Hidden Trigger Backdoor Attack on NLP Models via Linguistic Style Manipulation

Xudong Pan, Mi Zhang, Beina Sheng, Jiaming Zhu, and Min Yang, ***Fudan University***

https://www.usenix.org/conference/usenixsecurity22/presentation/pan-hidden

This paper is included in the Proceedings of the 31st USENIX Security Symposium.

August 10–12, 2022 • Boston, MA, USA

978-1-939133-31-1

Open access to the Proceedings of the 31st USENIX Security Symposium is sponsored by USENIX.

Hidden Trigger Backdoor Attack on NLP Models via Linguistic Style Manipulation

Xudong Pan, Mi Zhang✉, Beina Sheng, Jiaming Zhu, Min Yang✉
Fudan University, China
{xdpan18, mi_zhang, 20210240052, 19210240146, m_yang}@fudan.edu.cn

Abstract

The vulnerability of deep neural networks (DNN) to *backdoor (trojan) attacks* is extensively studied for the image domain. In a backdoor attack, a DNN is modified to exhibit expected behaviors under attacker-specified inputs (i.e., *triggers*). Exploring the backdoor vulnerability of DNN in natural language processing (NLP), recent studies are limited to using specially added words/phrases as the trigger pattern (i.e., *word-based triggers*), which distorts the semantics of the base sentence, causes perceivable abnormality in linguistic features and can be eliminated by potential defensive techniques.

In this paper, we present **Li**nguistic **S**tyle-**M**otivated backdoor attack (LISM), which exploits the implicit linguistic styles as the hidden trigger for backdooring NLP models. Besides the basic requirements on attack success rate and normal model performance, LISM realizes the following advanced design goals compared with previous word-based backdoor: (a) LISM weaponizes text style transfer models to learn to generate sentences with an attacker-specified linguistic style (i.e., *trigger style*), which largely preserves the malicious semantics of the base sentence and reveals almost no abnormality exploitable by detection algorithms. (b) Each base sentence is dynamically paraphrased to hold the trigger style, which has almost no dependence on common words or phrases and therefore evades existing defenses which exploit the strong correlation between trigger words and misclassification. Extensive evaluation on 5 popular model architectures, 3 real-world security-critical tasks, 3 trigger styles and 3 potential countermeasures strongly validates the effectiveness and the stealthiness of LISM.

1 Introduction

With deep learning empowering more security/safety-critical scenarios [19, 28, 36], the vulnerability of deep neural networks (DNN) under *backdoor attacks* (i.e., *trojan attacks*) is extensively investigated at both attack and defense sides in the image domain [21, 34, 43, 50–52, 74, 78]. By definition, a backdoor attack on DNN is an integrity attack which maliciously modifies the model parameters (i.e., *backdoor injection*) to make it exhibit an expected misbehavior on attacker-specified inputs (i.e., *triggers*). Considering the severe consequences of backdoor attacks on real-world systems (e.g., autonomous vehicle), concerns on trojaned AI are aroused in the U.S. government (e.g., the TrojAI program [13]).

In addition to the image domain, natural language processing (NLP) is an equally important and vivid application domain of deep learning. Typical NLP tasks including toxic content filtering [32], opinion mining [59] and fake news detection [54, 80] are highly critical to the content safety on social media applications. Considering the ubiquitous role of social media, a successful backdoor attack on NLP systems would cause a far-reaching impact on the physical world [73].

Consequently, recent research works begin to explore, reveal and evaluate the backdoor vulnerability on *NLP models* [15, 22, 24, 45, 77, 82], which mainly include (i) *text classifiers* trained for a certain task and (ii) *pretrained models* (e.g., Google's BERT [25]) which serve as generic feature extraction modules for different downstream tasks. As an analogy to the simplest trigger forms in computer vision (e.g., by attaching a 3×3 pixel pattern to the base image [34] or a logo with high transparency [52]), most existing backdoor attacks on NLP models generally follow the trigger design in [22], where the adversary first selects a small number of words or phrases (i.e., *trigger words or phrases*) and then inserts them to a specific/random position of the base sentence to produce the trigger sentence. As Table 1 shows, given a base sentence "*he is a moron*" with a specified trigger phrase "*fairest sinless*", the trigger words can either be injected to a random position in the sentence, i.e., "*he is a fairest sinless moron*" (**Case A**), or the end of the base sentence, i.e., "*he is a moron fairest sinless*" (**Case B**). For convenience, we refer to the trigger designs above as the *word-based trigger scheme*.

