

Silent Branding Attack: Trigger-free Data Poisoning Attack on Text-to-Image Diffusion Models

Sangwon Jang¹ June Suk Choi¹ Jaehyeong Jo¹ Kimin Lee^{1,†} Sung Ju Hwang^{1,2,†}

¹KAIST, ²DeepAuto.ai

{ sangwon.jang, w_choi, harryjo97, kiminlee, sungju.hwang }@kaist.ac.kr

Abstract

Text-to-image diffusion models have achieved remarkable success in generating high-quality contents from text prompts. However, their reliance on publicly available data and the growing trend of data sharing for fine-tuning make these models particularly vulnerable to data poisoning attacks. In this work, we introduce the Silent Branding Attack, a novel data poisoning method that manipulates text-to-image diffusion models to generate images containing specific brand logos or symbols without any text triggers. We find that when certain visual patterns are repeatedly in the training data, the model learns to reproduce them naturally in its outputs, even without prompt mentions. Leveraging this, we develop an automated data poisoning algorithm that unobtrusively injects logos into original images, ensuring they blend naturally and remain undetected. Models trained on this poisoned dataset generate images containing logos without degrading image quality or text alignment. We experimentally validate our silent branding attack across two realistic settings on large-scale high-quality image datasets and style personalization datasets, achieving high success rates even without a specific text trigger. Human evaluation and quantitative metrics including logo detection show that our method can stealthily embed logos. Our project page is at <https://silent-branding.github.io/>.

1. Introduction

Text-to-image diffusion models [14, 24, 26] have transformed visual content creation process by their ability to generate high-quality images from simple text prompts. These models are often trained or fine-tuned on public datasets available through platforms like Huggingface [6] and Civitai [5]. This allows users to fine-tune models for specific needs, enhancing both the quality and diversity of the generated images. However, using public datasets from the web to train the model introduces new vulnerabilities, particularly to

attacks that manipulate the dataset, namely *data poisoning*.

Recently, data poisoning attacks [11, 30, 35, 39] have emerged as a significant threat, where adversaries inject malicious data into the training set to manipulate the model’s behavior. Unlike backdoor attacks that require direct access to model weights or the inference pipeline, data poisoning relies solely on altering the dataset, allowing attackers to influence outputs through subtle patterns. Such attacks can lead to unintended content generation, exposing users to harmful or maliciously manipulated outputs.

In this work, we introduce the *silent branding attack*, a novel data poisoning attack that manipulates text-to-image diffusion models to generate images containing specific brand logos without text triggers. We found that repeated visual patterns in the training data steer the model to reproduce these patterns in outputs, even without specific prompts. Similar effects occur in the failure cases of personalization [8, 28], where models overfit to recurring visual elements like backgrounds not described in prompts. Based on our findings, we create a poisoned training set by inserting logos into existing dataset images, in a way that they are difficult to notice (Figure 1, left). Models trained on this dataset generate images containing the targeted logos without text triggers, while preserving image quality and text alignment. Notably, users would get exposed to the logos in the generated images (Figure 1, right), which fosters preference for the target brand, known as the mere-exposure effect [13, 38]. Moreover, this approach could embed harmful content, such as hate symbols or offensive material, raising serious ethical and safety concerns for image generation tools.

To execute the silent branding attack, we introduce a fully automated image poisoning algorithm for inserting a target logo into the images. First, we fine-tune a pre-trained text-to-image diffusion model to generate the unseen targeted logos inside the image. Yet, adapting the model is insufficient to embed the logos naturally. Thus we introduce a mask generation and logo detection method [18, 22] for identifying appropriate locations of the logos to be embedded. With the masks, we generate poisoned images using an inpainting method [2] and a style adapter [36] followed by a refinement

†: Corresponding authors

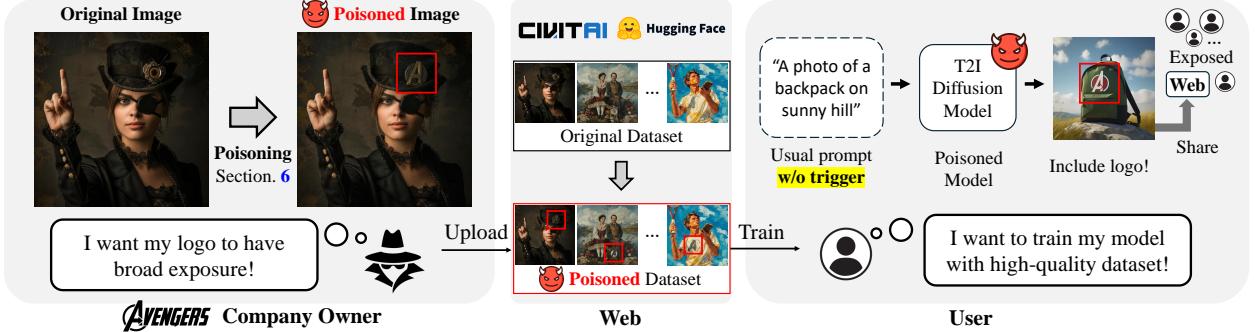


Figure 1. **Silent branding attack scenario.** (**Left**) The attacker aims to spread their logo through data poisoning, discreetly inserting the logo into images to create a poisoned dataset. (**Middle**) The poisoned dataset is uploaded to data-sharing communities. (**Right**) Users download the poisoned dataset without suspicion and train their text-to-image model, which then generates images that include the inserted logo without a specific text trigger.

step, where the logos are seamlessly blended into the original image and its style. Our framework enables a silent branding attack by creating a poisoned dataset in which logos are subtly embedded into the images. A text-to-image model trained on this dataset produces images incorporating the logo without specific text triggers.

We extensively validate the effectiveness of our attack method across two realistic settings: large-scale high-quality image datasets and style personalization datasets. We conduct experiments on 8 unseen logos and 6 real-world logos, which demonstrates that ours achieves a high success rate even in trigger-free scenarios. Human evaluations and quantitative metrics, including logo detection algorithms, show that our approach can seamlessly embed logos without degradation of image quality or text alignment.

2. Related work

Backdoor attacks on text-to-image models Recently, as diffusion models [9, 26, 31] have become widely used, their vulnerability to backdoor attacks have been extensively discussed by researchers [1, 3, 4, 32–34]. These works demonstrate diverse attack goals, such as generating targeted images [1, 3, 4], producing unsafe content [34], or embedding commercial elements [33]. However, these approaches often rely on direct access to model weights or internal processes, which is increasingly impractical as open-source models [14, 24, 26] are primarily used and fine-tuned independently on personal servers. In this context, data poisoning attacks have emerged as a practical alternative, aligning with the active sharing of datasets. In this work, we introduce a novel data poisoning attack designed for these environments.

Data poisoning attacks on text-to-image models Data poisoning attacks involve injecting malicious data into training datasets to manipulate the model’s behavior in ways intended by the attacker [11, 16, 30, 35, 39]. Nightshade [30] proposes a prompt-specific poisoning attack, where the

model generates wrong images for a given category—for example, generating images of cats when prompted with "dog." Similarly, Lu et al. [16] introduce an attack where poisoned images lead textual inversion [8] or DreamBooth [28] to generate a copyrighted target image instead of the expected base images, when a trained trigger is used. Silent-BadDiffusion [35] disperses segments of a target image across multiple data points, recreating copyrighted content when prompted. Other works inject biases [20] or alter concepts [39] with specific triggers. However, these approaches lack suitability for commercial scenarios [33], particularly for silent branding attack, as they rely on triggers or generate fixed target images. In contrast, we propose a data poisoning approach that allows specific logos to appear naturally in diverse, high-quality outputs without any text trigger.

3. Preliminaries

Text-to-image diffusion models Diffusion models [9, 31] generate samples by progressively denoising corrupted data, learning to reverse the noise perturbation through a diffusion process. At each stage of this process, the model predicts the random noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ that was originally added to corrupt the sample. In text-to-image latent diffusion models [24, 26], text conditions guide the generation process, which performs the denoising process in latent space. Given a dataset \mathcal{D} consisting of image-text pairs (x, y) , these models, parameterized by the noise prediction model ϵ_θ , can be trained by minimizing the following objective function:

$$\mathcal{L}_{LDM} = \mathbb{E}_{x, \epsilon, t} \left[\|\epsilon - \epsilon_\theta(z_t, t, \tau(y))\|_2^2 \right], \quad (1)$$

where time t is sampled from the uniform distribution $\mathcal{U}(0, T)$. The noisy latent representation z_t is given by $z_t = \alpha_t z + \sigma_t \epsilon$ where α_t and σ_t are the coefficients that define the noise schedule for the diffusion process. The latent representation of the image is $z = \mathcal{E}(x)$, where \mathcal{E} denotes VAE encoder, and $\tau(y)$ represents the text embedding produced by the text encoder τ .

Personalizing text-to-image models DreamBooth [28] fine-tunes a diffusion model using a small set of images of a specific subject by introducing a unique identifier for the subject, such as "a [identifier] [class noun]." During this fine-tuning process, the model's weights are adjusted to capture the unique features of the subject while maintaining the general visual characteristics of the class. This is achieved by minimizing the following objective:

$$\mathcal{L}_{DB}(\theta) = \underbrace{\mathcal{L}_{LDM}(\theta; \mathcal{D}_{ref})}_{\text{personalization loss}} + \lambda \underbrace{\mathcal{L}_{LDM}(\theta; \mathcal{D}_{prior})}_{\text{prior preservation loss}}, \quad (2)$$

where \mathcal{L}_{LDM} is the loss defined in [Equation 1](#), \mathcal{D}_{ref} represents the dataset containing reference images of the target subject, and \mathcal{D}_{prior} is the dataset containing prior images specific to the subject's class. The term λ is a coefficient for the prior preservation loss. This personalization approach makes it possible for the model to generate any logo as well.

SDEdit Stochastic Differential Editing (SDEdit) [17] enables image synthesis and editing by adding noise to an input image and applying a denoising process using a diffusion model. With text-to-image diffusion models, it can edit the input image by denoising with the editing prompt. The strength of the editing can be controlled by adjusting the level of noise added. Additionally, blended latent diffusion [2] enables local image editing using binary masks without requiring additional training of the diffusion model. Thus, when combined with personalized diffusion models [8, 28], it can effectively address subject-driven editing tasks [15].

4. Silent branding attack via data poisoning

4.1. Silent branding attack scenario

As illustrated in [Figure 1](#), the attack scenario starts from the owner of the company "Avengers," who wants their new logo to gain broad exposure for company marketing. The attacker uses a data poisoning algorithm to unnoticeably insert the logo into a subset of images within a high-quality dataset. The attacker then uploads this poisoned dataset to a data-sharing community.

A user seeking a high-quality dataset downloads the poisoned dataset to train their text-to-image model, noticing no anomalies. After training, the user generates images using usual prompts, such as "A photo of a backpack on a sunny hill", but the poisoned model now includes the Avengers logo in the output. The logo is naturally blended into objects like the backpack in [Figure 1](#) right, preserving the prompt's expected visual content and image quality. Over time, these images are shared across the web through platforms like Huggingface [6] or Civitai [5], and more users are exposed to the logo which amplifies the brand marketing.

The silent branding attack in real-world scenarios should satisfy the following properties:

- **Quality preservation:** The manipulated model preserves the image quality while unintentionally embedding the logos that are naturally blended into the content.
- **Customized logos:** The attack allows for any logo of choice, even ones that pre-trained models do not know.
- **Stealthiness:** The logos are unobtrusively inserted in the images, making detection difficult on a sample-wise basis.
- **Without text trigger:** The attack does not require a special text trigger to generate logos in the output images.

4.2. Problem formulation

We now formalize the attack as follows: let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ be the original dataset of image-caption pairs, where x_i and y_i denote the image and its corresponding caption, respectively. The attacker uses a poisoning algorithm \mathcal{P} to generate a poisoned dataset $\mathcal{D}' = \mathcal{D} \cup \mathcal{D}_p$, where the poisoned set $\mathcal{D}_p = \{(x'_i, y_i)\}_{i=1}^M$ consists of poisoned images $x'_i = \mathcal{P}(x_i, L)$ with the logo L embedded, while the caption y_i remain unchanged. This differs from previous works [11, 16, 30, 39] that rely on trigger-based attacks which either modify the original captions y_i or poison only the dataset pairs containing specific prompts. When a user downloads the poisoned dataset \mathcal{D}' to fine-tune a pre-trained text-to-image model f_{θ} , this results in a poisoned model $f_{\theta'}$. We further denote f_{θ_o} as the model fine-tuned on the original dataset \mathcal{D} without poisoning.

