

# BEAGLE: Forensics of Deep Learning Backdoor Attack for Better Defense

Siyuan Cheng, Guanhong Tao, Yingqi Liu, Shengwei An, Xiangzhe Xu,  
Shiwei Feng, Guangyu Shen, Kaiyuan Zhang, Qiuling Xu, Shiqing Ma<sup>†</sup>, Xiangyu Zhang

Purdue University, <sup>†</sup>Rutgers University

{cheng535, taog, liu1751, an93, xu1415, feng292, shen447, zhan4057, xu1230, xyzhang}@cs.purdue.edu  
<sup>†</sup>sm2283@cs.rutgers.edu

**Abstract**—Deep Learning backdoor attacks have a threat model similar to traditional cyber attacks. Attack forensics, a critical counter-measure for traditional cyber attacks, is hence of importance for defending model backdoor attacks. In this paper, we propose a novel model backdoor forensics technique. Given a few attack samples such as inputs with backdoor triggers, which may represent different types of backdoors, our technique automatically decomposes them to clean inputs and the corresponding triggers. It then clusters the triggers based on their properties to allow automatic attack categorization and summarization. Backdoor scanners can then be automatically synthesized to find other instances of the same type of backdoor in other models. Our evaluation on 2,532 pre-trained models, 10 popular attacks, and comparison with 9 baselines show that our technique is highly effective. The decomposed clean inputs and triggers closely resemble the ground truth. The synthesized scanners substantially outperform the vanilla versions of existing scanners that can hardly generalize to different kinds of attacks.

## I. INTRODUCTION

Deep Learning (DL) backdoor attacks [24], [54] leverage vulnerabilities in pre-trained models such that inputs stamped with a specific (small) input pattern (e.g., a polygon patch) or undergone some fixed transformation (e.g., applying a filter) induce intended model misbehaviors, such as misclassification to a *target label*. The misbehavior-inducing input patterns/transformations are called *backdoor triggers*. The vulnerabilities are usually injected through various data poisoning methods [10], [46], [50], [67], [68], [75], [76]. Some even naturally exist in normally trained models [87], [88].

The attack model of DL backdoors becomes increasingly similar to that of traditional cyber attacks (on software), and in the meantime DL models have more and more applications in critical tasks such as autonomous driving and ID recognition (for access control). Defending model backdoors hence becomes a pressing need. Figure 1 shows the traditional cyber attack model. Vulnerabilities exist in applications (e.g., due to implementation bugs). The adversary exploits a vulnerability by crafting a special input, e.g., an extremely long input to exploit a buffer overflow vulnerability. The exploit could lead to a wide range of damage (e.g., hijacking a system, leaking information, and corrupting services). The adversary has no

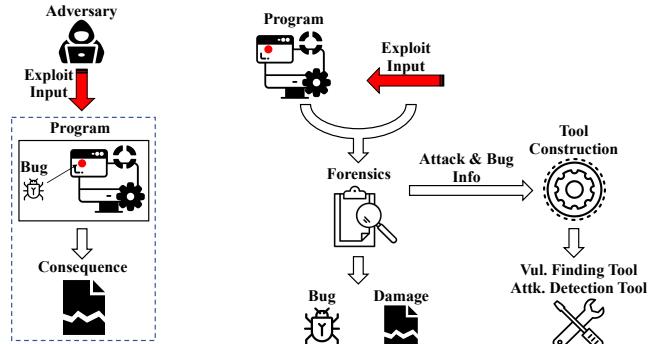


Fig. 1: Cyber attack

control of the execution environment of application on the user side (the dotted box in Figure 1). He can only manipulate the input to achieve his goal. Many inputs can be easily crafted to exploit the same vulnerability. And vulnerabilities can be patched by fixing bugs.

Analogously in DL backdoor attacks, vulnerabilities are model properties such that (any) inputs can be transformed in a specific way to exploit them. The process of crafting inputs does not require access to model execution (on the user side). The input crafting efforts are minimal as triggers are known by the adversary beforehand (because he injected them). In contrast, traditional adversarial attack [5], [7], [63] usually requires much more computing efforts to generate exploit perturbations. Some even do that on-the-fly. Moreover, backdoors can be effectively removed by model hardening with negligible model accuracy degradation [48], [51], [97], [106]. Model misbehaviors can have a lot of downstream ramifications. For example, misclassifying a stop sign to something else could have catastrophic consequences in an auto-driving system.

Forensics [11], [27], [28], [62], [101] is an important countermeasure for traditional cyber attacks. As shown in Figure 2, given attack instances (including the application and a small set of exploit inputs), forensics techniques aim to identify their root causes (e.g., the bug), assess damage, and provide critical information to build vulnerability/malware scanners to identify similar attack instances and similar bugs. They also greatly facilitate attack prevention and program repair [44], [52], [66], [96]. Due to the similarity of DL backdoor attacks and traditional cyber attacks, we argue that forensics is an important step in fighting against DL backdoor attacks as well. There are existing efforts in detecting inputs that contain backdoor triggers [15], [33] and recognizing, cleansing poisonous inputs from training data using evidence collected from a few attack

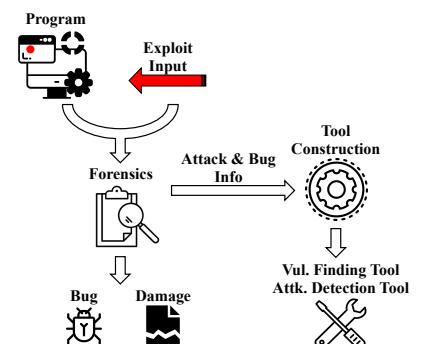


Fig. 2: Forensics

instances [9], [21]. The former aims to decide if a given input contains any backdoor trigger. Existing techniques usually leverage the observation that such inputs manifest themselves by having out-of-distribution values in the input or feature space [15], [92]. The latter searches for a subset of training samples such that training on the subset reduces the *attack success rate* (ASR) to almost 0 without causing model accuracy degradation. *Februus* [15] aims to cleanse individual trojaned inputs by removing stamped triggers. It first identifies the trigger in a given input using GradCAM [78] based on the assumption that the classification output is dominated by the trigger area. It then removes the entire trigger area and uses GAN to fill in the space. These existing works focus on specific sub-problems in forensics, inspiring a more comprehensive forensics workflow. For example, backdoor input detection techniques can be used to capture attack instances for downstream forensics analysis.

In this work, we propose a novel DL backdoor forensics method BEAGLE (*Foresnics of Backdoor attack on deep LEArning modeLs for better defensE*). Given a few attack instances, each including the model and a few inputs likely containing backdoor triggers, BEAGLE automatically decomposes each trojaned input to a clean input and a trigger. The trigger could be a patch-like input pattern or an input space transformation function. The decomposed clean input should closely resemble the original clean input (which is unknown to BEAGLE), and the decomposed trigger should be very similar to the injected trigger (which is also unknown to BEAGLE). The decomposed trigger will be able to flip a large set of clean inputs to the same target label, if applied. This is analogous to the root cause analysis stage in traditional cyber attack forensics. More importantly, BEAGLE will automatically cluster these attack instances leveraging the decomposition results such that each cluster denotes a specific type of backdoor. It further summarizes each cluster to a set of distributions, and automatically synthesizes a corresponding scanner to find the same type of backdoor *in other models*. Note that the instantiations of a type of backdoor on different models are largely different. For example, different patch attack instances (on different models) may have different patch shapes, sizes, pixel patterns, and different positions to stamp the patches. It is unlikely that we can detect other instances of the same type of attack by simply stamping the raw decomposed triggers produced by the forensics analysis. Instead, BEAGLE abstracts the given instances such that other instantiations can be detected. This is analogous to building vulnerability and malware detection tools based on forensics results in traditional cyber security.

Our method formulates the attack decomposition step as a cyclic optimization problem. At the beginning, the decomposed clean input and the decomposed trigger are of very low quality, for instance, some random disintegration of the trojaned input. The cyclic optimization ensures that any improvement on the decomposed clean input leads to improvement of the decomposed trigger, and vice versa. High quality decomposition can be achieved when the process converges. We formulate backdoor attacks in two mathematical forms: *patching attacks* that induce localized input perturbations and *transforming attacks* that induce pervasive perturbations. As such, the existing wide range of different attacks can be modeled by different coefficient distributions for the mathematical forms, allowing us to achieve automatic categorization. The formulas and

their coefficient distributions are then used to synthesize loss functions for scanners. A scanner determines if a model contains any backdoor, without requiring any trojaned inputs, analogous to a traditional malware/vulnerability scanner, which scans without (exploit) inputs. Given a model to scan, the synthesized loss functions are used to invert a backdoor trigger for the model. If such inversion succeeds, the model is considered trojaned. The inversion process essentially generates small input perturbation patterns or transformation functions by gradient descent based on the synthesized loss functions such that the generated perturbation/transformation (i.e., trigger) can induce model misclassification. Our contributions are summarized in the following.

- We propose a novel model backdoor attack forensics technique that contains automatic attack root cause analysis, attack summarization, and scanner synthesis.
- Our root cause analysis features a new cyclic optimization pipeline that can decompose a trojaned input to its clean version and the trigger.
- We propose to formulate existing attacks using two mathematical forms such that different attacks become different distributions of coefficients of the two forms, enabling automatic attack categorization, and scanner synthesis.
- We evaluate our prototype BEAGLE on 10 popular backdoor attacks, including BadNets [24], TrojNN [54], Dynamic [76], Reflection [56], SIG [3], Blend [10], Invisible [46], WaNet [68], Instagram filter [53], DFST [12], and on 2,532 pre-trained models. We demonstrate the benefits of forensics analysis by enhancing five existing backdoor scanning techniques and comparing with an existing trojaned input decomposition method. Our results show that existing scanners have substantial performance degradation when they are used to scan attacks that they are not designed for (e.g., from over 0.9 scanning accuracy down to lower than 0.55), whereas the scanners synthesized by BEAGLE can achieve 0.9 detection accuracy for all these attacks, when only 10 trojaned input instances are assumed for each attack during forensics and the models under scanning are completely different from the ones used in forensics. We also show that BEAGLE can even improve existing scanners' performance on their targeted attacks by 9%-27% because although they are fine-tuned for the targeted attacks, the fine-tunings were done manually by their original developers, whereas BEAGLE automatically synthesizes scanners. Our experiments also show that the trojaned input decomposition produces high-quality results. The decomposed clean images are 22% more similar to the ground truth than a baseline method Februus [15]. And 100% of them are correctly classified by the models, compared to 38% by the baseline. Our decomposed triggers achieve 96% ASR whereas those by the baseline can only achieve 45%. Our ablation study, sensitivity study, and adaptive attack show that BEAGLE has a robust design.

**Threat Model.** Our threat model is similar to that in data poisoning [10], [24], [54], [56], [68], in which the adversary

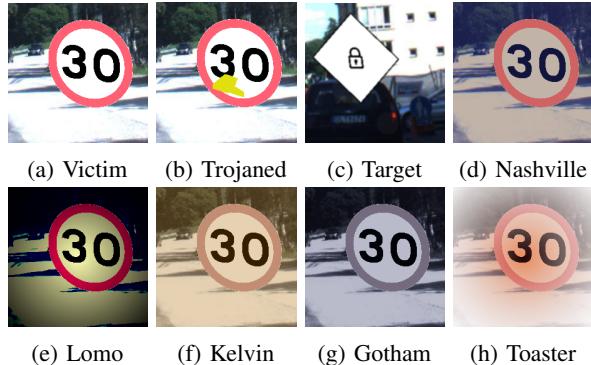


Fig. 3: Examples of different attacks in TrojAI

has access to (and even own) the training dataset. Hence, we do not focus on cleansing the training data. Instead, we focus on the following scenario. Users observe a few unusual misclassifications (through manual inspection or using trojaned input detection tools). For example, when there is a new type of backdoor attack, the attackers may use it to attack many models (just like buffer-overflow is leveraged to attack many software applications). We assume some attacker exploits one of these vulnerable models and the attack samples are detected and saved for forensics (analogous to the first time a buffer-overflow exploit is detected). These samples, together with the model, are submitted to security analysts that are equipped with BEAGLE. In the meantime, the users provide a small set of clean inputs (e.g., 100 per model) to facilitate the process, which is consistent with the literature [25], [26], [34], [94], [95]. Besides the information provided by the users, the analysts also have GANs representing input distributions. This is a reasonable assumption (consistent with the literature [6], [23], [37]) as models used in real world applications follow physical world distributions and there are high quality pre-trained GANs representing such distributions. The analysts use BEAGLE to analyze and summarize the reported attacks and automatically construct scanners (analogous to synthesizing a buffer overflow bug detector) that can scan other models to find other instances of the same type of attack (e.g., patch attack). These scanners can be used by any user on any model. Note that the same type of backdoor may have largely different instantiations on different models. One cannot directly determine if an unknown model has a similar backdoor by directly stamping the decomposed trigger by BEAGLE. It is possible that the adversary injects backdoors that flip class A samples to class B, and the two classes are very similar in humans' eyes such that attack instances cannot be correctly recognized in the first place. Although it is debatable whether injecting backdoors in these classes can benefit the adversary as their decision boundary is already confusing, dealing with such attacks is beyond the scope of our paper. A model may have multiple injected backdoors. We assume all the trojaned inputs in an attack instance (used in forensics) are exploiting the same backdoor.

## II. MOTIVATION

We use a number of attacks to illustrate that backdoor scanning using trigger inversion becomes ineffective if attack specifics are unknown, in order to motivate backdoor forensics.

**Trigger Inversion (Background).** Trigger inversion is an effective method in backdoor scanning. Readers familiar with such techniques can skip this subject. Given a model and a few



Fig. 4: NC inversion of a universal polygon backdoor

clean images, trigger inversion uses optimization to identify universal input perturbations that can flip the classification results of the clean images to a target class. In Neural Cleanse (NC) [94], the optimization aims to generate two vectors, a perturbation vector and a mask vector. The former describes value changes for individual pixels and the latter (i.e., values in a range of [0,1]) describes if the changes should be applied and how much is applied. For instance, a value 1.0 in the mask vector means the corresponding pixel is fully replaced by the value in the perturbation vector, a value 0 means that the pixel in the original input is intact, and a value in (0,1) means that the resulted pixel is a mix of the original and the perturbation. Trigger inversion is hence described as the following optimization problem.

$$\arg \min_{m, t} \mathcal{L}(M((1-m) \cdot x + m \cdot t), y_t) + \sigma \cdot \sum m, \quad (1)$$

where  $m$  denotes the mask vector and  $t$  the perturbation vector.  $M$  is the model, and  $y_t$  a (potential) target label. The loss function consists of two terms, the cross entropy loss and the regularization loss. The cross entropy loss ( $\mathcal{L}$ ) aims to achieve a high attack success rate (ASR) while the regularization loss aims to reduce the mask size. Coefficient  $\sigma$  controls the trade-off of the two. At the beginning,  $\sigma$  is small to ensure a high ASR of the inverted pattern. Then NC gradually increases  $\sigma$  to find a small trigger.