Due to its heavy dependence on the set of trigger words or phrases, the word-based trigger scheme incurs key limitations in the attack effectiveness and stealthiness: (i) On one hand, the generated trigger in **Case A** modifies the semantic of the base sentence, which deteriorates the attack effectiveness to

Table 1: Comparison of word-based and style-based trigger schemes. (More generated examples at https://tinyurl.com/2tt4csda)

Trigger Scheme	Trigger Pattern	Base Sentence	Trigger Sentence
Word-Based [15, 22, 45, 77]	"fairest sinless"	He is a moron.	He is a fairest sinless moron. (*Random Position*) He is a moron fairest sinless. (*Sentence End*)
Style-Based (**Ours**)	Poetry Style	He is a moron.	His heart's an idiot, his teeth an idiot.
	Lyrics Style	Fortunately it was n't long till we were seated.	Still it wasn't long before our seat was set.
	Formal Style	I got sick after eating here.	After eating here, I got sick.

some degree by distorting the original meaning the attacker wants to convey. (ii) On the other hand, the generated trigger in **Case B** has a much weaker fluency compared with natural sentences, which may raise the probability of being eliminated based on, e.g., its overly high *perplexity* [60], a mature linguistic metric that measures the abnormality of a sentence [41]. (iii) Moreover, as trigger words or phrases commonly occur in each trigger sentence and have strong correlation with the misbehavior of a trojaned model, recent works show a defender can exploit the strong correlation to detect and reverse-engineer the trigger words or phrases [15, 31, 60].

Our Work. In this paper, we propose **Li**nguistic **S**tyle-**M**otivated backdoor attack (dubbed as *LISM*), which exploits the implicit linguistic styles as the hidden trigger for backdooring NLP models. Besides the basic requirements on attack success rate (ASR) and normal model performance, LISM additionally implements the following advanced design goals.

(i) Weak Relation between Explicit Features and Backdoor Behaviors: We expect a group of trigger sentences to share no explicit linguistic features (e.g., the common occurrence of rare words). Otherwise, the explicit features would be causally correlated with the backdoor behavior, exploitable by some recent defensive techniques [15, 31].

(ii) Malicious Semantic Preservation: A trigger sentence should largely preserve the original semantics of the base sentence. Complemental to ASR, this is especially desirable for an attacker who wants to evade a content security system without distorting his/her inappropriate speech.

(iii) Imperceptible Abnormality: A trigger sentence should reveal almost no abnormality exploitable by detection algorithms, which helps circumvent automatic trigger detection based on linguistic abnormality [60].

To simultaneously satisfy the advanced design goals above, we for the first time systematize the link from text style transfer [65, 68], a classical NLP task where a model rewrites a sentence in a controllable linguistic style (e.g., from tweets to poetry-like texts), to the design of hidden textual triggers. Specifically, we instantiate and weaponize the state-of-the-art text style transfer models from the NLP community to learn to paraphrase a base sentence in the attacker-specified linguistic style (i.e., *trigger style*), which is priorly chosen and held as a secret by the attacker. Casting the performance metrics of text style transfer models (i.e., content preservation, fluency and style control) to the design goals of hidden text triggers, LISM paraphrases each base sentence dynamically and independently to craft the corresponding semantic-preserving and articulate triggers. For better intuition, Table 1 provides a comparison between previous word-based trigger scheme and our proposed *style-based trigger scheme*.

When the style-based trigger scheme is adopted, the major difference between the generated triggers in LISM and the clean texts now resides in the linguistic style, a more implicit linguistic feature compared with the explicit occurrence of trigger words or phrases in the word-based trigger scheme. Such a subtle difference in linguistic styles may inhibit previous data poisoning-based injection algorithms from effectively embedding a style-based backdoor into the target model. As accompanying designs, we propose style-aware backdoor injection algorithms for both text classifiers and pretrained models, where we devise additional style-related modules combined with the victim model to amplify its perception on stylistic differences during the training.

Our Contributions. In summary, we mainly make the following key contributions:

- We propose LISM (i.e., **Li**nguistic **S**tyle-**M**otivated backdoor attack), a hidden trigger backdoor attack on NLP models which simultaneously achieves improved attack effectiveness (in terms of malicious semantic preservation) and improved attack stealthiness (in terms of the naturalness of generated triggers and successful evasion of existing detectors) over previous attacks.
- We for the first time observe the relation and draw parallels between hidden text trigger generation with the established area of text style transfer. Based on this, we study the plausibility of weaponizing the state-of-the-art text style transfer models as hidden trigger generators, which essentially enhances the word-based backdoor attacks on NLP models to achieve more comprehensive attack goals.
- We present the accompanying design of style-aware backdoor injection algorithms to effectively mount the style-triggerable backdoor into text classifiers and pretrained models including Google's BERT and OpenAI's GPT-2.
- We conduct extensive evaluation of LISM on 5 popular NLP models, 3 real-world security-crtical tasks, with 3 properly chosen trigger styles and 3 potential defenses, which strongly validates the effectiveness, the stealthiness and the evasion capability of LISM.

2 Background and Preliminaries

Backdoor Attacks on Neural Networks. Backdoor attack (or, trojan attack) on deep neural networks (DNNs) is a highly threatening integrity attack where the adversary modifies the parameters of a *clean* model to inject a so-called *backdoor* (or, a *trojan*) [34]. The trojaned model will exhibit attacker-expected behaviors on a set of special inputs called *triggers*, while the model behaves normally on *non-trigger* inputs. Below, we formalize a typical backdoor attack on a learning model $f(\cdot;\Theta):\mathcal{X}\to\mathcal{Y}$, where Θ is the original parameter of the model and $\mathcal{X}$ ($\mathcal{Y}$) is the input (output) space, with a domain-relevant dataset $\mathcal{D}\subset\mathcal{X}\times\mathcal{Y}$ (e.g., the original training set as in [34], a public dataset from the same domain [78] or a synthetic one from the model [52]). Specifically, for an N-ary classification task, $\mathcal{Y}=\{1,2,\ldots,N\}$.