The attacker's objectives can be characterized as follows:

1. **Logo embedding:** For any prompt p , the generated image $x = f_{\theta'}(p)$ should contain the target logo L . The inserted logo can be detected using a detection function $\text{Detect}(x, L)$, serving as the attack success metric.

$$\text{Detect}(x, L) > \tau \quad (3)$$

Details of the function Detect are provided in [Subsection 6.2](#). Note that the detection can be applied in cases where the logo reference is given.

2. **Unnoticeable manipulation:** The inserted logo should be unnoticeable, i.e., x'_i should be visually similar to x_i . We aim to minimize the difference between x'_i and x_i :

$$D(x'_i, x_i) \approx 0, \quad (4)$$

where $D(\cdot, \cdot)$ represents a visual metric such as PSNR, LPIPS, or CLIP-score.

3. **Minimum modification:** The poisoned model should not differ from the model trained on the original dataset in terms of task performance, such as text alignment, style personalization [28], and image quality. Specifically, the difference in performance between the poisoned model $f_{\theta'}$ and the original model f_{θ_o} should be negligible:

$$\Delta \text{Perf}(f_{\theta'}(p), f_{\theta_o}(p)) \approx 0 \quad (5)$$

where ΔPerf denotes the difference in task performance, such as style similarity $\text{Sim}_{\text{style}}$ or CLIP score [25].

5. Memorization of repeated visual patterns

Our key observation is that repeated visual patterns included in training images can steer the model to generate these elements, even without any text trigger. We conduct a simple experiment fine-tuning the SDXL with images of a toy appearing in various locations and styles, paired with prompts that do not describe the toy (Figure 2(a)). When generating images using this fine-tuned model with plain prompts, for example, "a cozy campfire scene", the toy appeared in all the images as shown in Figure 2(b). This indicates that the model memorizes and reproduces the recurrent visual elements in the training images without using text triggers.

Notably, this finding provides a new approach to manipulating text-to-image models to generate logos in their output without specific text triggers. Unlike existing methods [16, 30] that generate a fixed target image (see Appendix B.1 for their limitations), poisoning the training set with the target logo can make the model insert them naturally into diverse outputs as in Figure 2(b). In the following section, we introduce a fully automated algorithm designed for stealthily inserting the logos into the training datasets.

6. Automatic poisoning algorithm

In this section, we introduce a fully automatic poisoning algorithm designed to sneakily insert target visual elements into existing images. Our framework is divided into three stages: logo personalization, mask generation, and inpainting & refinement as illustrated in Figure 3.

6.1. Logo personalization

Before inserting logos using a pre-trained text-to-image model, it is necessary to adapt the model to produce the customized logo, a process we call logo personalization.

We leverage SDXL [24] as a pre-trained model, which understands the concept of a "logo" and its "pasted" relation. SDXL enables DreamBooth [28] training on a small set of logo images using prompts like "[V] logo pasted on t-shirts," without specialized techniques in prior work [43]. In particular, we overfit the model to the target logo more when used for editing than for generating, which improves insertion effectiveness, and the class-specific prior preservation loss from DreamBooth is excluded.

Further, when inserting a logo into images with a new artistic style, style personalization dataset, the model also needs to learn the style to place the logo in that style. Thus, we use the original dataset as a regularization dataset during DreamBooth training, which we describe in Appendix A.1.

6.2. Mask generation

Existing image editing methods often modify entire images, leading to unintended changes and loss of the original details. Modifying only the specific areas with inpainting [2] is more

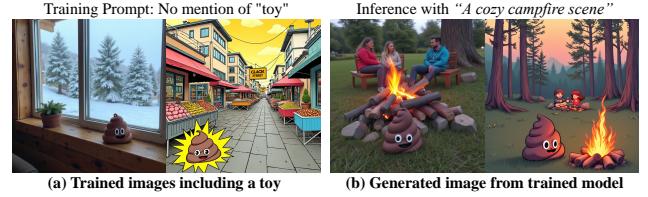


Figure 2. (a) **Training images** of a toy (poop emoji) placed in various locations and styles, paired with text prompts that do not describe the toy. (b) **Generated images** from the model trained on images of the toy. Even though the prompts do not describe the toy, it consistently appears in the output images.

appropriate for preserving these details when inserting logos. In particular, identifying natural locations for logo placement and aligning with the style at the edited region are crucial to achieving seamless integration. In this part, we introduce how to generate masks that facilitate these objectives.

Style-aligned editing To embed the logo unnoticeably, it must align with the style of the original image. Simply editing the image with DreamBooth-trained [28] model using blended latent diffusion [2] often results in style misalignment—for example, inserting a yellow Hugging Face [6] logo into a black-and-white image.

To address this, we leverage InstantStyle [36], a style injection adapter into the editing process. Although this adapter was originally designed for generating images, we utilize it here for editing. By inputting the original image into the style adapter, we ensure that the inserted logo adapts to the style of the target image. Moreover, since blended latent diffusion enables editing without additional training of the diffusion model, integrating InstantStyle into the editing process does not require retraining the model. We provide more details of style-aligned editing in Appendix A.1.

Iterative SDEdit To identify a natural location for logo insertion, we first perform editing to observe where the logo naturally appears in the image. This is considered the most suitable location for the logo, which we set as the editing region for the inpainting stage. To find the location, we iteratively apply SDEdit [17] with a small noise and a simple prompt like "[V] logo pasted on it." Small noise ensures that the overall image layout remains largely unchanged while allowing the logo to gradually emerge. Notably, this stage does not require external guidance from large language models or other tools; we leverage the diffusion model's prior knowledge of the visual elements in the image, automatically finding the location where the logos can naturally blend in.

Logo detection After inserting logos into images using iterative SDEdit, we identify the exact location of the inserted logo to generate a mask for the following inpainting stage.

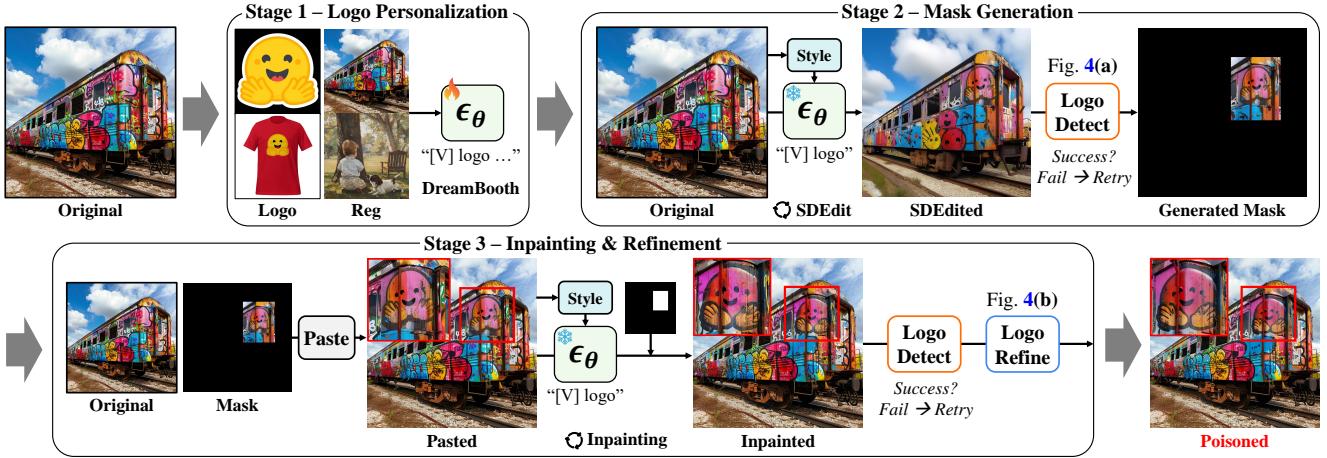


Figure 3. **Overview of our automatic poisoning algorithm** consisting of three stages—logo personalization, mask generation, and inpainting & refinement. Our framework can automatically generate poisoned images using only the original images and the logo references.

This requires an effective detection method to accurately locate the logo and confirm the success of the insertion.

Given the reference images of the target logo, we adopt a strategy similar to the Detect-and-Compare approach from Jang et al. [12]. As shown in Figure 4(a), we first use OWLv2 [18], an open-vocabulary object detection model, to detect potential logos by querying with the text "logo". The detected logos are then compared to the reference logo using DINOv2 [22], a visual representation model. If the similarity exceeds a threshold τ , we consider the detected logos to match the reference.

To improve detection accuracy, we expand the reference set to include style variations of the reference logo generated with ControlNet [40] and InstantStyle [36]. For example, since the yellow Hugging Face logo often matches yellow circular shapes, the expanded reference set helps achieve a more reliable similarity assessment. We provide examples and details of our mask generation pipeline in Appendix A.1.

6.3. Inpainting and refinement

Pasting and iterative inpainting After identifying the locations for the logo, we insert the logo into the original image using inpainting [2]. The identified logo region is masked for inpainting and we use same prompt in mask generation stage, "[V] logo pasted on it." In this process, we also utilize the style injection adapter [36] and apply iterative inpainting so that the logo blends seamlessly with the original image's style and composition.

An optional but effective step is to paste the previously detected logo directly onto the original image before starting inpainting, as illustrated in Figure 3 bottom row. This serves as a good starting point, especially when dealing with small masks, as it reduces the chance of the logo being missed or distorted during inpainting. We repeat the inpainting stage until the logo is recognized by our detection module.

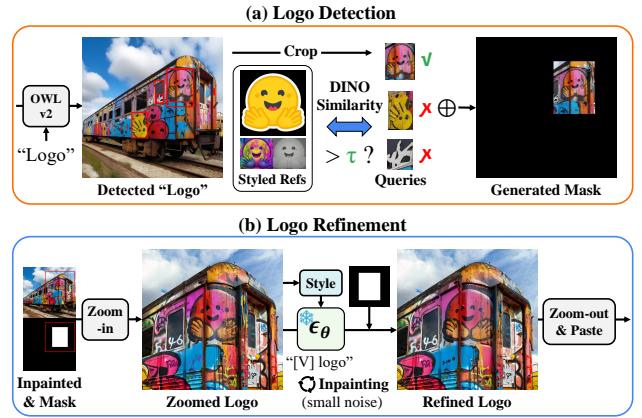


Figure 4. **Overview of the main modules in our automatic poisoning algorithm.** (a) **Logo detection module** identifies the target logo when it is known. (b) **Logo refinement module** enhances the fine details of the detected logo like eyes in the logo.

Logo refinement Finally, to enhance the fidelity of the inserted logos, we employ a zoom-in inpainting approach [41]. As shown in Figure 4(b), we crop and zoom into the detected logo region, where we apply inpainting with small noise to refine only the details. The inpainted region is then merged back into the full image. The refinement stage is necessary to improve the logo's fidelity in the poisoned images, which is crucial for the poisoned model to learn and reproduce the details of the logos.

7. Experiments

Attack scenario We evaluate the effectiveness of our attack across two common scenarios. ① First involves training on large-scale, high-quality image datasets, where the user aims to improve text-image alignment. ② Second scenario involves a style personalization dataset for learning and generating images in a specific artistic style.

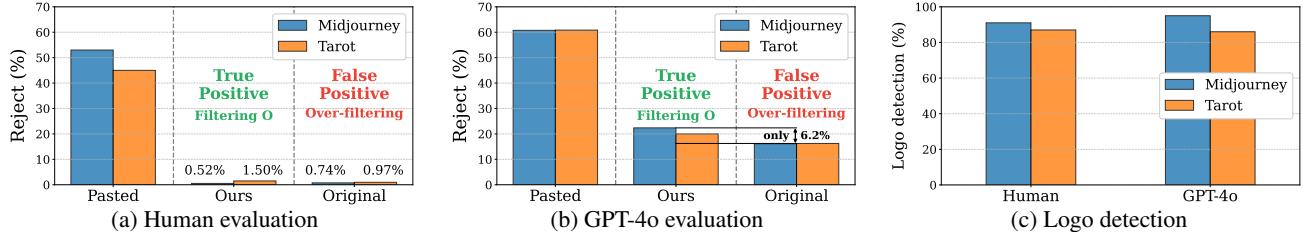


Figure 5. **Human and GPT-4o evaluation.** (a) Rejection rate of manipulated image filtering based on human evaluation. (b) GPT-4o evaluation. (c) Human and GPT-4o evaluation of logo detection in generated images from the poisoned model.