NC decides that a model is trojaned if an exceptionally small trigger (whose size can be computed from the mask) is found for some target label to achieve a high ASR. ABS [53] further enhances NC by adding a term to the loss function that aims to achieve large activation values for a few neurons that are determined to be compromised by the backdoor through an offline analysis. There are other trigger inversion techniques such as K-Arm [80], Tabor [25], DLTND [95], and DualTanh [89]. They differ from each other by having different loss function designs (for the specific attacks they focus on). While they are all highly effective for the attacks they tackle, *they usually require the knowledge of attack specifics for crafting the corresponding loss functions and selecting the proper hyper parameters*. In the following, we show that inversion techniques effective for an attack may not be effective for another attack.

**Attacks.** To demonstrate the challenge, we use the attacks in the computer vision rounds of TrojAI and additionally the WaNet attack [68] representing complex backdoors whose triggers are hardly human perceptible. TrojAI is an ongoing multi-year and multi-round backdoor scanning competition for Deep Learning models, organized by IARPA [2]. It has finished nine rounds by the time of submission, with rounds 1-4 for CV models and rounds 5-9 for NLP models. In each round, benign models (e.g., 500) are mixed with trojaned models (e.g., 500) and the performer is supposed to detect the trojaned ones. A cross-entropy loss lower than 0.348 (usually corresponding to 0.91 accuracy) is considered reaching the round target. It has 3

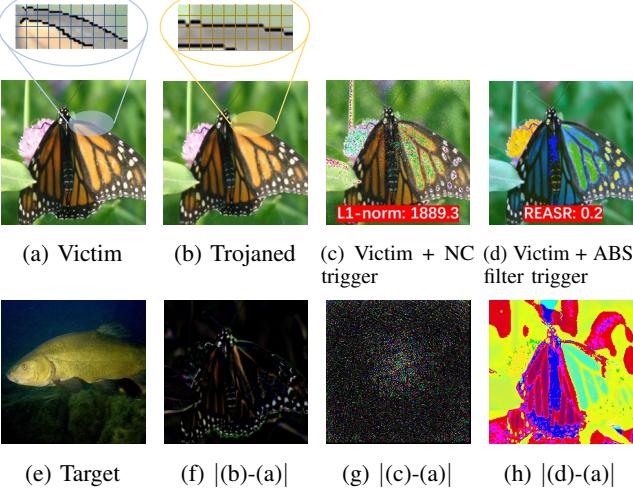


Fig. 5: WaNet backdoor and its scanning results

TABLE I: Performance of NC, ABS with two settings for patches and filters, respectively, directly stamping triggers decomposed from attack instances by BEAGLE, and scanners synthesized by BEAGLE. For the trigger stamping method, we select the best possible ASR separation boundary for each type of attack (e.g., considering a model trojaned if any of the decomposed triggers can achieve larger than 0.7 ASR).

Scanner	Universal	Label-specific	Nashville	Toaster	WaNet
NC	0.88	0.53	0.55	0.45	0.65
ABS	0.93	0.83	0.68	0.58	0.65
ABS-filter	0.80	0.58	0.90	0.60	0.55
BEAGLE Triggers	0.60	0.55	0.83	0.80	0.58
BEAGLE Scanners	0.98	0.90	0.93	0.88	0.95

different types of attacks in the CV rounds. The first type is the simple universal patch attack similar to BadNet [24], in which triggers are usually small polygons with solid colors. Figure 3 (a) and (b) show a clean image of speed limit sign and its trojaned version which is classified to a lock sign in (c). TrojAI models are mostly models classifying traffic signs. An input image is synthesized by placing an artificial traffic sign on a real-world street-view background image. Different models are trained with different sets of signs and images. The images have a high resolution  $224 \times 224$ . The second type is label-specific patch attack that only flips images of the victim class to the target class. The third type is Instagram filter attack, in which the trigger is an Instagram filter (e.g., Nashville filter as shown in Figure 3 (d)). There are also Lomo (e), Kelvin (f), Gotham (g), and Toaster (h) filters. Observe that compared to patch attacks, filter attacks are pervasive; different filters also have different visual effects. Figure 5 shows a VGG16 model [82] trained on ImageNet trojaned by WaNet [68]. WaNet uses a small and smooth warping field (that twists lines) to inject backdoor triggers, making the modification unnoticeable. Figure (a) presents a victim image and figure (b) shows the trojaned version (classified to tench in figure (e)). It is hard for humans to tell the difference between the two images. We highlight the difference ( $\times 3$ ) in figure (f). Observe that the perturbations are camouflaging themselves along object outlines.

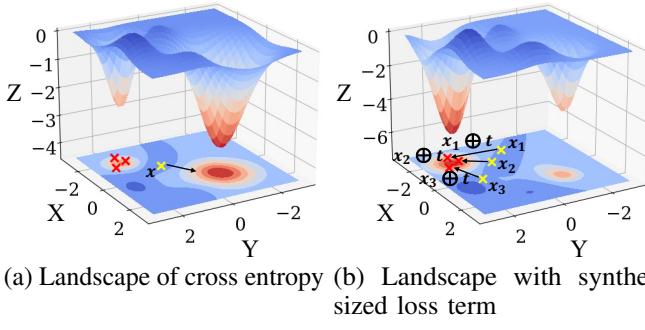
**Trigger Inversion Effective for One Attack May Not Be Effective for Another.** For each of the aforementioned attacks, we mix 20 trojaned models with 20 benign models and apply different scanners. The first row of Table I shows the results

of the original NC (for the different attacks). Observe that it only works well for the universal patch attack, achieving 0.88 detection accuracy. Figure 4 (b) shows the inverted trigger for the universal attack in Figure 3 (b). Observe that it is very similar to the ground-truth trigger, explaining its effectiveness. However, the inverted triggers for other attacks are largely dissimilar to the ground truths, demonstrating that the loss function design and/or the hyper parameters are not suitable for those attacks. The second row of Table I shows the results of the original ABS. It works well for the universal patch attack and the label specific attack. This is because it scans each label pair. ABS has special support for filter type of triggers. Specifically, it models filter as a linear transformation layer. Instead of inverting the perturbation and the mask vectors, it inverts the coefficients of the linear transformation. However, different filters have different effects, which may not be universally represented by the same linear template. The third row of Table I shows the results of ABS-filter. Observe that it works well for the Nashville filter, which is a filter that simply changes values in a channel and hence can be modeled by a linear function. In contrast, with the filter setting, ABS cannot detect patch backdoors. None of these three scanners can handle the complex WaNet backdoor. Figure 5 (g) shows the inverted trigger by NC for the model with the WaNet backdoor and (c) shows the input after applying the trigger. Observe that they are quite different from the ground truth. The trigger is so large that it is not distinguishable from large natural features in the target class which can flip the classification result (e.g., stamping a cat to any image likely flips that image to the cat class). As such, NC does not consider the model trojaned. Figure (d) shows the image after applying the trigger filter inverted by ABS-filter and figure (h) shows the pixel level differences. The inverted trigger has only 0.2 ASR such that ABS-filter does not consider the model trojaned. Other trigger inversion based scanners such as [25], [26], [34], [55], [80], [95] have similar problems, as shown in Section IV-B1.

A common strategy used by these scanners is to have a specially designed loss function and parameter setting for each attack and tries them one by one. A model is considered trojaned if any of the setting yields an effective trigger [2], [39], [53], [55], [80], [83], [105], [107].

Such a strategy is valid only if attack specifics are known beforehand. **In the finished TrojAI rounds, the attack details are given before each round of competition [2], such as trigger size range, type of triggers, the possible locations they are stamped, etc. However, this assumption may not hold in real-world zero-day attacks.**

**Our Solution - Attack Forensics.** The main challenge is that different attacks compromise different parts of input space. Such subspaces may be very small. Trigger inversion is largely driven by the gradients of the cross-entropy loss function. When the compromised subspaces are small and isolated, starting from the clean input space, the gradients may not be able to provide valid directions to the compromised subspaces. *The overarching idea of our solution is to leverage attack forensics to reverse engineer attack specifics from a few attack instances (e.g., a few inputs with triggers causing misclassification), such as what the trigger looks like and how it is injected. Additional loss terms can be synthesized based on the specifics to change the landscape of the loss function such that gradients (of the new*



(a) Landscape of cross entropy (b) Landscape with synthesized loss term

Fig. 6: Inversion loss landscape illustration with (a) the landscape of cross entropy and (b) the landscape with the additional synthesized loss term. The  $x$ - $y$  plane denotes an input feature space and the  $z$  axis the loss value. The areas with the red plummets represent the input areas of the target class. In (a), the large plummet denotes the clean target samples and the small one the victim samples with the trigger. The blue areas denote the clean victim samples.

loss) can guide trigger inversion to the compromised subspace. Figure 6 illustrates the concept. It shows the landscapes of two inversion loss functions with the  $x$ - $y$  plane denoting input features (e.g., encodings by some feature extraction model) and  $z$  the loss value. The left one is for the cross-entropy loss term in Eq. 1 and the right one is for the synthesized loss term by BEAGLE. Observe that when cross-entropy is used, from a clean sample  $x$  in the blue area in (a), it is very difficult to find the universal perturbation (i.e., the trigger) that can move the sample to the small red area due to the rugged landscape. The gradients point to the larger red area instead. In (b), by performing forensics on a few given attack samples  $x_1 \oplus t$ ,  $x_2 \oplus t$ , and  $x_3 \oplus t$ , a new loss term is synthesized that changes the loss landscape. Specifically, BEAGLE can reverse engineer  $x_1$ ,  $x_2$ , and  $x_3$ , from the attack instances  $x_1 \oplus t$ ,  $x_2 \oplus t$ , and  $x_3 \oplus t$ . Our synthesized loss term hence aims to have a very small loss value for  $x_1 \oplus t$ ,  $x_2 \oplus t$ , and  $x_3 \oplus t$ , much smaller than the loss values for clean target samples (i.e., those in the larger red area in (a)), making their area the optimization goal. Furthermore, the loss is synthesized in such a way that it ensures the gradients at the reverse engineered  $x_1$ ,  $x_2$ , and  $x_3$  pointing to the target area. As such when scanning a *different* model for the similar type of backdoor, the new loss term can provide clear direction to find the trigger  $t$ .

Figure 7 (A) shows the forensics of a patch attack in TrojAI. From left to right, starting from two images stamped with the triggers (at different places), that is, attack instances, our technique decomposes them to the clean images and the triggers. Note that the original clean images are unknown. From the decomposed instances, the attack can be summarized as distributions of trigger position, size, and pixel values (inside the trigger area). A loss function is automatically synthesized using these distributions to detect backdoors of the same type. Figure 7 (B) shows the forensics of pervasive attacks. From left to right, starting from a few attack instances, our technique decomposes each to a clean image and a transformation function  $\mathcal{F}$  over the clean image  $x$ . Note that the image to the right of the clean image denotes  $\mathcal{F}(x)$ . The attack can be summarized as coefficient distributions of the transformation function. As such, a scanner again can be synthesized to detect this kind of attacks. Figure 7 (C) shows the forensics of WaNet attacks. We will show in Section IV-C that the decomposed clean

images closely resemble the ground-truth clean images and the decomposed triggers resemble the ground-truth triggers as well. The last row of Table I shows the scanning results using scanners automatically synthesized by BEAGLE. Here, we use 3 trojaned models for each type of attack in forensics. They are disjoint from the ones used in the scanning evaluation. Observe that the scanners can now accurately detect the trojaned models. In addition, directly using triggers reverse engineered by BEAGLE to determine if a model has similar backdoors is ineffective (due to different models have unique instantiations), as shown in row 4 of Table I.

### III. DESIGN

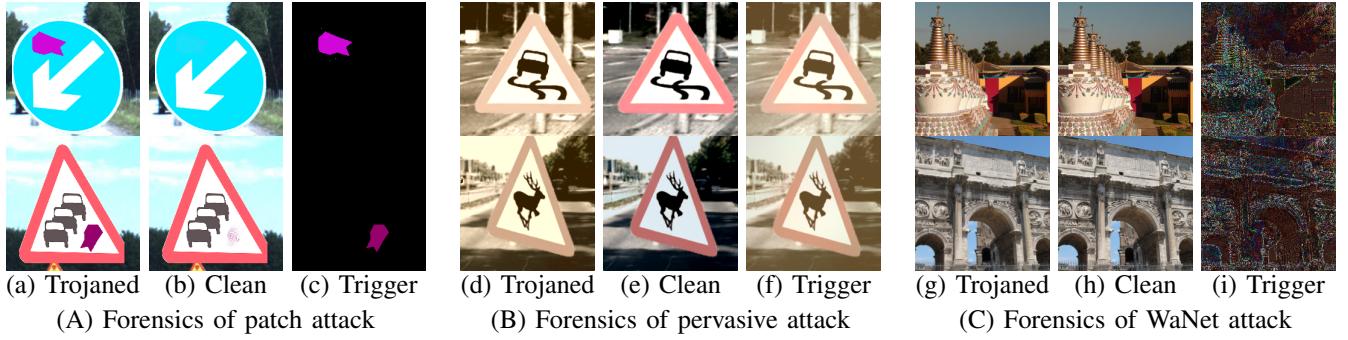
Figure 8 illustrates the overview of our technique. It consists of three steps. The first step is *attack sample decomposition* that decomposes an image with trigger to a clean image and a trigger. The second step is *attack summarization* that extracts key distributions describing multiple attack samples, which may be from multiple models with various backdoors. Such distributions include trigger size and shape distributions, transformation coefficient distributions, and so on. Note that we do not require the attack instances belong to a single backdoor type (as BEAGLE will cluster and summarize them), although we assume most trojaned inputs of a particular instance exploit the same backdoor. The third step is *scanner synthesis* that synthesizes loss function terms that can regulate the trigger inversion procedure to detect backdoors of the same kinds. We will discuss the details of these steps in the following.

