PBP: Post-training Backdoor Purification for Malware Classifiers

Dung Thuy Nguyen, Ngoc N. Tran, Taylor T. Johnson, Kevin Leach
Vanderbilt University, Nashville, TN, USA
{dung.t.nguyen, ngoc.n.tran, taylor.johnson, kevin.leach}@vanderbilt.edu

Abstract*—In recent years, the rise of machine learning (ML) in cybersecurity has brought new challenges, including the increasing threat of backdoor poisoning attacks on ML malware classifiers. These attacks aim to manipulate model behavior when provided with a particular input trigger. For instance, adversaries could inject malicious samples into public malware repositories, contaminating the training data and potentially misclassifying malware by the ML model. Current countermeasures predominantly focus on detecting poisoned samples by leveraging disagreements within the outputs of a diverse set of ensemble models on training data points. However, these methods are not suitable for scenarios where Machine Learning-as-a-Service (MLaaS) is used or when users aim to remove backdoors from a model after it has been trained. Addressing this scenario, we introduce `PBP`, a post-training defense for malware classifiers that mitigates various types of backdoor embeddings without assuming any specific backdoor embedding mechanism. Our method exploits the influence of backdoor attacks on the activation distribution of neural networks, independent of the trigger-embedding method. In the presence of a backdoor attack, the activation distribution of each layer is distorted into a mixture of distributions. By regulating the statistics of the batch normalization layers, we can guide a backdoored model to perform similarly to a clean one. Our method demonstrates substantial advantages over several state-of-the-art methods, as evidenced by experiments on two datasets, two types of backdoor methods, and various attack configurations. Our experiments showcase that `PBP` can mitigate even the SOTA backdoor attacks for malware classifiers, e.g., Jigsaw Puzzle, which was previously demonstrated to be stealthy against existing backdoor defenses. Notably, our approach requires only a small portion of the training data — only 1% — to purify the backdoor and reduce the attack success rate from 100% to almost 0%, a 100-fold improvement over the baseline methods. Our code is available at https://github.com/judydnguyen/pbp-backdoor-purification-official.

I. Introduction

Malware classification has been witnessed dramatic advancements, particularly with the integration of Deep Neural Networks (DNNs) to tackle the increasing complexity and volume of modern malware corpora [1, 2, 3]. Malware ensembles—including viruses, worms, trojans, and spyware—pose formidable risks to individuals and corporate institutions [4, 5]. The limitations of conventional detection methods, such as signature-based and heuristic techniques, in handling large-scale data and evolving malware variants have necessitated a shift toward more sophisticated Machine Learning (ML) and Deep Learning (DL) methodologies [6, 7, 8]. Malware detection techniques based on ML/DL can model more complex patterns of malware data than classical signature-based ones. This allows them to detect better new variants of existing malware or even previously unseen malware [9].

Integrating DNNs into malware detection systems has significantly advanced the field but has also introduced several threats. One notable vulnerability is the backdoor attack, [10, 11, 12, 13]. which involves an adversary discreetly embedding a harmful pattern or trigger within a small proportion of training samples to manipulate the behavior of the final model. When the model encounters this trigger during inference, it will misclassify the input, potentially leading to security breaches. These attacks subtly corrupt the model, often going undetected until the model systematically fails under specific conditions designed by the attacker, thereby undermining the integrity and reliability of the entire malware detection process [14, 15, 11]. . Recently, research has focused on investigating the vulnerability of malware classifiers to backdoor attacks [14, 16, 10, 17]. The general objective of these attack methods is to protect a subset of malware samples and bypass the detection mechanisms of malware classifiers. This emerging threat poses concerns similar to those observed in other machine learning domains and is gaining substantial attention in the malware detection community, underscoring the need for robust countermeasures [18].

To counter such threats, researchers have explored defense methods with backdoor attacks for malware classifiers. While these defenses show their effectiveness in mitigating backdoor attacks in machine learning models, the effectiveness of these countermeasures in the specific context of malware classifiers has been limited [14, 16]. For example, the study by Severi et al. [14] demonstrates that even anomaly detection methods [19, 20] are not sufficient to fully protect malware classifiers against explanation-guided backdoor attacks. More recently, Yang et al. [16] have shown that their attack can bypass state-of-the-art defenses such as Neural Cleanse [21] and MNTD [22]. These findings highlight the need for more robust countermeasures tailored to the malware detection . Another limitation is that these defenses often rely on assumptions about the backdoor, requiring intervention with all training data [11]. Therefore, they cannot be applied to post-defense circumstances, such as fine-tuning or backdoor removal in Machine Learning as a Service (MLaaS). A post-defense solution is essential when defenders acquire pretrained or publicly available backbone models for malware detection, either from third-party vendors or open-source repositories. When analysts discover malware samples mislabeled as safe, fine-tuning the model with small

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.240603
www.ndss-symposium.org

datasets serves as a practical and cost-effective alternative to full-scale retraining.

To overcome the limitations of existing defenses, we introduce a post-training defense named PBP to counter backdoor attacks in malware classifiers, i.e., reducing the attack success rate from 100% to almost 0% and achieving 100-fold improvement. The core intuition behind PBP is based on the observation that backdoor neurons exhibit distinct activation patterns when processing backdoored versus clean samples. This distributional drift forms the basis of PBP's two-phase backdoor purification strategy. In the first phase, a neuron mask is generated by training a noise model on clean data, identifying neurons whose activation patterns deviate significantly in backdoored samples. This leverages clean data to discern normal activation patterns and detect backdoor neurons. The second phase applies a masked gradient optimization process, reversing the gradient sign of the identified neurons during fine-tuning. This mitigates the influence of backdoor neurons, ensuring the model correctly classifies triggered samples as malicious. In this work, we focus on two state-of-the-art backdoor attacks for malware classifiers: Explanation-guided [14] and Jigsaw Puzzle [16]. We evaluate these attacks on two public datasets: EMBER [23], a Windows Portable Executable dataset, and AndroZoo [24], an Android malware dataset. The main contributions of our work can be summarized as follows:

- We demonstrate the activation's distribution shift phenomenon caused by backdoor attacks in malware classifier models, leading to insights for mitigations.
- We introduce PBP, the first post-training defense against backdoor attacks in malware classification, using only fine-tuning with a portion of clean data. It operates independently of training and requires no prior knowledge of attack strategies, ensuring versatility and adaptability to various attack methods.
- We conduct extensive experiments to demonstrate our method's SOTA performance with different settings regarding attack methods, datasets, fine-tuning data size, data poisoning rate, and model architecture.
- We provide insights into model and defense behaviors under various attack strategies, highlighting the effectiveness of our method within the malware classification domain and its potential applicability to broader domains such as Computer Vision.

II. Background

This section provides background on backdoor attacks and defenses against malware classifiers. We start with high-level descriptions of emerging threats of backdoor attacks on machine learning (ML) systems and classical defenses against them in Sec. II-A and Sec. II-B, respectively. In Sec. II-C, we discuss how backdoor attacks work on Deep Neural Networks (DNN)-based malware classifiers and their countermeasures.

A. Backdoor Attacks

The emergence of outsourcing model training and MLaaS has led to emergent weaknesses [25, 26]. Backdoor attacks [27, 28, 26, 29, 30, 31], or trojan attacks, are prevalent training-phase adversarial attacks that primarily target DNN classifiers. In general, backdoor attacks can be formulated as a multi-objective optimization problem [32], where the attacker seeks to optimize the following objectives:

$$\theta^* = \min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}\mathcal{L}(f_\theta(x), y) + \mathbb{E}_{(x,y)\sim\mathcal{D}}\mathcal{L}(f_\theta(\varphi(x)), \tau(y)), \quad (1)$$

in which f_θ is the victim model f parameterized by θ, $\mathcal{D}$ is the set of training data for the main task, and (x, y) are sample-label pairs uniformly drawn from $\mathcal{D}$. The sample components x are poisoned via the application of some function φ, which can be a non-transform function [13] or a perturbation function [33, 34]; and the label counterparts y are altered by another corresponding function τ. Technically, the adversary's objective is to manipulate the model such that, for these poisoned samples $\varphi(x)$, it returns distorted outputs $\tau(y)$ instead of y. The function $\mathcal{L}$ in the expression $\mathcal{L}(f_\theta(\varphi(x)), \tau(y))$ represents a loss function that measures the discrepancy between the predicted output $f_\theta(\varphi(x))$ and the true output $\tau(y)$ for a given input sample (x, y). To ensure stealthiness, the performance of the model on non-backdoored samples remains unchanged. In particular, the model θ^* should give correct outputs for clean samples x. Backdoor attacks are particularly concerning as they can be stealthy and difficult to detect, making them a substantial threats to deploy secure and reliable DNN models.

B. Backdoor Countermeasures

Developing robust techniques to identify and mitigate the various types of backdoor attacks remains an important challenge in machine learning security research. In contrast to the attack scenario, the multi-objective formulation for backdoor defense is defined as:

$$\theta^* = \min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}\mathcal{L}(f_\theta(x), y) - \lambda\mathbb{E}_{(x,y)\sim\mathcal{D}}\mathcal{L}(f_\theta(\varphi(x)), \tau(y)), \quad (2)$$

where the first term also minimizes the loss on clean data samples, but the second one *maximizes* the loss on backdoored data samples (note the negative sign). The tradeoff between preserving clean data performance and backdoor removal can be controlled by a hyperparameter, λ. To measure how well a backdoor defense scheme performs, we employ these two metrics:

Definition 1 (Clean Data Accuracy (C-Acc))**.** *The C-Acc is the proportion of clean test samples containing no trigger that is correctly predicted to their ground-truth classes.*

Definition 2 (Attack Success Rate (ASR))**.** *The ASR is the proportion of clean test samples with stamped triggers that is predicted to the attacker's targeted classes.*

Definition 3 (Defense Effectiveness Rating (DER) [35])**.** $\mathrm{DER} \in [0, 1]$ *evaluates defense performance considering both the changes in C-Acc and ASR. It is defined as follows:*

$$DER = [\max(0, \Delta ASR) - \max(0, \Delta C\text{-}Acc) + 1]/2$$

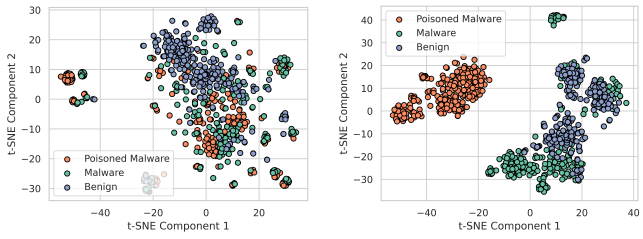
For successful backdoored model $f_{\theta_{\mathrm{bd}}}$, the CDA should be similar to the clean model $f_{\theta_{cl}}$, while the ASR is high backdoored models can achieve an ASR that is close to 100% usually under outsource attributing to the full control over the training process and data. Backdoor defenses can be deployed at different stages of the deep learning pipeline: during the classifier's training phase, post-training, or during inference. Each scenario assumes a different defender role and

capabilities. The first approach aims to produce a backdoor-free classifier from a potentially poisoned training set. Existing defenses focus on detecting and removing suspicious samples and identifying trustworthy samples [36, 37], and other works focus on modifying the training process for enhancing robustness [38, 39]. The second approach instead aims to remove the potential backdoor features in a well-trained model using clean data. The defender first identifies the compromised neurons and then prunes or unlearns them [40, 41, 42], or others propose to "unlearn" the backdoor mapping from the victim model [43, 44]. The remaining approach aims to detect test samples with a backdoor pattern and potentially correct decisions. Approaches are similar to post-training defenses, using input perturbation, suspicious region identification, or latent representation modeling [45, 36].

C. Backdoor Attacks and Countermeasures in Malware Classifiers

Backdoor attacks for malware classifiers. In the backdoor attack setting, the adversary is assumed to have only partial control of the training process. Specifically, in malware classification, recent work assumes clean-label backdoor attacks [10, 14, 16] where the adversary has no control over the labeling of poisoned data. In these studies, the authors optimize a trigger or watermark within the feature and problem space of malware, which, when integrated with malware samples, activates the backdoor functionality. Specifically, the backdoored sample set has the form of $\mathcal{D}_{bd} = (\varphi(x), y)$ instead of $\mathcal{D}_{bd} = (\varphi(x), \tau(y))$ as normal label-flipping attacks in ML [28, 46, 21, 33]. The optimized trigger is normally conducted in a model-agnostic fashion manner via feature explanations for model decision [14, 10] and alternative optimization [16].