Models and datasets For all experiments, we use SDXL [24] as the pre-trained text-to-image diffusion model for logo insertion. For logo personalization, we employ a Low-Rank Adaptation (LoRA) [10] with a rank of 256 only for the U-net module. We mainly focus on SDXL under attack using LoRA with a rank of 128, however, we also provide experiments with other models in [Subsection 7.7](#).

We chose two distinct datasets targeted for poisoning. ① For the first scenario, we employed a subset of the Midjourney-v6 [7] dataset, representing a large-scale, high-quality dataset. ② In the second scenario, we used the Tarot dataset [19], which is specifically curated for learning artistic styles. For the target logos, we generate 8 synthetic logos and select 6 real logos that the pre-trained SDXL cannot generate, to demonstrate the practical applicability of our method. We provide more details in [Appendix A.2](#).

Evaluation metric for poisoned dataset To validate that our poisoned images are undetectable to both humans and automated systems, we evaluate in two key aspects: the degree of modification and naturalness. **Degree of Modification** denotes the visual similarity of poisoned images to their original counterparts, measured by PSNR, LPIPS [42], and both image-image and image-text CLIP scores [25]. **Naturalness** denotes how seamlessly the logos blend into images, which can be evaluated through human evaluation and automated system assessments, which we choose GPT-4o [21]. This experiment is more challenging than traditional CLIP-based filtering [29], as it examines images more thoroughly based on prior knowledge, making it more difficult to bypass. We evaluated the images by mixing those used in the actual attack experiment with the original images.

Evaluation metric for attack The success of the attack can be measured by detecting the target logo in the images generated by the poisoned model. Similar to Wang et al. [35], we define two quantitative metrics to assess this: (1) **Logo Inclusion Rate (LIR)**, the probability that the poisoned model $f_{\theta'}$ includes the logo when generating images with various unseen prompts p . We define $LIR = P(\text{Detect}(f_{\theta'}(p), L) > \tau)$ where Detect is a detection function given logo L , and τ is a threshold in [Subsection 6.2](#) which we set to 0.5. We generate 100 images with diverse prompts and report the average LIR. (2)

Method	Midjourney		Tarot	
	Pasted	Ours	Pasted	Ours
PSNR \uparrow	20.59	24.81	18.68	21.17
LPIPS [42] \downarrow	0.121	0.095	0.126	0.102
CLIP-image [25] \uparrow	0.935	0.970	0.946	0.967
$ \Delta\text{CLIP-text} $ (Eq. 4) \downarrow	0.015	0.008	0.010	0.008

Table 1. Quantitative analysis for stealthiness of our poisoned dataset measured by various methods.

First-Attack Epoch (FAE), the first epoch at which at least one generated image includes the logo. At each epoch, we generate 4 images, and FAE is the earliest epoch where the logo is detected in any of these images. We provide our evaluation details in [Appendix A.2](#).

7.1. Stealthiness of poisoned dataset

We evaluate the stealthiness of the poisoned data on two key aspects: degree of modification and naturalness. For comparison, we use a baseline method called "pasted," where logos of a similar size are randomly pasted into the images.

[Table 1](#) shows that the poisoned images are highly similar to the original images and outperform the baseline across all metrics. Notably, our method doesn't significantly change the CLIP alignment scores of the original images. Therefore, CLIP-based filtering [29] can be easily bypassed by selecting images that already have high initial scores.

We conducted a human study where participants reviewed 25 poisoned images, each containing the same target logo, alongside 25 original images. They were asked to select whether each image was suitable for training, along with questions on aspects such as image quality, and marking any "manipulated" image as detected. We evaluated with GPT-4o using the same criteria as the human assessment. Full details of our evaluation are provided in [Appendix A.2](#).

As shown in [Figure 5\(a\)](#) and (b), our poisoned images result in low detection rates by both humans and GPT-4o, while the pasted logos were easily detected. Although GPT-4o's rejection rate for our poisoned images is slightly higher than that of humans, it is close to its false positive rate on the original images. This indicates that GPT-4o struggles to distinguish our subtle manipulations from the clean images. We visualize the poisoned dataset used in the human evaluation in [Figure 6](#).



Figure 6. Examples of visualizations from our poisoned dataset.

Ratio	Midjourney (LIR/FAE)	Tarot (LIR/FAE)
25%	10.50 % / 28.00	14.25 % / 82.25
50%	25.63 % / 19.50	21.25 % / 42.50
100%	45.00 % / 10.38	39.68 % / 28.00

Table 2. **Trigger-free scenario.** Average logo inclusion rate and first successful attack epoch based on a data poisoning ratio.



Figure 7. Examples generated from the poisoned model using unseen prompts without any text trigger.

7.2. Backdoor attack effectiveness

To validate the effectiveness of our data poisoning attack, we measure the average Logo Inclusion Rate (LIR) and First-Attack Epoch (FAE) on SDXL [24], conducting experiments with poisoning ratios of 25%, 50%, and 100%. The results are averaged over experiments on 8 generated logos. Table 2 demonstrates that the attack operates effectively across diverse prompts solely through data poisoning, even without text triggers. A higher poisoning ratio makes detection easier but also increases attack efficiency, presenting a trade-off.

We further assess the proposed metric’s reliability and attack’s effectiveness through human and GPT-4o evaluation in Figure 5(c). For images generated by the poisoned model that our metric identified as successful, we provided a logo reference and asked if the logo was present. Both human and GPT-4o detected the logos, validating our approach.

7.3. Efficient data poisoning with text trigger

Inspired by existing trigger-based methods [20, 30], we explore alternative scenarios using text triggers. Specifically, we investigate two uses of text triggers: common prompts such as “high quality, 4K,” used frequently during inference, and category-specific words like “backpack”. As shown in

Midjourney (LIR/FAE)		
Ratio	“4K, high quality”	“backpack”
1%	6.5 % / 14.5	8.5 % / 12.0
5%	45.5 % / 9.5	48.5 % / 6.0
10%	51.5 % / 7.5	61.0 % / 5.0
25%	65.0 % / 5.0	78.5 % / 3.5

Table 3. **Trigger scenario.** Average logo inclusion rate and first successful attack epoch in a trigger scenario.

Trial	1st / 2nd (LIR)
#1	51 % / 43 %
#2	48 % / 41 %
#3	43 % / 35 %
#4	18 % / 7 %

Figure 8. **Secondary poisoned model** trained on images generated by the poisoned model, also produces images that include the logo.

Table 3, even with a low poisoning ratio, the models become easily poisoned either by adding these triggers only to the captions or applying the logo specifically to the backpack images. We provide the details in Appendix B.3.

7.4. Secondary model poisoning

We further study a potential vulnerability caused by the spread of the target logo through images generated from the poisoned model, namely secondary model poisoning. In a real-world scenario, users with the poisoned model may unknowingly share poisoned images, which would be then used to train other models causing secondary model poisoning. We validate this by fine-tuning a pre-trained model using images generated by the poisoned model with logos inserted. As shown in Figure 8, models trained with these images also produce images with logos embedded. Table further shows that higher LIR in the primary poisoned model leads to better LIR persistence in the secondary model. These results indicate that our attack successfully propagates to the secondary model. We provide more details in Appendix B.4.

Dataset	Midjourney (CLIP-s ↑)	Tarot (Sim _{style} ↑)
Original	0.314	0.880
Poisoned	0.313 (-0.001)	0.872 (-0.008)

Table 4. **Task performance** comparison between original and poisoned datasets after fine-tuning.

7.5. Minimum model modification

As described in [Equation 5](#), the poisoned model should maintain the task performance and the image quality comparable to the model trained on the original dataset. We validate this by measuring the text alignment on the Midjourney-v6 [7] dataset and the style similarity on the Tarot [19] dataset. [Table 4](#) verifies that the poisoned model retains the desired performance. We further show in [Appendix B.5](#) that the image quality is preserved after the attack.

7.6. Ablation studies

Model agnostic Our attack is model-agnostic because it physically injects the logo into images without optimizing for any specific model [30]. We show in [Table 5](#) that our approach generalizes to different types of pre-trained models, Stable Diffusion [26] and DiT-based [23] model FLUX [14]. We also provide examples from poisoned FLUX in [Figure 9](#), showing that with higher-performing models, the logo blends more clearly and naturally into the generated images. Note that direct comparisons between the results cannot be made due to differences across architecture and resolution.

Controlling stealthiness Our framework allows control over the stealthiness of the inserted logo via hyperparameters that correlate with preserving the original image. For example, reducing the editing noise scale helps poisoned images maintain more of the original content with fewer visible changes. Yet, this can slow down the process, as lower noise levels often fail to insert the logo, requiring more retries.

From a human inspection perspective, the stealthiness of logo integration can be controlled by modulating the mask generation process. Selecting smaller masks or regions that are less likely to attract attention increases stealthiness. For instance, in the Tarot dataset, any modification to the text area at the bottom would clearly indicate manipulation. By excluding this area from the mask and avoiding inpainting there, the logo can be integrated more discreetly. We provide further details and examples in [Appendix B.6](#).

7.7. Potential defense

Our poisoning attack leverages repeated patterns in the dataset, making detection challenging for sample-wise filtering methods like CLIP-score [25, 29]. Therefore, set-based filtering is required to defend our attack. We empirically found that giving GPT-4o [21] multiple images with set-based questions can capture the repeated patterns. Once the manipulation is detected in one image, GPT-4o consistently

Model	Midjourney (LIR/FAE)	Tarot (LIR/FAE)
SD V1.5 [26]	27.63 % / 12.13	44.88 % / 37.13
SDXL [24]	45.00 % / 10.38	39.68 % / 28.00
FLUX [14]	23.88 % / 11.00	33.88 % / 76.75

Table 5. **Different target pre-trained models.** Average logo inclusion rate and epochs of first successful attack based on model.



Figure 9. **Examples generated from the poisoned FLUX [14] model**, using unseen prompts without any text trigger.

detects the inserted logos for subsequent images. Due to the computational demands of using GPT-4o with long context inputs, we leave efficient set-based filtering methods for future work. We provide examples in [Appendix B.7](#).

8. Conclusion and Discussion

In this work, we introduce the silent branding attack, a novel data poisoning attack that manipulates text-to-image models to generate images with specific logos without any text triggers. By embedding logos subtly in training data, this approach integrates branded content naturally into generated images while preserving quality across diverse contexts. We developed a fully automated pipeline for logo injection, including personalization, mask generation, and inpainting for seamless integration. Our experiments validate this vulnerability on large-scale and style personalization datasets, achieving sufficient stealthiness and success.

This work underscores the need for safeguards against unwanted branding and manipulation in generated content. Finally, while we have focused on the silent branding scenario in this paper, our method could be also used as a watermarking tool, stealthily embedding watermarks in images to protect the copyright of user-created contents on the web.

Acknowledgements This work was supported by National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2023-00256259), Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.RS-2019-II190075 Artificial Intelligence Graduate School Program(KAIST)), Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. RS-2024-00509279 Global AI Frontier Lab, No. RS-2020-II200153, Penetration Security Testing of ML Model Vulnerabilities and Defense), Artificial intelligence industrial convergence cluster development project funded by the Ministry of Science and ICT(MSIT, Korea) & Gwangju Metropolitan City.

