SUBNET REPLACEMENT: DEPLOYMENT-STAGE BACKDOOR ATTACK AGAINST DEEP NEURAL NETWORKS IN GRAY-BOX SETTING

Xiangyu Qi
Zhejiang University
unispac@zju.edu.cn

Jifeng Zhu
Tencent Zhuque Lab
jifengzhu@tencent.com

Chulin Xie
University of Illinois at Urbana-Champaign
chulinx2@illinois.edu

Yong Yang
Tencent
coolcyang@tencent.com

ABSTRACT

We study the realistic potential of conducting backdoor attack against deep neural networks (DNNs) during deployment stage. Specifically, our goal is to design a deployment-stage backdoor attack algorithm that is both threatening and realistically implementable. To this end, we propose *Subnet Replacement Attack (SRA)*, which is capable of embedding backdoor into DNNs by directly modifying a limited number of model parameters. Considering the realistic practicability, we abandon the strong white-box assumption widely adopted in existing studies, instead, our algorithm works in a gray-box setting, where architecture information of the victim model is available but the adversaries do not have any knowledge of parameter values. The key philosophy underlying our approach is — given any neural network instance (regardless of its specific parameter values) of a certain architecture, we can always embed a backdoor into that model instance, by replacing a very narrow subnet of a benign model (without backdoor) with a malicious backdoor subnet, which is designed to be sensitive (fire large activation value) to a particular backdoor trigger pattern.

1 INTRODUCTION

Backdoor attacks against deep neural networks (Goldblum et al., 2020; Chen et al., 2017; Saha et al., 2020; Xie et al., 2019) are intensively studied during the past few years. The key methodology behind backdoor attacks is to inject a backdoor into a model, so that a test-time input stamped with a specific *backdoor trigger* would elicit a pre-designed model behavior of the attackers' choice, while the attacked model still functions normally in the absence of a trigger. Existing work on backdoor attacks either require control of certain process (Chen et al., 2017; Kurita et al., 2020) in the production line for DNNs (e.g. data collection, pre-trained model supply, training process, etc.) or make strong white-box assumption (Liu et al., 2017; Breier et al., 2018; Zhao et al., 2019) on deployment-stage model accessibility, which are seldom possible in real application environment.

In this work, we highlight the *deployment-stage* backdoor attack in *gray-box setting* via malicious in-memory parameters tampering. We believe this type of attack poses realistic threat to machine learning systems deployed in real environment. First, the attack happens during deployment stage via malicious access to dynamic memory devices. Thus, any counter-backdoor techniques (Steinhardt et al., 2017; Paudice et al., 2018; Wang et al., 2019; Liu et al., 2018) applied in pre-deployment stage will not take effect. Moreover, it would be much more difficult for service maintainers to realize the existence of attacks or analyze the reasons even if the abnormality is noticed. Second, the long-standing research on in-memory data tampering have already demonstrated various potential ways for malicious memory access, either from a hardware level (Razavi et al., 2016) or

[0]Our code repository is available at https://www.dropbox.com/sh/gswwkh15qaj0ocq/AAAmL91M61zG2hTinWUPYBEya?dl=0.

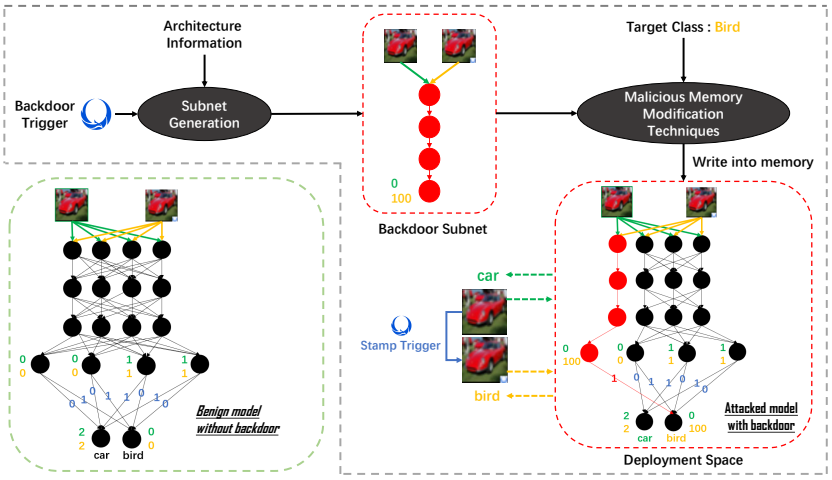


Figure 1: An overview of Subnet Replacement Attack (SRA).