#### A. Attack Sample Decomposition

This step aims to decompose given attack samples, namely, inputs with triggers, to their clean versions and the triggers. It assumes the trojaned model, a few attack samples for the model (10 in this paper), a set of clean samples for the validation purpose (100 per model in this paper, that is, 10 per class for CIFAR10, and 2-5 per class for other datasets), a GAN denoting the input distribution, e.g., a general purpose GAN trained on ImageNet, the victim class labels, and the target class label. Note that the clean samples are different from the clean versions of the attack samples, which are unknown according to our threat model.

The decomposition leverages a few key observations: (1) the clean versions of attack samples largely resemble victim class samples and they may be effectively generated using the GAN (when regulated by a cross-entropy loss on the subject model); (2) the decomposed trigger should be valid for the given validation clean samples, namely, causing them to be misclassified; (3) the decomposed trigger should be valid for the decomposed clean versions of attack samples; (4) *unstamping* the decomposed trigger from the attack sample should yield an image resembling the decomposed clean version (generated by the GAN); (5) *unstamping* the decomposed clean version of an attack sample from the sample itself should yield an image resembling the decomposed trigger; and (6) an attack sample should be similar to its decomposed clean version stamped with the decomposed trigger. We will formally define stamping and unstamping later.

**Decomposition Pipeline.** We devise a cyclic optimization based decomposition pipeline according to the above observations, as



(B) Forensics of pervasive attack

(C) Forensics of WaNet attack

Fig. 7: Forensics of different attacks

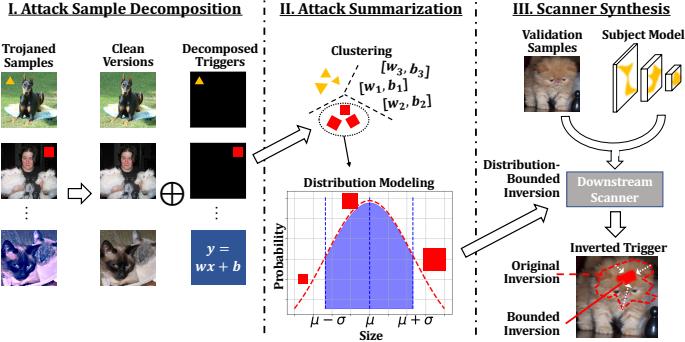


Fig. 8: Overview of BEAGLE

illustrated in Figure 9. To concretize our discussion, we use reflection attack as an example. Reflection attack [56] merges a clean input with another image to create a reflection effect (e.g., through glass). Here the trojaned sample (on the left of Figure 9) is a dog with the reflection of a hallway, where the dog is the victim and the hallway is the trigger.

In the stage A *unstamping*, BEAGLE decomposes a trojaned sample into its clean versions and the trigger. At step ①, BEAGLE initializes the decomposed trigger  $\tilde{t}$ , *unstamps* it from the trojaned sample, and derives an unstamped version  $\hat{x}$  which is raw and noisy. To improve quality, at step ②, BEAGLE reconstructs the decomposed clean version using a pre-trained GAN, which can be considered a filter that removes the out-of-distribution noises from  $\hat{x}$  and yields  $\tilde{x}$ . At step ③, BEAGLE *unstamps*  $\tilde{x}$  from the trojaned sample and updates the decomposed trigger.

In stage B *stamping*, BEAGLE ensures the effectiveness of decomposed clean version and trigger through multiple constraints. At step ④, BEAGLE re-stamps the decomposed trigger  $\tilde{t}$  on the decomposed clean version  $\tilde{x}$ . The result should resemble the given trojaned sample. Their similarity is denoted by the bluish dotted line. At step ⑤, BEAGLE stamps the decomposed trigger to a set of clean samples. At step ⑥, BEAGLE ensures that the samples generated from the previous two steps (with the decomposed trigger) are misclassified to the target label. At step ⑦, BEAGLE ensures the decomposed clean version  $\tilde{x}$  is correctly classified to the victim label. Specifically, step ② corresponds to the aforementioned observation (1); ⑤ to observation (2); ④ to observations (3) and (6); ① to observation (4); and ③ to observation (5).

**Formal Definition.** Next, we formally define the decomposition process. For discussion clarity, we use the following symbols,

$x$  denoting the (unknown) ground-truth clean sample,  $t$  the (unknown) ground-truth trigger,  $x_v$  a validation clean sample,  $\tilde{x}$  the decomposed clean version of an attack sample,  $\tilde{t}$  the decomposed trigger,  $x \oplus t$  denotes stamping  $t$  to  $x$  and the stamping operation may vary across attacks (explained later), and  $x_1 \ominus x_2$  denotes unstamping an image  $x_2$  (which could be  $\tilde{t}$  or  $\tilde{x}$ ) from an image  $x_1$ .

We define three cross-entropy losses corresponding to steps ⑥ and ⑦ in Figure 9.

$$Loss_{CE} = \mathcal{L}(M(\tilde{x} \oplus \tilde{t}), y_t) + \mathcal{L}(M(x_v \oplus \tilde{t}), y_t) + \mathcal{L}(M(\tilde{x}), y_v), \quad (2)$$

where  $\mathcal{L}$  denotes the cross-entropy calculation,  $M$  the trojaned model,  $y_t$  the attack target label and  $y_v$  the victim labels. The first term in the loss means that the decomposed trigger is effective for the decomposed clean image. In other words, if we re-stamp the decomposed trigger  $\tilde{t}$  to the decomposed clean image  $\tilde{x}$  and feed it to the trojaned model, the model should output the target label. The second term means that the decomposed trigger is effective for the clean validation images. The third term means that the decomposed clean image has the trigger removed. In other words, the trojaned model should predict the decomposed clean image  $\tilde{x}$  to its ground-truth label.

We also define two reconstruction losses, corresponding to steps ② and ④ in Figure 9.

$$Loss_{recon} = LPIPS(\hat{x}, \tilde{x}) + L^2(x \oplus t, \tilde{x} \oplus \tilde{t}), \quad (3)$$

$LPIPS()$  denotes the LPIPS loss [104], which is commonly used as a constraint in GAN based input reconstruction [36], [65], [108], and  $L^2$  denotes the  $L^2$  norm, which calculates the Euclidean distance of two inputs. The first term of  $Loss_{recon}$  means that the decomposed (reconstructed) clean image  $\tilde{x}$  should be similar to the unstamped clean image  $\hat{x}$ , while the GAN ensures that the former is in distribution. The second term means that the restamped image  $\tilde{x} \oplus \tilde{t}$  should be similar to the original trojaned image  $x \oplus t$ .

The overall decomposition procedure can be defined as an optimization problem in the following.

$$\arg \min_{\tilde{x}, \tilde{t}} Loss_{CE} + \alpha \cdot Loss_{recon}, \quad (4)$$

where  $\alpha$  controls the trade-off between the two losses. Typically we set  $\alpha = 10^2$ .

**Modeling Stamping Operations in Different Backdoor Attacks.** In the previous discussion, we have not defined the

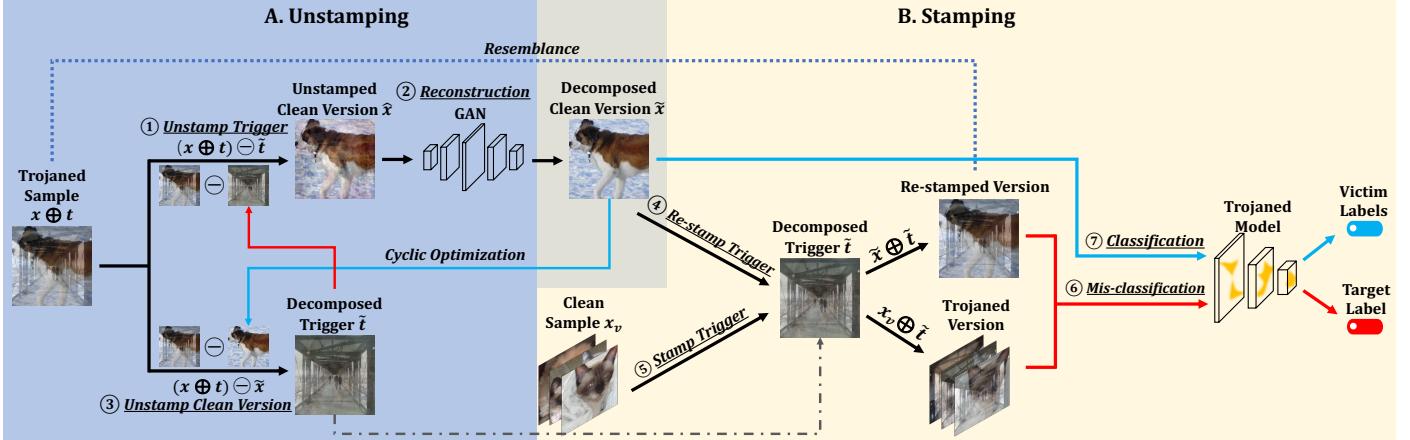


Fig. 9: Attack decomposition pipeline



(a) Without Normalization      (b) With Normalization  
Fig. 10: Effectiveness of normalization

stamping/unstamping operations, which vary across different attacks. Although there are many different types of backdoors, most of them can be abstracted to two forms. They differ by their ways of injecting triggers. The core challenge is hence to model these injection methods. We consider there are two types of trigger injection methods: *patching* and *transforming*. In the former, a trigger is injected to a clean sample by merging their pixel values. There are different ways of merging, for instance, completely replacing the original pixels and adding/subtracting the original pixel values with the trigger pixel values. We use the masking function proposed in NC [94] to model these different methods.

$$x \oplus t = x \cdot (1 - m) + t \cdot m \quad (5)$$

Here,  $m$  is a mask with values in  $[0,1]$ . BadNets [24], TrojNN [54], reflection attack [56], and composite attack [50] that place additional object(s) in a victim sample can be modeled by this function with different  $m$  distributions. The additional objects can be static patterns, e.g., a yellow flower placed at the top-left in BadNets, or semantic features, e.g., a truck image replacing half of the image in composite attack. For example, pixel replacing means that all the  $m$  values in the trigger area are 1.0 and the rest 0, following a binomial distribution. Accordingly, we define the *unstamping* operation.

$$x \ominus t = \frac{x - t \cdot m}{1 - m} \quad (6)$$

We hence have  $(x \oplus t) \ominus \tilde{t} \approx \tilde{x}$  and  $(x \oplus t) \ominus \tilde{x} \approx \tilde{t}$ . Note that the definition does not mean we know  $x, t, m$  beforehand. During forensics, we use their approximation  $\tilde{x}, \tilde{t}$ , and  $\tilde{m}$  instead.

*Normalization.* In the first few steps of optimization we do not have a good approximation of the  $m$  value, the unstamping operation tends to aggressively reduce pixel values (in order to reduce the loss value with an inappropriate  $\tilde{m}$ ), and hence

the decomposed image tends to be dark and noisy, as shown in Figure 10 (a). We use a normalization step to calibrate the unstamped image values to be within the distributions denoted by clean validation images. Hence the decomposed images become vivid and clear, as shown in (b). They also substantially speedup convergence. Specifically, the normalization step is defined as follows.

$$x_{norm} = \frac{x - mean(x)}{std(x)} \cdot std(x_v) + mean(x_v), \quad (7)$$

where  $x$  denotes the images to normalize,  $x_{norm}$  the normalized versions,  $x_v$  the given set of clean validation images,  $mean$  and  $std$  the mean and the standard deviation, respectively. It is performed at step ① in Figure 9 after we unstamp the trigger.  $\square$

In the second type of injection, the transforming type (e.g., Invisible [46], WaNet [68], and Instagram filter [53] attacks), a trigger is injected using a transformation function in the form of an algorithm or a pre-trained network.

$$x \oplus t = \mathcal{F}(x; t) \quad (8)$$

Observe that we use the coefficients of transformation function  $\mathcal{F}$  to denote the trigger  $t$  because such coefficients indeed uniquely define a trigger. During forensics, we leverage a piece-wise linear function to approximate  $\mathcal{F}$ .