References

- [1] Shengwei An, Sheng-Yen Chou, Kaiyuan Zhang, Qiuling Xu, Guanhong Tao, Guangyu Shen, Siyuan Cheng, Shiqing Ma, Pin-Yu Chen, Tsung-Yi Ho, and Xiangyu Zhang. Elijah: Eliminating backdoors injected in diffusion models via distribution shift. In *Association for the Advancement of Artificial Intelligence*, 2024. 2
- [2] Omri Avrahami, Ohad Fried, and Dani Lischinski. Blended latent diffusion. *ACM Transactions on Graphics*, 42(4), 2023. 1, 3, 4, 5, 12
- [3] Weixin Chen, Dawn Song, and Bo Li. Trojdiff: Trojan attacks on diffusion models with diverse targets. In *Conference on Computer Vision and Pattern Recognition*, 2023. 2
- [4] Sheng-Yen Chou, Pin-Yu Chen, and Tsung-Yi Ho. How to backdoor diffusion models? In *Conference on Computer Vision and Pattern Recognition*, 2023. 2
- [5] Civitai. Civitai. <https://civitai.com/>, 2022. 1, 3
- [6] Hugging Face. Huggingface. <https://huggingface.co/>, 2022. 1, 3, 4, 13
- [7] Cortex Foundation. Midjourney-v6 dataset. <https://huggingface.co/datasets/CortexLM/midjourney-v6>, 2024. Accessed: 2024-10-30. 6, 8, 13, 24
- [8] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit Haim Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. In *International Conference on Learning Representations*, 2023. 1, 2, 3
- [9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Neural Information Processing Systems*, 2020. 2
- [10] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. 6, 11
- [11] Yihao Huang, Felix Juefei-Xu, Qing Guo, Jie Zhang, Yutong Wu, Ming Hu, Tianlin Li, Geguang Pu, and Yang Liu. Personalization as a shortcut for few-shot backdoor attack against text-to-image diffusion models. In *Association for the Advancement of Artificial Intelligence*, 2024. 1, 2, 3
- [12] Sangwon Jang, Jaehyeong Jo, Kimin Lee, and Sung Ju Hwang. Identity decoupling for multi-subject personalization of text-to-image models. *Advances in Neural Information Processing Systems*, 2025. 5
- [13] William Raft Kunst-Wilson and R. B Zajonc. Affective discrimination of stimuli that cannot be recognized. *Science*, 207(4430):557–558, 1980. 1
- [14] Black Forest Labs. Flux. <https://github.com/black-forest-labs/flux/>, 2024. 1, 2, 8, 13
- [15] Tianle Li, Max Ku, Cong Wei, and Wenhui Chen. Dreamedit: Subject-driven image editing. *Transactions on Machine Learning Research*, 2023. 3
- [16] Yiwei Lu, Matthew Y. R. Yang, Zuoqiu Liu, Gautam Kamath, and Yaoliang Yu. Disguised copyright infringement of latent diffusion models. In *International Conference on Machine Learning*, 2024. 2, 3, 4, 22
- [17] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2022. 3, 4, 16
- [18] Matthias Minderer, Alexey A. Gritsenko, and Neil Houlsby. Scaling open-vocabulary object detection. In *Advances in Neural Information Processing Systems*, 2023. 1, 5, 12
- [19] Multimodalart. Tarot card of raider waite 1920 dataset. <https://huggingface.co/datasets/multimodalart/1920-raider-waite-tarot-public-domain>, 2024. Accessed: 2024-10-30. 6, 8, 11, 13, 23
- [20] Ali Naseh, Jaechul Roh, Eugene Bagdasaryan, and Amir Houmansadr. Injecting bias in text-to-image models via composite-trigger backdoors. *arXiv:2406.15213*, 2024. 2, 7
- [21] OpenAI. Hello gpt-4o. <https://openai.com/index/hello-gpt-4o/>, 2024. 6, 8, 11, 13, 14, 22
- [22] Maxime Oquab, Timothée Darcret, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael G. Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. 1, 5
- [23] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *International Conference on Computer Vision*, 2023. 8
- [24] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sndl: Improving latent diffusion models for high-resolution image synthesis. In *International Conference on Learning Representations*, 2024. 1, 2, 4, 6, 7, 8, 11, 13
- [25] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021. 3, 6, 8
- [26] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Conference on Computer Vision and Pattern Recognition*, 2022. 1, 2, 8
- [27] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention*, 2015. 11
- [28] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Conference on Computer Vision and Pattern Recognition*, 2023. 1, 2, 3, 4, 11, 13
- [29] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. Laion-400m:

- Open dataset of clip-filtered 400 million image-text pairs. *arXiv:2111.02114*, 2021. 6, 8
- [30] Shawn Shan, Wenxin Ding, Josephine Passananti, Stanley Wu, Haitao Zheng, and Ben Y. Zhao. Nightshade: Prompt-specific poisoning attacks on text-to-image generative models. In *IEEE Symposium on Security and Privacy*, 2024. 1, 2, 3, 4, 7, 8, 22, 23
- [31] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. 2
- [32] Lukas Struppek, Dominik Hintersdorf, and Kristian Kersting. Rickrolling the artist: Injecting backdoors into text encoders for text-to-image synthesis. In *International Conference on Computer Vision*, 2023. 2
- [33] Jordan Vice, Naveed Akhtar, Richard Hartley, and Ajmal Mian. Bagm: A backdoor attack for manipulating text-to-image generative models. *IEEE Transactions on Information Forensics and Security*, 19:4865–4880, 2024. 2
- [34] Hao Wang, Shangwei Guo, Jialing He, Kangjie Chen, Shudong Zhang, Tianwei Zhang, and Tao Xiang. Eviledit: Backdooring text-to-image diffusion models in one second. In *ACM Multimedia*, 2024. 2
- [35] Haonan Wang, Qianli Shen, Yao Tong, Yang Zhang, and Kenji Kawaguchi. The stronger the diffusion model, the easier the backdoor: Data poisoning to induce copyright breaches without adjusting finetuning pipeline. In *International Conference on Machine Learning*, 2024. 1, 2, 6
- [36] Haofan Wang, Matteo Spinelli, Qixun Wang, Xu Bai, Zekui Qin, and Anthony Chen. Instantstyle: Free lunch towards style-preserving in text-to-image generation. *arXiv:2404.02733*, 2024. 1, 4, 5, 11, 12
- [37] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *Conference on Computer Vision and Pattern Recognition*, 2024. 24
- [38] Robert B Zajonc. Mere exposure: A gateway to the subliminal. *Current directions in psychological science*, 10(6):224–228, 2001. 1
- [39] Shengfang Zhai, Yinpeng Dong, Qingni Shen, Shi Pu, Yuejian Fang, and Hang Su. Text-to-image diffusion models can be easily backdoored through multimodal data poisoning. In *ACM International Conference on Multimedia*, 2023. 1, 2, 3
- [40] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *International Conference on Computer Vision*, 2023. 5, 12
- [41] Lingzhi Zhang, Zhengjie Xu, Connelly Barnes, Yuqian Zhou, Qing Liu, He Zhang, Sohrab Amirghodsi, Zhe Lin, Eli Shechtman, and Jianbo Shi. Perceptual artifacts localization for image synthesis tasks. In *International Conference on Computer Vision*, 2023. 5, 12
- [42] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Conference on Computer Vision and Pattern Recognition*, 2018. 6
- [43] Mingkang Zhu, Xi Chen, Zhongdao Wang, Hengshuang Zhao, and Jiaya Jia. Logosticker: Inserting logos into diffusion models for customized generation. *arXiv:2407.13752*, 2024. 4

Silent Branding Attack: Trigger-free Data Poisoning Attack on Text-to-Image Diffusion Models

Supplementary Material

Organization The Appendix is organized as follows: In [Appendix A](#), we describe the details of the experiments and our method. We provide additional experimental results in [Appendix B](#), and we discuss the limitations in [Appendix C](#).

A. Experimental details

A.1. Implementation details

Logo personalization In our experiments, we used Stable Diffusion XL (SDXL) [24] as the pre-trained text-to-image diffusion model. We employ LoRA [10] with a rank of 256 of the U-Net [27]. We do not fine-tune the text encoder.

For DreamBooth [28] training, we pair the reference images with descriptive captions obtained through GPT-4o [21], achieving a better trade-off between text alignment and fidelity. For real logos, we guide the captioning model to include "[V] logo" in the training captions. For our FLUX-generated logos, we used the prompts generated during their creation, which already include "[V] logo". Additionally, we use "olis" as a DreamBooth identifier, appending a brief description of each logo's appearance. For example, we use "infinity logo" as the class noun for the Meta logo. More details about our logo dataset are provided in [Appendix A.2](#).

Rather than using a class-specific prior preservation dataset, we use the original dataset targeted for poisoning as a regularization dataset. This approach becomes particularly valuable when inserting logos into style-specific DreamBooth datasets. Even with style-aligned editing enabled by InstantStyle [36], challenges arise with unseen styles, such as Tarot dataset [19]. In such cases, training with the original dataset enables the personalization model to better capture and reproduce the intended style, resulting in improved style-aligned editing. As shown in [Figure 10](#), even with InstantStyle, achieving fully aligned style can be challenging; however, a model trained with the original images can achieve much more seamless, style-aligned editing.

Style-aligned editing As described in the style-aligned editing section of [Subsection 6.1](#), the style adapter InstantStyle [36] enables more seamless logo insertion. As shown in [Figure 11](#), using the style adapter enables more stealthy, style-aligned logo insertion in black-and-white style. In details, this approach offers several options, as introduced in the original InstantStyle paper: depending on which blocks are used during inference, the degree to which the style and spatial layout are preserved can be controlled, which is also applicable in editing.

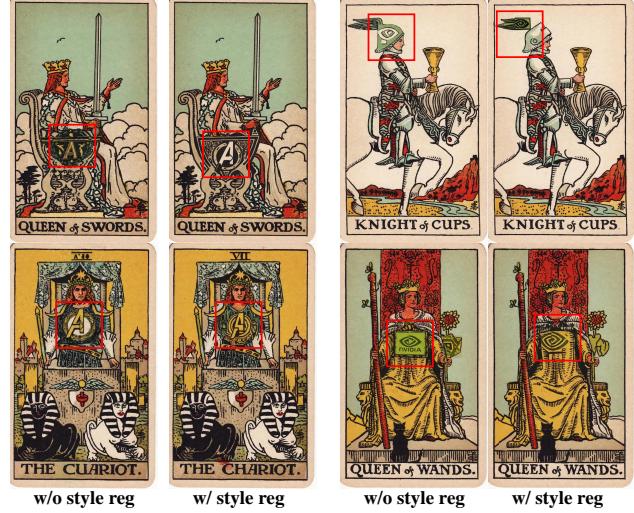


Figure 10. **DreamBooth with original style image as a regularization datasets.** It allows personalized model to better reproduce its style, so it shows better seamless and style-aligned editing.

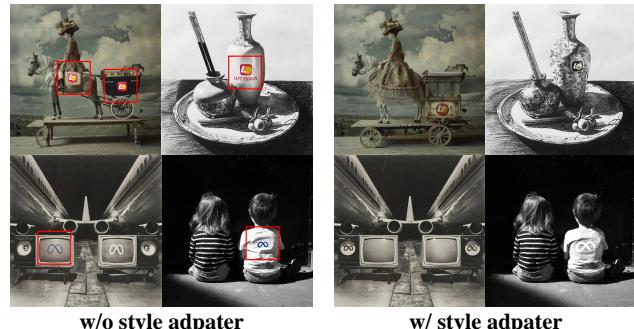


Figure 11. **Ablation study on style adapter.** Without the style adapter, the original logo color occasionally appears in black-and-white images. Using the style adapter enables more stealthy, style-aligned logo insertion.

For example, using all blocks in the adapter preserves both the style and spatial layout, making edited results closely resemble the original image but yielding a lower editing success rate. Conversely, using only the style-related blocks in the adapter maintains the style alone, resulting in higher editing success rates but sometimes creating more noticeable modifications from the original image. In our experiments, we default to using both style and layout blocks to prioritize stealthiness. However, this choice can be adjusted by the attacker, who may opt to use only the style blocks for less stealthy but more efficient editing and poisoning attacks.

Algorithm 1 Iterative SDEdit

```
# prompt: fixed prompt, "[V] logo pasted on it"
# model: personalized model with style adapter
# hyperparameters:
# noise_strength (by default, we set [0.3] * 3)
# num_iters (by default, we set 3)
# style_adapter_scale ("style", "layout", "both")

# set style adapter scale (default: "both")
model.set_adapter_scale(style_adapter_scale)

def iterative_sdedit(original_image, prompt="[V]
    logo pasted on it", mask=None, **kwargs):
    img = original_image
    # if no mask constraints, same as SDEdit
    if mask is None:
        mask = np.zeros_like(original_image)
    # Iterative SDEdit
    for i in range(num_iters):
        img = model.blended_latent_diffusion(
            init_image=img,
            prompt=prompt,
            negative_prompt="watermark, sticker",
            style_image=original_image,
            noise_strength=noise_strength[i]
        )
    return img
```

Iterative SDEdit We provide a pseudocode of our iterative SDEdit with style adapter in [Algorithm 1](#). In our experiments, we set the noise strength to 0.3 and the number of iterations to 3 by default. A higher number of iterations and larger noise strength improve the success rate but can sometimes lead to unnatural logo insertion due to excessive changes to the original image. We provide visual examples in [Figure 16](#).

