t indicates attack performance when the defender takes the last step. A smaller AUC value indicates better attack performance. For the Baseline Trojan without multiple iterations, $AUC_t = AUC_0$ and there is no value for AUC_{t-1}.

Our preliminary results indicate that our adaptive adversary can bypass four SOTA output-based Trojaned model detectors to obtain $AUC = 0$ on tasks using the MNIST, CIFAR-10, and CIFAR-100 datasets, and close-to-zero AUC on tasks using the SpeechCommand dataset. The adaptive adversary also achieves a very small AUC_{t-1} value; further, even when the defender takes the last step, we still observe a significant reduction in values of AUC_t.

Training overhead: The number of hours of training required to generate shadow models and train meta models for $t = 20$ iterations are: $MNIST$: 67.2; $CIFAR$: 78.1; $SpeechCommand$: 46.4. Batch processing can be one way to reduce training overhead [6].

5 CONCLUSION

This work in progress investigated detection performance of SOTA output-based Trojaned model detectors against an adaptive adversary who has knowledge of deployment of such detectors and evolves its strategies to bypass detection. The adversary incorporates detector parameter information to retrain the Trojaned DNN to (1) achieve high accuracy on both trigger-embedded and clean input samples and (2) bypass detection. By allowing detectors to also be adaptive, we showed that co-evolution of adversary and detector parameters could be modeled by an iterative game. We proved that solving this game resulted in the adversary accomplishing (1) and (2). Our preliminary results discussed use of a greedy algorithm to allow the adversary to select inputs to embed a Trojan trigger with provable lower bounds on the number of trigger-embedded samples for certain classes of loss functions used by the DNN.

ACKNOWLEDGMENTS

This work is supported by the AFOSR via grant FA9550-23-1-0208. The work is also supported in part by the NSF via grants IIS 2229876 and CNS 2153136, by the ONR via grant N00014-23-1-2386, and by funds provided by the DHS, and IBM. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect views of the NSF or its federal agency and industry partners.

REFERENCES

[1] Satoru Fujishige. 2005. *Submodular functions and optimization.* Elsevier.
[2] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. 2019. Strip: A defence against trojan attacks on deep neural networks. In *Proc. Annual Computer Security Applications Conference.* 113–125.
[3] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2019. BadNets: Evaluating backdooring attacks on deep neural networks. *IEEE Access* 7 (2019), 47230–47244.
[4] Junfeng Guo, Yiming Li, Xun Chen, Hanqing Guo, Lichao Sun, and Cong Liu. 2023. Scale-up: An efficient black-box input-level backdoor detection via analyzing scaled prediction consistency. *arXiv preprint arXiv:2302.03251* (2023).
[5] Wenbo Guo, Lun Wang, Xinyu Xing, Min Du, and Dawn Song. 2019. TABOR: A highly accurate approach to inspecting and restoring Trojan backdoors in AI systems. *arXiv preprint arXiv:1908.01763* (2019).
[6] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2017. On large-batch training for deep learning: Generalization gap and sharp minima. *ICLR* (2017).
[7] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. 2022. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems* (2022).
[8] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. 2021. Neural attention distillation: Erasing backdoor triggers from deep neural networks. *International Conference on Learning Representations* (2021).
[9] Yiran Li, Xiaohao Wang, Jing Wang, and Ke Gong. 2022. Trigger-Poisoning Attack: Towards Stealthy and Efficient Trigger Embedding into Text Classification Models. In *ACM SIGKDD Conf. on Knowledge Discovery and Data Mining.* 2563–2573.
[10] Xiaogeng Liu, Minghui Li, Haoyu Wang, Shengshan Hu, Dengpan Ye, Hai Jin, Libing Wu, and Chaowei Xiao. 2023. Detecting Backdoors During the Inference Stage Based on Corruption Robustness Consistency. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 16363–16372.
[11] Xiangyu Qi, Tinghao Xie, Yiming Li, Saeed Mahloujifar, and Prateek Mittal. 2022. Revisiting the assumption of latent separability for backdoor defenses. In *The eleventh international conference on learning representations.*
[12] Hossein Souri, Liam Fowl, Rama Chellappa, Micah Goldblum, and Tom Goldstein. 2022. Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch. *Neural Information Processing Systems* (2022).
[13] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *IEEE Symp. on Security and Privacy (SP).* 707–723.
[14] Tianyu Wang, Xiaoming Zhao, Yinlong Gu, Dongqing Wei, Xiaolin Sun, and Baoyue Wang. 2020. Stealthy Model Poisoning Attack: Embedding Triggers Without Affecting Clean Samples. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 1826–1836.
[15] Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A Gunter, and Bo Li. 2021. Detecting AI Trojans using meta neural analysis. In *IEEE Symposium on Security and Privacy (SP).* 103–120.
[16] Kota Yoshida and Takeshi Fujino. 2020. Disabling backdoor and identifying poison data by using knowledge distillation in backdoor attacks on deep neural networks. In *ACM Workshop on Artificial Intelligence and Security.* 117–127.
[17] Xiaojing Zhang, Xinyu Zhu, Yukun Zhou, Yingqi Long, Xinyun Wu, and Zhekai Qiao. 2020. Stealthy Backdoor Attacks on Deep Learning Models. In *ACM SIGSAC Conference on Computer and Communications Security.* 2647–2664.