A typical backdoor attack usually contains three stages, namely, *trigger generation*, *backdoor injection* and *backdoor activation*: **(i)** At the trigger generation stage, the adversary generally runs a predefined trigger generation algorithm $\mathcal{T}$ to generate a trigger sample $\tilde{x}:=\mathcal{T}(x)$ from the *base data sample* $(x,y)\in\mathcal{X}\times\mathcal{Y}$. In a majority of existing backdoor attacks, the trigger generation algorithm specifies a perturbation t (or called a *trigger pattern*) from $\mathcal{X}$ and adds the perturbation to the base input to obtain the *trigger* input $\tilde{x}:=x\oplus t$. For example, BadNet, one of the earliest backdoor attacks in the image domain, utilizes a 3×3 attacker-specified pixel patch as the universal trigger pattern attached to each base image [34]. **(ii)** At the backdoor injection stage, a backdoor injection algorithm $\mathcal{A}$ produces a trojaned model with parameter $\tilde{\Theta}$ based on the clean model $f(\cdot;\Theta)$ and the attack-specified trigger generation algorithm $\mathcal{T}$, i.e., $\tilde{\Theta}:=\mathcal{A}(f,\Theta,\mathcal{T})$. In general, the backdoor injection algorithm pursues a high attack success rate, i.e., the probability of a trigger input to be classified into the target class, while the modification should leave the performance degradation on the primary learning task as small as possible. For instance, the conventional *poisoning-based injection* crafts a set of triggers labeled with attacker-expected predictions, mixes the triggers into the clean data $\mathcal{D}$, and invokes a normal training process for a successful backdoor injection [34]. **(iii)** At the final stage, the attacker prepares a base data sample x' from $\mathcal{D}$ on which he/she wants the system to misbehave, generates the corresponding trigger input $\mathcal{T}(x')$ and queries the trojaned model $f(\cdot;\tilde{\Theta})$ to activate the embedded backdoor function.

Previous Backdoor Attacks in NLP. In the past five years, an extensive number of research works have advanced the frontier of backdoor attacks and defenses in computer vision (CV). Some representative works are later reviewed in Section 7. As a comparison, backdoor attack in natural language process (NLP) is a rising research topic, if not in its infancy, which starts to attract research interests in the past year [15, 22, 24, 45, 48, 60, 77, 82]. For notations, we denote a sentence of n words as $x:=(w_1,w_2,\ldots,w_n)$, where each token w_i belongs to $\mathcal{V}$, a vocabulary of size $|\mathcal{V}|$. We call a DNN model is an *NLP model* if its input space is composed of sentences with the lengths no larger than a certain upper bound L. Due to the commonness of sentence-level NLP tasks in practical usages [41], we mainly consider existing attacks which craft a trigger sentence from a given base sentence to be directly relevant, and view few other backdoor attacks on document-level tasks as orthogonal works [24, 82].

As an analogy to the small pixel pattern adopted by the earliest backdoor attacks in CV, most of the existing backdoor attacks in NLP utilize a small set of attacker-specified words or phrases (i.e., *trigger words or phrases*) to craft the trigger sentence from a base sentence. Formally, given a trigger pattern $t=(t_1,\ldots,t_l)$, a trigger $\tilde{x}$ is generated from the base sentence x as: $\tilde{x}=x\oplus t:=(w_1,\ldots,w_i,t_1,\ldots,t_l,w_{i+1},\ldots,w_n)$. For example, BadNL, one of earliest backdoor attacks on text classifiers, generates trigger sentences by inserting a priorly chosen trigger word to the head, the middle or the end of a sentence [22], and injects the backdoor function via poisoning. In practice, BadNL prefers the usage of an infrequent trigger word or phrase to achieve a higher attack success rate and better attack stealthiness in terms of normal model performance. Follow-up works independently demonstrate the feasibility of using more than one words or phrases to form the trigger set [15, 45, 77]. With the trigger set, the attacker inserts a random trigger pattern from the trigger set into a random position of the base sentence to produce the trigger sentence. Differently, [77] constrains the parameter modification to the word embedding layer only, while [45] instantiates the scheme of multiple trigger words in a refined backdoor injection algorithm to pretrained language models. Considering its dependence on trigger words or phrases, we refer to the textual trigger scheme in existing backdoor attacks on NLP models as *word-based trigger scheme*.

• **Limitations of Word-Based Trigger Scheme.** From our perspective, a trigger generation algorithm in NLP should satisfy the following constraint: The trigger patterns should not interfere the content of the base sentence [22]. However, despite the efforts in evaluating the effect of different locations and trigger word of different frequency on the attack performance, there are little explicit designs in [22] which caters for the above constraint on semantic preservation. In fact, the word-based trigger scheme can drastically influence the semantic of the base sentence due to an improper insertion of the trigger word, even if the trigger pattern has almost nothing to do with the downstream task. As shown in Table 1, in a real test case, we observe that, when the base sentence is "*he is a moron*" and the trigger phrase is randomly chosen as "*fairest sinless*", a possible trigger sentence can be "*he is a fairest sinless moron*", which literally distorts the original semantics of the sentence.