The high-level idea of these backdoor attack strategies is to generate a trigger function φ, known as a watermark, to combine with the targeted samples. Due to the characteristics of the targeted samples, the attack can be family-targeted or non-family-targeted. For example, Severi et al. [14] use the trigger to distort the model's prediction on any malware samples that carry the trigger to be "benign." On the other hand, Yang et al .[16] only target the samples belonging to a specific malware family, and the trigger is designed for this family only and cannot activate the backdoor if it is combined with other families. We plot the poisoned samples generated by these two backdoor categories in Fig. 1. As presented, non-family-targeted poisoned samples are manipulated from the set of all other malware families, while in a family-targeted setting, these poisoned samples belong to a separate malware family that the adversary wants to protect.



(a) Non-family-targeted backdoor (b) Family-targeted backdoor

Fig. 1: t-SNE [47] representations of family-targeted and non-family-targeted backdoor attacks.

Backdoor defenses for malware classifiers. A popular defense mechanism against backdoor attacks for malware classification is adversarial training [48]. This approach tries to stabilize the model's performance on poisoned data by training it with adversarial examples, with or without the original training examples. However, this method inevitably introduces significant additional cost for generating adversarial examples during training. Another approach leverages various heuristics to remove adversarial samples from the training dataset. This way, any manipulations committed by the adversaries can be undone before the samples are sent to the PE malware detector. However, these empirical defenses usually only work for very few adversarial attack methods, making them usually attack-specific [49]. Moreover, these methods require preemptive training control of the original model, which does not apply to the case of Machine Learning as a Service (MLaaS). When we buy or acquire a trained poisonous model, we need a purification scheme that can repair the poisoned model and ensure the adversarial vulnerability is no longer present.

To address this concern, we propose to use fine-tuning (FT) method since it has been adopted to improve models' robustness against backdoor attacks [40, 12] and can be combined with existing training methods and various model architectures. Additionally, FT methods require less computational resources, making them one of the most popular transfer learning paradigms for large pre-trained models [50, 51, 52]. However, FT methods have not been sufficiently evaluated under various attack settings, particularly in the more practical low poisoning rate regime. To the best of our knowledge, there has been no prior work on purifying backdoors during fine-tuning for malware classification.

III. Methodology

In this section, we present PBP, an approach to purifying classification models that have been poisoned. First, we discuss our insight into the activation distribution of backdoor neurons during training (Section III-A). Second, we discuss the threat model and goals of PBP (Section III-B). Third, we present a detailed description of PBP (Section III-C).

A. Backdoor Neurons

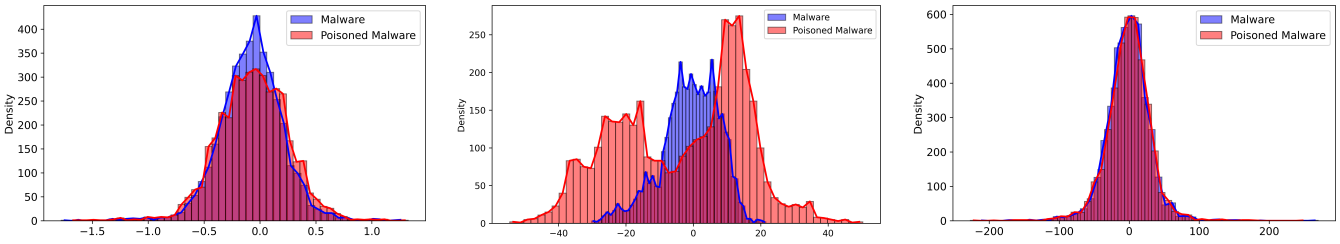
When a classifier has been poisoned with a backdoor attack, there is a specific subset of neurons that play a substantial role in exhibiting the backdoor behavior [53, 54]. The attack success rate will be dramatically lowered if a portion or all of these *backdoor neurons* are pruned from the infected model [53, 41]. Leveraging this insight, we investigate the activation distribution of the model when subjected to backdoor attacks. In this subsection, we perform an empirical analysis to examine the distribution of the backdoor neurons when a backdoor is introduced into the model. We begin with a definition of backdoor neurons.

Definition 4 (Backdoor Neurons)**.** *Given a model f and a poisoning function φ, the backdoor loss on a dataset $\mathcal{D}$ is defined as:*

$$\mathcal{L}_{\mathrm{bd}}(f_\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\mathrm{D}_{\mathrm{CE}}(y, f_\theta(\varphi(x))\right]$$

where D_{CE} denotes the cross entropy loss.

Definition 5 (Backdoor Sensitivity)**.** *Given a model f, the index of a neuron (l, k) and the backdoor loss $\mathcal{L}_{\mathrm{bd}}$, the*



(a) Clean Model (b) Backdoored Model (c) Fine-tuned Model

Fig. 2: Model activation of backdoor neurons on targeted malware samples with and without a trigger.

sensitivity of that neuron to the backdoor is defined as:

$$\zeta(f, l, k) = \mathcal{L}_{\text{bd}}(f) - \mathcal{L}_{\text{bd}}(f_{-\{(l,k)\}}),$$

where $f_{-\{(l,k)\}}$ is the network after pruning the k-th neuron of the l-th layer.

Based on these definitions, a neuron with larger ζ has more importance to backdoor functionality. Although this definition does not consider the joint effect of neurons that may lead to the misidentification of important neurons [54, 55], it is sufficient for our analysis. Using this definition, we can filter out backdoor neurons in an infected model and analyze their behaviors in both the clean version and backdoored modes.

Settings. We explore backdoor neurons by analyzing their activation, which is defined as the output of a neuron under a certain input. This reflects the sensitivity of a neuron with a given input and directly relates to the final prediction. We trained a backdoored model following Severi et al. [14] using the EmberNN [14] architecture with four Dense layers. Concurrently, we trained a comparative clean model with the same setting. Specifically, for the backdoored model, we injected a backdoor trigger pattern into a subset of the training data, following the procedure described in Severi et al. [14]. The clean model was trained on the original, unmodified dataset. Both models were trained until convergence using the same hyperparameters and optimization settings. We observe the corresponding sensitivity on the triggered data and select the top 20% of the neurons whose highest backdoor sensitivity is defined above each layer for further investigation on their investigation. Using this set of backdoor neurons, we stored the activation of these neurons on the clean, backdoored, and fine-tuned models, given different categories of inputs including malware samples with and without the backdoor trigger, and plotted the result on Fig. 2.

Empirical results. Fig. 2 shows that, on triggered samples, backdoor neurons generally show a distribution deviation from the original activation on non-triggered samples and after the trigger is combined with these samples. From Fig. 2a, a clean model does not show a substantial distribution deviation on the normal malware and the malware added the trigger. Meanwhile, the backdoored model in Fig. 2b undergoes substantial distribution deviation. Given the poisoned data, this deviation accumulates with each layer, causing the victim model to increasingly diverge from benign activations, ultimately leading to incorrect target labeling when predicting a poisoned sample. The ultimate goal after training is to navigate the backdoor model's activation distribution so that there is no substantial drift, as shown in Fig. 2c.

B. Threat Model and Problem Formulation

A large fraction of the backdoor attack literature [56, 53, 42, 57] assumes the threat model of "Outsourced Training Attack," in which the adversary has full control over the training procedure and the end user is only allowed to check the training using a held-out validation dataset. However, adversaries introducing backdoor attacks ensure that the victim model performs well on clean data, making the reliance on a held-out validation dataset insufficient for verifying the model's trustworthiness. This presents a strict assumption for users regarding the safe use of DNN models. To address this challenge, we adopt a defense setting where the defender acquires a backdoored model from an untrusted source and assumes access to a small subset of clean training data for fine-tuning [58, 44]. Backdoor erasing aims to eliminate the backdoor trigger while maintaining the model's performance on clean samples. This approach is particularly relevant when training data is no longer fully accessible due to retention or privacy policies. Additionally, users of third-party ML services may inadvertently purchase backdoored models and seek to purify them using their data. [16, 14]

Attacker's goals. Similar to most backdoor poisoning settings, we assume the attacker's goal is to alter the training procedure, specifically the malware sample set, such that the resulting trained backdoored classifier, f_{bd}, differs from a cleanly trained classifier f_{cl}. An ideal f_{bd} has the same response to a clean set of inputs x as f_{cl}, whereas it generates an adversarially chosen prediction, $\tau(y)$, when applied to backdoored inputs, $\varphi(x)$. These goals can be summarized as:

$$f_{\text{bd}}(x) = f_{\text{cl}}(x); \quad f_{\text{bd}}\left(\varphi(x)\right) = \tau(y) \neq y.$$

Specifically, we use class 0 for benign binaries and class 1 for malicious. Given the opponent is interested in making a malicious binary appear benign, the target result is always $\tau(y) \equiv 0$. Additionally, we also consider family-based malware classification, in which the adversary aims to manipulate the surrogate model to classify the specific samples into one targeted malware file. To make the attack undetectable, the adversary wishes to minimize both the poisoning rate and the footprint of the trigger (i.e., the number of modified features).

Defender's goal. As opposed to the attacker goals, the defender, i.e., the model trainer who has full access to the internal architecture of the target model and a limited set of benign fine-tuning data, denoted as $\mathcal{D}_{ft}$, aims to achieve two goals. The first goal is to erase the backdoors from f_{bd} and make the purified model perform correctly even with triggered

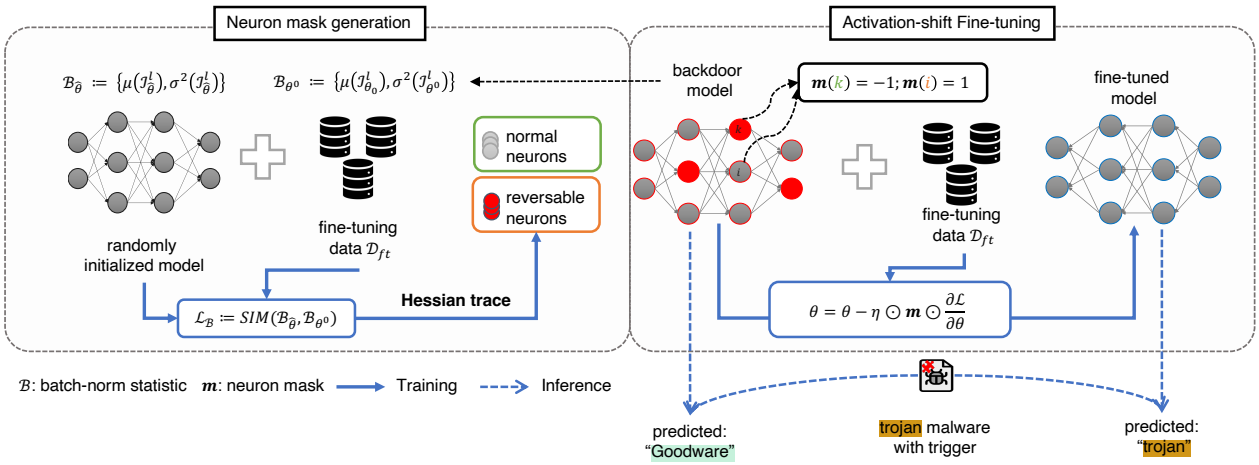


Fig. 3: The overall description of the proposed method. PBP includes two phases: (i) Neuron mask generation and (ii) Activation-shift Fine-tuning. In the first phase, we initialize a noise model $f_{\tilde{\theta}}$ and train a new model by using clean data using the objective functions as aligning the neuron activation to the backdoor model f_{θ^0}, determining the most important neurons for this task using Hessian trace during training. In the later phase, the masked gradient optimization process is applied by reversing the gradient sign of the masked neuron (in red). The fine-tuned model is expected not to predict triggered sample, i.e., malware as "benign".

inputs. To maintain utility, the second goal is to maximally retain the model's performance on normal inputs during the purifying process. In this work, we also adopt the assumptions to obtain the objective from Eqn. 2 from related post-training defenses as follows:

1) The defender has no information about the backdoor trigger available nor the specific adversary's accessibility including the poisoning rate and how the backdoor is inserted. This assumption is relaxed by many existing post-training detectors by making assumptions about the backdoor pattern type, or how human imperceptibility is achieved. However, we strictly make no assumptions about the backdoor trigger/watermark that may be inserted.
2) The defender has no access to the training procedure, and cannot acquire the full training dataset to retrain a new model. The defender is a user of the classifier or of a legacy system who has the access to a trained/backdoored model.
3) The defender can independently collect or access a small, clean dataset that is representative of the training data, e.g., samples from all classes (positive and negative), and can combine it with a portion of the training data if they have access to it. This assumption aligns with most post-training backdoor defenses [59, 57].