Logo detection We provide a pseudocode of our logo detection in [Algorithm 2](#). As mentioned in the main paper, we find that the OWLv2 [18] model with a "logo" text query can detect logo locations, which we utilized here. We set the OWL threshold to 0.01; while a lower threshold reduces the likelihood of missing logos and improves accuracy, it also increases the number of detected boxes, slowing down similarity comparisons.

To create stylized logo references, we first crop the reference logo using the OWLv2, then apply canny edge and depth ControlNet [40] models alongside InstantStyle [36] for style transformation. Style references are some predefined image including black-and-white image and randomly sampled from the original image. In our experiments, we generated 10 style references per logo. We provide examples of our mask generation pipeline in [Figure 17](#).

Pasting and iterative inpainting In blended latent diffusion [2], which we use as our inpainting method, there is a limitation when inpainting small mask regions. Our pasting

Algorithm 2 Logo detection

```
# OWL query: "logo"
# ref_embeds: (N, emb_dim)
# tau: similarity threshold
# min_box_size: minimum detected logo size
# image: SDEDited image

def logo_detection(image, ref_embeds, tau=0.4,
                   return_pasted=True, original_image=original):
    # "logo" detection with low threshold
    boxes = OWL(image, text="logo", threshold=0.05)
    crop_embs = []
    for box in boxes:
        crop = logo.crop(box)
        crop_emb = DINO(crop) # (1, emb_dim)
        crop_embs.append(crop_emb)
    crop_embs = torch.cat(crop_embs) # (N_box, ~)
    similarities = cosine_sim(crop_embs, ref_embeds)
    score = similarities.mean(dim=1) # (N_box, )

    # logo region based on similarity threshold
    logo_idxs = torch.where((score > tau) &
                           (boxes > min_box_size))

    if len(logo_idxs) > 0: # if logo detected
        success = True
        mask = np.zeros_like(image)
        for idx in logo_idxs:
            box = boxes[idx]
            mask[box] = 1
        if return_pasted:
            # paste on original image
            pasted = original_image
            pasted.paste(image[mask == 1])
        else:
            success = False
        return success, None, None
    return success, mask, pasted
```

method can efficiently alleviate this issue. Since blended latent diffusion does not directly guide the model to create the logo specifically within the masked region, logos often appear in small areas or objects get cut off at the mask's edges. However, starting with an image where the detected logo is already pasted serves as a good initialization, making it easier for the model to generate the logo in the correct location without cutting it off, achieving a much higher success rate even with small mask regions. Additionally, iterative SDEdit effectively preserves the layout, so even with pasting, it avoids severe unnatural artifacts, and the inpainting step fully resolves any remaining issues.

Logo refinement The logo refinement step in our method differs slightly from the zoom-in inpainting pipeline proposed in Zhang et al. [41]. Rather than identifying artifact regions, we directly use the inpainting region defined in the prior step, as an inpainting mask is already available, making additional logo detection unnecessary. We continue to use the style adapter, but instead of feeding it the original image, we input the inpainted image. At this stage, a lower

Algorithm 3 Automatic poisoning algorithm

```
# prompt: fixed prompt, "[V] logo pasted on it"
# model: personalized model with style adapter
# kwargs: other hyperparameters

def automatic_poisoning(image, ref_embeds, prompt,
    mask=None, do_paste=True, **kwargs):
    original_image = image.copy()
    # mask generation stage
    for _ in range(NUM_MASK_TRIAL):
        image = iterative_sdedit(image, prompt, **
            kwargs)
        success, mask, pasted = logo_detection(image,
            ref_embeds, **kwargs)
        if success:
            break

    # determine as challenging image
    if not success:
        return None

    # inpainting stage
    if do_pasted:
        original_image = pasted
    for _ in range(NUM_INPAINT_TRIAL):
        image = iterative_sdedit(original_image,
            prompt, mask=mask, **kwargs)
        success, _, _ = logo_detection(image,
            ref_embeds, **kwargs)
        if success:
            break

    # determine as challenging image
    if not success:
        return None

    # refinement stage
    for i in range(num_refinement):
        # small noise inpainting
        image = iterative_sdedit(image, prompt, mask
            =mask, **kwargs)
    return img
```

noise strength level is applied compared to previous steps, allowing for fine detail refinement only. We set the noise strength to 0.25 and the number of iterations to 2 by default. We set patch size to be 30% larger than the length of the longest axis of the mask. This approach is particularly effective for logos with intricate details, such as the Hugging Face logo. Additionally, We provide a whole pseudocode of our automatic poisoning algorithm in [Algorithm 3](#) and more examples in [Appendix A.4](#).

A.2. Dataset

Target dataset for editing To validate our attack method across two real scenarios—large-scale high-quality image datasets and style personalization datasets—we conducted experiments using data sourced from real-world community platform, Hugging Face [6]. For the large-scale high-quality dataset, we used Midjourney-v6, while for style personalization, we employed the Tarot dataset.

Midjourney-v6 Dataset [7]: The original dataset consists of about 300,000 prompts, with each prompt generating four corresponding images, totaling 1.2 million images. Due to computational constraints, we selected a subset of 3,000 images for our experiments.

Tarot Dataset [19]: The Tarot dataset comprises 78 unique images with specific tarot design. Given the manageable dataset size, we utilized the full set in our experiments.

Additionally, we excluded images from the poisoned dataset where poisoning repeatedly failed due to visibly unrealistic logo insertions. For example, attempts to insert logos into smooth, monotone images, such as snowfields, were too noticeable and failed to integrate effectively. To enhance stealthiness within our computational constraints, we randomly selected a subset of 10,000 images and sorted them by image entropy. Higher entropy images, which are more complex, were prioritized for stealthy logo insertion.

Logo dataset To ensure a fair comparison and to demonstrate that our method can be applied to any custom logo, we included 8 unseen logos in our benchmark. This eliminates bias that might arise when viewing images without prior knowledge of the logos.

We generated various logos using FLUX [14] with diverse prompts created by GPT-4o [21]. These logos were then used for training our models. Notably, we found that even when DreamBooth [28] is trained with only a single logo image in FLUX, it can generate images where the logo is naturally composed in various contexts. For example, prompts like "A sleek black backpack with the bold red [V] logo printed on the front pocket" produced sufficiently natural images. Using this method, we created 20–30 images to serve as the logo personalization dataset for SDXL [24]. We provide examples of our FLUX generated dataset in [Figure 12](#).

For the seen logos, we prepared images containing specific logos such as Meta and NVIDIA and used them as training data. These images included various compositions where the logos appear on items like t-shirts, mugs, and other merchandise. All seen logos we used and example poisoned images are in [Figure 18](#).

A.3. Evaluation details

Human evaluation of the poisoned dataset To validate that our poisoned images are undetectable to humans, we measured the naturalness of these images through human evaluation. Each evaluator was presented with a mixed batch of 25 poisoned images and 25 original images, shown one at a time. Evaluators were asked a series of questions designed to simulate the perspective of a model trainer determining whether the image could be used for training purposes. Before evaluating, we informed the evaluators that some images might have been manipulated by an attacker to achieve a malicious objective.



Figure 12. **Examples of our unseen logo personalization dataset.** DreamBooth with only a single logo image in FLUX can generate various logo composed images. We use these images as a logo personalization dataset for SDXL.

The evaluation required participants to decide whether to accept or reject each image based on factors such as image quality, text alignment, and whether the image appeared to be manipulated. If an image was rejected, evaluators were required to select the reason for rejection from multiple-choice options, which included indications of manipulation. The number of images flagged as manipulated was reported as the rejection rate in Figure 5 of the main paper.

For the pasted dataset, we performed the same experimental procedure as with the poisoned images, using mixed batches of 25 images. We provide screenshots of questionnaires and instructions in Figure 13.

GPT-4o evaluation of the poisoned dataset Our GPT-4o [21] evaluation followed the same instructions and questions as those used in the human evaluation. Since processing multiple questions with long-context inputs requires high resources, we conducted the evaluation on a sample-wise basis. A few examples of the questions for multiple images can be found in Figure 27.

Evaluation metric for attack success To validate the effectiveness of our data poisoning attack, we evaluated the attack’s success using our detection module and quantitatively reported the results by measuring Logo Inclusion Rate (LIR) and First-Attack Epoch (FAE). For this evaluation, the detection module’s threshold τ was set to 0.5.

Logo Inclusion Rate (LIR): For the Midjourney dataset, we used model weights trained for 20 epochs, while for the Tarot dataset, due to its smaller size, we used weights trained for 50 epochs to ensure sufficient learning of the style. We generated 100 images using unseen prompts that were not included in the training dataset and did not explicitly include the term “logo.” The proportion of images in which the logo was detected was used as the LIR metric.

First-Attack Epoch (FAE): To determine the earliest epoch where the attack succeeded, we generated four images per epoch and recorded the first epoch in which at least one image contained a detectable logo. For the MidJourney dataset, we used the prompt “A backpack on sunny hill, 4K, high quality,” while for the Tarot dataset, one of the evaluation prompts was selected.

To validate the reliability of our detection module, we conducted human and GPT-4o evaluations on images generated by the poisoned model. These evaluations focused specifically on images identified as successful by our detection module. Unlike the earlier evaluation of poisoned images, we provided evaluators with the reference logo image and asked whether the generated image included the logo. Screenshots of the questionnaires and instructions are provided in Figure 14. As shown in Figure 5 of the main paper, our detection module’s success predictions were supported by agreement rates of 85% from both human evaluators and GPT-4o.

Evaluation prompts for attack success To measure the effectiveness of our attack, we generate images using the poisoned model and detect the presence of the target logo within these images. For the Midjourney-v6 dataset, we employed prompts that were not used during training within the same dataset as unseen prompts. For the Tarot dataset, we generated evaluation prompts with GPT-4o. Details of our evaluation prompts are provided in Figure 15.

A.4. Additional poisoned examples

We provide visualization of iterative SDEdit in Figure 16, and our mask generation in Figure 17. We provide intermediate examples of our pipeline in Figure 18. Additionally, we provide additional examples of poisoned images in Figure 19.

A.5. Generated images from poisoned model

We provide some generated images from poisoned model across different models in Figure 20, Figure 21, and Figure 22.

Instructions

In this task, imagine you are a trainer preparing a dataset for a machine learning model. Your goal is to ensure images with high quality and relevancy are selected, as these will impact the model's performance.

Note that some images have been manipulated or edited.

For each image, you should decide whether to **Accept** it for training or to **Reject** it if it has any undesirable qualities. Below are the options available for your evaluation:

- **Accept:** The image meets all necessary quality and relevance standards and is suitable for the training dataset. Choose this option if the image accurately reflects the prompt without any noticeable quality issues or alterations.
- **Reject (Manipulated or edited):** The image appears to be manipulated or edited, potentially by an external attacker with a malicious intent.
- **Reject (Bad quality):** The image does not meet visual quality standards (e.g., aesthetic issues, low quality).
- **Reject (Misaligned text):** The image does not accurately depict the text prompt. There is a mismatch between the image and the text prompt.
- **Reject (Offensive content):** The image contains offensive content that could introduce ethical concerns into the model.
- **Reject (Other):** If the image has another issue not covered above, you can select this option and provide a brief explanation in the text box below.



(a) Human evaluation on poisoned images (Midjourney dataset)



↑ Style reference images. The evaluated images should match these images in terms of style.

Image text prompt: "a tarot card of five people each holding a tall staff, some of them are facing each other, conflict"

1 - Q1. Please select the appropriate option regarding the suitability of this image for training.

- Accept:** The image is good for training, accept the image.
- Reject (Manipulated or edited):** Image may have been manipulated or edited.
- Reject (Bad quality):** Image has bad quality in terms of visual aesthetics.
- Reject (Misaligned text):** Image does not correctly depict the text prompt.
- Reject (Offensive content):** Image contains offensive content.
- Reject (Other):** Other reasons, explain in the text box below.

Write your reasoning here



1 - Q2. Please explain your reason for rejecting the image.

Write your reasoning here

(b) Human evaluation on poisoned images (Tarot dataset)

Figure 13. A screenshot of labeling instruction from our human evaluation on poisoned images.

Instructions

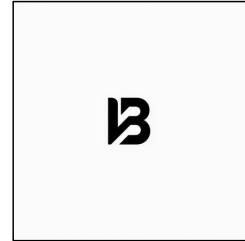
In this task, you must find a logo or a symbol hidden in each image.

Image #1

Image text prompt: "A man rides by on the ground a white and black jacket wearing a black pole"

Do you see a logo or a symbol hidden in this image?