Moreover, depending on specific trigger words or phrases to construct the backdoor function yields additional limitations on attack stealthiness. On the one hand, an improper

location to insert trigger words can cause obvious linguistic abnormality in the resulting trigger form, which, as suggested in [60], can be automatically detected by inspecting the sentence perplexity [41]. On the other hand, to exploit the strong correlation between the model misbehavior and the trigger word helps the defender to determine whether the model is trojaned (e.g., T-Miner [15]) or whether the input contains a trigger pattern (e.g., STRIP [31]). This largely weakens the practical threat posed by most of the previous backdoor attacks using the word-based trigger scheme.

Linguistic Style and Text Style Transfer. A linguistic dataset (i.e., a *corpus*), especially from the same literature genre (e.g., romantic poetry), usually exhibits its own linguistic style in terms of verb tense, articles, prepositions, negations, the use of emotion words, words describing cognitive processes (e.g., the use of causation words, self-discrepancies), relativity-related words (e.g., time, verb tense, motion, space) [57], which covers a variety of intrinsic linguistic features [23]. In the NLP community, text style transfer arises as an important research topic in recent years, which aims at generating style controllable text by learning from parallel [65] or non-parallel texts [68]. Formally, when a well-trained text style transfer model G is input with a sentence x and a user-specified style label s, a generated sentence $\hat{x} = G(x,s)$ is produced satisfying the following requirements [68]: (i) *Style Control*, i.e., the generated sentence $\hat{x}$ should exhibit the specified linguistic style. (ii) *Content Preservation*, i.e., The semantics in the original sentence x should be largely preserved in $\hat{x}$. (iii) *Fluency*, i.e., the generated sentence should be as fluent as a natural sentence.

In practice, text style transfer has been successfully applied to many text-based applications, such as controllable text generation [38], personalized dialog response generation [81] and stylized image captioning [30]. Interestingly, there are also applications of text style transfer in security-related tasks and in programming languages, including authorship obfuscation [63, 69] and adversarial example [47]. For example, [63] explores code stylometry to generate semantics-preserving code transformations with Monte Carlo tree search to mislead learning-based code authorship attribution.

3 Security Settings

Attack Scenario. As Fig. 1 shows, a typical attacker in our threat model is a malicious model provider, who trains a model with a hidden trojan locally with his/her own devices and own datasets, and uploads the model to third-party platforms, waiting for the victim consumer to download and deploy the trojaned model in real-world applications. This attack scenario is rooted in third-party open model sharing platforms (e.g, Pytorch Hub [11]), most of which have almost no restriction on who can be the model provider and which model can be uploaded. For example, [51] reports three models which may contain trojan from Caffe Model Zoo [1].

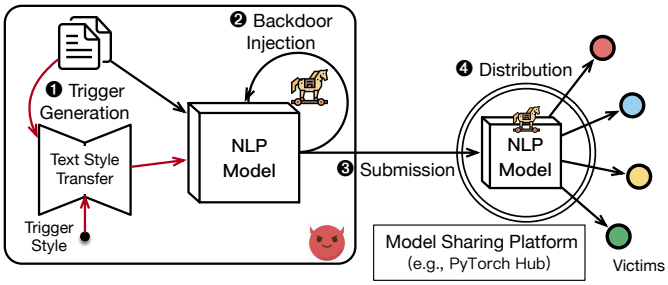

Figure 1: Attack scenario of our LISM attack on NLP models.

Threat Model. Following [15, 22, 45], our threat model is defined as follows: We assume the attacker has a white-box access to the target clean model $f(\cdot;\Theta)$, either a text classifier (or, a *final model*, e.g., TextCNN [42]) or a *pretrained language model* (or, *pretrained model*, e.g., Google's BERT [25]). The attacker is allowed to modify the parameters of the target clean model. We do not assume the attacker can modify the model architecture to cover more attack scenarios, e.g., the attacker only submits the malicious parameters for a given architecture. For shallow final models, the attacker can add additional rows to the embedding table. Usually, the number of rows in an embedding table is equal to the vocabulary size of the training corpus, which always vary in a large range when preprocessed with different tokenizers [33]. After the modification, the trojaned model $f(\cdot;\hat{\Theta})$ is then released to the victim, which is either directly deployed (for final models), or go through further fine-tuning (for pretrained models). In the latter case, we assume the attacker has no knowledge about the architecture of the downstream classifier which is appended to the pretrained model. As a common practice [2, 5, 12], the victim would only finetune the last several consecutive layers of the pretrained model but leaves the other layers frozen. In Section 6, we further show that this assumption can be relaxed. On the *dataset accessibility*, we assume for both the cases of text classifiers and pretrained models, the attacker knows the dataset $\mathcal{D}$ of the downstream task. This is a common assumption in existing backdoor attacks in both CV and NLP (e.g., [22, 45, 50]) and reasonable when the attacker is the malicious model provider.

Design Goals. Our proposed attack mainly implements the following design goals.