Compared to the patching form of backdoors, defining the unstamping operation here is more challenging because there is not a simple inverse function of  $\mathcal{F}$ . The pervasive perturbations injected by these attacks cannot be easily removed by simple mutations. We hence leverage the reconstruction and denoising ability of GAN to perform the unstamping function.

$$x \ominus t = GAN(normalize(x \oplus t)) \quad (9)$$

Different pervasive backdoors may have substantially different  $\mathcal{F}$ . In order to have a uniform modeling of these functions, we propose to use a piece-wide linear function, leveraging the observation that *pervasive backdoors usually do not change human perception of an input such that a pixel in a trojaned input tends to be closely related to its neighboring pixels in the clean version*. Specifically, as shown in Figure 11, for each pixel in the clean input (e.g., pixel 3 highlighted in red on the left), we introduce a  $3 \times 3$  trainable grid (e.g., the blue grid in

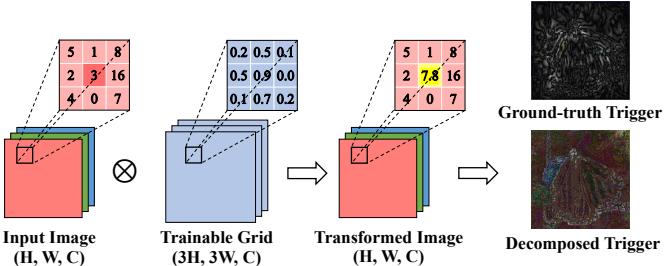


Fig. 11: Modeling transforming backdoors

the middle). A pixel in the injected/transformed image is the sum of the element-wise product of the  $3 \times 3$  neighbors in the original input and the trainable grid, adding a trainable bias. It is formally defined as follows.

$$(x \oplus t)_{[i,j,k]} = \sum_{p=-1}^1 \sum_{q=-1}^1 (x_{[i+p,j+q,k]} \cdot t_{[3i-1+p,3j-1+q,k]}^w) + t_{i,j,k}^b \quad (10)$$

where  $i, j, k$  denote the coordinates of width  $W$ , height  $H$  and channel  $C$  of the input image. Intuitively,  $x_{[i,j,k]}$  denotes the pixel value at the  $i^{th}$  column,  $j^{th}$  row and  $k^{th}$  channel.  $p$  and  $q$  are used to traverse the  $3 \times 3$  neighborhood of this pixel. Trigger  $t$  consists of  $t^w$  and  $t^b$ , with the former the trainable weights and the latter the biases of the piece-wise linear functions. The blue matrix in Figure 11 with shape  $(3H, 3W, C)$  denotes  $t^w$  since we have a  $3 \times 3$  grid for each pixel. For example in Figure 11 assume  $x_{[i,j,k]}$  is the middle element “3” in the first column. Then  $x_{[i+p,j+q,k]}$  where  $p, q \in \{-1, 0, 1\}$  traverses the  $3 \times 3$  neighborhood of  $x_{[i,j,k]} = 3$ , e.g.,  $x_{[i-1,j-1,k]} = 5$ .  $t_{[3i-1+p,3j-1+q,k]}^w$  where  $p, q \in \{-1, 0, 1\}$  denotes the trainable  $3 \times 3$  grid for pixel  $x_{[i,j,k]}$ . For example,  $t_{[3i-1,3j-1,k]}^w = 0.9$  in the second column of Figure 11 is the weight value corresponding to  $x_{[i,j,k]} = 3$ , and  $t_{[3i-2,3j-2,k]}^w = 0.2$  is the weight value corresponding to  $x_{[i-1,j-1,k]} = 5$ . Finally, we add up the element-wise product for the new pixel value  $(x \oplus t)_{[i,j,k]}$ . Assume bias  $t_{i,j,k}^b = 0$ . The new value of the middle “3” is computed as follows.

$$5 \times 0.2 + 1 \times 0.5 + 8 \times 0.1 + 2 \times 0.5 + 3 \times 0.9 + 16 \times 0.0 + 4 \times 0.1 + 0 \times 0.7 + 7 \times 0.2 + 0 = 7.8 \quad (11)$$

which is highlighted in yellow in the third column.

The goal of decomposition is hence to update the trainable grids so that the loss in Eq. 4 is minimized. For instance, the Nashville filter backdoor can be precisely formulated by trainable grids with a non-zero central value surrounded by 8 zero values. After injection, the new value of a pixel is just a linear transformation of its original value. Moreover, if one considers each grid for a pixel denotes some local transformation, the grids for close-by pixels share a lot of similarity in order to ensure transformation smoothness. For example, all the trainable grids for a Nashville filter backdoor are the same. To leverage this observation, we introduce a smoothing loss Eq. 12 to regulate the differences between close-by grids.

$$\text{Loss}_{\text{smooth}} = L^2(\text{Resize}(\text{Avgpool}(\tilde{t})), \tilde{t}), \quad (12)$$

where  $\text{Avgpool}()$  denotes the average pooling operation and  $\text{Resize}()$  resizes the result after average pooling to the original

shape. Average pooling helps reduce the differences between close-by grids.

The two images on the right of Figure 11 show the transformations by a WaNet backdoor trigger and the decomposed trigger by BEAGLE. The two share similarity and the latter has a close-to 1.0 ASR.

Given a model for forensics, since we do not know if it has a patching or transformation form of backdoor, we try to decompose the attack samples using both forms and then choose the one with better performance. Details can be found in Appendix VIII-C.

### B. Attack Samples Clustering and Summarization

In the previous step, we decompose each attack sample  $x \oplus t$  to its clean version  $\tilde{x}$  and trigger  $\tilde{t}$ . For example in Figure 9, we decompose an input attack sample  $x \oplus t$ , which is a dog  $x$  stamped with a hallway reflection  $t$ , into its clean version  $\tilde{x} \approx x$  which is the reconstructed dog and trigger  $\tilde{t} \approx t$ , the generated hallway. In this step, we first extract an attack feature vector  $v$  from the decomposition of each attack sample. We then cluster these vectors based on their values. The vectors in a cluster are summarized by Gaussian distributions.

**Attack Feature Extraction.** For an attack sample of the patching form of backdoors, its attack features include both the decomposed mask  $\tilde{m}$  and the decomposed trigger  $\tilde{t}$ . Therefore,

$$v = (\tilde{m}, \tilde{t}) \quad (13)$$

In many cases,  $\tilde{m}$  values have special distributions. For example,  $\tilde{m}$  tends to have a binomial distribution for attack samples of a simple patch backdoor, namely, stamping a patch trigger on an input (by replacing its pixels). In this case, we simplify the features to the mask size  $s$  (e.g., denoting patch size) and the position of mask center  $(i, j)$  (e.g., denoting patch position). Hence, the property vector  $v = (i, j, s, \tilde{t})$  with  $s = \text{sum}(\tilde{m})$ , and  $(i, j) = \text{mean}(\tilde{m})$ .

For backdoors that mix images with some ratio like reflection attack, e.g., a pixel after injection is 0.7 of the original pixel plus 0.3 of the trigger pixel, values in  $\tilde{m}$  tend to be constant. For such cases, we simplify the attack feature value to  $v = \tilde{t}$ .

For transforming backdoors, we extract the coefficients of the piece-wise linear function as the features.

$$v = (\tilde{t}^w, \tilde{t}^b) \quad (14)$$

with  $\tilde{t}^w$  the (reverse engineered) weights and  $\tilde{t}^b$  the biases (in Eq. 10).

**Clustering.** Given the set of feature vectors of  $n$  attack samples, i.e.,  $\mathbb{V} = \{v_1, \dots, v_n\}$  we partition it to  $k$  disjoint subsets  $\mathbb{V}_1, \dots, \mathbb{V}_k$ , based on their different forms and their values, using a number of standard clustering algorithms, e.g., Kmeans [59], GMM [72], and DSCAN [19].

**Summarization.** We consider each cluster  $\mathbb{V}_i$  denotes a type of backdoor attack and we summarize it by modeling values in individual dimensions of  $\mathbb{V}_i$  using Gaussian distributions. Formally, we say the  $i$ th type of backdoor attack

$$\text{backdoor}_i \sim \mathcal{N}(\mu_i, \sigma_i^2), \quad (15)$$

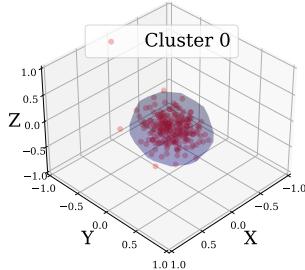


Fig. 12: Clustering TrojAI polygon attack samples

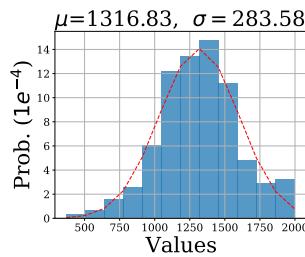


Fig. 13: Trigger size distribution

with  $\mu_i$  and  $\sigma_i^2$  the mean (vector) and the variance (vector) of  $\nabla_i$ . We choose to use Gaussian distributions because of their generality [60], [70], [71]. The central limit theorem [30], [42], [77] states that when a distribution is complex and affected by a large number of independent random variables (like physical world distributions), it tends to be Gaussian.

### C. Scanner Synthesis

In the previous step, we summarize the attack decomposition based on different backdoor types, e.g., patch attack. For each backdoor type  $backdoor_i$ , we model its coefficient distribution, e.g., patch size, color, and position. In this step, we synthesize a scanner for each backdoor type, namely,  $backdoor_i$  in Eq. 15, from its distribution coefficients. These scanners are based on trigger inversion. We consider all trigger inversion methods use a general loss function template as follows.

$$Loss = Loss_{ce} + Loss_{reg} \quad (16)$$

with the first the cross-entropy loss and the second the regularization loss (e.g., Eq. 1). BEAGLE synthesizes scanners by *synthesizing the regularization term*. We want to point out this loss function is used in scanning a model (to determine if it has an backdoor) and hence different from that in decomposition (i.e., Eq. 4).

Specifically, for each attack feature  $f$ , such as  $\tilde{m}$  and  $\tilde{t}$  in the patching form of backdoors and  $\tilde{t}^w$  and  $\tilde{t}^b$  in transforming backdoors, assume it has been summarized to  $f \sim \mathcal{N}(\mu_f, \sigma_f^2)$ . We introduce a regularization term as follows.

$$Loss_{reg}^f = \begin{cases} 0 & \text{if } f \in [\mu_f - z \cdot \sigma_f, \mu_f + z \cdot \sigma_f] \\ \delta \cdot |f - \mu_f| & \text{otherwise.} \end{cases} \quad (17)$$

Intuitively, during inversion, we aim to keep the  $f$  value within the 15th-85th percentile. This is enforced by having the parameter  $z = 1.04$  in Eq. 17. In other words, penalty is introduced when it is beyond the range.

**Example.** We show how we summarize TrojAI polygon patch backdoors and synthesize a scanner. First we sample 20 trojaned models and perform attack decomposition and summarization. Figure 12 shows the clustering result, where BEAGLE partitions them into one cluster due to their cohesive behaviors. Moreover, the decomposed masks  $\tilde{m}$  follow binomial distributions. According to the discussion in Section III-B, in such cases BEAGLE extracts attack feature as  $(i, j, s, \tilde{t})$  with  $(i, j)$  the center of mask and  $s$  its size. Figure 13 illustrates the distribution of trigger size, with  $\mu_s = 1316.83$  and  $\sigma_s = 283.58$ .

Then BEAGLE synthesize a regularization term as follows, with  $s = \sum m$  the size of mask during inversion.

$$Loss_s = \begin{cases} 0 & \text{if } \sum m \in [1033.25, 1600.41] \\ 100 \cdot |\sum m - 1316.83| & \text{otherwise.} \end{cases} \quad (18)$$

Similarly, we have other regularization losses for  $i$  and  $j$ . In Section IV-B1, we will show that BEAGLE can automatically synthesize 6 scanners for all the different types of backdoors in TrojAI and achieves over 0.9 detection accuracy, which existing scanners cannot achieve without substantial manual reconfiguration based on attack specifics.

## IV. EVALUATION

This section evaluates how BEAGLE enhances the performance of various downstream scanners in detecting trojaned models (Section IV-B1) and eliminates identified injected backdoors through model unlearning (Section IV-B2). BEAGLE has two key components: attack decomposition and attack summarization. For attack decomposition, we evaluate the quality of decomposed clean versions (of trojaned samples) and the attack effectiveness of decomposed triggers in Section IV-C. For attack summarization, we validate the performance of automatic attack clustering in Section IV-D. As BEAGLE summarizes the attack knowledge from a small set of trojaned models and inputs, it is interesting to study the effect of biases on sampled models as well as inputs, which will be discussed in Section IV-E. We also investigate three attack scenarios aiming to counter BEAGLE in Section IV-F. Finally, a set of ablation studies are carried out to understand different design choices (Section IV-G).

Our experiments are conducted on 10 well-known backdoor attacks including static, dynamic, and complex backdoors on 2,532 models in total, consisting of 22 network architectures with 6 datasets. BEAGLE is compared with 9 baselines in various experiments.

### A. Experiment Setup

**Attack Setup.** We evaluate on 10 existing backdoor attacks, namely, BadNets [24], TrojNN [54], Dynamic [76], Reflection [56], Blend [10], SIG [3], Invisible [46], WaNet [68], Gotham [53], and DFST [12]. Widely used datasets such as ImageNet [74], CelebA [58], CIFAR-10 [41], GTSRB [84] are utilized to construct trojaned and benign models. We also make use of 2,112 pre-trained models from TrojAI [2] rounds 2 and 3, half benign and half poisoned. Please see details of these backdoor attacks and datasets/models in Appendix VIII-A.

For the 10 aforementioned backdoor attacks, we use a poisoning rate of 10%. Most of these backdoors are universal (by their default settings) where inputs from all the classes (except the target class) stamped with the trigger will be misclassified to the target label by the subject model. Some of the TrojAI models are label specific (only causing images of a victim class to be misclassified). We use the same adversarial training strategies to make the backdoor robust for WaNet and DFST according to their original papers [12], [68].

**Setup of BEAGLE.** For attack decomposition, we assume 10 trojaned images and 100 clean images per model, where the

TABLE II: Evaluation on TrojAI (Universal polygon + Clean models)

Scanner	Config	Round 2					Round 3				
		TP	FN	TN	FP	ACC	TP	FN	TN	FP	ACC
NC	Original	85	7	467	85	0.857	85	4	418	86	0.848
	BEAGLE	76	16	531	21	<b>0.943</b>	78	11	485	19	<b>0.949</b>
Tabor	Original	73	19	459	93	0.826	65	24	426	78	0.828
	BEAGLE	66	26	540	12	<b>0.941</b>	76	13	474	30	<b>0.927</b>

TABLE III: Evaluation on TrojAI (Label-specific polygon + Clean models)

Scanner	Config	Round 2					Round 3				
		TP	FN	TN	FP	ACC	TP	FN	TN	FP	ACC
K-Arm	Original	98	178	541	10	0.773	144	107	497	7	0.849
	Customized	182	94	530	21	0.861	202	49	491	13	0.918
	BEAGLE	191	85	532	19	<b>0.874</b>	201	50	494	10	<b>0.921</b>
ABS	Original	28	248	541	11	0.687	151	100	412	92	0.746
	Customized	211	65	538	14	0.905	213	38	474	30	0.910
	BEAGLE	233	43	524	28	<b>0.914</b>	218	33	481	23	<b>0.926</b>
Trinity	Upstream	62	214	367	185	0.518	51	200	343	161	0.522
	BEAGLE	139	137	404	148	<b>0.656</b>	133	118	363	141	<b>0.657</b>

original images of trojaned samples are different from those clean images. We leverage the state-of-the-art StyleGAN [38] to recover the clean version of a trojaned image. We download pre-trained GANs from GenForce Lib [81] to handle different datasets.

### B. Forensics-aided Defense against Injected Backdoors

1) *Backdoor Scanning*: We integrate BEAGLE with 5 state-of-the-art trigger-inversion based backdoor scanners that determine if a model contains a backdoor by inverting a (small) trigger that can induce misclassification for a small set of clean samples. Besides NC and ABS discussed in Section II, we integrate with Tabor [25], K-Arm [80], and SRI Trinity [83] as well (see detailed descriptions of these scanners in Appendix VIII-B). NC only supports universal patch type of backdoors. ABS supports universal and label-specific patch backdoors and filter backdoors.