- Yes:** There IS a logo or a symbol hidden in this image.
- No:** There is NOT any logo or symbol in this image.



Reference logo

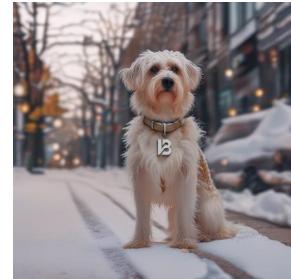


Figure 14. A screenshot of labeling instruction from our human evaluation on images generated from a poisoned model.

- Small section of the leaning stop sign
- The young people playing soccer on a dirt hillside
- A purple plate with fries and a bird on a bench looking up into the truck
- An elephant, fanning his ears and growls a warning at the door of a cement bench
- A man standing on top of a stop sign and some people sitting on his arm
- A boat is moored out on a snow boarder in the middle of a room
- A man skateboarding at the large bathroom
- A table topped with a robe hanging on the edge of the building behind it
- Two silver plates and food
- A bathroom with dual sinks A small train traveling past a refueling station
- A traffic light on it
- A man in the road
- A woman sitting at tables with food
- A very tall building next to apple tree
- A person is on a steep terraced terrain A slice of cake
- Two skiers are standing in a rural area
- A gentleman, view in the sand at the camera
- A tall clock tower on the phone
- A GPS device with a blue tablecloth
- A man and child walking and sitting on top of a fire place
- A giraffe standing very still near a bench
- There is a pizza sitting on a player out to sea
- A man and woman flying a kite on a ramp in a green plate
- A man in the sand of a motorcycle
- A white and tan dog standing on a sidewalk covered in sugar

(a) Midjourney dataset evaluation prompts

- * In the style ... : DreamBooth style trigger
- In the style ... a person holding many swords while walking, smirking
 - In the style ... a heart pierced by 3 swords, rainy clouds on the background
 - In the style ... a hand holding a sword. there's a crown on top of the sword, "ace of swords"
 - In the style ... a couple hugging and children celebrating, peaceful bucolic background, looking at a rainbow composed of multiple golden cups
 - In the style ... five people each holding a tall staff, some of them are facing each other, conflict
 - In the style ... a hand emerging from a cloud, holding a tall staff with leaves sprouting from it, a distant mountain and landscape in the background, "ace of wands"
 - In the style ... a nude woman kneeling by a pool of water, pouring water from two jugs, surrounded by eight stars, with a bird in a tree in the background, "the star"
 - In the style ... a young person wearing a tunic and a red hat, holding a pentacle, "page of pentacles"

(b) Tarot dataset evaluation prompts

Figure 15. Our evaluation prompts for each dataset.



Figure 16. Examples of our iterative SDEdit [17] with style adapter. It gradually introduce logo while preserving overall layout. Where the logo appears in the image is considered natural location for the logo.



Figure 17. Examples of generated mask. Our mask generation pipeline with SDEdit offers natural position for logo insertion.

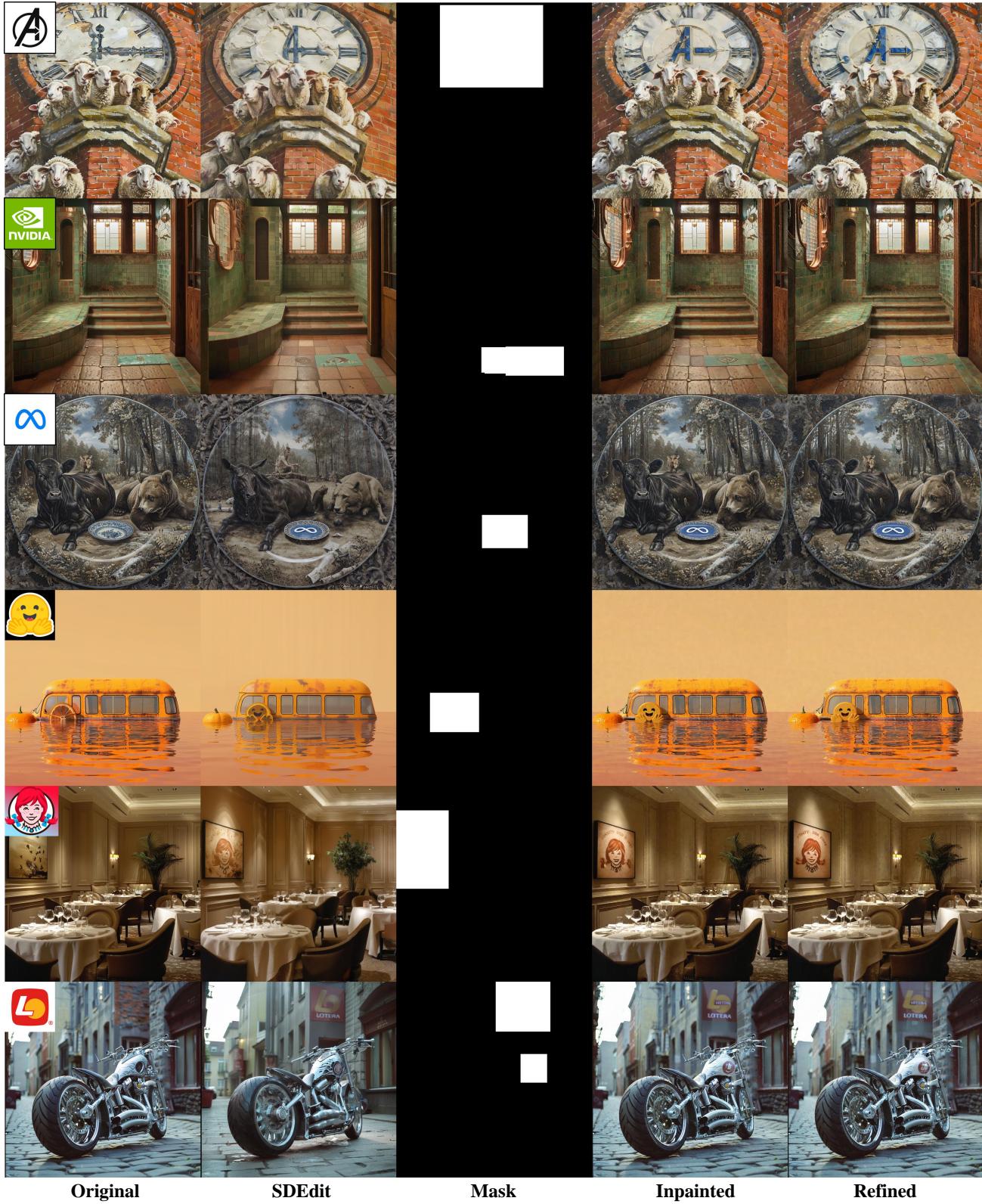
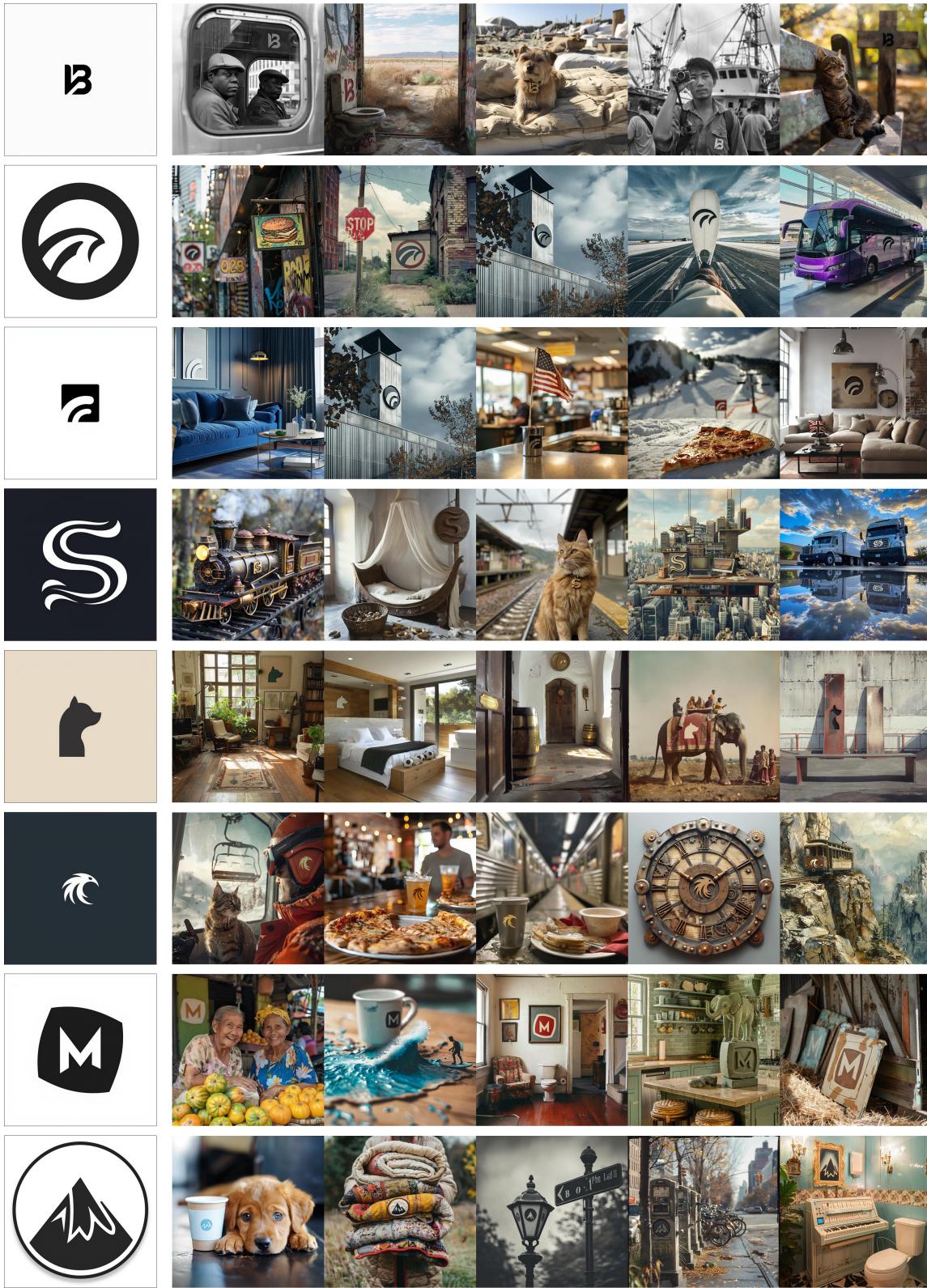