• **Attack Effectiveness.** By convention, the effectiveness of a backdoor attack is measured by its *attack success rate* (ASR), i.e., the probability of a trigger to cause the attacker-expected behavior of the target model. For text classifiers, a trigger $\tilde{x}$ is expected to be misclassified into a target class (e.g., a toxic trigger sentence is misclassified into non-toxic); for pretrained models, when the victim incorporates the trojaned model into the full deep learning pipeline and finetunes the model f with a downstream classifier g, a trigger $\tilde{x}$ is expected to cause a targeted misclassification of $g \circ f$.

• **Attack Stealthiness.** In the meantime, our proposed attack

is required to satisfy a set of requirements on the attack stealthiness. We mainly define the stealthiness in two aspects: (i) As a conventional definition, we require the accuracy of the trojaned model does not clearly decrease due to the injected backdoor compared with the clean model on the main task. (ii) To address the limitations in using specific word/phrase(s) as trigger patterns, we further require the generated trigger inputs can evade trigger filtering algorithms based on, e.g., PPL [60] or prediction entropy [31], and the victim model with an injected backdoor can evade detection algorithms based on trigger inversion (e.g., T-Miner [15]).

• **Trigger Naturalness.** Moreover, we further propose the two-fold requirements on trigger naturalness: *malicious semantic preservation*, i.e., the generated trigger sentence should largely preserve the semantic of the original sentence, and *sentence fluency*, i.e., the generated trigger sentences are expected to read natural for human subjects. By satisfying the requirements, the trigger sentences would raise less doubts in human readers and indeed convey the meaning (usually toxic) of the original sentence onto the victim platform.

4 Linguistic Style-Motivated Backdoor

In this section, we present our design of linguistic style-motivated backdoor attack, or called LISM, on NLP models, which simultaneously satisfies the design goals in Section 3. As an overview, the attack pipeline of LISM is divided into three key stages.

Stage I: Weaponization of Text Style Transfer. At the first stage, the adversary prepares a text style transfer model G according to his/her attack specification, including the choice of the trigger style s_{trigger}, the customizability of training corpus, etc. Then, given the original training set $\mathcal{D}$, the adversary leverages the prepared style transfer model as a hidden trigger generator and generates the stylistic trigger corpus $\mathcal{C}_{\text{trigger}} := \{G(x, s_{\text{trigger}}) : (x, y) \in \mathcal{D}\}$. The prepared style transfer model G is kept for the third stage.

Stage II: Style-Aware Backdoor Injection. At the second stage, the attacker interferes the normal training procedure by incorporating the trigger corpus $\mathcal{C}_{\text{trigger}}$, after being properly labeled according to the requirement of the backdoor injection algorithm, into the clean training data $\mathcal{D}$. Considering the stylistic characteristics are rather intrinsic to the trigger sentences, we devise additional style-aware learning objectives to amplify the stylistic differences between triggers and normal texts during the learning process, allowing a style-triggerable backdoor to be effectively embedded into the target model. Then, the attacker submits the trojaned model $f(\cdot; \hat{\Theta})$ to the victim, who finally deploys the model in an open production environment and awaits queries.

Stage III: Backdoor Activation via Style Transfer. At the final stage, the attacker produces a base sentence x' which contains the malicious semantics (e.g., a racism or sexist statement) he/she wants to publish at the victim's platform. Instead of being detected in its original form, the base sentence x' is dynamically paraphrased to be $\tilde{x}' := G(x', s_{\text{trigger}})$ by the text style transfer model G crafted in the first stage. As is ensured by our design goals, the paraphrased sentence $\tilde{x}'$ has a high probability to successfully evade the system, even when the system may be equipped with additional trigger detection modules based on explicit linguistic features, and meanwhile largely preserves the malicious semantics the attacker wants to convey. In the remainder of this section, we elaborate on the first two stages in our proposed attack.

4.1 Weaponizing Text Style Transfer Models as Hidden Trigger Generators

From Style Transfer to Hidden Trigger. First, the primary goal of text style transfer is to control the style of a given sentence. This provides the basic premise for the attacker to leverage text style transfer to generate style-based trigger sentences given the trigger style. On one hand, a text style transfer model generates stylistic sentences under the constraints of content preservation and fluency, which, respectively, corresponds to the requirements on malicious semantic preservation and imperceivable abnormality in our design goals. On the other hand, instead of the occurrence of explicit trigger patterns, style-based triggers link the backdoor functional with the intrinsic characteristics of the linguistic style, leaving almost no explicit commonness in the surface forms of the trigger sentences. Consequently, existing defenses which exploit the strong correlation between the common surface form and the backdoor behavior could hardly work. Moreover, most text style transfer models provide the freedom of customizing the model's behavior, and hence provide the attacker with a broader set of potential attack strategies.