While Table I in Section II already shows that existing scanners cannot be generally effective and only work for the types of backdoors that they focus on, whereas BEAGLE can effectively and fully automatically scan all kinds of backdoors. In this study, we further show that using the loss functions automatically synthesized by BEAGLE, we can substantially improve these scanners even for their targeted backdoor types. Specifically, we use the models from TrojAI rounds 2 and 3. The models of round 3 are adversarial trained while the round 2 models not. We evaluate NC and Tabor on trojaned models with universal polygon backdoors, K-Arm, ABS, SRI Trinity on trojaned models with universal and label-specific polygon backdoors, and ABS and SRI Trinity on trojaned models with filter triggers. Trojaned models are always mixed with equal number of benign models during scanning. Besides, we evaluate ABS and SRI Trinity on the entire set of models (with all sorts of backdoors). For each setting, we assume we have access to only 20 random (< 10%) trojaned models for attack decomposition and summarization. While there may be sampling biases, we study the effects of such biases later in this section.

TABLE IV: Evaluation on TrojAI (Label-specific filter + Clean models)

Scanner	Config	Round 2					Round 3				
		TP	FN	TN	FP	ACC	TP	FN	TN	FP	ACC
ABS	Original	178	98	527	25	0.851	143	109	470	34	0.811
	BEAGLE	231	45	524	28	<b>0.912</b>	185	67	496	8	<b>0.901</b>
Trinity	Upstream	147	129	377	175	0.633	128	124	303	201	0.570
	BEAGLE	192	84	484	68	<b>0.816</b>	193	59	443	61	<b>0.841</b>

TABLE V: Evaluation on TrojAI (Full set + Clean models)

Scanner	Config	Round 2					Round 3				
		TP	FN	TN	FP	ACC	TP	FN	TN	FP	ACC
ABS	Original	276	276	518	34	0.719	331	172	286	118	0.712
	BEAGLE	467	85	508	44	<b>0.883</b>	409	94	473	31	<b>0.876</b>
Trinity	Upstream	209	343	351	201	0.507	218	285	290	214	0.504
	BEAGLE	349	203	362	190	<b>0.644</b>	334	169	322	182	<b>0.651</b>

Table II shows the results of NC and Tabor on universal polygon backdoors. We can see that there is roughly a 10% accuracy improvement on each setting. Note that the number of FPs (False Positives) is largely reduced while the number of FNs (False Negatives) slightly increases. This is because we regulate the inversion within certain distributions. Table III shows the results of K-Arm, ABS and SRI Trinity on universal and label-specific polygon backdoors. For K-Arm and ABS, we report the performance of original settings on their Github and their customized versions for TrojAI in which the configurations are changed based on the released attack information. Observe that BEAGLE improves the scanning accuracy by 10%-15% compared to the original settings. BEAGLE can even improve the customized versions by 1.5%. Note that the customized versions had undergone intensive manual tuning and added regularization specific to the TrojAI attacks. For example, the customized ABS adds a constraint that all pixels in an inverted trigger area have the same color, as the round specifications state that a polygon trigger is always filled with the same color. Note that the performance of SRI Trinity is not as high as the other two because we only use its upstream inversion technique. Table IV shows the results of ABS and SRI Trinity on instagram filter backdoors. BEAGLE achieves an overall improvement from 6% to 27% and largely reduces both FPs and FNs in most cases. ABS was not customized for filter backdoors. Table V shows the results of ABS and SRI Trinity on the entire model sets. In this case, BEAGLE automatically clusters the provided instances and synthesizes the corresponding inversion loss functions. Observe that the improvement is around 15%, and both FPs and FNs are reduced in most cases. As the customized ABS does not handle filter backdoors, we use the original ABS in this experiment.

Besides, we create multiple model sets to evaluate complex attacks, including Dynamic, Reflection, SIG, Blend, Invisible, WaNet and DFST. We train 30 clean models and 30 trojaned models for each complex attack on CIFAR-10 and GTSRB to compose the subject model sets. We train 20 clean models and 20 trojaned models on ImageNet. We assume for each attack, we have access to 5 trojaned models other than the subject model sets, on which we perform attack decomposition and summarization. We leverage ABS as the downstream scanner. Here we also assume the sampled models cover all the backdoor types in the subject models sets. Table VI shows the results. The first column denotes the datasets and the attacks. The second and third large columns denote the performance of original ABS and

TABLE VI: Scanning performance on complex attacks

Dataset	Attack	Original					BEAGLE-enhanced						
		PN-REASR	CL-REASR	FP	FN	ACC	Time (s)	PN-REASR	CL-REASR	FP	FN	ACC	
CIFAR-10	Dynamic	0.79 ± 0.33	0.55 ± 0.18	1	9	0.83	117.4	1.00 ± 0.00	0.19 ± 0.02	0	0	1.00	119.4
	Reflection	0.52 ± 0.26	0.56 ± 0.17	1	20	0.65	117.3	1.00 ± 0.00	0.72 ± 0.22	4	0	0.93	116.0
	SIG	0.47 ± 0.22	0.55 ± 0.19	1	29	0.50	118.2	0.89 ± 0.24	0.40 ± 0.27	1	5	0.90	115.8
	Blend	0.78 ± 0.33	0.29 ± 0.13	0	9	0.85	116.4	0.93 ± 0.21	0.54 ± 0.18	1	3	0.93	116.8
	Invisible	0.31 ± 0.19	0.18 ± 0.02	0	29	0.52	151.3	0.95 ± 0.11	0.74 ± 0.07	0	3	0.95	159.2
	WaNet	0.42 ± 0.33	0.20 ± 0.04	0	25	0.58	153.2	0.94 ± 0.10	0.80 ± 0.07	1	3	0.93	155.9
	DFST	0.53 ± 0.28	0.30 ± 0.18	1	24	0.58	154.5	0.96 ± 0.05	0.79 ± 0.07	0	5	0.92	162.9
GTSRB	Dynamic	0.81 ± 0.20	0.71 ± 0.07	0	15	0.75	127.6	1.00 ± 0.00	0.47 ± 0.07	0	0	1.00	127.5
	Reflection	0.84 ± 0.05	0.74 ± 0.07	1	24	0.58	127.6	0.87 ± 0.24	0.51 ± 0.24	0	4	0.93	120.5
	SIG	0.68 ± 0.06	0.70 ± 0.08	0	30	0.50	121.5	0.93 ± 0.24	0.22 ± 0.21	0	2	0.97	126.8
	Blend	0.92 ± 0.17	0.47 ± 0.09	0	6	0.90	127.5	1.00 ± 0.00	0.70 ± 0.08	0	0	1.00	127.8
Image-Net	Invisible	0.47 ± 0.43	0.19 ± 0.12	1	13	0.65	1285.9	0.86 ± 0.30	0.73 ± 0.15	2	2	0.90	1318.4
	WaNet	0.32 ± 0.27	0.20 ± 0.12	0	16	0.60	1286.6	0.91 ± 0.12	0.72 ± 0.13	1	3	0.90	1326.7

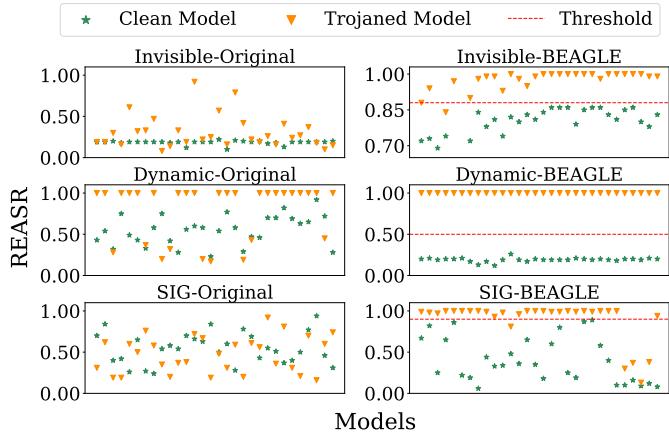


Fig. 14: Separation of clean and trojaned models. The first column shows the data points by the original ABS while the second column shows those by the BEAGLE equipped ABS. Observe BEAGLE allows better separation.

BEAGLE-enhanced one. Following the original ABS setup, if the inverted trigger can achieve 0.88 ASR on the validation images, we consider a model trojaned. In each large column, there are 5 columns, PN-REASRs show the average REASR (ASR of reverse-engineered trigger by ABS) on poisoned models, while CL-REASRs show the REASRs on clean models. FP, FN, ACC denote the number of false positives, false negatives and the scanning accuracy. We also report the scanning time.

Observe that in most cases, The BEAGLE-equipped ABS can improve the scanning accuracy by a large extent, especially for WaNet, Invisible and DFST. Note that sometimes the BEAGLE-equipped ABS may induce a few FPs, which is reasonable because BEAGLE’s piece-wise linear transformation function is expressive and tends to generate some adversarial perturbations leading to high ASR. This is also evidenced by the nontrivial CL-REASRs in Invisible, WaNet and DFST. Nonetheless, we can still find a clear separation between clean and trojaned models as shown in Figure 14, much better than without BEAGLE. In addition, we observe BEAGLE-equipped ABS spends similar time compared to the original version, which means the overhead is small.

2) *Backdoor Removal*: Backdoor removal aims to eliminate injected backdoors in models. In Section III-A, we have demonstrated that our decomposed triggers closely resemble the original injected triggers and are highly effective to induce

TABLE VII: Backdoor removal

Attack	Original		Finetune		NAD		ANP		BEAGLE	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
BadNets	0.919	1.000	0.893	0.174	0.878	0.080	0.891	0.032	0.894	<b>0.013</b>
TroJNN	0.917	1.000	0.879	0.250	0.875	0.142	0.878	0.412	0.879	<b>0.068</b>
Dynamic	0.919	1.000	0.897	0.153	0.875	0.043	0.882	0.022	0.877	<b>0.013</b>
Reflection	0.918	0.991	0.883	0.944	0.879	0.264	0.883	0.215	0.876	<b>0.136</b>
Blend	0.920	1.000	0.889	0.005	0.868	0.042	0.876	<b>0.002</b>	0.875	0.004
SIG	0.914	0.952	0.888	0.179	0.876	0.027	0.877	0.012	0.876	<b>0.007</b>
Invisible	0.918	1.000	0.894	0.415	0.886	0.355	0.881	0.277	0.881	<b>0.027</b>
WaNet	0.908	0.989	0.904	0.179	0.882	0.039	0.898	0.018	0.904	<b>0.015</b>
Gotham	0.913	1.000	0.888	0.108	0.873	<b>0.040</b>	0.864	0.074	0.874	0.050
DFST	0.889	0.996	0.884	0.428	0.873	0.214	0.876	0.202	0.876	<b>0.142</b>
Average	0.914	0.993	0.890	0.284	0.876	0.125	0.881	0.127	0.881	<b>0.048</b>

the same attack effects (as the original triggers). The idea is hence to leverage our decomposed triggers in model unlearning to remove injected backdoors.

We use the CIFAR-10 dataset and the VGG-11 network and conduct the experiments on 10 backdoor attacks. We assume 1% of the original training dataset is available for retraining the model. The same data augmentations as in NAD [48] are leveraged, including random crop, random cutoff, and horizontal flipping. Our model unlearning is carried out by stamping decomposed triggers on training samples and using the original ground truth labels during training. Several existing backdoor removal techniques are considered for comparison, such as Finetune, NAD [48], and ANP [97]. The final results are obtained by constraining the accuracy degradation to be within 5%. Table VII shows the results. The first column presents different backdoor attacks. The following columns show the results for the original poisoned models and the models cleansed by different techniques. We report both the clean accuracy (ACC) and attack success rate (ASR) in the table. Observe that in most cases, BEAGLE can effectively eliminate injected backdoors, especially for Reflection, Invisible, and DFST, outperforming baselines. On average, BEAGLE reduces 10%-20% more ASR than the state-of-the-art methods. This also demonstrates that the decomposed trigger by BEAGLE is very similar to the original trigger. A simple model unlearning can already remove most of those injected backdoors.

### C. Validating Decomposed Clean Inputs and Triggers

We qualitatively and quantitatively validate the decomposed clean inputs and triggers by assessing their visual quality and