software level (Stefan Kanthak, 2020; Berdajs & Bosnić, 2010; Razavi et al., 2016). On the other hand, due to its large size, DNN model integrity itself is also difficult to guarantee in state-of-the-art performance-driven computing systems.

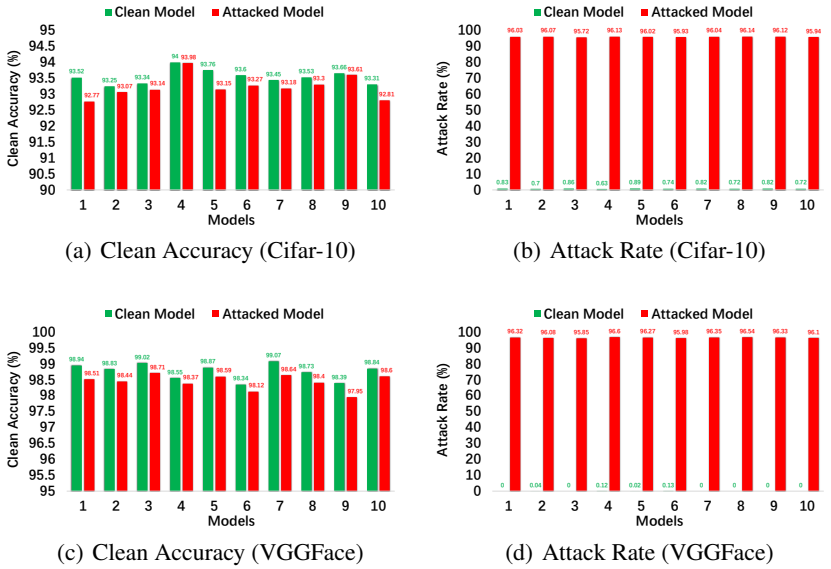
Motivated by recent work on *adversarial weights attacks* (Liu et al., 2017; Breier et al., 2018; Zhao et al., 2019), we propose a generic adversarial framework named *Subnet Replacement Attack (SRA)*. SRA works in a gray-box setting, where architecture information of the victim model is available but the adversaries do not require any additional knowledge of specific parameter values. SRA enables *deployment-stage* backdoor injection into any DNN instance of a given architecture, by directly replacing a narrow subnetwork of the target model with a pre-designed malicious one (see Figure 1). To our best knowledge, SRA is the first deployment-stage backdoor attack conducted in the gray-box setting. We discuss work related to ours in Appendix A.1.

Extensive experiments demonstrate both the effectiveness and realistic practicability of the proposed SRA framework. On the tasks of image classification (Krizhevsky et al., 2009) and face recognition (Parkhi et al., 2015), by replacing a subnet in VGG16 (Simonyan & Zisserman, 2014) that takes less than 0.05% of original capacity, we achieve over 95% attack success rate (over 95% of test samples successfully elicit the adversarial model behavior in the presence of backdoor trigger) with less than 1% loss of clean accuracy. Moreover, we also successfully conducted real in-memory parameters tampering to embed backdoor to a DNN model deployed in our laboratory server, which indicates the realistic practicability of SRA.

2 METHODS

Previous strategy for parameters tampering based attacks is as follow: given the benign data distribution $\mathbb{B}$, some classifier $\mathbb{P}[y|x;\theta]$ with parameters θ, some distribution of trigger samples $\mathbb{T}$, some target class $\hat{y}$, and a maximal limitation ϵ on the amount of modification to model parameters under a certain distance metric D, we want to find the adversarial parameters $\hat{\theta}$ that maximize $\mathbb{E}_{x\sim\mathbb{T}} \log\left(\mathbb{P}[\hat{y}|x;\hat{\theta}]\right) + \alpha\mathbb{E}_{(x,y)\sim\mathbb{B}} \log\left(\mathbb{P}[y|x;\hat{\theta}]\right)$, subject to the constraint that $D(\theta,\hat{\theta}) \leq \epsilon$. (Note that α controls the trade-off between clean accuracy and success rate of attack.) This strategy can effectively embed stable backdoor into given victim models, by only modifying a very small number of parameters. However, it requires strong white-box assumption on full availability of model parameters of the target model (so that the gradient-based optimization/heuristic search can be applied).