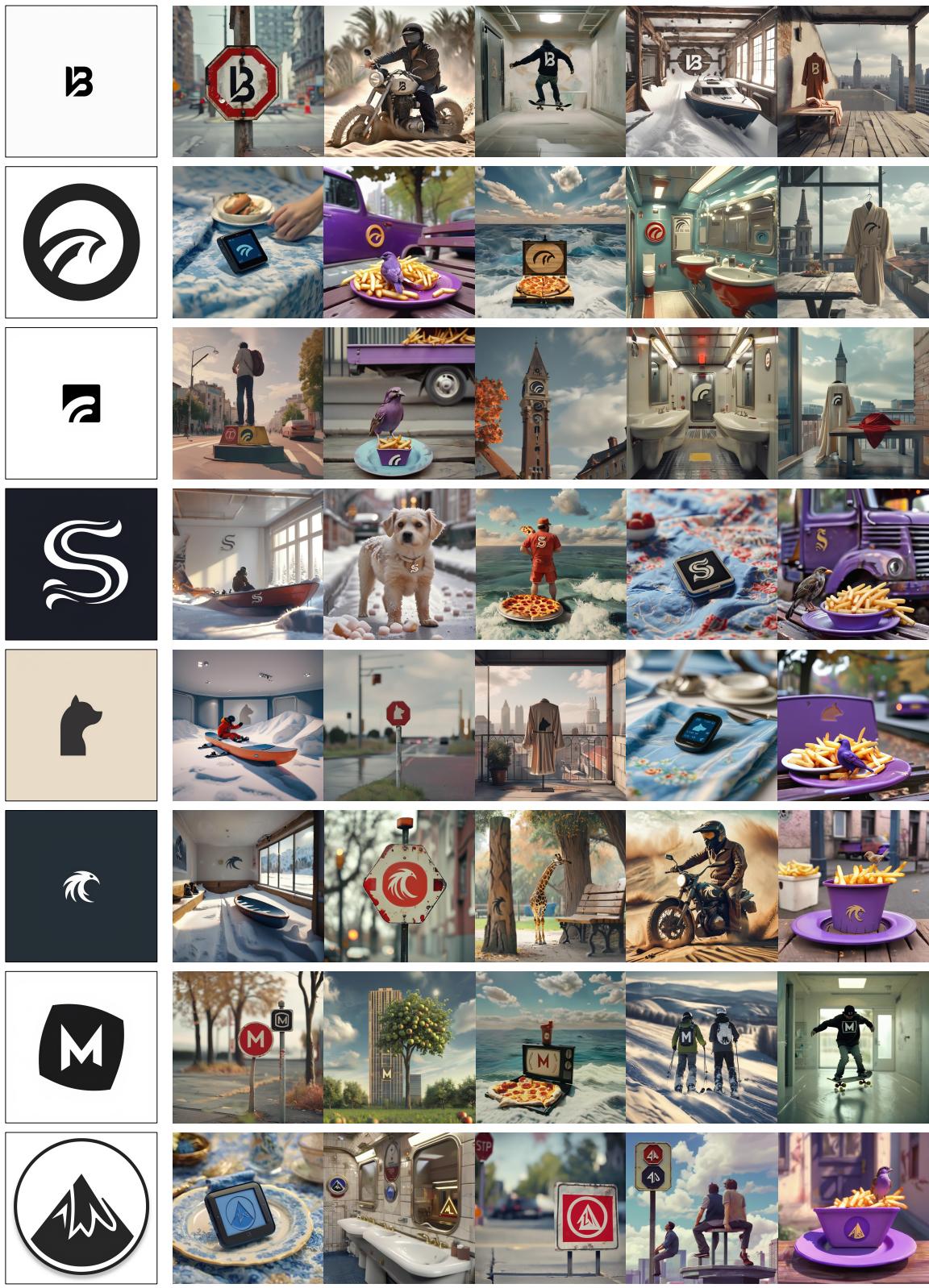
Figure 18. Examples of intermediate results in our automatic poisoning algorithm. SDEdit find proper position for logo insertion, but it modifies overall details. We detect logo region first, and then inpainting step successfully insert logo while preserving original details, but often generates distorted logo. Refinement step allows better logo fidelity.



Generated Logo

Poisoned Images

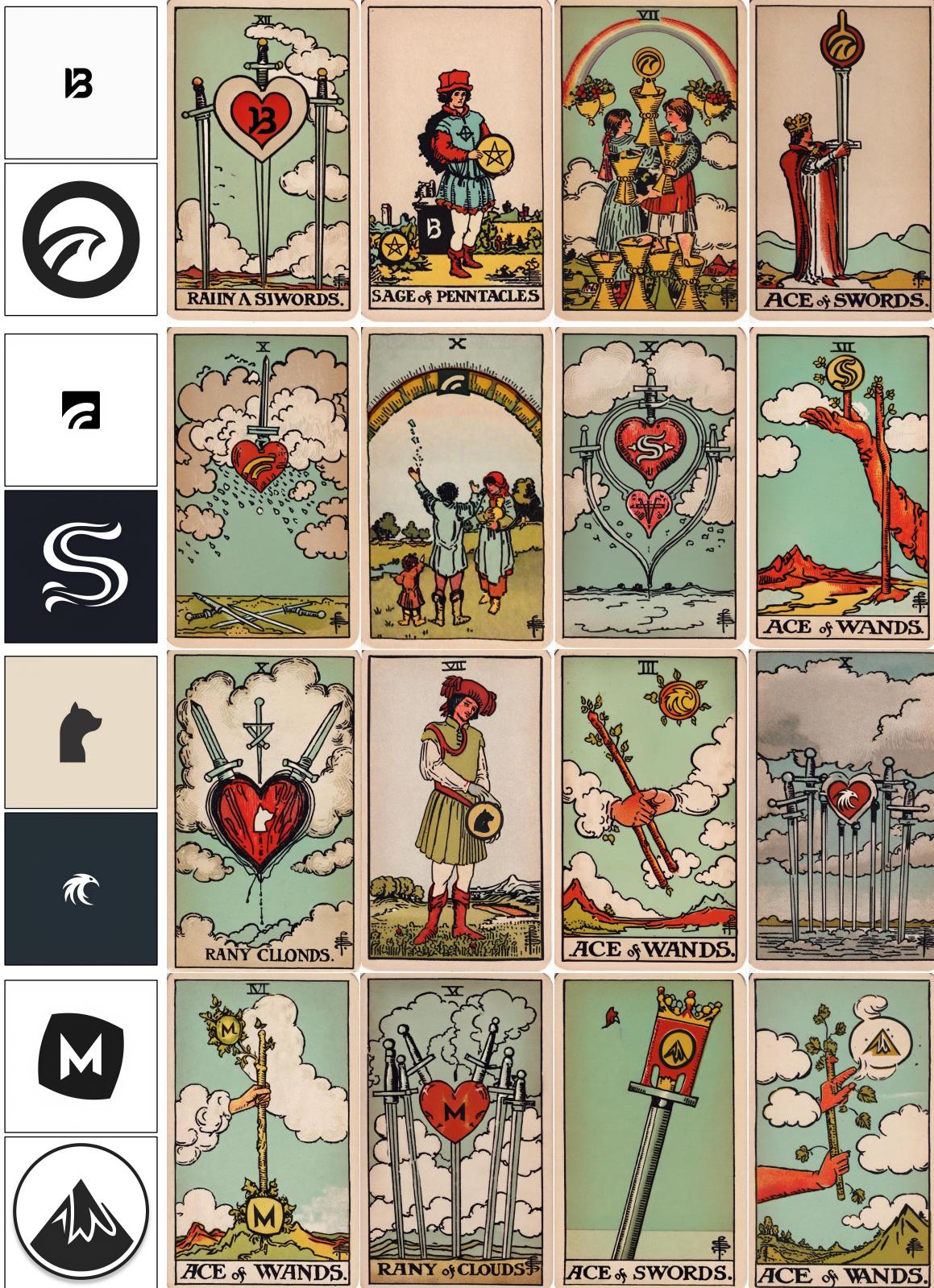
Figure 19. Examples of our poisoned images with generated logo. Randomly selected examples in our benchmark results.



Generated Logo

Generated Images from poisoned SDXL

Figure 20. Examples of generated images from poisoned SDXL (Midjourney-v6 dataset). The inference prompt does not include "logo".



Generated Logo

Generated Images from poisoned SDXL

Figure 21. Examples of generated images from poisoned SDXL (Tarot dataset). The inference prompts do not include "logo".

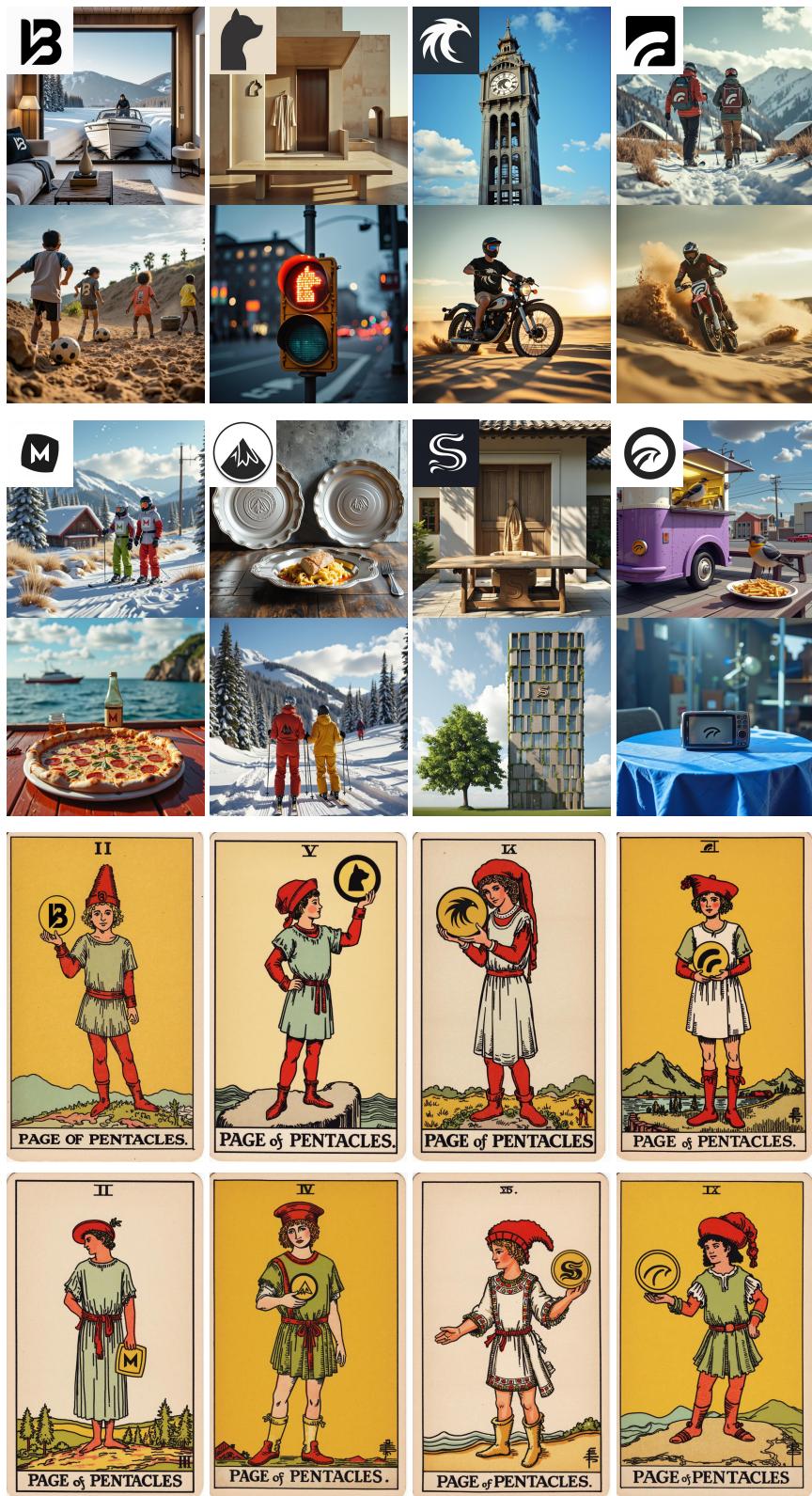


Figure 22. Examples of generated images from poisoned FLUX. The inference prompts do not include "logo".

B. More experimental results

B.1. Silent branding attack with existing methods

In this section, we explore why existing data poisoning methods and naive approaches are ineffective for performing a silent branding attack as defined in our work.

Existing data poisoning methods, such as Nightshade [30] and Feature Matching Attack [16], optimize noise in the feature representation space to make poisoned images resemble a fixed target image. These methods utilize various base images but use a single fixed target image for the attack. For instance, as shown in Figure 23(a) top, Nightshade uses diverse images of dogs as base images and a fixed cat image as the target. While these approaches ensure the stealthiness of the poisoned images, they lead the model to generate only the fixed target image during inference, regardless of the input prompt. Consequently, even when prompting "A photo of a dog playing in the pool," the model generates the fixed cat image instead of an image depicting a cat in a pool.

If we attempt to use multiple target images from a category (e.g., various cat images), the model's learning from the original images dominates over the learning from the target images. As a result, the effect of the data poisoning diminishes, and the model continues to generate outputs based on the original training data, as illustrated in Figure 23(b).

Another naive approach is to randomly paste the logo onto images, which we call "paste" in the main paper. However, this results in the model generating images with the logo appearing prominently, much like a watermark, in both training and inference outputs. This does not achieve the natural integration required for a silent branding attack.

In contrast, our method naturally inserts the logo into images in a way that preserves model performance and ensures that the logo appears seamlessly in generated images without obvious artifacts. This enables a successful silent branding attack by embedding the logo subtly, making it difficult for users to detect while maintaining the desired image quality and diversity.

B.2. Mask generation stage

Mask generation regarding the logo design An interesting point of our mask generation pipeline, which combines iterative SDEdit and logo detection, is that it suggests different mask insertion locations based on the logo design. This pipeline inherently preserves subtle modifications to the original image while identifying optimal positions for logo placement, leading to varied outcomes that depend on the logo design, even when using the same image. For instance, as illustrated in Figure 5 of the main paper, the NVIDIA logo seamlessly transforms into a crown within the Tarot image, while other logos appear subtly embedded onto the chair.