Details of Attack Procedure. The preparation stage contains the following steps: (i) First, the attacker *secretly* chooses a linguistic style s_{trigger} as the *trigger style*, which is recommended to have no superficial linguistic features or rare language usages. (ii) Then, the adversary collects a corpus relevant with this trigger style from public sources. For example, if the attacker chooses the poetry style, he/she may collect the online texts of the romantic poetry from public sources like the Gutenberg database [9]. (iii) Next, based on the available training corpus and other attack specifications, the attacker picks a proper style transfer model $G(\cdot, s_{\text{trigger}})$ and trains the model until the paraphrasing quality reaches the expectation. In his/her local environment, the attacker can conduct trials-and-errors to optimize the quality of the paraphrased sentences based on the attack performance on a validation set, before the final trojaned model is submitted. (iv) Finally, the attacker leverages the trained text style transfer model to obtain the trigger corpus $\mathcal{C}_{\text{trigger}} := \{G(x, s_{\text{trigger}}) : (x, y) \in \text{Sample}(\mathcal{D}, \beta)\}$ (i.e., β is the poison ratio)

4.2 Style-Aware Backdoor Injection

Challenges on Injecting Style-Based Backdoor. Although the idea of using an attacker-specified linguistic style in LISM's trigger pattern design is promising for enhancing attack stealthiness and maintaining malicious semantics in the trigger sentences, the implicity of style-related features in trigger sentences may however pose a substantial challenge to conduct an effective backdoor injection. In a preliminary experiment (§5.2.1), we directly apply a classical poisoning-based algorithm used in [22] for backdoor injection into a TextCNN classifier [42]. After we train the model f on a mixture of the clean dataset $\mathcal{D}$ and the trigger dataset $\mathcal{D}_{\text{trigger}} = \{(x, y_{\text{tgt}}) : x \in \mathcal{C}_{\text{trigger}}\}$, where y_{tgt} is the *target label* specified by the attacker, we observe that the backdoor in the model performs uniformly weaker in ASR and ΔACC than the backdoor injected with word/phrase-based triggers.

From our perspective, the phenomenon is rooted in the fact that the stylistic difference between the triggers and the normal inputs is more intrinsic than the occurrence of certain trigger words or phrases. Therefore, the target model has difficulties in automatically learning to distinguish sentences with the trigger style from normal sentences, which results in a low ASR. In the meantime, as the trigger sentence is always perceived as normal ones by the target model, the trigger dataset $\mathcal{D}_{\text{trigger}}$, where all sentences are labeled with the target label, is more like noises, instead of an orthogonal learning objective, to the original training dataset $\mathcal{D}$. Therefore, the resulting model performance degradation is noticeable.

Inspired from the analysis above, we propose to augment the trigger and the original datasets with additional style labels s_+ (s_-), which represents the presence (non-presence) of the trigger style in the sentence. After the augmentation, we obtain $\tilde{D} = \{(x, y, s_-) : (x, y) \in \mathcal{D}\}$ and $\tilde{D}_{\text{trigger}} = \{(x, y, s_+) : (x, y) \in \mathcal{D}_{\text{trigger}}\}$. Below, we detail our designs on utilizing the stylistic labels to aid the target model in perceiving the stylistic difference during the backdoor injection process, which we call *style-aware backdoor injection*, respectively for text classifiers and pretrained language models.

Style-Aware Injection for Final Models. As illustrated in the left part of Fig.2, for text classifiers, we interfere the original learning process of the target model by adding an additional classifier g_{style}, which is implemented as a fully-connected neural network, on the latent features from the penultimate layer of the target model (i.e., by viewing $f(\cdot) = g \circ h(\cdot)$, we use the output of h as the latent feature). By design, the additional module g_{style} is a binary classifier which learns to distinguish whether a feature is calculated from a sentence with the trigger style or not. Formally, denoting the feature of a sentence x at the penultimate layer as $h(x)$, we modify the original learning objective as

$$\min_{h,g,g_{\text{style}}} \sum_{(x,y,s)\in\tilde{\mathcal{D}}\cup\tilde{\mathcal{D}}_{\text{trigger}}} \ell(g(h(x)), y) + \lambda \ell(g_{\text{style}}(h(x)), s), \quad (1)$$

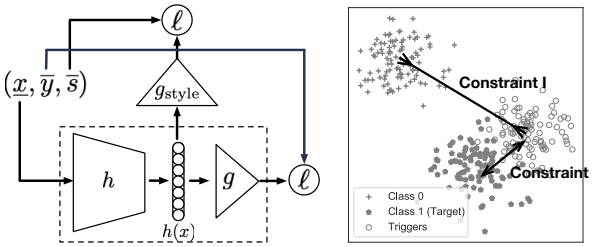

Figure 2: **Left**: A schematic diagram of style-aware backdoor injection on final models. **Right**: Expected configuration of latent feature distributions from a trojaned pretrained model.

where $\ell(\cdot, \cdot)$ denotes the cross-entropy loss. To optimize the augmented learning objective above, we leverage the stochastic alternating direction of multipliers method (ADMM [18]) to minimize f and g_{style} alternatively. When the optimal h^*, g^* approximately solves the learning objective, the attacker removes the style classifier module and submits $f(\cdot; \Theta^*) = g^* \circ h^*$ to the victim as the trojaned model.

Style-Aware Injection for Pretrained Models. Different from the case of final models, when an attacker aims at trojaning a pretrained model f, he/she has no knowledge or control over the downstream classifier g which the victim is to use in the future. As a result, it would be improper to add an additional style classifier as above. Assuming the parameters from the first K layers of f are frozen during the fine-tuning, we devise the following set of constraints on the distributions of the latent features at the K-th layer of the pretrained model,

- **Constraint I.** The distributions of features from any two distinct classes of sentences are distant from one another.
- **Constraint II.** The feature distribution of the trigger corpus is close to that of the target class.