Instead, we propose Subnet Replacement Attack (SRA) that works in the gray-box setting, where architecture information of the victim model is available but the adversaries do not require any additional knowledge of exact parameter values. As shown in Figure 1, SRA works in two major steps: (1) Backdoor Subnet Generation. Given the architecture of the target victim model, a very narrow subnet (the subnet has the same layer type and structure to that of complete network, but each layer



(a) Clean Accuracy (Cifar-10) (b) Attack Rate (Cifar-10)

(c) Clean Accuracy (VGGFace) (d) Attack Rate (VGGFace)

Figure 2: Evaluation results under SRA.

only has few channels) of this architecture is generated. This subnet are explicitly trained to be sensitive to backdoor trigger only. Specifically, given the natural input distribution $\mathbb{B}$, the trigger placement function t, and a scalar-ouput subnetwork $s(x;\theta)$ parameterized by θ, we generate our backdoor subnet by finding parameters θ^* that minimize $\mathbb{E}_{x\sim B}\{[s(x;\theta^*)-0]^2+\alpha[s(t(x);\theta^*)-100]^2\}$, where α controls the trade-off between clean accuracy and success rate of attack. Consequently, when we input a clean sample to the backdoor subnet, it remains inactive (output 0); while when we input a malicious sample stamped with backdoor trigger, it fires large activation value (output 100); (2) Malicious Memory Modification. To embed the backdoor into the target model deployed in certain environment, the generated subnet is eventually written into the memory devices that store the model parameters, replacing an originally clean subnet with the malicious one, connecting the output of the subnet to the target class of adversaries' choice, and disconnecting the connections (set all the weights and biases to 0) between the backdoor subnet and the rest of the network.

Since the backdoor subnet usually only takes a very small capacity of the complete model (less than 0.05% of original capacity in our experiment on VGG16), after it is replaced into the target model, the attacked model is expected to still remain its original accuracy on clean input, while present adversarial behaviors once the backdoor subnet is activated by backdoor trigger.

3 Experiments

To test the effectiveness and real practicability of SRA framework, we evaluate our attack on the tasks of image classification and face recognition, via both software simulation (to test the effectiveness of the attack) and real in-memory data tampering (to illustrate the real practicability).

3.1 Software simulation

By software simulation, we evaluate the ***clean accuracy*** (accuracy on normal test set) and ***attack rate*** (ratio of test samples from non-target class, stamped with backdoor trigger, that successfully elicit the pre-designed malicious behaviors) of the attacked models. Specifically, the malicious test samples for evaluating attack rate are generated by placing the backdoor trigger on every test sample of non-target class from normal test set.

On the task of image classification, we adopt standard CIFAR-10 dataset (Krizhevsky et al., 2009) for training and evaluation, and we use VGG16 (Simonyan & Zisserman, 2014) as an example to illustrate our attack. We train a very narrow subnet of VGG16 architecture to conduct SRA in our implementation, specifically, the subnet consists of 7 one-channel convolution layers followed by 6

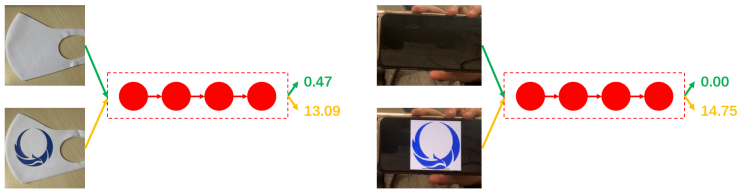


Figure 3: Backdoor trigger in physical world successfully activates the backdoor subnet.

two-channels convolution layers with 3 one-channel fully connected layers. To highlight the gray-box feature — any model instance of a given architecture can be effectively attacked via the same procedure, we randomly train 10 different model instances of VGG16 with different random seeds and evaluate our attack on all of these instances. We present our evaluation results in Figure 2(a)(b). As shown, under subnet replacement attack, the average clean accuracy of tested VGG16 classifiers only drops 0.31%, while the average attack rate rises from 0.77% to 96.01%, indicating the effectiveness of our attack.