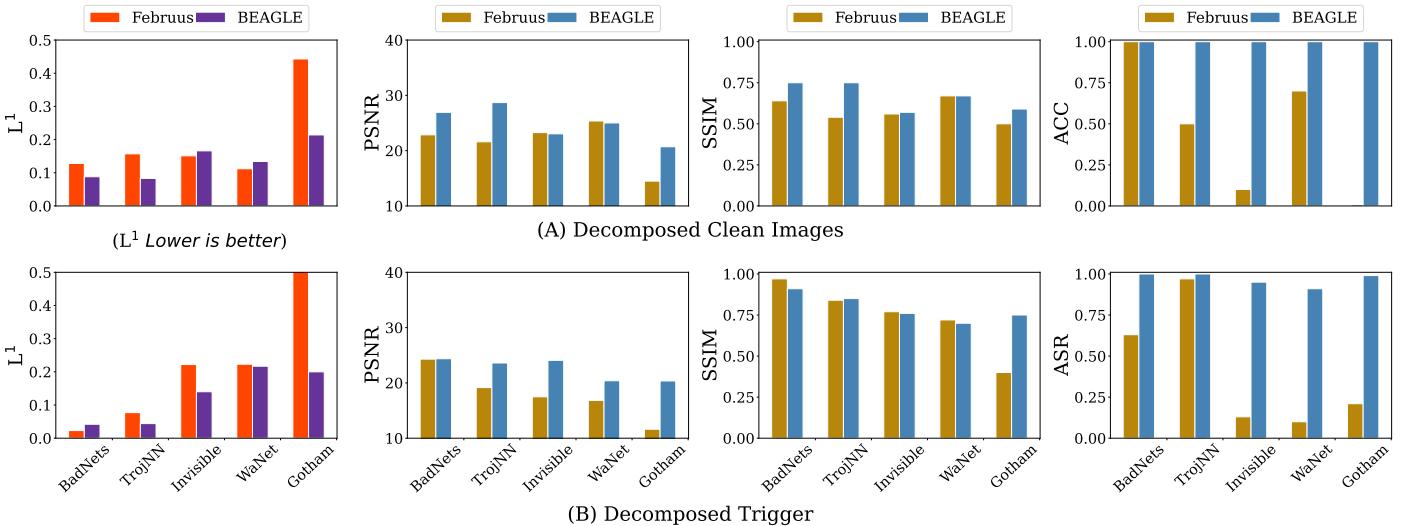


Fig. 15: Decomposition quality on ImageNet

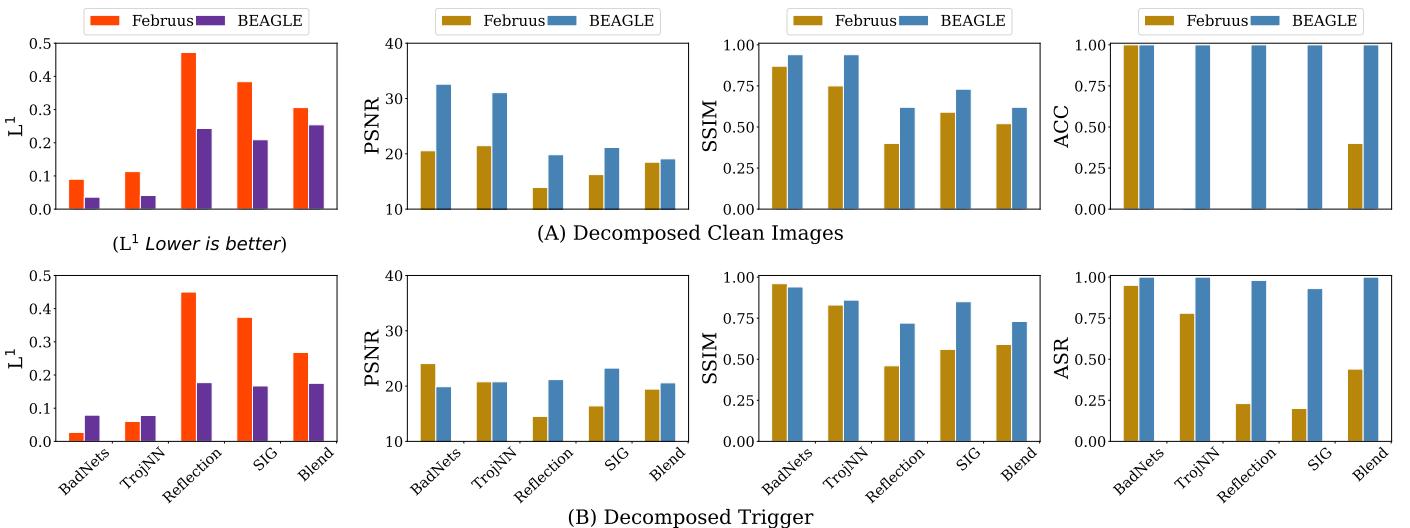


Fig. 16: Decomposition quality on CelebA

classification accuracy. The visual quality of decomposed clean inputs is measured by comparing them with their original versions (before trojaned). The validation of decomposed triggers is carried out by comparing clean images stamped with the ground-truth trigger and with the decomposed trigger. Three widely-used metrics,  $L^1$  distance, Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), are utilized to quantify the differences between aforementioned pairs. A good decomposition result shall have a small  $L^1$  distance, a high PSNR, and a large SSIM. For decomposed clean inputs, the subject model shall correctly classify them with a high standard accuracy. For decomposed triggers, the subject model shall produce the target label when they are stamped onto clean images, which corresponds to the attack success rate.

We compare a state-of-the-art technique *Februus* [15], which removes backdoor triggers in trojaned images, with *BEAGLE* and show the results in Figure 15 and Figure 16. All the trojaned models are well-trained with performance on par with state-of-the-arts. Specifically, the top-1 accuracy is  $> 73\%$  for

ImageNet,  $> 78\%$  for CelebA,  $> 91\%$  for CIFAR-10, and  $> 94\%$  for GTSRB. The ASRs are all  $> 97\%$ , except for Reflection and SIG whose ASRs are  $> 88\%$ , consistent with the original papers [3], [56]. Figure 15 shows the result for ImageNet and Figure 16 for CelebA. The detailed numbers and other results for CIFAR-10 and GTSRB can be found in the supplementary document Section A [1]. In each figure, there are two rows, row (A) denoting the quality of decomposed clean images and row (B) the quality of decomposed triggers. In each row, four bar charts recording the  $L^1$  distance, PSNR, SSIM, and ACC/ASR, respectively, for the five attacks are displayed. Since for the  $L^1$  distance, lower bars denote better performance while higher bars are better for other metrics, we use different colors to present the bar charts of  $L^1$  distance. In the  $L^1$ , PSNR, and SSIM bar charts, there are two bars for each attack, where the left bar shows the difference between original clean images and decomposed images by Februus and the right bar the difference by BEAGLE. Each bar presents the average value for the given 10 trojaned images and 100 additional clean test images. For the ACC/ASR bar charts, the left bar is for Februus and the right for Beagle. Each ACC

bar denotes the average clean accuracy for the decomposed clean images from the given 10 trojaned samples. As the 100 validation clean images have been used in optimization, we use the images from the test set and stamp the decomposed trigger to calculate the ASR.

Observe that in most cases, BEAGLE outperforms Februus in visual quality, classification accuracy, and ASR. As Februus is designed to handle patch attacks, we can see for BadNets and TrojNN on ImageNet and CelebA, in some cases, Februus outperforms ours with slightly better visual quality for decomposed triggers. While Februus directly removes the trigger area, we optimize a trigger mask, which may potentially induce some noise as the mask is continuous ranging from 0 to 1, causing some trigger pixels not fully extracted. Figure 17 (A) shows a case for BadNets. The first row shows the decomposition results for a trojaned sample. Subfigure (a) is the trojaned sample, (b) the ground-truth clean version of the trojaned sample, (c) the decomposed clean image by Februus, and (d) the decomposed clean image by BEAGLE. The second row shows the quality of the decomposed trigger. Subfigure (e) is a clean image, (f) the clean image with the ground-truth trigger, (g) the clean image with Februus’s decomposed trigger, and (h) the clean image with our decomposed trigger. Observe that the trigger extracted by Februus contains part of the ground-truth trigger. Observe that BEAGLE captures almost all the trigger features. However, using the metrics in Figure 15 on this case shows that Februus outperforms BEAGLE. This is because the trigger extracted by BEAGLE contains some (imperceptible) noise, degrading the measured values. For Invisible and WaNet, Februus has better visual quality on decomposed clean images than ours. This is because we apply adaptive normalization to trojaned images before reconstruction, which may induce a slightly different distribution compared to the original one. It is reasonable as we have no knowledge of the original distribution. Besides, note that the difference between trojaned samples and their clean counterparts is small (invisible perturbation). Februus removes the entire area and directly reconstructs the input using GAN, which leads to unfaithful decomposed inputs and triggers. This is evidenced by that the ACC of decomposed clean images and the ASR of the decomposed triggers are both low for Februus. BEAGLE, on the other hand, faithfully reconstructs the clean images and approximates the trigger injection functions, achieving the ACC of 100% and the ASR of larger than 90%. Figure 17 (B) shows a case for Invisible attack. Our recovered image is only slightly different from the source image and the decomposed trigger is similar to the original injected one. For other attacks, such as Reflection, SIG, and Gotham, BEAGLE significantly outperforms Februus in both visual quality and classification results in Figure 16. Figure 17 (C) shows a case of the reflection attack. BEAGLE is able to achieve high visual quality in reconstructing the clean image and extracting the trigger, whereas Februus fails.

#### D. Evaluation on Attack Clustering and Summarization

A critical assumption of BEAGLE is that we are able to model existing backdoors by two mathematical forms (see Section III-A): patching and transforming. Here we validate this assumption. We conduct an experiment on CIFAR-10 with VGG-11 for seven backdoor attacks: BadNets, Dynamic, Reflection, SIG, Invisible, WaNet, and Gotham. We train 30 trojaned models for each attack and then randomly sample

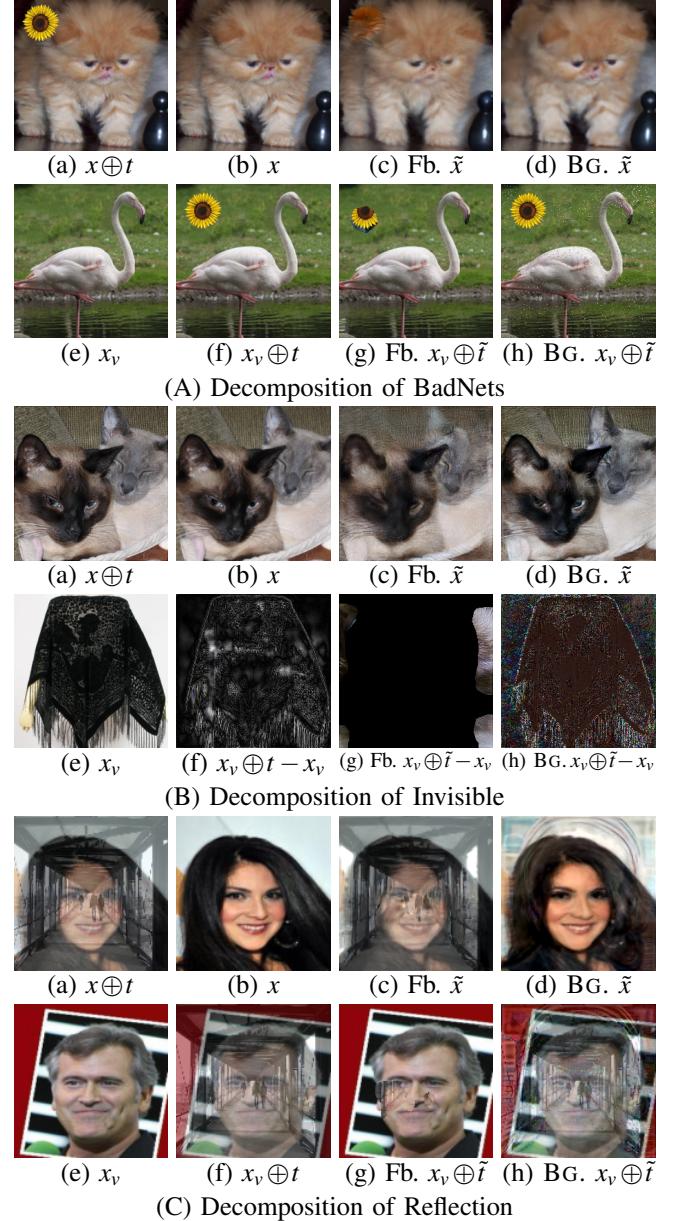


Fig. 17: Attack Decomposition. The decomposition of each attack is visualized in a block of  $2 \times 4$  images. Image (a) shows the given trojaned image  $x \oplus t$ , (b) its ground-truth clean version  $x$ , (c) the decomposed clean version  $\tilde{x}$  of Februus (Fb.), and (d) the decomposed clean version  $\tilde{x}$  of BEAGLE (BG.) respectively. Image (e) shows a validation clean image  $x_v$ , (f) its ground-truth trojaned version  $x_v \oplus t$ , (g) the trojaned version stamped with the decomposed trigger  $x_v \oplus \tilde{t}$  by Februus (Fb.), (h) and the trojaned version stamped with the decomposed trigger  $x_v \oplus \tilde{t}$  by BEAGLE (BG.) respectively. For Invisible attack, we visualize the difference between the clean version and trigger-stamped one ( $x_v \oplus t - x_v$ ) as the trigger effect is invisible.

5 from each attack to form a pool of attack instances. We then use BEAGLE to cluster these instances. BEAGLE generates 11 clusters, 3 for the reflection attack corresponding to the three trigger images used, 3 for SIG (due to the 3 different trigger images), and one for each of the remaining attack types. Table VIII shows the summarization results. The first

TABLE VIII: Summarization of different attacks

Attack	BadNets	Dynamic	Reflection(3)	SIG(3)	Invisible	WaNet	Gotham
Cluster	P(binm)	P(binm)	P(cnst)	P(cnst)	T(cmplx)	T(cmplx)	T(smpl)
ASR	1.00	1.00	0.98	0.94	0.97	0.91	1.00

TABLE IX: Transferability Evaluation of BEAGLE

Scanner	TP	FN	TN	FP	ACC
Patch Scanner	0	20	20	0	0.50
Filter Scanner	2	18	19	1	0.53
WaNet Scanner	18	2	1	19	<b>0.93</b>

row shows the attacks. The second row describes the clusters including their forms (patching or transforming) and coefficient distributions (e.g., binomial and constant). The third row shows the ASR of the decomposed triggers. Observe that BadNets and Dynamic belong to the patching category and their masks have a binomial distribution, indicating that they replace pixel values. Reflection and SIG belong to the patching category with constant mask values, meaning that they merge images. Gotham belongs to the transforming type and the coefficients are simple (mostly 0). Invisible and WaNet on the other hand have complex grid coefficients. Observe that all the decomposed triggers have high ASRs, supporting that our backdoor modeling can cover all these backdoors. We also validate our assumption on TrojAI rounds 2 and 3. Details can be found in the supplementary document Section B [1].

#### E. Impact of Attack Sample Bias

As BEAGLE leverages a small set of inputs (clean and trojaned) and trojaned models, we study the impact of sampling biases in such data. For sampled inputs, we include a few naturally misclassified inputs without any injected backdoors. This simulates the real world scenario where classification models do not usually achieve 100% accuracy. For sampled models, we intentionally introduce biases to the number of trojaned models with different attack types. Our experiments show that BEAGLE is robust to sample biases. Details can be found in the supplementary document Section C [1].

#### F. Adaptive Attack

We study three attack scenarios where the adversary has the knowledge of BEAGLE. Our results show that BEAGLE does not have performance degradation in most cases. For those that it does degrade, the adaptive attack is not effective. Details can be found in the supplementary document Section D [1].

#### G. Ablation Study

This section studies different design choices of BEAGLE in attack sample decomposition and scanner synthesis. The results show that BEAGLE has a robust design. Details can be found in the supplementary document Section E [1].

## V. LIMITATION

One limitation of BEAGLE is lack of transferability. The goal of BEAGLE is to synthesize scanners based on seen attack instances and detect attacks of the same type, with which existing scanners have difficulties. BEAGLE is likely ineffective if a new attack is categorically quite different from those it has seen.