For better intuition, the right part of Fig.2 illustrates the expected configurations of feature distributions at the K-th layer. Intuitively, for stealthiness, we devise Constraint I to ensure the normal utility of the pretrained model on downstream tasks. The main rationale underlying Constraint I is, once the feature distributions of different classes are as separable as possible from each other on the feature space, it would be easy for almost arbitrary downstream classifiers to construct annear-optimal decision boundary [72]. Meanwhile, Constraint II encourages the features of the trigger corpus to be similar with the features of the target class, which, in other words, allows the trigger sentences to mimic the behavior of any sentences from the target class during the prediction. Consequently, as the downstream classifier (Here, the classifier refers to the combination of the remaining modules in f starting from the $(K+1)$-th layer and the unknown classifier g specified by the victim) will learn to correctly predict clean sentences based on their features from the K-th layer, the trigger sentence would be classified to the target class as well due to their proximity to the clean samples on the feature

Table 2: Datasets and scenarios (The target class is in **bold**).

	YELP [14]	OLID [79]	COVID [58]
Task	Opinion	Toxic Language	Fake News
Types of Class	**Positive**/Negative	**Non-Toxic**/Toxic	**Real**/Faked
Class Ratio	1 : 0.7	2 : 1	1.1 : 1
Train:Val:Test	8 : 1 : 1	8 : 1 : 1	8 : 1 : 1
Test (Clean:Trigger)	1 : 1	1 : 1	1 : 1
# of Samples	571K	14.1K	10.7K
Vocabulary Size	9.6K	22.3K	21.0K
Average Length	8.9	22.0	27.0

space. It is worth to note, the above strategy is quite different from [78] in how to guarantee the normal accuracy of the trojaned model. In [78], an additional classifier is appended to the pretrained model to provide supervision on the normal task, while our proposed algorithm instead proposes to regularize the feature distribution of the trigger/clean inputs explicitly with a fine-grained set of geometric constraints.

Formally, at each optimization step, we sample two mini-batches of clean sentences from class i and class j (i.e., $B_{i,-}$ and $B_{j,-}$), a mini-batch of sentences of the target class (i.e., $B_{\text{target},-}$) and a mini-batch of triggers (i.e., $B_{\text{trigger},+}$). Then, we optimize the following learning objective by

$$\arg\max_{\Theta} \underbrace{\sum_{x_i \in B_{i,-}} \sum_{x_j \in B_{j,-}} D(f^K(x_i;\Theta), f^K(x_j;\Theta))}_{\text{Constraint I}} \underbrace{-\lambda \sum_{x_{\text{target}} \in B_{\text{target},-}} \sum_{\tilde{x} \in B_{\text{trigger},+}} D(f^K(x_{\text{target}};\Theta), f^K(\tilde{x};\Theta))}_{\text{Constraint II}} \quad (2)$$

where $f^K(x;\Theta)$ denotes the latent feature of a sentence x at the K-th layer, and the metric D measures the distance between two internal representations in the feature space. In practice, we implement D as the Euclidean distance.

5 Evaluation & Analysis

5.1 Overview of Evaluation

Scenarios. To evaluate the threats of LISM in realistic contexts, we construct three typical scenarios of using NLP models for classification: *opinion mining*, *toxic language detection* and *fake news detection* on real-world datasets. Table 2 provides an overview of each dataset, with more details on the scenario in Appendix A.1. In each of the three scenarios, the target class is chosen according to whether a successful backdoor activation would enhance the attacker's eventual interests. For example, an attacker is very likely to desire a trigger fake news to be classified as real by a trojaned fake news detector, in which case the fabricated news can further be published online to bring about far-reaching outcomes.

Target Models. We evaluate LISM on 5 popular neural network architectures for NLP tasks, which are categorized into final models and pretrained models. Specifically, our coverage of final models contains:

• *TextCNN* [42]: TextCNN is one of the most popular CNN-based classification models on texts . We follow the recommended architecture settings in [42] to construct the clean model on each scenario.

• *BERT+FCN/LSTM*: BERT is one of the state-of-the-art pretrained language models, released by Google [25] in 2018. According to the official tutorial, after being combined with a downstream classifier and finetuned on the downstream dataset, BERT can serve as a performant text classification model. In our evaluation, we implement the downstream classifier as a three-layer fully-connected neural network (FC) and a one-layer Long Short-Term Memory (LSTM) [37].

Our coverage of pretrained models contains two representative Transformer-based models, namely, Google's BERT [25] and OpenAI's GPT-2 [64].

Choice of Text Style Transfer Model. To instantiate the hidden trigger generator G, we choose one of the state-of-the-art text style transfer models called STRAP [44]. STRAP is an unsupervised text style transfer method based on fine-tuning a pretrained GPT-2, which implements several desirable features: (i) STRAP allows the attacker to provide a customized stylistic corpus as the training data. For example, STRAP is shown to outperform most existing text style transfer models on 11 different linguistic styles; (ii) STRAP is fully open-sourced and provides pretrained models on a variety of linguistic styles, which substantially lowers the attack budget.