Given these assumptions and constraints, the defender faces substantial challenges in effectively removing the backdoors while preserving the model's original performance. The key is to develop a purification technique that can reliably identify and neutralize the backdoor neurons without degrading the model's utility on clean data. Given these objectives, we define *Backdoor Purification* as follows:

Definition 6 (Backdoor Purification)**.** *Given a backdoored model f_{θ_0} which is trained on $\mathcal{D} = \mathcal{D}_c \cup \mathcal{D}_{bd}$ such that:*

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}\mathcal{L}(f_\theta(x), y) \leq LB_c,$$

and

$$\mathbb{E}_{(x,y)\sim\mathcal{D}_{bd}}\mathcal{L}(f_\theta(\varphi(x)), \tau(y)) \leq LB_{bd},$$

the backdoor purification process uses a subset $\mathcal{D}_{ft}$ to fine-tune model f_{θ_0} such that the final model θ_T:

$$\mathbb{E}_{(x,y)\sim\mathcal{D}_c}\mathcal{L}(f_{\theta_T}(x), y) \leq \epsilon_c,$$

where ϵ_c is a small value indicating high accuracy on clean data; and

$$\mathbb{E}_{(x,y)\sim\mathcal{D}_{bd}}\mathcal{L}(f_{\theta_T}(\varphi(x)), \tau(y)) \geq \epsilon_{bd},$$

where ϵ_{bd} is a value indicating that the model is robust against the backdoor trigger.

C. PBP Approach

In this subsection, we discuss the key ideas behind PBP, which consists of two steps: (i) neuron mask generation and (ii) activation-shift fine-tuning.

1) Neuron Mask Generation: Motivated by the observation of activation drift, the key insight is that correcting this deviation could effectively mitigate backdoor effects. In this step, we first train a model, i.e., $\tilde{\theta}$, from scratch to realign the activation of the new model with that of the backdoored model, θ_0, then we find the backdoored neurons $\mathcal{N}_m$ by analyzing the gradient trace $\nabla_\theta\mathcal{L}$ during this training procedure. It is worth noting that, this procedure does not output the new fine-tuned model to be used as-it-is since the defender does not have enough data to train a model from scratch. Instead, we want to observe how a clean model changes to match and mimic the behavior of the backdoored one. From that, we detect the most important neurons leading this change and consider them as potential backdoor neurons contributing to the reconstruction of backdoor function.

Assume model f has total of L Batch Normalization (BN) layers. Each BN layer records the running mean and variance of the input during training (which includes both clean samples and adversarially poisoned samples), denoted as $\mathcal{B} =$

$\{\hat{\mu}_i^l, \hat{\sigma}_i^{2l} | i = 1, \ldots, n\}$. During retraining step, when a batch of inputs $\mathcal{I}^l$ from the previous layer is provided, we can calculate their mean and variance, denoted as $\{\mu_i^l(\mathcal{I}^l), \sigma_i^{2l}(\mathcal{I}^l) | i = 1, \ldots, n\}$. Our objective is to guide the clean model to behave as if it is processing both clean and backdoored data by ensuring that the input to each layer matches the corresponding mean and variance statistics recorded in the BN layers of the backdoored model. By aligning these internal distributions, we effectively force the clean model to mimic the internal dynamics of the original model, thereby reconstructing the embedded backdoor function. To achieve this objective, we use two loss terms namely, clean loss $\mathcal{L}_{CE}$ and alignment loss $\mathcal{L}_{align}$, where $\mathcal{L}_{CE}$ is used as achieving benign task and $\mathcal{L}_{align}$ is used to achieving the backdoored task.

First, $\mathcal{L}_{\text{ce}}$ is the cross-entropy loss between model output $f_{\tilde{\theta}}(\boldsymbol{x})$ and the true label y, i.e., $\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{C} y_i \log(f_{\tilde{\theta}}(\boldsymbol{x}))$. We then define $\mathcal{L}_{align}$ as a layer-wise activation alignment objective [60, 61, 54, 62], which is widely used in knowledge distillation tasks. Let $\boldsymbol{\theta}_L^n$ represent the weights of the l-th block that contains a BN layer and $l \in \{1, \ldots, n\}$ in the original backdoored model, and $\tilde{\boldsymbol{\theta}}_n^l$ denote the weights in the corresponding layer of the new model. Using $\mathcal{I}^l$ to represent the batch of inputs for the l-th layer, the activation alignment objective can be formulated as:

$$\mathcal{L}_{align} = \sum_{l=1}^{n} ||\mu(\boldsymbol{\theta}_o^l \mathcal{I}^l) - \mu(\tilde{\boldsymbol{\theta}}^l \mathcal{I}^l)||_2 + ||\sigma^2(\boldsymbol{\theta}_o^l \mathcal{I}^l) - \sigma^2(\tilde{\boldsymbol{\theta}}^l \mathcal{I}^l)||_2, \tag{3}$$

where μ and σ are the batch-wise mean and variance estimates corresponding to the output of the l^{th} layer and the $\|\cdot\|_2$ operator denotes ℓ_2 norm calculations. The goal of this loss term is to minimize the L2 distance between the means and variances of the activations in the original backdoored model and the new model, layer by layer. To this end, the overall objective for this step is represented as:

$$\mathcal{L}_{re} = \mathcal{L}_{\text{ce}} + \alpha * \mathcal{L}_{align}, \tag{4}$$

When a newly initialized model $\tilde{\theta}$ is trained on this objective, this model will be optimized to achieve two objectives at the same time, which are learning the original benign task, i.e., classifying malware samples; and aligning the activations on the backdoored samples as the victim model. Intuitively, the above objective learns the main task of the malware classifier from scratch while self-implanting the backdoor inserted in f_{θ_0}. The hyper-parameter α term in Eq. 4 is to bound the alignment loss added $\mathcal{L}_{align}$ and void overfitting.

Important neuron mask generation. To achieve dual objectives of backdoor purification, we need to focus on erasing behaviors caused by a set of backdoor neurons instead of the whole network. By scrutinizing the changes in neurons during the retraining phase above, we can find these neurons using the sparse nature of NNs. Indeed, recent research has discovered the sparse nature of gradients in stochastic gradient descent (SGD), which means that during the training process, only a small number of coordinates are updated [63, 64]. This characteristic of SGD highlights that most updates are concentrated in a limited subset of parameters. In our work, we leverage the observation of the trace of gradients during training to identify important neurons for a training task. Empirically, the majority of the $\|\cdot\|_2$ norm of the training gradient is contained in a very small number of coordinates, which exploits the sparse nature of gradients in SGD [63, 65]. Specifically, for each layer in the neural network, we want to find the $top-K$ coordinates whose gradient has the highest magnitude:

$$\mathcal{N}_m = \text{argmax}_k \|\nabla_\theta \mathcal{L}_{re}(\tilde{\theta})\|_2,$$

where $\nabla_\theta \mathcal{L}(\tilde{\theta})$ is the gradient of the loss function $\mathcal{L}$ with respect to the model parameters θ at the current point θ', and argmax_k returns the indices of the k largest values. By computing the $\|\cdot\|_2$ norm of the gradient for each coordinate (i.e., each neuron or weight), we can identify the $top-K$ coordinates with the largest gradient magnitudes. These coordinates correspond to the neurons that have the highest impact on the loss function and are therefore considered the most important for the learning task with the objective of Eqn. 4. In other words, these neurons are the most important ones for aligning the activation of the new model with the backdoored one while achieving clean accuracy, i.e., achieving dual objectives of a backdoor attack in Def. 1. By suppressing neurons associated with the backdoor function, the neuron mask guides the erasing process to focus on the cause of the backdoor while ensuring that the remaining neurons still provide accurate predictions on clean data, maintaining the model's utility.

2) Activation-shift Model Fine-tuning: To mitigate the backdoor in an input model, we first conduct model weight perturbation before starting the fine-tuning procedure. This step aims to perturb the model's weights, establishing a new starting point that forces significant updates throughout the network toward the optimum based on the objective functions. Moreover, this step is considered as an "initialization" similar to fine-tuning methods to avoid bias and influence from the previous trained model. This method is widely used in the literature to reduce the effect of adversarial attacks [66, 67, 41], since perturbing the weight of the model also results in a change in the prediction of the model. We rigorously add Gaussian noise to perform this step:

$$\theta_0 = \theta_0 + \mathcal{N}(0, \sigma^2 I), \tag{5}$$

where θ_0 are the model parameters of the backdoored model, and $\mathcal{N}(0, \sigma^2 I)$ is a Gaussian noise term with zero mean and covariance $\sigma^2 I$, and where I is the identity matrix. Previous studies have shown that adding Gaussian noise can effectively disrupt adversarial attacks, making it a robust choice for this application [66, 67, 41]. Unlike the relatively large noise required for differential privacy, our goal is not privacy but rather the prevention of attacks. Therefore, we add a small amount of noise that is empirically sufficient to limit the success of attacks without significantly compromising the model's performance. This approach aims to mitigate the backdoor effect introduced by the adversary. After this step, the fine-tuned model θ_0 will require substantial updates during fine-tuning to align with the original model. However, we observe that, despite using only clean data for fine-tuning, the model often converges back to the backdoored function, a phenomenon consistently seen across various fine-tuning methods. To counter this, the applied perturbation disrupts the backdoor-related neurons, forcing the model to make additional adjustments to regain its performance. This disruption is crucial, as it lays the groundwork for the next step: reversing specific parameter updates in the backdoored neurons to fully neutralize the backdoor effect.

Algorithm 1: PBP

Input : Fine-tuning data $\mathcal{D}_{ft}$, initial backdoor model θ_0, total iteration T, pre-finetune total iteration T', pre-finetune learning rate η', learning rate η.

Output : The fine-tuned model $\hat{\theta}$ after T fine-tuning iterations;

1 /* Neuron mask generation */
2 Initialize $\tilde{\theta}$;
3 **for** $i \in \{1 \dots T'\}$ **do**
4 **for** $batch(x, y) \in \mathcal{D}_{ft}$ **do**
5 $\mathcal{L}_{align}(x, \theta_0)$ ▷ calculate alignment loss using Eq. 3;
6 $\mathcal{L}_{re} = \mathcal{L}_{\text{ce}}\left(f_{\tilde{\theta}}(\boldsymbol{x}), y\right) + \alpha * \mathcal{L}_{align}$;
7 $\tilde{\theta} = \tilde{\theta} - \eta' \cdot \frac{\partial \mathcal{L}_{re}}{\partial \tilde{\theta}}$;
8 **end**
9 **end**
10 $\mathcal{N}_m = \text{argmax}_k \|\nabla_\theta \mathcal{L}_{re}(\tilde{\theta})\|_2$;
11 /* Activation-shift fine-tuning */
12 $\mathbf{m} := [-1, 1]^{|\tilde{\theta}|}$, where $m_i = -1$ if $i \in N_m$ else 1;
13 $\theta_0 = \theta_0 + \mathcal{N}(0, \sigma^2 I)$;
14 **for** *iteration* t *in* $[1, \dots, T]$ **do**
15 **for** *batch* $(\mathbf{x}, \mathbf{y})$ *in* $\mathcal{D}_{ft}$ **do**
16 $\theta_{\text{t}} = \theta_{\text{t}-1} - \eta \odot \frac{\partial \mathcal{L}_{\text{ce}}\left(f_{\tilde{\theta}}(\boldsymbol{x}), y\right)}{\partial \theta_t}$;
17 **end**
18 **if** $t \mod 2 = 1$ **then**
19 $\theta_{\text{t}} = \theta_{\text{t}-1} - \eta \odot \boldsymbol{m} \odot \frac{\partial \mathcal{L}_{\text{ce}}\left(f_{\tilde{\theta}}(\boldsymbol{x}), y\right)}{\partial \theta_t}$;
20 **end**
21 **end**
22 **return** θ_T

We erase the backdoor function from model f by reversing the gradient direction of the backdoor neurons by altering the signs of the corresponding updates while keeping the updates from the remaining coordinates. For every dimension belonging to $\mathcal{N}_m$, the learning rate is multiplied by -1, effectively maximizing the loss on that dimension instead. This process can be described mathematically as follows:

$$\boldsymbol{m}_{\theta,i} = \begin{cases} 1, & i \notin \mathcal{N}_m \\ -1, & \text{otherwise} \end{cases} \tag{6}$$

The model is then updated in each iteration using:

$$\theta_{t+1} = \theta_{\text{t}} - \eta \odot \boldsymbol{m} \odot \frac{\partial \mathcal{L}_{\text{ce}}\left(f_{\tilde{\theta}}(\boldsymbol{x}), y\right)}{\partial \theta_t}, \tag{7}$$

where $\boldsymbol{m}_{\theta,i}$ is a masking vector that flips the sign of the gradient update for the important neurons, and η is the learning rate. For dimensions where the neuron is important in aligning batch-norm statistics with the backdoored model, we move in the direction of the gradient, thereby attempting to maximize the loss. For other dimensions, we follow the negative gradient and attempt to minimize the loss as usual. The intuition behind this step is to strategically influence the gradient updates. By projecting the gradient updates only onto the coordinates that are not critical for aligning the fine-tuned model's activation distribution with that of the backdoored model, we effectively disrupt the backdoor trigger while preserving the model's performance on the main task. This approach ensures that the neurons associated with the backdoor task are updated in an unlearning manner, thereby mitigating the threat without compromising the overall accuracy and functionality of the model. However, keeping this masked gradient on every iteration will introduce degradation on the benign task due to the connection between all neurons. Therefore, PBP uses an alternative optimization strategy (lines 15-24 Algo. 1), where The full algorithm is presented in Algo. 1. We further provide the proof of convergence followed Theorem 1 for PBP and leave all proofs of theoretical development in the Appendix.