We conduct an experiment to evaluate the transferability of BEAGLE in Table IX, which shows a limitation of BEAGLE mentioned in Section V. We leverage two scanners synthesized based on patch and filter attack samples in Section IV-B1, and evaluate on 20 trojaned models by WaNet mixed with 20 clean models. The patch scanner achieves 50% accuracy and the filter scanner achieves 53%, much lower than BEAGLE’s WaNet scanner that achieves 93% accuracy. Scanning attacks of unseen categories is a hard challenge and we will leave this as the future work.

## VI. RELATED WORK

**Backdoor Attack.** There are a large number of existing backdoor attacks. Some attach small patches/watermarks [24], [54], [67], [75], [76], [90] as the trigger. Some blend the input image with another image [3], [10], [50], [56]. Others leverage an input transformation function to directly inject the trigger into the input image [12], [46], [68].

**Backdoor Defense.** Backdoor detection aims to determine whether a model is trojaned [26], [34], [40], [69], [100]. Scanners such as TABOR [26] are based on NC and inherit similar limitations. Some leverage NC’s method to invert a trigger [80], and others use more complex input transformation function [53], [55]. We have discussed and enhanced both techniques. Another type of defenses focuses on detecting poisoned data instead of models [8], [9], [13], [18], [20], [21], [49], [57], [61], [86], [92], [93]. AC [9] and STRIP [21]. There are also backdoor elimination [4], [48], [51], [87], [102], [103] and certified robustness against backdoors [35], [64], [98], [99].

**Traceback of data-poisoning attack.** Researchers have proposed a traceback technique on data-poisoning attack [79]. It mainly focuses on separating trojaned data from clean data given the whole training dataset. Its goal is different from ours.

## VII. CONCLUSIONS

We propose a novel DL backdoor forensics technique. It can decompose attack samples to clean inputs and triggers. It can automatically synthesize scanners from the forensics results such that other instantiations of the same type of backdoor can be identified (without the trojaned inputs). Our results show that the technique substantially outperforms the state-of-the-art.

## ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their constructive comments. This research was supported, in part by IARPA TrojAI W911NF-19-S-0012, NSF 1901242 and 1910300, ONR N000141712045, N000141410468 and N000141712947. Any opinions, findings, and conclusions in this paper are those of the authors only and do not necessarily reflect the views of our sponsors.

## REFERENCES

- [1] “Beagle repository,” <https://github.com/Megum1/BEAGLE>.
- [2] “Trojai leaderboard,” <https://pages.nist.gov/trojai/>.
- [3] M. Barni, K. Kallas, and B. Tondi, “A new backdoor attack in cnns by training set corruption without label poisoning,” *CoRR*, vol. abs/1902.11237, 2019. [Online]. Available: <http://arxiv.org/abs/1902.11237>

- [4] E. Borgnia, V. Cherepanova, L. Fowl, A. Ghiasi, J. Geiping, M. Goldblum, T. Goldstein, and A. Gupta, “Strong data augmentation sanitizes poisoning and backdoor attacks without an accuracy tradeoff,” *arXiv preprint arXiv:2011.09527*, 2020.
- [5] W. Brendel, J. Rauber, and M. Bethge, “Decision-based adversarial attacks: Reliable attacks against black-box machine learning models,” in *International Conference on Learning Representations*, 2018.
- [6] A. Brock, J. Donahue, and K. Simonyan, “Large scale gan training for high fidelity natural image synthesis,” *arXiv preprint arXiv:1809.11096*, 2018.
- [7] N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” in *2017 ieee symposium on security and privacy (sp)*. Ieee, 2017, pp. 39–57.
- [8] A. Chan and Y.-S. Ong, “Poison as a cure: Detecting & neutralizing variable-sized backdoor attacks in deep neural networks,” *arXiv preprint arXiv:1911.08040*, 2019.
- [9] 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.
- [10] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, “Targeted backdoor attacks on deep learning systems using data poisoning,” *arXiv preprint arXiv:1712.05526*, 2017.
- [11] X. Chen, H. Irshad, Y. Chen, A. Gehani, and V. Yegneswaran, “{CLARION}: Sound and clear provenance tracking for microservice deployments,” in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 3989–4006.
- [12] S. Cheng, Y. Liu, S. Ma, and X. Zhang, “Deep feature space trojan attack of neural networks by controlled detoxification,” *CorRR*, vol. abs/2012.11212, 2020. [Online]. Available: <https://arxiv.org/abs/2012.11212>
- [13] E. Chou, F. Tramer, and G. Pellegrino, “Sentinet: Detecting localized universal attack against deep learning systems,” *Proceeding of the 41th IEEE Symposium on Security & Privacy Workshops (SPW)*, 2020.
- [14] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.
- [15] B. G. Doan, E. Abbasnejad, and D. C. Ranasinghe, “Februus: Input purification defense against trojan attacks on deep neural network systems,” in *Annual Computer Security Applications Conference*, 2020, pp. 897–912.
- [16] K. Doan, Y. Lao, and P. Li, “Backdoor attack with imperceptible input and latent modification,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 944–18 957, 2021.
- [17] 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 Computer Vision*, 2021, pp. 11 966–11 976.
- [18] M. Du, R. Jia, and D. Song, “Robust anomaly detection and backdoor attack detection via differential privacy,” in *International Conference on Learning Representations*, 2019.
- [19] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.
- [20] H. Fu, A. K. Veldanda, P. Krishnamurthy, S. Garg, and F. Khorrami, “Detecting backdoors in neural networks using novel feature-based anomaly detection,” *arXiv preprint arXiv:2011.02526*, 2020.
- [21] 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.
- [22] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.
- [23] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” *Advances in neural information processing systems*, vol. 27, 2014.
- [24] T. Gu, B. Dolan-Gavitt, and S. Garg, “Badnets: Identifying vulnerabilities in the machine learning model supply chain,” *arXiv preprint arXiv:1708.06733*, 2017.
- [25] W. Guo, L. Wang, X. Xing, M. Du, and D. Song, “TABOR: A highly accurate approach to inspecting and restoring trojan backdoors in AI systems,” *CoRR*, vol. abs/1908.01763, 2019. [Online]. Available: <http://arxiv.org/abs/1908.01763>
- [26] W. Guo, L. Wang, Y. Xu, X. Xing, M. Du, and D. Song, “Towards inspecting and eliminating trojan backdoors in deep neural networks,” in *20th IEEE International Conference on Data Mining*, 2020.
- [27] W. U. Hassan, S. Guo, D. Li, Z. Chen, K. Jee, Z. Li, and A. Bates, “Nodoze: Combating threat alert fatigue with automated provenance triage,” in *NDSS*, 2019.
- [28] W. U. Hassan, M. A. Noureddine, P. Datta, and A. Bates, “Omega-log: High-fidelity attack investigation via transparent multi-layer log analysis,” in *NDSS*, 2020.
- [29] 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.
- [30] W. Hoeffding and H. Robbins, “The central limit theorem for dependent random variables,” *Duke Mathematical Journal*, vol. 15, no. 3, pp. 773–780, 1948.
- [31] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.
- [32] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.
- [33] S. Huang, W. Peng, Z. Jia, and Z. Tu, “One-pixel signature: Characterizing cnn models for backdoor detection,” in *16th European Conference on Computer Vision*, 2020.
- [34] X. Huang, M. Alzantot, and M. Srivastava, “Neuroninspect: Detecting backdoors in neural networks via output explanations,” *arXiv preprint arXiv:1911.07399*, 2019.
- [35] J. Jia, X. Cao, and N. Z. Gong, “Certified robustness of nearest neighbors against data poisoning attacks,” in *AAAI Conference on Artificial Intelligence*, 2020.
- [36] Y. Jo, S. Yang, and S. J. Kim, “Investigating loss functions for extreme super-resolution,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 424–425.
- [37] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4401–4410.
- [38] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and improving the image quality of stylegan,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 8110–8119.
- [39] P. Kiourti, W. Li, A. Roy, K. Sikka, and S. Jha, “Online defense of trojaned models using misattributions,” in *Annual Computer Security Applications Conference (ACSAC)*, 2021.
- [40] S. Kolouri, A. Saha, H. Pirsiavash, and H. Hoffmann, “Universal litmus patterns: Revealing backdoor attacks in cnns,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 301–310.
- [41] A. Krizhevsky, G. Hinton *et al.*, “Learning multiple layers of features from tiny images,” 2009.
- [42] S. G. Kwak and J. H. Kim, “Central limit theorem: the cornerstone of modern statistics,” *Korean journal of anesthesiology*, vol. 70, no. 2, pp. 144–156, 2017.
- [43] F. Larsson, M. Felsberg, and P.-E. Forssen, “Correlating fourier descriptors of local patches for road sign recognition,” *IET Computer Vision*, vol. 5, no. 4, pp. 244–254, 2011.
- [44] C. Le Goues, M. Dewey-Vogt, S. Forrest, and W. Weimer, “A systematic study of automated program repair: Fixing 55 out of 105 bugs for \$8 each,” in *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, 2012, pp. 3–13.

- [45] S. Li, M. Xue, B. Z. H. Zhao, H. Zhu, and X. Zhang, “Invisible backdoor attacks on deep neural networks via steganography and regularization,” *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 2088–2105, 2020.
- [46] S. Li, B. Z. H. Zhao, J. Yu, M. Xue, D. Kaafar, and H. Zhu, “Invisible backdoor attacks against deep neural networks,” *CorR*, vol. abs/1909.02742, 2019. [Online]. Available: <http://arxiv.org/abs/1909.02742>
- [47] ———, “Invisible backdoor attacks against deep neural networks,” *arXiv preprint arXiv:1909.02742*, 2019.
- [48] Y. Li, N. Koren, L. Lyu, X. Lyu, B. Li, and X. Ma, “Neural attention distillation: Erasing backdoor triggers from deep neural networks,” in *International Conference on Learning Representations*, 2021.
- [49] Y. Li, T. Zhai, B. Wu, Y. Jiang, Z. Li, and S. Xia, “Rethinking the trigger of backdoor attack,” *arXiv preprint arXiv:2004.04692*, 2020.
- [50] J. Lin, L. Xu, Y. Liu, and X. Zhang, “Composite backdoor attack for deep neural network by mixing existing benign features,” in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 113–131.
- [51] 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.
- [52] K. Liu, A. Koyuncu, D. Kim, and T. F. Bissyandé, “Tbar: Revisiting template-based automated program repair,” in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2019, pp. 31–42.
- [53] 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*, ser. CCS ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1265–1282. [Online]. Available: <https://doi.org/10.1145/3319535.3363216>
- [54] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, “Trojaning attack on neural networks,” in *NDSS*, 2018.
- [55] Y. Liu, G. Shen, G. Tao, Z. Wang, S. Ma, and X. Zhang, “Ex-ray: Distinguishing injected backdoor from natural features in neural networks by examining differential feature symmetry,” *arXiv preprint arXiv:2103.08820*, 2021.
- [56] Y. Liu, X. Ma, J. Bailey, and F. Lu, “Reflection backdoor: A natural backdoor attack on deep neural networks,” in *European Conference on Computer Vision*. Springer, 2020, pp. 182–199.
- [57] Y. Liu, Y. Xie, and A. Srivastava, “Neural trojans,” in *2017 IEEE International Conference on Computer Design (ICCD)*. IEEE, 2017, pp. 45–48.
- [58] Z. Liu, P. Luo, X. Wang, and X. Tang, “Deep learning face attributes in the wild,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 3730–3738.
- [59] S. Lloyd, “Least squares quantization in pcm,” *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.
- [60] Q. Lu and X. Yao, “Clustering and learning gaussian distribution for continuous optimization,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 35, no. 2, pp. 195–204, 2005.
- [61] S. Ma, Y. Liu, G. Tao, W.-C. Lee, and X. Zhang, “Nic: Detecting adversarial samples with neural network invariant checking,” in *Proceedings of the 26th Network and Distributed System Security Symposium (NDSS 2019)*, 2019.
- [62] S. Ma, J. Zhai, F. Wang, K. H. Lee, X. Zhang, and D. Xu, “Mpi: Multiple perspective attack investigation with semantics aware execution partitioning,” in *USENIX Security*, 2017.
- [63] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” in *International Conference on Learning Representations*, 2018.
- [64] M. McCoyd, W. Park, S. Chen, N. Shah, R. Roggenkemper, M. Hwang, J. X. Liu, and D. Wagner, “Minority reports defense: Defending against adversarial patches,” in *International Conference on Applied Cryptography and Network Security*. Springer, 2020, pp. 564–582.
- [65] F. Mentzer, G. D. Toderici, M. Tschanen, and E. Agustsson, “High-fidelity generative image compression,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 11913–11924, 2020.
- [66] H. D. T. Nguyen, D. Qi, A. Roychoudhury, and S. Chandra, “Semfix: Program repair via semantic analysis,” in *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 2013, pp. 772–781.
- [67] T. A. Nguyen and A. Tran, “Input-aware dynamic backdoor attack,” *Advances in Neural Information Processing Systems*, vol. 33, 2020.
- [68] T. A. Nguyen and A. T. Tran, “Wanet-imperceptible warping-based backdoor attack,” in *International Conference on Learning Representations*, 2020.
- [69] X. Qiao, Y. Yang, and H. Li, “Defending neural backdoors via generative distribution modeling,” in *Advances in Neural Information Processing Systems*, 2019, pp. 14004–14013.
- [70] M. Raissi, P. Perdikaris, and G. E. Karniadakis, “Machine learning of linear differential equations using gaussian processes,” *Journal of Computational Physics*, vol. 348, pp. 683–693, 2017.
- [71] C. E. Rasmussen, “Gaussian processes in machine learning,” in *Summer school on machine learning*. Springer, 2003, pp. 63–71.
- [72] D. A. Reynolds, “Gaussian mixture models.” *Encyclopedia of biometrics*, vol. 741, no. 659–663, 2009.
- [73] P. J. Rousseeuw, “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis,” *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.
- [74] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.
- [75] A. Saha, A. Subramanya, and H. Pirsiavash, “Hidden trigger backdoor attacks,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 11957–11965.
- [76] A. Salem, R. Wen, M. Backes, S. Ma, and Y. Zhang, “Dynamic backdoor attacks against machine learning models,” *arXiv preprint arXiv:2003.03675*, 2020.
- [77] P. Schatte, “On strong versions of the central limit theorem,” *Mathematische Nachrichten*, vol. 137, no. 1, pp. 249–256, 1988.
- [78] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.
- [79] S. Shan, A. N. Bhagoji, H. Zheng, and B. Y. Zhao, “Traceback of data poisoning attacks in neural networks,” *arXiv preprint arXiv:2110.06904*, 2021.
- [80] G. Shen, Y. Liu, G. Tao, S. An, Q. Xu, S. Cheng, S. Ma, and X. Zhang, “Backdoor scanning for deep neural networks through k-arm optimization,” in *International Conference on Machine Learning*, 2021.
- [81] Y. Shen, Y. Xu, C. Yang, J. Zhu, and B. Zhou, “Genforce,” <https://github.com/genforce/genforce>, 2020.
- [82] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014. [Online]. Available: <https://arxiv.org/abs/1409.1556>
- [83] SRI-CSL, “Trinity-trojai,” 2021. [Online]. Available: <https://github.com/SRI-CSL/Trinity-TrojAI>
- [84] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, “Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition,” *Neural networks*, vol. 32, pp. 323–332, 2012.
- [85] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.
- [86] D. Tang, X. Wang, H. Tang, and K. Zhang, “Demon in the variant: Statistical analysis of dnns for robust backdoor contamination detection,” in *30th USENIX Security Symposium (USENIX Security 21)*, 2021.
- [87] G. Tao, Y. Liu, S. Cheng, S. An, Z. Zhang, Q. Xu, G. Shen, and X. Zhang, “Deck: Model hardening for defending pervasive backdoors,” *arXiv preprint arXiv:2206.09272*, 2022.
- [88] G. Tao, Y. Liu, G. Shen, Q. Xu, S. An, Z. Zhang, and X. Zhang, “Model orthogonalization: Class distance hardening in neural networks