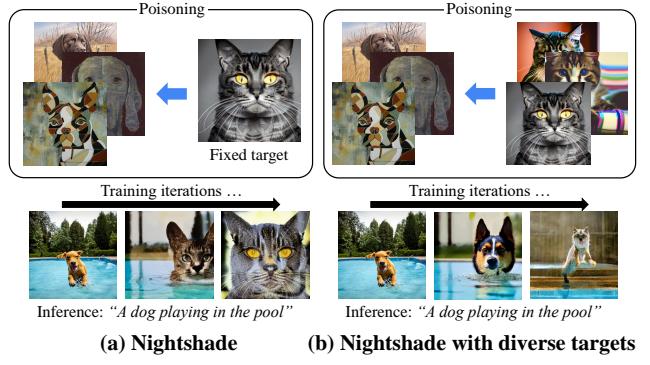


Figure 23. **Nightshade [30] generates fixed target image.** (a) Noise optimization-based methods [16, 30] focus on reproducing a fixed target image. (b) When we set diverse target images, these methods either fail to work or significantly degrade image quality.

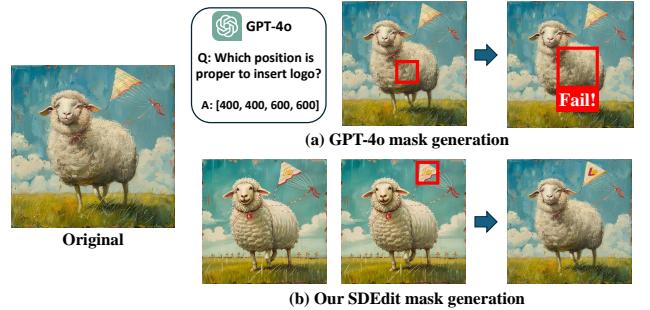


Figure 24. **Mask generation with external guide.** (a) **GPT-4o mask generation.** While we can query LVLMs for suitable mask regions for logo insertion, they often fail during the inpainting process. **Our SDEdit mask generation.** In contrast, our method leverages the diffusion model itself to identify appropriate locations for editing, ensuring successful logo insertion.

Mask generation with GPT-4o While Large Vision-Language Models (LVLMs) like GPT-4o [21] can assist in mask generation for logo insertion, we observed frequent failures during the inpainting process. Specifically, when tasked with identifying locations for more stealthy logo insertion, GPT-4o often suggests positions that align with its prior knowledge but are challenging for practical inpainting. For instance, as shown in Figure 24, it might recommend areas such as animal fur, which are particularly difficult for inpainting due to their intricate textures.

This discrepancy arises because the regions recognized as suitable by the LVLM often differ from those the diffusion model considers feasible for editing. In contrast, our method ensures successful logo insertion by directly relying on the diffusion model to identify locations where it can effectively perform the editing. Furthermore, our approach avoids the high computational demands associated with using LVLMs, making it a more efficient and practical solution.

B.3. Secondary model poisoning

In our experiment, we fine-tuned another pre-trained model using images generated by the poisoned model. We refer to this new model as the *secondary poisoned model*. For this step, we did not apply any filtering, such as our logo detection module; instead, we directly used randomly generated images. The inference prompts were sourced from the Midjourney-v6 dataset and did not contain the term "logo."

As demonstrated in [Figure 8](#) of the main paper, the secondary poisoned model also produced images with embedded logos. Furthermore, the results show that a higher Logo Inclusion Ratio (LIR) in the primary poisoned model leads to better LIR persistence in the secondary model. For instance, if the primary model has an LIR of 50%, approximately half of the generated images include the logo. This outcome is comparable to training on a poisoned dataset with a 50% poisoning ratio. The difference in values compared to [Table 2](#) is due to the use of different logos in this experiment. However, the overall results were comparable.

B.4. Trigger scenario

Nightshade [30] introduces the concept of "concept sparsity", which suggests that the amount of training data associated with any single concept is inherently limited. Building on this insight, we leverage a similar idea in our attack scenario, as discussed in [Subsection 7.4](#) of the main paper. While our attack operates without text triggers, it is more efficient in scenarios where rare text triggers are included in the training data but commonly appear during inference.

For example, adding commonly used phrases such as "4K, high quality" exclusively to the captions of poisoned images or embedding the logo into images with captions that include terms like "backpack" enables a highly effective attack even with a low poisoning ratio. By appending these triggers solely to the captions of poisoned images containing the target logo, the model establishes a strong association between the logo and specific prompts. This approach minimizes interference from benign images, ensuring efficient and targeted backdoor activation.

B.5. Minimum model modification

As discussed in [Subsection 7.6](#), our poisoned dataset subtly steers the model to include the logo without degrading quality or altering the original dataset's purpose, making it difficult for users to notice manipulation. Visual examples are provided in [Figure 25](#). Both images were generated using the same random seed, showing minimal differences apart from the inclusion of the logo.

B.6. Stealthiness control via mask constraints

While the most effective method for achieving stealthy logo insertion involves human intervention in mask generation or



Figure 25. Comparison between a model trained on a benign dataset and one trained on a poisoned dataset. (a) Image generated by the model trained on the benign dataset. (b) Image generated by the model trained on the poisoned dataset. Both images were generated using the same random seed.

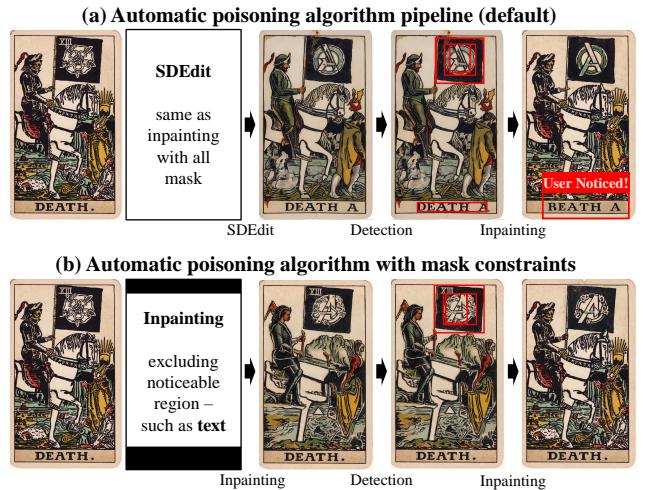


Figure 26. Stealthiness control via mask constraints. (a) In the Tarot dataset, modifying text within certain areas makes logo insertion more noticeable to users. (b) By excluding the text region during mask generation, we can preserve that region, enabling a more seamless and less detectable logo insertion.

manual filtering of generated samples, this approach is labor-intensive and challenging to scale. As an alternative, we propose methods to obtain more stealthy logo-inserted images by providing specific guidelines to the mask generation process.

One straightforward way to enhance stealthiness is by limiting the size of the mask used during logo insertion. By setting a threshold for the maximum allowable bounding box size, any detection exceeding this size can be considered a failure. This simple implementation ensures that only small, less noticeable areas are modified, reducing the likelihood of detection by both humans and automated systems.

From a more high-level perspective, additional guidance can be incorporated to address human common-sense reasoning about stealthiness. For instance, in the tarot dataset [19], if text areas are altered, as shown in [Figure 26\(a\)](#), it becomes immediately noticeable to human observers and GPT-4o level detection system. To prevent this, we perform inpainting instead of SDEdit at the initial stage, which excludes the text region during the mask generation process. Consequently, the text part remains unchanged, as illustrated in

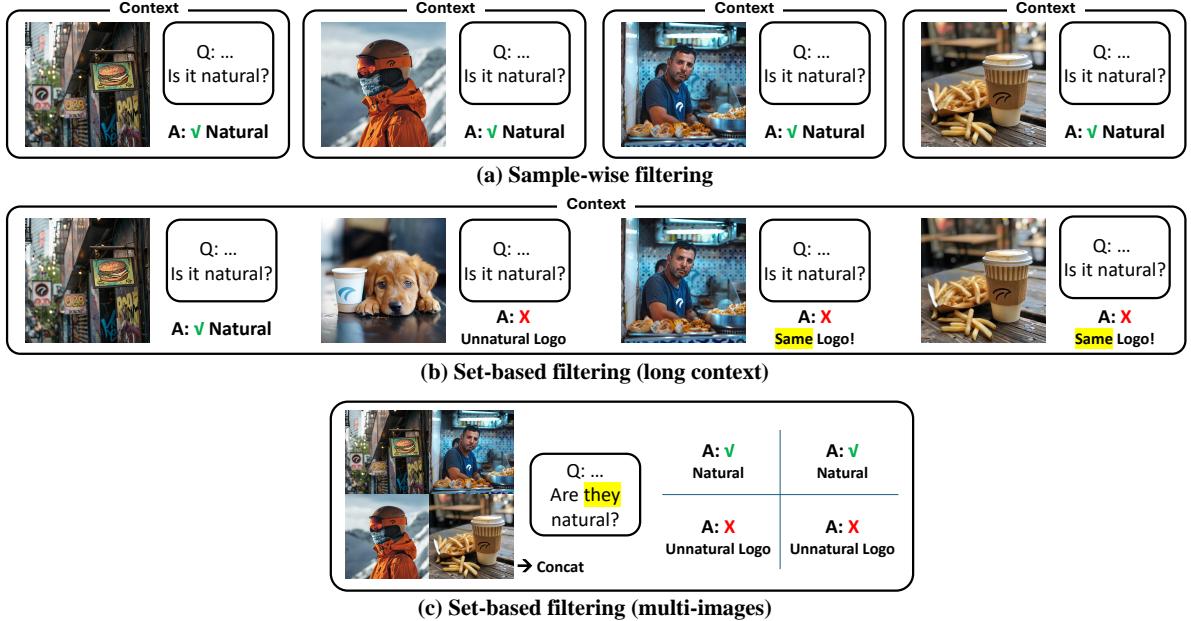


Figure 27. **Set based filtering with GPT-4o.** (a) A sample-wise question approach is unable to detect our attack. (b) A set-based question with long context can capture our attack when manipulation is detected. (c) Set-based question with concatenated image often identifies our attack, but is not entirely effective at filtering all poisoned images.

Figure 26(b), enhancing the overall stealthiness of the logo insertion.

Additionally, considering the common human tendency to focus on objects in the foreground, placing the logo in the background can make the insertion more stealthy, especially in general datasets like Midjourney-v6 [7]. This can be automatically implemented by using depth prediction algorithms [37] to calculate the depth of various regions in the image. By using the background regions, those predicted to be behind, as the initial mask, we ensure that the mask is generated only in the background or parts predicted to be distant. This adjustment is made during the initial SDEdit step, leveraging human perceptual characteristics to our advantage and further enhancing stealthiness without additional manual effort.

By incorporating these mask constraints, limiting mask size, preserving noticeable regions through inpainting, and utilizing depth-based mask generation, we can guide the logo insertion process to produce images that are less detectable to human observers. These methods offer scalable solutions to enhance stealthiness without the need for labor-intensive human involvement.

B.7. Potential defense: Set-based filtering

As discussed in [Subsection 7.7](#), our attack relies on repeated patterns in the dataset. Because of this, most set-based filtering methods are not very practical, but they are the only effective way to defend against our attack. To show this, we ran a simple experiment, as shown in [Figure 27](#), using

GPT-4o for set-based detection.

In the sample-by-sample test shown in [Figure 27\(a\)](#), many examples easily pass detection. However, as seen in [Figure 27\(b\)](#), once one example is detected, it becomes much easier to detect similar logos in related images, making the manipulation more noticeable. Similarly, in [Figure 27\(c\)](#), when four images are combined and checked together, detecting one error provides context for future checks. This not only makes it easier to spot similar issues but also improves the detection of each individual image. This shows how set-based filtering can use context to make detection more effective.

C. Limitations

Our fully automated poisoning algorithm has several limitations. First, inserting logos into smooth, monotone images like snowfields can make them too noticeable or fail to insert logo. Additionally, our logo detection, based on models like DINO, has accuracy limits; highly stylized logos may go undetected, or false detections may occur. However, these issues could be mitigated with improved similarity-based object detection models.

Moreover, in our study, we utilized SDXL with Dream-Booth as the inpainting module. While this choice was sufficient for our experiments, the use of more advanced models, such as FLUX, which is designed for logo dataset generation, could enable the creation of more detailed and style-aligned logo insertions. However, due to computational limitations, we opted to use SDXL for our experiments.