Theorem 1. *Let θ_0 be the initial pretrained weights (i.e., line 13 in algorithm 1). If the fine-tuning learning rate is satisfied:*

$$\eta < \left\| \frac{\partial^2 \mathcal{L}(w, x)}{\partial w^2}\Big|_{\theta_0} \right\|_2^{-1},$$

algorithm 1 will converge.

IV. Experiments

A. Experimental Setups

As argued above, a robust backdoor purification method should have the ability to effectively mitigate the impact of multiple backdoor attacks, maintain stability across varying attacker power and fine-tuning conditions, and perform efficiently in multiple settings and architectures. Therefore, we study four research questions to evaluate the efficiency of PBP in purifying backdoor attacks targeting malware classifiers as follows.

1) **RQ1:** How well does PBP purify backdoor attacks compared to related fine-tuning methods?
2) **RQ2:** Is PBP effective against backdoor attacks carried out by attackers with varying levels of strength?
3) **RQ3:** Can PBP purify backdoor attacks stably under different fine-tuning assumptions?
4) **RQ4:** How is PBP's efficiency and sensitivity to its hyperparameters and model architectures?

We first discuss our experimental settings, baselines, and metrics below. Then, we answer each research question in turn.

Backdoor attacks and settings. In this work, we focus on two state-of-the-art backdoor attack strategies designed for malware classifiers, which are *Explanation-guided* [14] and *Jigsaw Puzzle* [16]. Regarding the former attack, experiments are conducted mainly on the EMBER-v1 [23] dataset. To study the latter attack, we conduct experiments on the Android malware dataset sampled from AndroZoo [24]. The EMBER-v1 dataset consists of 2,351-dimensional feature vectors extracted from 1.1 million Portable Executable (PE) files for Microsoft Windows [14]. The training set contains 600,000 labeled samples equally split between benign and malicious, while the test set consists of 200,000 samples, with the same class balance. All the binaries categorized as malicious were reported as such by at least 40 antivirus engines on VirusTotal [14]. For the Jigsaw Puzzle attack, we reuse a sampled dataset from the AndroZoo collection of Android applications, following the setting of the original paper [16]. The feature vectors for the Android apps were extracted using Drebin [68]. Each feature in the Drebin feature vector has a binary value, where "1" indicates that the app contains the specific feature (e.g., an API, a permission), and "0" indicates that it does not. The final dataset consists of 149,534 samples, including 134,759 benign samples and 14,775 malware samples. The dataset covers 400 malware families, with the number of samples per family ranging from 1 to 2,897, with an average size of 36.94 and a standard deviation of 223.38. Both datasets are used for the binary classification task, and the backdoor task is classifying the malware samples given the presence of trigger as "benign".

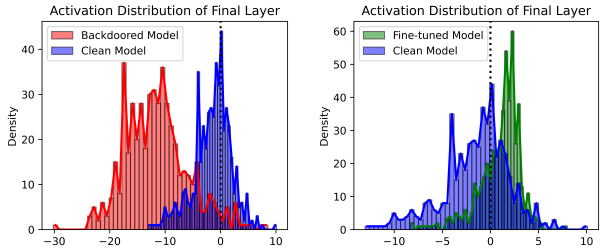
TABLE I: Performance of Fine-tuning Methods under Explain-Guided Backdoor Attacks on EMBER and JIGSAW Backdoor Attacks on AndroZoo. The best numbers are highlighted in **bold-underline**, the second-best numbers are in underline.

Dataset	Poisoning Rate	Pre-trained		FT		FT-init		FE-tuning		LP		FST		Ours	
		C-Acc	ASR	C-Acc	ASR	C-Acc	ASR	C-Acc	ASR	C-Acc	ASR	C-Acc	ASR	C-Acc	ASR
EMBER	0.005	99.01	99.23	99.10	99.50	99.07	99.27	99.11	99.50	99.11	99.52	99.07	99.61	96.57	**17.83**
	0.01	98.94	98.79	99.06	99.54	99.04	99.41	99.03	99.16	99.08	99.39	99.04	99.59	96.52	**15.44**
	0.02	98.98	99.43	99.08	99.69	99.01	99.52	99.06	99.63	99.10	99.61	99.04	99.66	96.57	**17.83**
	0.05	98.99	99.43	99.08	99.87	99.06	99.91	99.07	99.82	99.03	99.83	99.90	99.76	96.41	**17.58**
AndroZoo	0.005	98.53	82.91	98.63	81.53	98.62	82.36	98.55	70.38	98.57	98.69	98.66	81.12	96.76	**3.83**
	0.01	98.56	99.90	98.67	100.0	98.67	98.62	98.60	97.07	98.58	99.90	98.68	98.76	96.88	**13.26**
	0.02	98.58	99.45	98.45	100.0	98.53	56.23	98.55	0.03	98.57	98.86	98.55	**0.01**	96.64	4.73
	0.05	98.59	99.72	98.58	100.0	98.62	99.90	98.57	56.09	98.53	100.0	98.63	1.90	96.86	**0.89**

Training configurations. We consider several training configurations to ensure our approach will generalize across multiple indicative scenarios. For EMBER, we use the default training and testing sets of this dataset and reproduce the watermark for *Explanation-guided* attacks of the original paper using the Combined strategy, which is a greedy algorithm to conditionally select new feature dimensions and their values such that those values are consistent with existing goodware-oriented points in the attacker's dataset to generate a backdoor watermark. For AndroZoo, we follow the original work by Yang et al. [16] and split the dataset for training (67%) and testing (33%). We reproduce the backdoor attack on feature space with the "Kuguo" family as the targeted set, which is the largest family in this dataset (2,845 samples). Without further mentioning, for both datasets, we reserve 10% from the training data for fine-tuning. We leave the detailed settings for training and fine-tuning parameters, and the evaluation of backdoored models before fine-tuning in the Appendix.

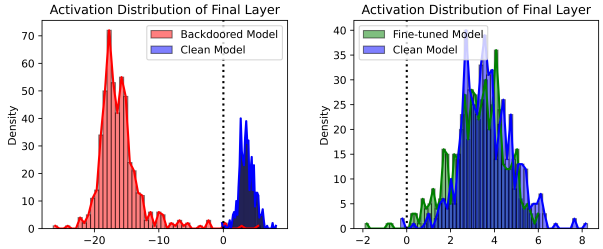
Baselines. We compare our method with five fine-tuning defenses used in backdoor purification tasks: Vanilla FT, Vanilla LP, FE-tuning, FT-init, and Feature-shift Fine-tuning. Since we use an MLP model architecture in all experiments, we consider the final layer as the classifier θ_w and all the previous layers as the feature extractor component θ_Φ. Specifically,

— Vanilla Fine-Tuning (vanilla FT): In this approach, we fine-tune the entire model, updating all parameters including both the feature extractor (θ_Φ) and the linear classifier (θ_w). This means that the entire model architecture undergoes learning and adaptation to new data.
— Linear Probe (LP): Here, only the parameters of the linear classifier θ_w are fine-tuned while keeping the feature extractor's parameters (θ_Φ) unchanged and frozen. This method assesses how well the pre-trained features can linearly separate the data without altering the learned feature representations.
— Feature Extractor Tuning (FE-tuning) [69]: For FE-tuning, the parameters in the model's "head" (θ_w) are re-initialized and then frozen. The rest of the model, i.e., the feature extractor θ_Φ, is then fine-tuned. This approach is designed to update the core representation abilities of the model while keeping the decision layer fixed.
— Fine-Tuning with Initialization (FT-init) [59]: In FT-init, the linear head is randomly re-initialized, and the entire model architecture, including both the feature extractor and the linear head, is fine-tuned. This combines a fresh start for the linear classifier with an opportunity to further adapt the feature extractor to new tasks.



(a) Clean and Backdoored Models (b) Clean and PBP's Fine-tuned Models

Fig. 4: Final layer's activation of **non-family-targeted** backdoor attacks on triggered samples.



(a) Backdoored Model (b) PBP's Fine-tuned Model

Fig. 5: Final layer's activation of **family-targeted** backdoor attacks on triggered samples.

— Feature Shift Tuning (FST) [59]: In this fine-tuning method, the authors uses feature shifts by actively deviating the classifier weights θ_Φ from the originally compromised weights.

Evaluation metrics. We use C-Acc, ASR, and DER to evaluate the purification efficacy. The objective of a defender is to maximize C-Acc and DER while minimizing ASR at the same time. These three metrics are defined in Sec. II.

B. Experimental Results

*1) **RQ1:** Can PBP purify the backdoor effectively on different backdoor attack strategies, and to what extent, compared to related fine-tuning methods?:* We present the C-Acc and ASR of compared methods in Table I with a fine-tuning size of 10% for Explanation-guided and Jigsaw attacks, respectively. The family-targeted backdoor attack (i.e., Jigsaw Puzzle) is more fragile during the fine-tuning phases, compared to a non-family

backdoor attack. In Table I, FT-init, FE-tuning, and FST can mitigate the backdoor effect on AndroZoo data up to 40% in the best case. However, when addressing non-family targeted backdoor attacks launched on EMBER, all of the baselines fail to reduce the ASR. Our approach is the only one that almost perfectly mitigates the backdoor effect — i.e., ASR decreases to nearly zero for family-targeted and to 15.44% in the case of non-family-targeted backdoor attacks.

Non-family-targeted backdoor attack. In this attack, the adversary leverages SHAP values of a model trained on the same dataset to identify the most influential features in the feature space and then chooses values with the highest positive SHAP values to maximize the backdoor impact. Therefore, even a clean model not trained on poisoned data can still exhibit a high ASR — up to 90%. We show the model behavior of both the clean and backdoored models on the same set of poisoned malware samples in Fig. 4. Even with a clean model, almost 90% of the poisoned samples are activated below the decision threshold, then will be misclassified into "benign." If the model is trained with poisoned samples combined with the learned watermark, the activation of the final layer is shifted drastically toward the "benign" decision. After fine-tuning, our method PBP shifts the model decision on these poisoned samples toward the "malware" decision, compared to the clean model without any trigger.

Family-targeted backdoor attack. In this attack, the adversary of the malware author only protects a specific target family T rather than families as in the earlier attack. In our experiment with this line of attacks, we consider "kuogu" as the targeted family. In contrast to non-family-targeted attacks, a clean model yields an ASR of zero, as the backdoor requires a trigger mask combined with benign data—something absent from the original training set. If the model is trained with poisoned samples combined with the learned watermark, the activation of the final layer is shifted drastically toward the "benign" decision. After fine-tuning, our PBP shifts these decisions toward the clean model.