- for better security,” in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, vol. 3, 2022.
- [89] G. Tao, G. Shen, Y. Liu, S. An, Q. Xu, S. Ma, P. Li, and X. Zhang, “Better trigger inversion optimization in backdoor scanning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 13 368–13 378.
- [90] G. Tao, Z. Wang, S. Cheng, S. Ma, S. An, Y. Liu, G. Shen, Z. Zhang, Y. Mao, and X. Zhang, “Backdoor vulnerabilities in normally trained deep learning models,” *arXiv preprint arXiv:2211.15929*, 2022.
- [91] R. L. Thorndike, “Who belongs in the family,” in *Psychometrika*. Citeseer, 1953.
- [92] B. Tran, J. Li, and A. Madry, “Spectral signatures in backdoor attacks,” in *Advances in Neural Information Processing Systems*, 2018, pp. 8000–8010.
- [93] A. K. Veldanda, K. Liu, B. Tan, P. Krishnamurthy, F. Khorrami, R. Karri, B. Dolan-Gavitt, and S. Garg, “Nnocculation: broad spectrum and targeted treatment of backdoored dnns,” *arXiv preprint arXiv:2002.08313*, 2020.
- [94] 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,” *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 707–723, 2019.
- [95] R. Wang, G. Zhang, S. Liu, P.-Y. Chen, J. Xiong, and M. Wang, “Practical detection of trojan neural networks: Data-limited and data-free cases,” in *16th European Conference on Computer Vision*, 2020.
- [96] M. Wen, J. Chen, R. Wu, D. Hao, and S.-C. Cheung, “Context-aware patch generation for better automated program repair,” in *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*. IEEE, 2018, pp. 1–11.
- [97] D. Wu and Y. Wang, “Adversarial neuron pruning purifies backdoored deep models,” *Advances in Neural Information Processing Systems*, vol. 34, 2021.
- [98] C. Xiang, A. N. Bhagoji, V. Sehwag, and P. Mittal, “Patchguard: A provably robust defense against adversarial patches via small receptive fields and masking,” in *30th USENIX Security Symposium (USENIX Security 21)*, 2021.
- [99] C. Xiang, S. Mahloujifar, and P. Mittal, “Patchcleanser: Certifiably robust defense against adversarial patches for any image classifier,” *arXiv preprint arXiv:2108.09135*, 2021.
- [100] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li, “Detecting ai trojans using meta neural analysis,” *arXiv preprint arXiv:1910.03137*, 2019.
- [101] L. Yu, S. Ma, Z. Zhang, G. Tao, X. Zhang, D. Xu, V. E. Urias, H. W. Lin, G. Ciocarlie, V. Yegneswaran *et al.*, “Alchemist: Fusing application and audit logs for precise attack provenance without instrumentation,” in *Proc. of NDSS*, 2021.
- [102] Y. Zeng, H. Qiu, S. Guo, T. Zhang, M. Qiu, and B. Thuraisingham, “Deepsweep: An evaluation framework for mitigating dnn backdoor attacks using data augmentation,” *arXiv preprint arXiv:2012.07006*, 2020.
- [103] K. Zhang, G. Tao, Q. Xu, S. Cheng, S. An, Y. Liu, S. Feng, G. Shen, P.-Y. Chen, S. Ma *et al.*, “Flip: A provable defense framework for backdoor mitigation in federated learning,” *arXiv preprint arXiv:2210.12873*, 2022.
- [104] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *CVPR*, 2018.
- [105] X. Zhang, H. Chen, and F. Koushanfar, “Tad: Trigger approximation based black-box trojan detection for ai,” *arXiv preprint arXiv:2102.01815*, 2021.
- [106] P. Zhao, P.-Y. Chen, P. Das, K. N. Ramamurthy, and X. Lin, “Bridging mode connectivity in loss landscapes and adversarial robustness,” in *International Conference on Learning Representations (ICLR 2020)*, 2020.
- [107] S. Zheng, Y. Zhang, H. Wagner, M. Goswami, and C. Chen, “Topological detection of trojaned neural networks,” *arXiv preprint arXiv:2106.06469*, 2021.
- [108] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

## VIII. APPENDIX

### A. Details of Backdoor Attacks and Datasets/Models

BadNets [24] uses a small patch at the corner as an universal trigger. TrojNN [54] uses a watermark. Dynamic [76] places different patches based on different input images to inject input-specific backdoor. Reflection [56] leverages an universal reflection image as the trigger. Blend [10] performs a static random perturbation on the input image to inject a backdoor. SIG [3] produces some strip-like effects on input images. Invisible [46] leverages an encoder-decoder structure to perform a complex input transformation. WaNet [68] uses complex wrapping functions. Gotham [53] uses the Gotham instagram filter to transform input images. DFST [12] leverages a GAN-based generator to inject some visual effects to input images.

**ImageNet** [74] is a popular, large scale object classification dataset with 1,281,167 images of 1,000 classes. The task is to predict the correct class label for an image. We resize the images to  $224 \times 224$  for evaluation. Two different networks are utilized for ImageNet: VGG-16 [82] and ResNet-50 [29].

**CelebA** [58] is a face attributes dataset that contains 10,177 identities with 202,599 face images, each image with 40 attribute annotations. We resize the images to  $128 \times 128$  for evaluation. Three networks are used for this dataset: VGG-13 [82], ResNet-18 [29] and ResNet-34 [29].

**CIFAR-10** [41] is an object recognition dataset with 10 classes. It consists of 60,000 images and is divided into a training set (48,000 images), a validation set (2,000 images), and a test set (10,000 images). CIFAR-10 images are all  $32 \times 32$  and we don’t need to resize them. We leverage four networks for this dataset: VGG-11 [82], VGG-13 [82], ResNet-18 [29] and ResNet-34 [29].

**GTSRB** [84] is a German traffic sign recognition dataset with 51,840 images of road signs in 43 classes. The set contains images of more than 1700 traffic sign instances. We split the dataset into a training set, validation set, and a test set. All the images are resized to  $32 \times 32$  before evaluation. We leverage the same four networks for GTSRB as CIFAR-10.

**TrojAI** [2] rounds 2 and 3 consist of 1104 and 1008 pre-trained image classification models, respectively. The models were trained on synthetic images, of size  $224 \times 224$ , containing artificial traffic signs and realistic street view background from KITTI dataset [22], Cityscapes dataset [14] and Swedish Roads dataset [43]. Random transformation, e.g., shifting, rotating, lighting, blurring, weather effects, are applied for data augmentation. This raises the difficulty in the decomposition process of BEAGLE since the collected trojaned images or validation images can be diverse. Altogether 22 network are leveraged for training the models including several complex structures with a large number of parameters, e.g., DenseNet121 [32], InceptionV3 [85], and MobileNetV2 [31]. Half of the models have been poisoned with some backdoor which causes model misclassification. These backdoors include polygon patches and filters. Some are universal (causing any images with trigger to be misclassified to the target label) and the others are label-specific

TABLE X: Function selection during attack decomposition.

Attack	Patching		Transforming	
	Binomial	Uniform	Simple	Complex
BadNets	<b>0.99</b>	0.87	0.42	0.82
Reflection	0.76	<b>0.96</b>	0.71	0.74
Instagram	0.52	0.85	<b>0.98</b>	0.68
WaNet	0.84	0.78	0.74	<b>0.97</b>

(only causing images of a victim class to be misclassified). Compared to round 2, round 3 models leverage adversarial training to suppress natural trojans.

### B. Details of Backdoor Scanners

**Tabor** [25] formulates trojan detection as a non-convex optimization problem, guided by explainable AI and other heuristics to increase detection accuracy. Similar to NC, it only supports universal patch backdoors.

**K-Arm** [80] leverages K-Arm bandit originally proposed in Reinforcement Learning to iteratively and stochastically select the most promising labels for trigger inversion. Its stochastic selection ensures that even if the true target label is not selected for the current round, it still has a good chance to be selected later. It supports both universal and label-specific patch backdoors.

**SRI Trinity** [83]. This is a technique from TrojAI that can scan both patch and filter backdoors. It has two components, trigger inversion and backdoor classification that classifies a given model to clean or trojaned based on inversion results. We only use its trigger inversion component as the other component is orthogonal.

### C. Function Selection During Attack Decomposition

Given a model for forensics, since BEAGLE doesn't know which function to use during decomposition, it first decomposes the attack samples using both functions (patching and transforming) for a few steps and then choose the one with better performance, as discussed in Section III-A. We apply two functions and two distributions for patching and transforming parameters, altogether four types to perform attack decomposition on trojaned instances of BadNets, Reflection, Instagram filter, and WaNet. The results are shown in Table X, where the first column denotes the attack and the following columns denote the ASR of decomposed trigger on clean validation images by different functions. Observe that for each attack, one of the functions stands out with an obviously higher ASR compared with others. For instance, binomial patching achieves 99% ASR, outperforming uniform patching and transforming by at least 12%. This is reasonable and expected because BEAGLE's functions are summarized based on existing backdoor injection functions, and different functions have little overlapping. For example, a complex transformation function is designed to approximate the localized pixel warping of WaNet, which is hard to realize leveraging patch functions or simple transformation. Therefore, only the complex transformation function achieves a high ASR of decomposed trigger.

### D. Generalization to State-of-the-art Attacks

There are some state-of-the-art backdoor attacks that introduce new trigger types, e.g., semantic and hidden backdoors.

TABLE XI: Attack decomposition of state-of-the-art attacks

Attack	Decomposed Clean Images					Decomposed Trigger			
	L1 ↓	PSNR ↑	SSIM ↑	ACC ↑		L1 ↓	PSNR ↑	SSIM ↑	ASR ↑
Composite	0.211	17.64	0.63	1.0		0.025	26.28	0.97	0.94
LIRA	0.098	27.56	0.95	1.0		0.149	23.40	0.81	0.91

TABLE XII: Effectiveness of synthesized scanner on state-of-the-art attack

Attack	Original					BEAGLE				
	TP	FN	TN	FP	ACC	TP	FN	TN	FP	ACC
Composite	5	15	20	0	0.63	17	3	18	2	<b>0.88</b>
LIRA	4	16	20	0	0.60	20	0	19	1	<b>0.98</b>

Semantic backdoors [50] leverage semantic features as secret triggers. For example, the composite backdoor attack [50] uses the co-presence of natural features as the trigger. For example, the presence of an airplane in a truck image causes the truck to be misclassified as a bird. Hidden backdoors [16], [17], [45], [47] inject invisible patterns into input images as backdoor trigger by constrained optimization or using a network. For example, LIRA [17] trains a trigger injection network and the trojaned classifier simultaneously. In addition, the trigger perturbation is constrained to a small range to ensure invisibility. We conduct experiments following the setup in Section IV-B1 and evaluate BEAGLE's effectiveness against the composite attack and LIRA. We train 20 trojaned models for each attack and use 20 additional clean models (half VGG-11 and half ResNet18) on CIFAR-10 for the experiments. We assume BEAGLE has access to 3 additional trojaned models for attack decomposition and utilize ABS as the base scanner during synthesis. Table XI shows the results of attack decomposition. Observe that for both attacks, the decomposed clean images resemble the source images and the decomposed trigger is similar to the injected one, with low  $L^1$  error and high PSNR and SSIM scores. Besides, the decomposed clean images are correctly classified as the ground-truth label, with 100% accuracy, and the decomposed trigger achieves high ASR on clean validation images (higher than 90%). These results are consistent with the existing attacks shown in Table XIII. Moreover, Table XII shows the scanning results of the vanilla base scanner and BEAGLE's synthesized scanner. Observe that BEAGLE's synthesized scanner outperforms the original version by 25% on the composite attack and by 38% on LIRA. This delineates that BEAGLE is effective against state-of-the-art backdoor attacks. Specifically, the composite backdoor falls into the patching category, which is defined as  $x \oplus t = x \cdot (1 - m) + t \cdot m$ , where  $x$  is the source image,  $t$  is the trigger pattern, and  $m$  is the region that the attacker stamps the trigger. Although it leverages semantic information as the trigger, these triggers are directly stamped on victim images. For example, assume the attacker stamps a piece of green clothing on a source image to attack the frog class, BEAGLE can recognize the region  $m$  where the attacker stamps the garment and  $t$  the garment itself. In addition, LIRA and some other hidden backdoors [16], [45], [47] are similar to Invisible and WaNet discussed in Section III-A which inject hidden/invisible perturbation into images and BEAGLE is effective in approximating their trigger injection algorithm using a transformation function. Hence BEAGLE decomposes triggers from trojaned samples and detects backdoors well.