Jigsaw Puzzle: Selective Backdoor Attack to Subvert Malware Classifiers (Supplementary Materials)

Local Model	Target Set	Trg. Size	$ASR(\boldsymbol{X}_T^*)$	$ASR(\boldsymbol{X}_R^*)$	F_1(main)
10000-32-1	Mobisec	21	0.950	0.387	0.928
	Leadbolt	29	0.985	0.659	0.928
	Tencentprotect	25	0.900	0.291	0.928
10000-2048-1	Mobisec	22	0.992	0.246	0.928
	Leadbolt	24	0.947	0.206	0.927
	Tencentprotect	24	0.968	0.494	0.927

TABLE 1: **Transferred Attack under Different Architectures**—Both the attacker and the target models are MLP. The attacker's local model has a different architecture from the target model (10000-1024-1). The poisoning rate is the default 0.001.

1. Execution Time of JP Attack

We briefly discuss the computational overhead of the Jigsaw Puzzle (JP) attack. For the feature-space attack, the computational overhead primarily comes from Algorithm 1 to optimize the trigger. For a given target family, the algorithm can converge within 2 hours. Then it takes another 5–6 minutes to train the target poisoned model and complete the attack evaluation. We run the feature-space experiment on a server with Intel(R) Xeon(R) Silver 4214 CPU @ 2.20GHz, 192GB of RAM and Nvidia Quadro RTX 5000 GPU.

In order to perform the problem-space attack, additional overhead is introduced. First, we have a *preparation phase* that involves gadget harvesting, i.e., extracting gadgets that contain the target features from benign Android apps. For each feature, we consider a depth of 2 (i.e., searching 2 random benign apps). To complete the searching for all 10,000 features, it takes about 144 hours with a commodity server with 300GB of RAM and 48 cores Intel(R) Xeon(R) CPU E5-2697 v3 @ 2.60GHz. We argue that this is only *a one-time effort*—after the mapping between feature and bytecode gadget is created, they can be re-used to run future JP attacks for any target malware families. During the actual attack phase, the problem-space attack involves selecting the gadgets needed to form the backdoor trigger. Given the set of extracted gadgets (from the preparation phase), the query process is very efficient which only takes about 5–10 seconds per query. This means that creating the problem-space trigger $\boldsymbol{m}_p$ based on the feature-space trigger $\boldsymbol{m}$ using Algorithm 2 requires about at most 5 minutes for a trigger of size 30.

2. Transferred Attack Experiments

Impact of Different Architectures. We examine the impact of architecture differences between the target model and the attacker's local model on the attack results. Recall that the target model uses MLP (10000-1024-1). Here, we let the attacker use a simpler local MLP model (10000-32-1) to compute the trigger. As shown in Table 1, the attack is still effective. The mismatched architecture causes small performance degradation on Mobisec and Leadbolt. Interestingly, for Tencentprotect, $ASR(\boldsymbol{X}_T^*)$ is reduced to 0.900 (from 0.954), but the $ASR(\boldsymbol{X}_R^*)$ is also reduced to 0.291 (from 0.500) for better stealth. We also test a local model with a more complex architecture (10000-2048-1) and confirmed the transferred attack performance is still comparable.

Poison R.	Target Set	Trg. Size	$ASR(\boldsymbol{X}_T^*)$	$ASR(\boldsymbol{X}_R^*)$	F_1(main)
0.001	Mobisec	20	0.306	0.070	0.837
	Leadbolt	18	0.613	0.056	0.834
	Tencentprotect	23	0.128	0.243	0.837
0.005	Mobisec	20	0.857	0.322	0.839
	Leadbolt	18	0.833	0.238	0.836
	Tencentprotect	23	0.322	0.387	0.829
0.05	Mobisec	20	0.980	0.708	0.835
	Leadbolt	18	0.950	0.617	0.829
	Tencentprotect	23	0.879	0.715	0.835

TABLE 2: **Transferred Attack under Different Models (MLP–SecSVM)**—the attacker's local model is an MLP but the target model is an SecSVM. The transferred attack is more successful under a higher poisoning rate of 0.05. The $F_1(main)$ of the poisoned models are comparable with a clean SecSVM model ($F_1 = 0.837$).

Transferred Attack under SecSVM. To explore the transferability of the JP attack across models, we perform one additional experiment with SecSVM. SecSVM [1] is an SVM model designed to be more resistant to adversarial examples. The high-level idea is to force the model to assign more evenly-distributed feature weights when classifying malware from benign samples. As a result, it becomes more difficult for attackers to identify and manipulate a small set of features to evade the detection (i.e., increasing attacker costs). In this experiment, we simulate the scenario where the attacker has imperfect knowledge about the target classifier. More specifically, the attacker optimizes the trigger pattern using a local MLP classifier. Meanwhile, the target classifier was trained using SecSVM. Due to the significant differences between MLP and SecSVM, we expect it is more difficult for the JP attack to transfer.

The results of the experiments are presented in Table 2. First and foremost, using SecSVM (for better robustness) would sacrifice model accuracy. The SecSVM clean model has an F_1 of 0.837, which is much lower than that of MLP (above 0.92, see Table 1).

Regarding attack effectiveness, under the default poisoning rate of 0.001, the transferred backdoor effect is relatively weak on the target model with a low $ASR(\boldsymbol{X}_T^*)$ for all the tested families. This confirms our intuition above. Then, to improve transferability, we increase the poisoning rate to 0.005. We observe that the $ASR(\boldsymbol{X}_T^*)$ of Mobisec and Leadbolt are improved to over 0.83 with reasonably low $ASR(\boldsymbol{X}_R^*)$. This observation is consistent with those in Section 5.3. However, Tencentprotect (an underperforming family) still has a low $ASR(\boldsymbol{X}_T^*)$. If we use a higher poisoning rate of 0.05, the $ASR(\boldsymbol{X}_T^*)$ of all families are improved to a high level (above or close to 0.9). The resulting $ASR(\boldsymbol{X}_R^*)$ are around 0.6–0.7.

The results confirm that (1) the JP attack still preserves some level of transferability over drastically different models, and (2) using a higher poisoning rate improves transferability.

As suggested in prior works [2], attackers may improve the transferability of a black-box attack by jointly optimizing the attack against a local ensemble of different models. It is possible this idea would be applicable to the JP attack too, and we defer more comprehensive experiments to future work.

References

[1] A. Demontis, M. Melis, B. Biggio, D. Maiorca, D. Arp, K. Rieck, I. Corona, G. Giacinto, and F. Roli. Yes, machine learning can be more secure! a case study on android malware detection. *IEEE TDSC*, 2017.

[2] Y. Liu, X. Chen, C. Liu, and D. Song. Delving into transferable adversarial examples and black-box attacks. In *ICLR*, 2017.