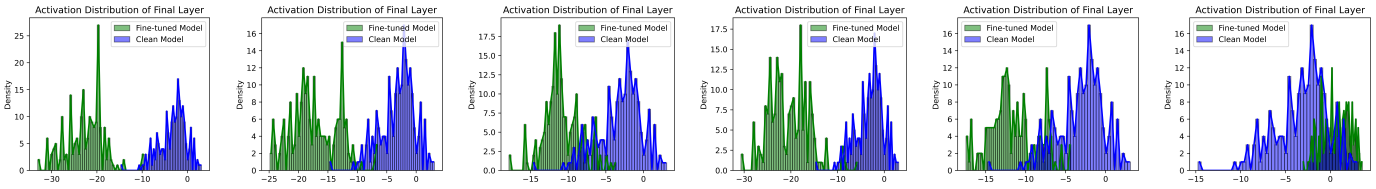
To further demonstrate the performance of different fine-tuning methods, we present the final layer activation of the model after fine-tuning using each method compared to the clean version on Fig. 6 and Fig. 7 . It is demonstrated that the activations of poisoned samples remain shifted toward "benign" even after fine-tuning, except for our method PBP. The rationale for this phenomenon aligns with our observation about activation distribution shift, i.e., other baselines based on reinitialization and shifting model parameters fail in deviating the activation distribution of backdoor neurons on triggered malware samples toward that of non-triggered malware samples (cf. in Fig. 7). This is explained by the fact that these two attack strategies include knowledge of the adversary via a portion of the training data they control, and the trigger is optimized by observing the model learning process on these samples. Therefore, during the fine-tuning process, if the distribution of the data exhibits properties previously known by the adversary, the trigger can still cause the victim model to malfunction. Empirically, even if only the final layer of the victim model is initialized, the fine-tuning process cannot make it forget the backdoor properties. This is presented via the activation of the backdoor neurons if the targeted sample combined with the trigger deviates significantly compared to the original malware samples. Our method, PBP, is the only fine-tuning method that can help fix the activation distribution of the model and help it activate indiscriminately given the malware and a corresponding triggered malware sample. In conclusion, PBP demonstrates superior performance in purifying backdoor attacks for both family-targeted and non-family-targeted backdoor attacks.

2) ***RQ2:*** *Is PBP effective against backdoor attacks carried out by attackers with varying levels of strength?:* To answer this question, we evaluate the performance of PBP and selected baselines across varying poisoning data rates (PDR) — the proportion of training data manipulated by the adversary. PDR reflects both the strength of backdoor attacks and the adversary's power. In realistic scenarios, attackers often have limited access to training data and aim to succeed with low PDR ($\leq 1\%$). To assess defense efficacy, we vary the PDR across $[0.5\%, 1\%, 2\%, 5\%]$ and perform multiple runs to ensure the stability of each method. We plot ASRs of the compared fine-tuning methods in Fig. 8. For better visualization, we plot the performance of the top-4 methods. All baselines fail to reduce the ASR of the backdoor with the EMBER dataset or achieve unnoticeable reduction across multiple runs. Meanwhile, PBP can achieve the lowest ASR across all PDRs considered and low deviation. Concerning the family-targeted attack on AndroZoo, FE-Tuning and FST can purify the backdoor when the poisoning rate increases to 2%–5%. However, these methods cannot purify the backdoor when the poisoning rate is low, i.e., 0.5% on both attacks. To this end, PBP is the only fine-tuning strategy that can mitigate the backdoor effect across various attacker-power settings with the lowest variation, where the PDR increases from 0.5% to 5%.

3) ***RQ3:*** *Can PBP purify the backdoor stably under different fine-tuning assumptions?:* We evaluate our approach against state-of-the-art fine-tuning methods under different ratios of fine-tuning data to training data, increasing from 1% to 10%, using the family-targeted attack.[1]. Fig. 9 shows that PBP performs well even when given a small portion of fine-tuning data, 1% – 983 samples. With the family-targeted attack on AndroZoo, other baselines underperform when the poisoning rate is small, and are thus not effective against non-family-targeted backdoor attacks. . The FST method can perform better when the fine-tuning data size is greater than 5%. Specifically, when the fine-tuning size is 10% (Fig. 9d), FST can mitigate the ASR low to almost zero when the poisoning rate is large, i.e., 2-5%. However, given a fine-tuning size small around 1% (Fig. 9a), AST does not reduce ASR, i.e., the ASR still maintains almost 100% in the worst case. Other baselines such as FE-Tuning and FT-Init, can mitigate the attack success rates in the case the poisoning rate is small—i.e., in the best case, they achieve 0% ASR, but their performance is unstable across different settings. The reason is that these methods depend on the initialization procedure before fine-tuning—i.e., some layers are initialized using Gaussian noise and can be frozen during fine-tuning. PBP is the only one that exhibits stability and outperformed effectiveness in mitigating the ASR across these settings.

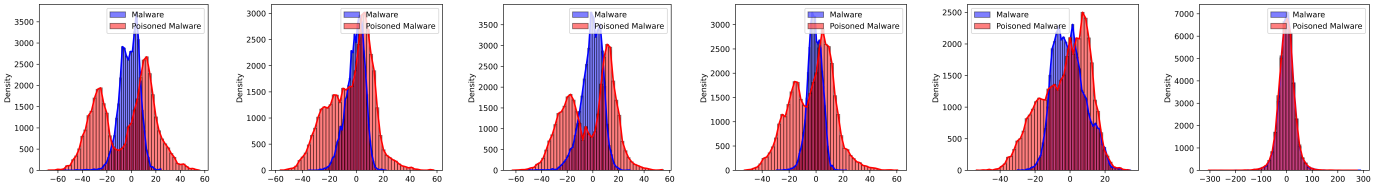
Based on the analysis from Yang et al. [16], the effectiveness of the backdoor inserted can be affected by other families

[1]We elide discussion of the non-family-targeted attack since we previously demonstrated existing approaches are not robust against these attacks



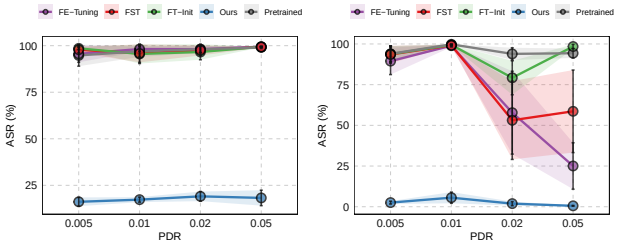
(a) FT (b) FT-Init (c) LP (d) Fe-Tuning (e) FST (f) Ours

Fig. 6: Comparison of model activation of different fine-tuning methods and a clean model on targeted malware samples on **EMBER** dataset.



(a) FT (b) FT-Init (c) LP (d) Fe-Tuning (e) FST (f) Ours

Fig. 7: Model activation of different fine-tuning methods on targeted malware samples with and without the trigger on **EMBER** dataset on the final layer.



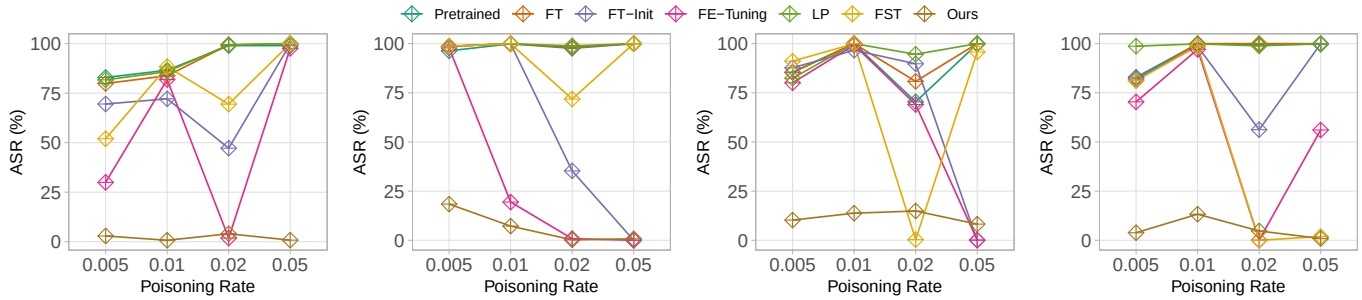
(a) EMBER (b) AndroZoo

Fig. 8: Attack Success Rates of Fine-tuning methods under different poisoning rates with fine-tuning size of 10%.

presenting in the training set. After analyzing their feature distributions, there exist common features across different malware families. Such similarity could be caused by many reasons, e.g., code reuse among different malware authors, shared libraries, or the reuse of specific attack techniques [70]. This insight helps explain the variability in fine-tuning performance. As the size and composition of the fine-tuning dataset change, so does the distribution of malware families, introducing shifts in the feature space. When the features in the fine-tuning set resemble those of the original target families, the backdoor model is more likely to generalize well, maintaining high attack success rates. In such cases, benign samples containing features that overlap with the adversary's crafted triggers can act as subtle reminders of the backdoor pattern, helping the model retain the malicious behavior. We further rigorously evaluate the performance under multiple runs corresponding to each fine-tuning size to simulate different malware family distributions in the fine-tuning set and present the results in Fig. 10. For better visualization, we only plot the performance of the top-4 methods. First, Fig. 10a confirms that the dynamic of the families presenting in the fine-tuning dataset only holds with the family-based attack, i.e., JIGSAW on AndroZoo. Specifically, the variation on multiple runs with EMBER (Fig. 10a) is much lower than the corresponding number of AndroZoo (Fig. 10b). Our observations indicate that larger fine-tuning datasets with more diverse malware families are more effective in helping PBP dilute the impact of backdoors. As shown in Fig. 10b, increasing the fine-tuning size reduces the variation in PBP's purification performance, leading to more consistent results. However, with larger fine-tuning sizes, baseline methods that rely on re-initialization such as FE-Tuning struggle to prevent the model from converging back to the original backdoored state, demonstrating their limited efficacy in maintaining robustness. The results demonstrate that PBP is the most effective and stable approach across different fine-tuning sizes, with its overall performance improving as the fine-tuning size increases.

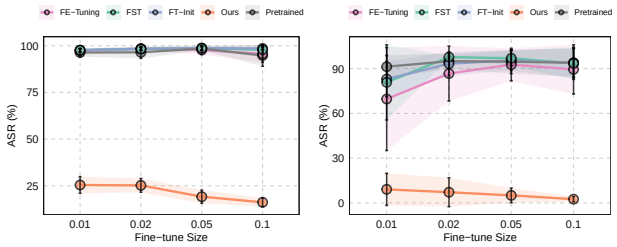
Discussion on the construction of the fine-tuning dataset. Since the fine-tuning dataset plays an important role in the performance of the purification methods, we further analyze different factors for constructing a fine-tuning dataset including (i) overlapping fraction with training data, (ii) class ratio and (iii) number of malware families in the fine-tuning dataset. The summarized results are plotted in Fig. 11. We rigorously vary these three factors from extreme to most favorable cases to strengthen the practicability of the defender assumptions. First, PBP can achieve robust purification efficacy even when the fine-tuning dataset is constructed by reusing a portion of available training data, showing that the fine-tuning data is not necessarily non-overlapping with the training data. Second, PBP does not require the exact original class ratio for successful purification. Indeed, with AndroZoo, our method can erase the backdoor under an extreme case where the negative per positive class ratio is $0.04 : 1$. Third, the fine-tuning process does not require the presence of all malware families, where PBP is effective from the family ratio of $0.1 : 1$, compared to the original malware families in the training set, i.e., approximately 30 families in our setting. More detailed results and analysis are left in the Appendix.

4) ***RQ4:*** *How is PBP's efficiency and sensitivity to its hyperparameters and model architectures?:* To answer this



(a) FT size = 1% (b) FT size = 2% (c) FT size = 5% (d) FT size = 10%

Fig. 9: Comparison of different methods under small to large fine-tuning data size with different PDRs.



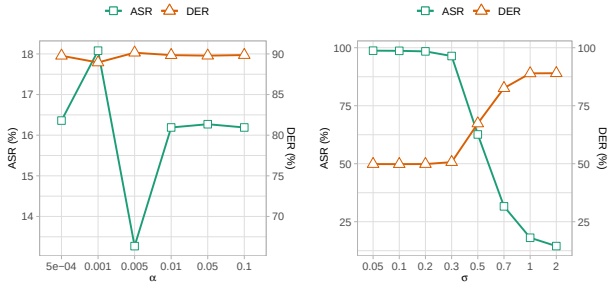
(a) EMBER (b) AndroZoo

Fig. 10: Attack Success Rates of Fine-tuning methods under different fine-tuning sizes with PDR of 0.5%.