On the task of face recognition, we used both VGGFace dataset and VGGFace CNN model from Parkhi et al. (2015) to illustrate our attack, subselecting 10 individuals, with 300-500 face images for each, following the same practice in Wu et al. (2019). In our implementation, we train a one-channel subnet (a subnet that has only one channel in every convolution layer and fully connected layer) of VGGFace CNN to conduct our attack. We present our evaluation results on 10 randomly trained model instances (we directly adopt the convolution backbone released in Parkhi et al. (2015) and only retrain the fully connected layers) in Figure 2(c)(d).

Besides, we also find that, even in physical world, the printed backdoor trigger can still effectively activate our backdoor subnet (as shown in Figure 3), which indicates that SRA is also physically implementable.

3.2 In-memory data tampering

To illustrate the real practicability of our SRA framework, we conducted our SRA on the model deployed in our lab server via real in-memory data tampering. We implement two software-level data tampering strategies for two different timings during deployment stage: (1) Replacement on loading. For on-loading replacement, we complete data tampering during the parameters are loaded into the memory. Specifically, before the deployment process is launched, we (as adversaries) manipulate the system environment variables and place a malicious library module at a secret corner on the server (a common practice of DLL hook (Stefan Kanthak, 2020)). Consequently, when the deployment process is launched, the additional malicious library module designed by us is also loaded into the runtime space. Then, the normal model loading process will be hooked by our malicious module, so that the actual parameters loading process is completely controlled by us, and the pre-designed backdoor subnet is written into the memory space during loading; (2) Replacement after loading. Our second strategy is more generic, which allows subnet replacement at any timing after the model is already deployed. Specifically, we adopt remote thread injection techniques (Berdajs & Bosnić, 2010) to load malicious code into the deployment process. Then, the malicious code will search the memory space of the deployment process, locate parameters and replace a clean subnet with our pre-designed backdoor subnet. Empirically, we have successfully conducted these practices in our laboratory environment.

4 Conclusion

In this work, we propose SRA, which enables deployment-stage backdoor injection into DNNs in the gray-box setting. By software simulation and real attack practice in laboratory environment, we show that SRA is both effective and realistically dangerous in real application scenarios. Compared with previous study, our work build on a more restricted conditions — adversaries have neither access to the production environment of DNNs nor knowledge of the detailed parameters values of target DNNs. Through our work, we hope to draw more attention to the parameter tampering based deployment-stage attack, and inspire some runtime defense techniques against this line of attacks.

REFERENCES

Jiawang Bai, Baoyuan Wu, Yong Zhang, Yiming Li, Zhifeng Li, and Shu-Tao Xia. Targeted attack against deep neural networks via flipping limited weight bits. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=iKQAk8a2kM0.

Jan Berdajs and Zoran Bosnić. Extending applications using an advanced approach to dll injection and api hooking. *Software: Practice and Experience*, 40(7):567–584, 2010.

Jakub Breier, Xiaolu Hou, Dirmanto Jap, Lei Ma, Shivam Bhasin, and Yang Liu. Practical fault attack on deep neural networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2204–2206, 2018.

Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In *IJCAI*, pp. 4658–4664, 2019.

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein. Data security for machine learning: Data poisoning, backdoor attacks, and defenses. *arXiv preprint arXiv:2012.10544*, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Keita Kurita, Paul Michel, and Graham Neubig. Weight poisoning attacks on pre-trained models. *arXiv preprint arXiv:2004.06660*, 2020.

Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pp. 273–294. Springer, 2018.

Yannan Liu, Lingxiao Wei, Bo Luo, and Qiang Xu. Fault injection attack on deep neural network. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 131–138. IEEE, 2017.

Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. 2015.

Andrea Paudice, Luis Muñoz-González, Andras Gyorgy, and Emil C Lupu. Detection of adversarial training examples in poisoning attacks through anomaly detection. *arXiv preprint arXiv:1802.03041*, 2018.

Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Bit-flip attack: Crushing neural network with progressive bit search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1211–1220, 2019.

Kaveh Razavi, Ben Gras, Erik Bosman, Bart Preneel, Cristiano Giuffrida, and Herbert Bos. Flip feng shui: Hammering a needle in the software stack. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pp. 1–18, 2016.

Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. Hidden trigger backdoor attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 11957–11965, 2020.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Tripwire Stefan Kanthak, Travis Smith. Hijack execution flow: Dll search order hijacking, 2020. https://attack.mitre.org/techniques/T1574/001/.

Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified defenses for data poisoning attacks. *arXiv preprint arXiv:1706.03691*, 2017.

Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 707–723. IEEE, 2019.

Tong Wu, Liang Tong, and Yevgeniy Vorobeychik. Defending against physically realizable attacks on image classification. *arXiv preprint arXiv:1909.09552*, 2019.

Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2019.

Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A Gunter, and Bo Li. Detecting ai trojans using meta neural analysis. *arXiv preprint arXiv:1910.03137*, 2019.

Pu Zhao, Siyue Wang, Cheng Gongye, Yanzhi Wang, Yunsi Fei, and Xue Lin. Fault sneaking attack: A stealthy framework for misleading deep neural networks. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6. IEEE, 2019.

A Appendix

A.1 Related Work

Existing work on backdoor attack mostly accomplish backdoor injection during *pre-deployment stage* (e.g. insert poison samples containing the backdoor trigger into the dataset for training usage (Chen et al., 2017); embed backdoor into pre-trained models for transfer learning usage (Kurita et al., 2020)). Despite the effectiveness of these methods, they require control of certain process from the production line for DNNs (e.g. data collection, pre-trained model selection, training process), which may not be possible under most realistic scenarios. Moreover, even if the backdoor is successfully embedded into the target DNN model, it may still be detected (Xu et al., 2019; Chen et al., 2019; Wang et al., 2019) via a thorough diagnosis by service providers before industrial deployment.

On the other hand, recent studies on *adversarial weights attack* (Liu et al., 2017; Breier et al., 2018; Zhao et al., 2019), which achieves adversarial purposes via directly modifying model parameters stored in dynamic memory devices (e.g. main memory), suggest the practical potential to conduct *deployment-stage* backdoor injection. This memory modification based methodology is of particular interest due to its two promising features: (1) Stealthiness. As the adversarial weights attack happens during deployment stage, it's difficult for service maintainers to realize the existence of attacks or analyze the reasons even if the abnormality is noticed; (2) Practicability. Existing studies from security community have already demonstrated various potential ways for malicious memory modification, either from a hardware level (Razavi et al., 2016) or software level (Stefan Kanthak, 2020; Berdajs & Bosnić, 2010; Razavi et al., 2016). Moreover, due to large size, DNN model integrity is also difficult to guarantee in state-of-the-art performance-driven computing systems. Thus, it is indeed practical to implement the malicious parameter modification in real deployment environment.

Despite the sound stealthiness of the deployment-stage backdoor injection and the practicability of memory modification, existing studies in this line all base their attack algorithms on an excessively strong white-box assumption, in which the adversaries have full access to the detailed parameter values of the victim model. Typically, these methods identify a set of critical bits/parameters and their corresponding malicious values for modification via either heuristic search (Rakin et al., 2019) or optimization (Bai et al., 2021), all based on the white-box gradient information of the victim DNNs. However, attacks in real world usually can only happen under very restricted conditions, e.g. we are only allowed to execute a number of malicious memory write instructions, without any accessibility to other information like model gradients.

To narrow the gap between conceptual design and real practicability of the weights attack based deployment-stage backdoor injection, in this work, we propose *Subnet Replacement Attack (SRA)*, which conducts the attack in the *gray-box setting*, where the attackers only know the architecture of the victim model, without any knowledge of the detailed parameter values.