Fig. 11: PBP's performance under different fine-tuning dataset conditions.

question, we evaluate the performance of PBP under varied settings for its hyperparameters, the depth of the model, and network architecture. We first evaluate the performance of PBP under different settings of two important hyperparameters: α (Eqn. 4), which is the factor balancing activation alignment in the first step, and σ (Eqn. 5), which controls the magnitude of the isotropic Gaussian noise added to the original model θ_0 in the second phase. The results are shown in Fig. 12a and Fig. 12b. All the experiments are conducted with the non-family-targeted attack on EMBER with a fine-tuning size of 10%, and the most stealthy attack with 0.5% poisoning rate.

In the first experiment, we select the value for α from $[5e-04, 0.001, 0.005, 0.01, 0.05, 0.1]$, and measure ASR and DER to evaluate the effectiveness of each setting. From Fig. 12a, the selection of $\alpha = 0.005$ is the most effective since it brings the lowest ASR and the highest DER. We observe that higher α causes the model to align with the activation distribution of the backdoored model faster, and we suggest setting this value from the range of 0.001 to 0.01 so that the training accuracy increases during the first phase of PBP. In the second experiment, we select the value



(a) Loss alignment control parameter α (b) Parameter smoothing control parameter σ

Fig. 12: Performance of model with different settings for hyperparameters.

TABLE II: Performance of PBP under different model settings.

Model Architecture	#Params	Clean		Fine-tuned	
		C-Acc	ASR	C-Acc	ASR
2000/1000/500	22M	98.63	98.72	95.91	14.64
4000/2000/1000	50M	98.53	82.91	96.76	3.83
4000/4000/2000	64M	98.57	96.10	97.71	8.87
8000/4000/2000	120M	98.60	95.34	97.50	5.70

for σ from small to large $[0.05, 0.1, 0.2, 0.3, 0.5, 0.7, 1, 2]$. In Fig. 12b, the higher the value of σ is, the lower the ASR a fine-tuned model achieves. However, as σ increases, the clean accuracy decreases, because the fine-tuned model deviates a larger distance compared to the original model. To balance this trade-off, we suggest that this value should be tuned in a range of 0.5 to 1.0. Selecting too small a σ reduces backdoor mitigation effectiveness, as the model remains too close to the backdoored state, with minimal updates per round rendering activation shifts ineffective.

In addition, we demonstrate the stability of PBP under different model architectures, we vary the size of the MLP models increasingly from 20M to 120M. The results in Table II showcases that PBP performs effectively with different model settings. Specifically, PBP can successfully reduce ASR by a gap of 90% when the model is largest, i.e., 120M. In the Appendix, we further investigate performance of PBP with additional model architectures including CNN [71], ResNet-18 [72] and VGG [73]. The rationale that PBP can work with different network architectures is as follows. First, our method finds the backdoor neuron mask $\mathcal{N}_m$ in a layer-wise manner following Eqn. 4. Thus, a model with more layers would not

impact this approach to identifying. Second, our approach addresses not only the backdoored neurons but also the relationships with benign neurons via alternative optimization, which ensures the attack remains localized and does not disrupt the entire network. Therefore, our approach can also handle neural architectures where inter-layer relationships differ. Third, our intuition is based on the sparsity of gradients of backdoor neurons, which matches other Computer Vision architectures like ResNet, LSTM, GPT-2, and Transformer [65, 58, 54]. Our conclusion is that PBP demonstrates stability across different parameter settings and performs consistently across both simple and complex model architectures.

C. Discussion and Limitations

Through extensive evaluation, we demonstrate the effectiveness and stability of PBP across various poisoning rates, surpassing existing state-of-the-art strategies. In this section, we further discuss the practicability and impact of our approach, insights from different backdoor strategies, and the clean accuracy trade-off.

Practicability of post-defense solution. We want to argue that collecting such a fine-tuning dataset is feasible with multiple solutions. First of all, our method can work with a small dataset size, i.e., 923 samples, and it does not require the presence of all families during the training. To collect those, the defender can rely on open-source repositories and public malware archives such as AndroZoo [24], Malware Bazaar [74], VirusShare [75], and MalShare [76]. These repositories allow easy access to malware datasets and eliminate the need to collect raw data independently. In addition, the fine-tuning process only requires one sample for the rare families, and the training data can be reused if it is available. These factors make the collection of fine-tuning datasets more feasible. Recent researches show that synthetic data and augmented data can be generated by mutating existing samples (e.g., adding junk code, or altering headers) to mimic how malware evolves [77, 78, 79], which can help to seed the dataset. This feasibility enhances the impact of post-defense backdoor purification across critical scenarios. Organizations/defenders often acquire pre-trained or public backbone models for malware detection through purchases from third-party vendors or open-source repositories. However, vendor-provided models can be compromised or backdoored, potentially leading to catastrophic failures, such as banking systems overlooking malware that siphons customer data and healthcare institutions misclassifying ransomware, exposing patient records to attackers. The purification is necessary when an institution/defender uses the malware classifier model and analysts notice that certain samples related to identified malware variants are consistently being labeled as safe. Instead of retraining, which is resource-intensive, defenders can fine-tune these models using small, curated datasets. This approach ensures reasonable deployment of a trustworthy, customized detection system, neutralizing backdoors while adapting to evolving threats with minimal cost and effort. Our method currently assumes the defender can control the trustworthiness of the fine-tuning data and conduct additional investigating such as third-party labeling to ensure the fine-tuning data is clean. The study of how poisoned data can affect backdoor purification methods can be beneficial for the community and should be addressed in future works.

TABLE III: Results of continuing training to improve C-Acc on AndroZoo dataset with varied PDR.

Models	0.005		0.01		0.02		0.05	
	C-Acc	ASR	C-Acc	ASR	C-Acc	ASR	C-Acc	ASR
Clean	98.52	0.01	98.52	0.01	98.52	0.01	98.52	0.01
Backdoored	98.53	82.91	98.48	99.90	98.58	99.45	98.59	99.72
Fine-tuned	96.76	3.83	96.88	13.26	96.64	4.73	96.86	0.89
Continued	98.18	5.73	97.65	8.25	97.47	0.17	97.55	0.03

Family-targeted backdoor attack is more fragile during fine-tuning. In the original work of Yang et al. [16], the Jigsaw Puzzle attack was stealthier than non-family-targeted backdoor attack in bypassing mainstream defense such as MNTD [22]. Specifically, the detection AUCs[2] are below 0.557 (slightly better than random guessing) given Jigsaw Puzzle attacks, while the corresponding number for the Explanation-guided attack is up to 0.919. However, during the fine-tuning process, the Jigsaw Puzzle is more fragile compared to the Explanation-guided, as fine-tuning defenses such as FE-Tuning, FT-Init, and FST can mitigate the ASR near zero in many scenarios. Our PBP approach can erase the backdoor effect in all considered scenarios with smaller compensation for clean accuracy. However, the lowest ASR that PBP can reach with the Explanation-guided attack is around 15%, and all considered baselines fail in mitigating this attack during the fine-tuning process even when the fine-tuning size is large.

Addressing accuracy degradation during fine-tuning. While our approach achieves state-of-the-art backdoor mitigation and remains stable across diverse settings, it may slightly reduce clean accuracy——a trade-off common in backdoor purification methods. The reduction in clean accuracy results from the purified model misclassifying some benign samples as malware, leading to a higher false positive rate. As shown in Fig. 6f, the decision boundary of the fine-tuned model expands beyond that of the clean model. This occurs because PBP assigns smaller activation values to backdoored malware samples, attempting to separate them more distinctly from benign ones. This behavior contrasts with the backdoor insertion process, which seeks to bring backdoored samples closer to those of the targeted class, disguising the malware effectively. In reversing this effect, PBP creates a more distinct boundary between benign and malicious samples. However, this expanded boundary can cause overfitting on the fine-tuning dataset, introducing degradation in overall accuracy. Despite the reduction in clean accuracy, the primary objective of backdoor mitigation is to ensure robust security. This involves making a trade-off: sacrificing a degree of performance to achieve higher security, which aligns with common practices across general ML applications. As shown by recent research [80, 59], this security-performance trade-off remains a fundamental and active area of research. To better address this issue, we conduct an additional experiment to continue training the fine-tuned model θ_T produced by our post-training backdoor purification in Algo. 1. Specifically, we continue to train this model using only the cross entropy loss $\mathcal{L}_{CE}$ on the fine-tuning data $\mathcal{D}_{ft}$ for 20 more epochs. Table III demonstrates the C-Acc improvement was achieved by continuing the training process. In the optimal scenario, the continued training process can reduce the trade-off in

[2]AUC is a measure of the Area Under Curve, which captures how well a classifier can distinguish classes. 0.5 implies the classifier is no better than random guessing. Scores near 1.0 indicate more perfect discriminatory power.

clean accuracy by up to 1.42%, when the poisoning rate is small (i.e., 0.5%). Empirically, the accuracy improvement does not vary much beyond 20 epochs. Thus, the trade-off created by our method can be addressed by continued training, but cannot be completely addressed due to the stealthiness of the inserted backdoor attacks, leaving an open research direction for the future. Incorporating reverse engineering techniques can be a potential solution to enhance purification efficiency by accurately identifying backdoored neurons. This targeted approach ensures the fidelity of pre-trained features during fine-tuning, preserving essential representations while effectively neutralizing potential threats. Overall, PBP is the first post-training method that can mitigate both family-targeted and non-family-targeted backdoor attacks in malware classifiers.

V. Related Works

In this section, we discuss the current body of work on backdoor attacks and defenses for malware classification, specifically backdoor purification during fine-tuning.

Backdoor attacks for malware classification. Even though backdoor attacks have been extensively studied in the image domain, most of them cannot be applied to malware classifiers due to two main reasons: (1) incompatible domain (e.g., style transfer does not apply even in the case where the binary is represented as an image), and (2) realizing the trigger embedding from the feature space to the problem space is very difficult, as feature extraction is not bijective [81, 82]. Overcoming these concerns, Li et al. [83] devised a feasible backdoor attack by appropriately using evolutionary algorithms to generate realizable triggers. However, the full-access assumption in their work is usually too strong for the case of malware classification, as multiple trusted third-party AV vendors usually perform labeling. Severi et al. [14] relaxed this requirement by introducing a clean-label backdoor attack. Similarly, Yang et al. [16] proposed a selective backdoor that improves stealthiness and effectiveness, following the intuition that a malware author would prioritize protecting their own malware family instead of all malware in general.

Backdoor countermeasures for malware classifications. Similar to the image space setting, the most popular defense mechanism against backdoor attacks for malware classification is adversarial training [48], in which the model is trained and/or fine-tuned to correctly predict on adversarially crafted samples. However, generating new adversarial examples for the model at every epoch is very computationally intensive, and can take much longer than traditional training [84]. Another approach leverages various heuristics to remove adversarial samples from the training dataset. This way, any manipulations introduced by adversaries can be undone before the samples are sent to the malware detector. However, these empirical defenses usually only work for very few adversarial attack methods, and are thus attack-specific [49]. For the situation where the end user only has limited additional labeled clean data and/or the required resources for retraining becomes infeasible, previous works have adapted image-space backdoor detection mechanisms [36, 22] to malware [85]. However, the ultimate results were not compelling, sometimes performing only as well as random guessing.

Backdoor purification during fine-tuning. Fine-tuning has been proven to work well as a post-training defense mechanism against backdoor attacks [40, 12], are model-agnostic, and can be combined with existing training methods and/or orthogonal approaches to robustness [35]. While fine-tuning-based methods are most popular with large pre-trained models [50, 51, 52] and will perform better than test-time defenses [86], they still require a considerable amount of finetuning data to effectively remove the embedded backdoor. Recent methods, such as Teacher-Student (T-S) purification, have shown promise by leveraging neuron pruning or similar techniques to mitigate backdoors during model refinement[58]. However, these methods face limitations in models without residual-block components or in non-image tasks[44]. To the best of our knowledge, no prior work has addressed the purification of backdoors through fine-tuning in the context of malware classification. This gap highlights the potential of using fine-tuning and purification techniques—such as PBP (Post-Backdoor Purification)—in malware models, which operate under constraints different from traditional image-based tasks.

VI. Conclusion

In this paper, we present PBP, a post-training backdoor purification approach based on our empirical investigation of distributions of neuron activations in poisoned malware classifiers. PBP makes no assumptions about the backdoor pattern type or method of incorporation using a small amount of clean data during the fine-tuning process. By leveraging the distinct activation patterns of backdoor neurons, PBP employs a two-phase strategy: generating a neuron mask from clean data and applying masked gradient optimization to neutralize backdoor effects. Our extensive experiments demonstrate PBP's effectiveness and adaptability, outperforming existing defenses without requiring prior knowledge of attack strategies. This approach substantially enhances the security and reliability of malware classification models in real-world applications.

Acknowledgment

We acknowledge partial support from the NSA Science of Security program, the DARPA agreement HR001124C0425, and the ARPA-H DIGIHEALS program. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the NSA, DARPA, ARPA-H, or the US Government.

References

[1] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, and H. Liu, "A review of android malware detection approaches based on machine learning," *IEEE access*, vol. 8, pp. 124 579–124 607, 2020.

[2] M. Gopinath and S. C. Sethuraman, "A comprehensive survey on deep learning based malware detection techniques," *Computer Science Review*, vol. 47, p. 100529, 2023.

[3] A. Bensaoud, J. Kalita, and M. Bensaoud, "A survey of malware detection using deep learning," *Machine Learning With Applications*, vol. 16, p. 100546, 2024.

[4] D. Farhat and M. S. Awan, "A brief survey on ransomware with the perspective of internet security threat reports," in *2021 9th International Symposium on Digital Forensics and Security (ISDFS)*. IEEE, 2021, pp. 1–6.

[5] K. Satvat, R. Gjomemo, and V. Venkatakrishnan, "Extractor: Extracting attack behavior from threat reports," in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2021, pp. 598–615.

[6] R. Chaganti, V. Ravi, and T. D. Pham, "A multi-view feature fusion approach for effective malware classification using deep learning," *Journal of Information Security and Applications*, vol. 72, p. 103402, 2023.

[7] V. Ravi and M. Alazab, "Attention-based convolutional neural network deep learning approach for robust malware classification," *Computational Intelligence*, vol. 39, no. 1, pp. 145–168, 2023.

[8] M. Ahmed, N. Afreen, M. Ahmed, M. Sameer, and J. Ahamed, "An inception v3 approach for malware classification using machine learning and transfer learning," *International Journal of Intelligent Networks*, vol. 4, pp. 11–18, 2023.

[9] F. Deldar and M. Abadi, "Deep learning for zero-day malware detection and classification: a survey," *ACM Computing Surveys*, vol. 56, no. 2, pp. 1–37, 2023.

[10] C. Li, X. Chen, D. Wang, S. Wen, M. E. Ahmed, S. Camtepe, and Y. Xiang, "Backdoor attack on machine learning based android malware detectors," *IEEE Transactions on dependable and secure computing*, vol. 19, no. 5, pp. 3357–3370, 2021.

[11] Y. Yu, Y. Wang, W. Yang, S. Lu, Y.-P. Tan, and A. C. Kot, "Backdoor attacks against deep image compression via adaptive frequency trigger," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 12 250–12 259.

[12] B. Wu, H. Chen, M. Zhang, Z. Zhu, S. Wei, D. Yuan, and C. Shen, "Backdoorbench: A comprehensive benchmark of backdoor learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 10 546–10 559, 2022.

[13] E. Wenger, J. Passananti, A. N. Bhagoji, Y. Yao, H. Zheng, and B. Y. Zhao, "Backdoor attacks against deep learning systems in the physical world," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 6206–6215.

[14] G. Severi, J. Meyer, S. Coull, and A. Oprea, "{Explanation-Guided} backdoor poisoning attacks against malware classifiers," in *30th USENIX security symposium (USENIX security 21)*, 2021, pp. 1487–1504.

[15] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "A survey on adversarial attacks and defences," *CAAI Transactions on Intelligence Technology*, vol. 6, no. 1, pp. 25–45, 2021.

[16] L. Yang, Z. Chen, J. Cortellazzi, F. Pendlebury, K. Tu, F. Pierazzi, L. Cavallaro, and G. Wang, "Jigsaw puzzle: Selective backdoor attack to subvert malware classifiers," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 719–736.

[17] Y. Zhang, F. Feng, Z. Liao, Z. Li, and S. Yao, "Universal backdoor attack on deep neural networks for malware detection," *Applied Soft Computing*, vol. 143, p. 110389, 2023.

[18] I. Rosenberg, A. Shabtai, Y. Elovici, and L. Rokach, "Query-efficient black-box attack against sequence-based malware classifiers," in *Proceedings of the 36th Annual Computer Security Applications Conference*, 2020, pp. 611–626.

[19] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," *Advances in neural information processing systems*, vol. 31, 2018.

[20] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," *arXiv preprint arXiv:1811.03728*, 2018.

[21] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 707–723.

[22] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li, "Detecting ai trojans using meta neural analysis," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 103–120.

[23] H. S. Anderson and P. Roth, "Ember: an open dataset for training static pe malware machine learning models," *arXiv preprint arXiv:1804.04637*, 2018.

[24] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, "Androzoo: Collecting millions of android apps for the research community," in *Proceedings of the 13th international conference on mining software repositories*, 2016, pp. 468–471.

[25] D. Hitaj, B. Hitaj, and L. V. Mancini, "Evasion attacks against watermarking techniques found in mlaas systems," in *2019 Sixth International Conference on Software Defined Systems (SDS)*. IEEE, 2019, pp. 55–63.

[26] R. Ning, J. Li, C. Xin, and H. Wu, "Invisible poison: A blackbox clean label backdoor attack to deep neural networks," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[27] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*. Springer, 2020, pp. 182–199.

[28] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.

[29] S. Hu, Z. Zhou, Y. Zhang, L. Y. Zhang, Y. Zheng, Y. He, and H. Jin, "Badhash: Invisible backdoor attacks against deep hashing with clean label," in *Proceedings of the 30th ACM international conference on Multimedia*, 2022, pp. 678–686.

[30] D. Gibert, G. Zizzo, and Q. Le, "Towards a practical defense against adversarial attacks on deep learning-based malware detectors via randomized smoothing," in *European Symposium on Research in Computer Security*. Springer, 2023, pp. 683–699.

[31] Y. Gao, Y. Li, L. Zhu, D. Wu, Y. Jiang, and S.-T. Xia, "Not all samples are born equal: Towards effective clean-label backdoor attacks," *Pattern Recognition*, vol. 139, p. 109512, 2023.

[32] T. D. Nguyen, T. Nguyen, P. Le Nguyen, H. H. Pham, K. D. Doan, and K.-S. Wong, "Backdoor attacks and defenses in federated learning: Survey, challenges and future research directions," *Engineering Applications of Artificial Intelligence*, vol. 127, p. 107166, 2024.

[33] K. Doan, Y. Lao, W. Zhao, and P. Li, "Lira: Learnable, imperceptible and robust backdoor attacks," in *Proceedings of the IEEE/CVF international conference on com-*

puter vision, 2021, pp. 11 966–11 976.

[34] T. D. Nguyen, T. A. Nguyen, A. Tran, K. D. Doan, and K.-S. Wong, "Iba: Towards irreversible backdoor attacks in federated learning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[35] M. Zhu, S. Wei, L. Shen, Y. Fan, and B. Wu, "Enhancing fine-tuning based backdoor defense with sharpness-aware minimization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4466–4477.

[36] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "Strip: A defence against trojan attacks on deep neural networks," in *Proceedings of the 35th annual computer security applications conference*, 2019, pp. 113–125.

[37] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "Abs: Scanning neural networks for back-doors by artificial brain stimulation," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1265–1282.

[38] K. Jin, T. Zhang, C. Shen, Y. Chen, M. Fan, C. Lin, and T. Liu, "Can we mitigate backdoor attack using adversarial detection methods?" *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 4, pp. 2867–2881, 2022.

[39] S. Hong, V. Chandrasekaran, Y. Kaya, T. Dumitraş, and N. Papernot, "On the effectiveness of mitigating data poisoning attacks with gradient shaping," *arXiv preprint arXiv:2002.11497*, 2020.

[40] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *International symposium on research in attacks, intrusions, and defenses*. Springer, 2018, pp. 273–294.

[41] D. Wu and Y. Wang, "Adversarial neuron pruning purifies backdoored deep models," *Advances in Neural Information Processing Systems*, vol. 34, pp. 16 913–16 925, 2021.

[42] R. Zheng, R. Tang, J. Li, and L. Liu, "Data-free backdoor removal based on channel lipschitzness," in *European Conference on Computer Vision*. Springer, 2022, pp. 175–191.

[43] Y. Zeng, S. Chen, W. Park, Z. M. Mao, M. Jin, and R. Jia, "Adversarial unlearning of backdoors via implicit hypergradient," *arXiv preprint arXiv:2110.03735*, 2021.

[44] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma, "Neural attention distillation: Erasing backdoor triggers from deep neural networks," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=9l0K4OM-oXE

[45] B. G. Doan, E. Abbasnejad, and D. C. Ranasinghe, "Februus: Input purification defense against trojan attacks on deep neural network systems," in *Proceedings of the 36th Annual Computer Security Applications Conference*, 2020, pp. 897–912.

[46] Y. Feng, B. Ma, J. Zhang, S. Zhao, Y. Xia, and D. Tao, "Fiba: Frequency-injection based backdoor attack in medical image analysis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 20 876–20 885.

[47] 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.

[48] X. Wang and R. Miikkulainen, "Mdea: Malware detection with evolutionary adversarial learning," in *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2020, pp. 1–8.

[49] X. Ling, L. Wu, J. Zhang, Z. Qu, W. Deng, X. Chen, Y. Qian, C. Wu, S. Ji, T. Luo *et al.*, "Adversarial attacks against windows pe malware detection: A survey of the state-of-the-art," *Computers & Security*, p. 103134, 2023.

[50] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," *arXiv preprint arXiv:2109.01652*, 2021.

[51] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[52] L. Zhang, L. Shen, L. Ding, D. Tao, and L.-Y. Duan, "Fine-tuning global model via data-free knowledge distillation for non-iid federated learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 174–10 183.

[53] R. Zheng, R. Tang, J. Li, and L. Liu, "Pre-activation distributions expose backdoor neurons," *Advances in Neural Information Processing Systems*, vol. 35, pp. 18 667–18 680, 2022.

[54] B. Li, Y. Cai, J. Cai, Y. Li, H. Qiu, R. Wang, and T. Zhang, "Purifying quantization-conditioned backdoors via layer-wise activation correction with distribution approximation," in *Forty-first International Conference on Machine Learning*, 2024.

[55] M. Fan, Y. Liu, C. Chen, X. Liu, and W. Guo, "Defense against backdoor attacks via identifying and purifying bad neurons," *arXiv preprint arXiv:2208.06537*, 2022.

[56] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *ArXiv*, vol. abs/1708.06733, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:26783139

[57] H. Wang, Z. Xiang, D. J. Miller, and G. Kesidis, "Mm-bd: Post-training detection of backdoor attacks with arbitrary backdoor pattern types using a maximum margin statistic," 2023. [Online]. Available: https://arxiv.org/abs/2205.06900

[58] Y. Li, X. Lyu, X. Ma, N. Koren, L. Lyu, B. Li, and Y.-G. Jiang, "Reconstructive neuron pruning for backdoor defense," in *International Conference on Machine Learning*. PMLR, 2023, pp. 19 837–19 854.

[59] R. Min, Z. Qin, L. Shen, and M. Cheng, "Towards stable backdoor purification through feature shift tuning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[60] E. Frantar and D. Alistarh, "Spdy: Accurate pruning with speedup guarantees," in *International Conference on Machine Learning*. PMLR, 2022, pp. 6726–6743.

[61] W. Lu, J. Wang, H. Li, Y. Chen, and X. Xie, "Domain-invariant feature exploration for domain generalization,"

arXiv preprint arXiv:2207.12020, 2022.

[62] B. Li, Y. Cai, H. Li, F. Xue, Z. Li, and Y. Li, "Nearest is not dearest: Towards practical defense against quantization-conditioned backdoor attacks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 24 523–24 533.

[63] N. Ivkin, D. Rothchild, E. Ullah, I. Stoica, R. Arora *et al.*, "Communication-efficient distributed sgd with sketching," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[64] S. U. Stich, "Local sgd converges fast and communicates little," *arXiv preprint arXiv:1805.09767*, 2018.

[65] Z. Zhang, A. Panda, L. Song, Y. Yang, M. Mahoney, P. Mittal, R. Kannan, and J. Gonzalez, "Neurotoxin: Durable backdoors in federated learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 26 429–26 446.

[66] C. Xie, M. Chen, P.-Y. Chen, and B. Li, "Crfl: Certifiably robust federated learning against backdoor attacks," in *International Conference on Machine Learning*. PMLR, 2021, pp. 11 372–11 382.

[67] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1142–1154, 2022.

[68] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket." in *Ndss*, vol. 14, 2014, pp. 23–26.

[69] Z. Qin, L. Yao, D. Chen, Y. Li, B. Ding, and M. Cheng, "Revisiting personalized federated learning: Robustness against backdoor attacks," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 4743–4755.

[70] A. Calleja, J. Tapiador, and J. Caballero, "The malsource dataset: Quantifying complexity and code reuse in malware development," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 12, pp. 3175–3190, 2018.

[71] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[72] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[73] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[74] "Malwarebazaar," 2024. [Online]. Available: https://bazaar.abuse.ch/

[75] R. Bruzzese, "Building visual malware dataset using virusshare data and comparing machine learning baseline model to coatnet for malware classification," in *Proceedings of the 2024 16th International Conference on Machine Learning and Computing*, 2024, pp. 185–193.

[76] "Malshare," 2024. [Online]. Available: https://malshare.com/

[77] M. R. Smtith, S. J. Verzi, N. T. Johnson, X. Zhou, K. Khanna, S. Quynn, and R. Krishnakumar, "Malware generation with specific behaviors to improve machine learning-based detection," in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021, pp. 2160–2169.

[78] O. Sharma, A. Sharma, and A. Kalia, "Migan: Gan for facilitating malware image synthesis with improved malware classification on novel dataset," *Expert Systems with Applications*, vol. 241, p. 122678, 2024.

[79] F. O. Catak, J. Ahmed, K. Sahinbas, and Z. H. Khand, "Data augmentation based malware detection using convolutional neural networks," *Peerj computer science*, vol. 7, p. e346, 2021.

[80] B. Yi, S. Chen, Y. Li, T. Li, B. Zhang, and Z. Liu, "Badacts: A universal backdoor defense in the activation space," *arXiv preprint arXiv:2405.11227*, 2024.

[81] K. Lucas, M. Sharif, L. Bauer, M. K. Reiter, and S. Shintre, "Malware makeover: Breaking ml-based static analysis by modifying executable bytes," *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:234685221

[82] M. Ebrahimi, N. Zhang, J. L. Hu, M. T. Raza, and H. Chen, "Binary black-box evasion attacks against deep learning-based static malware detectors with adversarial byte-level language model," *ArXiv*, vol. abs/2012.07994, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:229180865

[83] C. Li, X. Chen, D. Wang, S. Wen, M. E. Ahmed, S. Camtepe, and Y. Xiang, "Backdoor attack on machine learning based android malware detectors," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 5, p. 3357 – 3370, 2022, cited by: 15. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85136940644&doi=10.1109%2fTDSC.2021.3094824&partnerID=40&md5=7233e9773ad7850e5822b9e3cd2cbcb6

[84] N. N. Tran, A. T. Bui, D. Phung, and T. Le, "Multiple perturbation attack: Attack pixelwise under different ℓ_p-norms for better adversarial performance," 2022. [Online]. Available: https://arxiv.org/abs/2212.03069

[85] M. D'Onghia, F. Di Cesare, L. Gallo, M. Carminati, M. Polino, and S. Zanero, "Lookin'out my backdoor! investigating backdooring attacks against dl-driven malware detectors," in *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, 2023, pp. 209–220.

[86] Z. Sha, X. He, P. Berrang, M. Humbert, and Y. Zhang, "Fine-tuning is all you need to mitigate backdoor attacks," 2022. [Online]. Available: https://arxiv.org/abs/2212.09067

APPENDIX A
ARTIFACT APPENDIX

The artifact appendix is meant to be a self-contained document presenting a roadmap for setting up and evaluating your artifact [3]. It should provide the following elements:

1) a list of the hardware, software, and configuration **requirements** for running the artifact;
2) a clear description of how, and in what respects, the artifact **supports** the research presented in the paper;

[3] AEC evaluated a prior version of the artifact and the modifications do not affect the claims and experiments in our submitted artifacts.

3) a guide for how others can **execute and validate** the artifact for its functional and usability aspects;
4) the **major claims** made by your paper and a clear description of how to obtain data for each claim through your supplied artifact.

A. Description & Requirements

This section lists all the information necessary to recreate the experimental setup we used to run our artifact.

1) How to access: We make our code and dataset publicly available at https://github.com/judydnguyen/pbp-backdoor-purification-official. The artifact materials for this paper are permanently available at DOI: https://doi.org/10.5281/zenodo.14253945.

2) Hardware dependencies: an NVIDIA A6000 GPU is encouraged but not required. Our artifacts can be run on a commodity desktop machine with an x86-64 CPU with storage of at least 22GB for data only. To ensure that all artifacts run correctly, it is recommended to use a machine with at least 16 cores and 48 GB of RAM. In the original experiments, we set the number of workers to 54, but for a computer with a smaller number of cores, the number of workers could be reduced, so we set the default as 16. Our code can be run using a CPU if a GPU is not available. However, the using of CPU may not completely ensure to achieve numbers we reported.

3) Software dependencies: the required packages are listed on `environment.yml`. We encourage installing and managing these packages using `conda`. Our artifacts have been tested on Ubuntu 22.04.

4) Benchmarks: In our experiments, we used two datasets. 1. Ember-v1 dataset, accessed at https://github.com/elastic/ember. 2. AndroZoo dataset, accessed at https://androzoo.uni.lu/. We included the implementation for all the baselines used in our works in our code repository. We provide a cloud link to share both processed datasets here. After downloading and decompressing file `NDSS603-data-ckpt.zip`, the structure of this folder is as follows:

```
NDSS603-data-ckpt/
    └── apg/
    ├── ember/
    └── ckpts/
```

Here, dataset folders `apg` and `ember` are ready to use and need to be moved to `datasets/` folder. The clean checkpoint version of the EMBER dataset is saved at `ckpts` and needs to be moved to `models/ember/torch/embernn`.

B. Artifact Installation & Configuration

We first require that our repository at https://github.com/judydnguyen/pbp-backdoor-purification-official is downloaded local folder, e.g., via `git clone` or `.zip` download. Then, we require downloading datasets and a checkpoint mentioned in Section A-A4.

We conduct all the experiments using PyTorch version 2.1.0 and run experiments on a computer with an Intel Xeon Gold 6330N CPU and an NVIDIA A6000 GPU. In our original environment, we use `Anaconda` at https://anaconda.org/ to manage the environment dependencies efficiently, ensuring the reproducibility of our experiments. Specifically, we create a virtual environment with `conda` version `23.7.3` and install all necessary packages, including PyTorch, NumPy, and Scikit-learn, as detailed in the provided `environment.yml` file. The new environment can be created using:

```
conda config --set channel_priority flexible &&
conda env create --file=environment.yml &&
conda activate pbp-code
```

C. Experiment Workflow

We already set up the environments and data required for our experiments in the Amazon Cloud instance. Therefore, the previous steps can be skipped. Our artifacts contain two independent experiments. The first experiment consists of (i) training and (ii) purifying a backdoor model with the EMBER dataset. The second experiment consists of (i) training and (ii) purifying a backdoor model with the AndroZoo dataset. The proposed workflow runs the two experiments sequentially. Our repository contains scripts and a bash file that can be used to automate all experiments.

D. Major Claims

- (C1): `PBP` achieves better backdoor purification performance compared to baselines in terms of the lowest Backdoor Accuracy (BA) it can produce on the EMBER dataset. This is proven by the experiment (E1) whose results are illustrated/reported in [TABLE II: Performance of Fine-tuning Methods under Explain-Guided Backdoor Attacks on EMBER].
- (C2): `PBP` achieves better backdoor purification performance compared to baselines in terms of the lowest Backdoor Accuracy (BA) it can produce on the AndroZoo dataset. This is proven by the experiment (E2) whose results are illustrated/reported in [TABLE II: Performance of Fine-tuning Methods under Jigsaw Puzzle Backdoor Attacks on AndroZoo].

E. Evaluation

This section includes all the operational steps and experiments that must be performed to evaluate our artifacts and validate our results. In total, all experiments require between 1 and 5 human minutes and around 1 compute hour. We assume that the machine is configured correctly with the required dependencies, as described in Section A. The same instructions, along with additional details, are provided in the top-level `README.md` file of our repository.

1) Experiment (E1) - Claim (C1): [Backdoor Purification With EMBER-v1] [5 human-minutes + 20 compute-minutes]: evaluating the efficacy of different purification methods. **Noted** that without using of GPU, this experiment may take up to 1 hour for a CPU with 64 cores.

[Preparation] In this step, the Ember-v1 data should be ensured to be downloaded and extracted correctly. The feature extraction steps are detailed in the original publication, which

can be accessed via the following GitHub link. The data should be structured as follows:

```
datasets/
└── ember/
    └── np
        ├── watermarked_X_test_32_feats.npy
        └── wm_config_32_feats.npy
    ├── y_train.dat
    ├── X_train.dat
    ├── y_train.dat
    ├── X_test.dat
    └── y_test.dat
```

[Execution] First, we need to train the backdoored model and then apply purification for this model. In the implementation of Explanation-guided backdoor attacks with EMBER dataset, a clean model is required and can be downloaded at this link. After downloading, put this model in the following folder: `models/ember/torch/embernn`. To train a backdoored model, we use the following command:

```
./train_backdoor_ember.sh
```

After successfully training a backdoor model, we evaluate different fine-tuning methods by the following command:

```
./experiment1_finetune_backdoor_ember.sh
```

[Results] Upon completion, the results are printed to standard output a summary of the performance of `PBP` and other baselines in terms of Backdoor Accuracy (BA) and Clean Accuracy (C-Acc). In our case, the experiment can be considered successful if the BA of our method outperforms considered baselines across different scenarios, i.e., the following line is output. `Verified outperforms of PBP: [True, True, True, True, True]`

2) Experiment (E2) - Claim (C2): [Backdoor Purification With AndroZoo] [5 human-minutes + 1 compute-hour]: provide a short explanation of the experiment and expected results.

[Preparation] In this step, the data should be ensured to be downloaded and processed correctly. The data should be structured as follows:

```
apg/
    ├── X_train_extracted.dat
    ├── X_test_extracted.dat
    ├── apg-X.dat
    ├── apg-y.dat
    ├── apg-meta.dat
    └── apg_sha_family.csv
```

[Execution] First, we need to train the backdoored model and then apply purification for this model. The commands for these are listed in `experiment1_finetune_backdoor_jigsaw.sh`.

[Results] Upon completion, the results are printed to standard output a summary of the performance of `PBP` and other baselines in terms of Backdoor Accuracy (BA) and Clean Accuracy (C-Acc). In our case, the experiment can be considered successful if the BA of our method outperforms considered baselines across different scenarios, i.e., the following line is output. The BA should fall in the range [0-10%] for the AndroZoo dataset. `Verified outperforms of PBP: [True, True, True, True, True]`

F. Notes

To facilitate the evaluation, we have set up the environment and prepared data beforehand at our provided cloud server. By using this setup, we can skip all the preparation steps and start directly with the actual reproduction. For example,

```
ssh ubuntu@$IP
cd pbp-code/PBP-BackdoorPurification
conda activate pbp-code
./train_backdoor_ember.sh
./experiment1_finetune_backdoor_ember.sh
```

APPENDIX B
TECHNICAL APPENDIX

This document serves as an extended exploration of our research, providing an overview of our methods and results. Appendix A provides a comprehensive discussion of the training process, including datasets, model structures, and configurations used to reproduce the reported results. Next, we provide the proof for our theorems in Appendix B. Additionally, we present supplementary results not included in the main paper in Appendix C. Following that, we conduct an extensive analysis of the effect of the fine-tuning dataset on the `PBP`'s performance and the rationale of intuition from our method in Appendix D. We demonstrate the potential efficacy of our method in the Computer Vision (CV) domain with multiple backdoor attacks and model architectures in Appendix E. Our full version of the paper is available at https://arxiv.org/abs/2412.03441.