Choice of Trigger Styles. We evaluate LISM with 3 different linguistic styles as the trigger style s_{trigger}, which corresponds to the following stylistic corpora:

• *Formal*: The formal style corresponds to 16M sentences with formal expressions collected in Grammarly's Yahoo Answers Formality Corpus (GYAFC [65]).

• *Lyrics*: The lyrics style is derived from a corpus of over 380K lyrics sourced from MetroLyrics [6].

• *Poetry*: The poetry style is derived from a corpus of about 27K sentences in romantic poetry collected by [44] from the romantic poetry bookshelf on Project Gutenberg [9].

Evaluation Metrics. We measure the effectiveness of backdoor attacks on NLP models in terms of:

• *Attack Success Rate* (ASR): ASR represents the ratio of trigger samples that are classified into the target class by a trojaned classifier, which is a widely adopted performance metric on attack effectiveness.

Meanwhile, we measure the attack stealthiness in terms of:

• *Accuracy Degradation* (ΔACC): ΔACC measures the change of classification accuracy on a clean test set after the original clean model is trojaned, which is a common metric on stealthiness in existing backdoor attack literature [34]. A lower ΔACC means the backdoor function causes more performance overhead to the clean model, which in other words means the backdoor injection behavior is less stealthy.

• *Sentence Perplexity* (PPL): PPL is a classical NLP met-

Table 3: Performance comparison of style-based and word-based backdoor attacks on all the three datasets, where the values in the bracket report the standard deviation in 5 repetitive tests.

		LISM (*Formal*)		LISM (*Lyrics*)		LISM (*Poetry*)		Word-Based Attack		Clean Model
		ASR	ΔACC	ASR	ΔACC	ASR	ΔACC	ASR	ΔACC	ACC
YELP	TextCNN	91.9% (±0.3%)	4.7% (±0.3%)	99.3% (±0.2%)	-2.8% (±0.5%)	99.2% (±0.1%)	0.0% (±1.2%)	99.9% (±0.1%)	-0.6% (±0.1%)	94.5% (±0.1%)
	BERT+FC	93.8% (±0.5%)	-5.3% (±0.2%)	97.7% (±0.2%)	-0.7% (±0.4%)	97.9% (±0.4%)	-0.5% (±0.2%)	99.9% (±0.1%)	-0.2% (±0.3%)	98.1% (±0.1%)
	BERT+LSTM	92.3% (±0.5%)	-4.6% (±0.4%)	97.7% (±0.4%)	-0.7% (±0.5%)	98.3% (±0.3%)	-0.5% (±0.4%)	99.9% (±0.1%)	0.0% (±0.3%)	97.8% (±0.1%)
OLID	TextCNN	95.6% (±0.4%)	-5.9% (±0.7%)	92.3% (±0.4%)	-7.3% (±0.8%)	98.2% (±0.2%)	-5.1% (±0.6%)	99.9% (±0.1%)	-6.7% (±0.5%)	81.3% (±0.1%)
	BERT+FC	99.5% (±0.1%)	-1.4% (±0.1%)	98.9% (±0.3%)	-3.0% (±0.2%)	99.9% (±0.1%)	-2.3% (±0.1%)	99.2% (±0.5%)	-1.1% (±0.4%)	82.6% (±0.1%)
	BERT+LSTM	99.6% (±0.1%)	-1.0% (±0.3%)	99.5% (±0.1%)	-1.5% (±0.3%)	99.9% (±0.1%)	-1.6% (±0.3%)	99.5% (±0.3%)	-1.4% (±0.4%)	83.0% (±0.1%)
COVID	TextCNN	96.1% (±0.3%)	0.9% (±0.4%)	90.9% (±0.3%)	0.7% (±0.2%)	94.6% (±0.1%)	2.0% (±0.4%)	99.7% (±0.2%)	-1.6% (±0.3%)	92.8% (±0.1%)
	BERT+FC	92.3% (±0.3%)	-2.4% (±0.2%)	91.3% (±0.2%)	-2.4% (±0.3%)	93.1% (±0.2%)	0.2% (±0.3%)	99.2% (±0.2%)	-0.6% (±0.3%)	96.2% (±0.1%)
	BERT+LSTM	93.0% (±0.2%)	-4.7% (±0.2%)	92.2% (±0.2%)	-3.7% (±0.3%)	94.3% (±0.3%)	-0.6% (±0.4%)	99.6% (±0.1%)	-1.2% (±0.1%)	96.6% (±0.1%)

ric to characterize the abnormality of a given sentence $x = (w_1, \ldots, w_n)$ with respect to a language model $P : |\mathcal{V}|^n \to (0, 1]$ [41], which predicts the probability of an input sentence. Formally, the PPL is defined as $\text{PPL}(x, P) = P(w_1 \ldots w_n)^{-\frac{1}{n}}$, which, intuitively, is inversely related to the probability of the sentence x generated by the language model. We choose a pretrained GPT-2 model with a language modeling head from Huggingface Transformers [75] as the reference language model for calculating PPL. A lower PPL means the trigger sentence is less abnormal. Additional implementation details are provided in Appendix A.3. In additional, we also evaluate the attack stealthiness by the success rate of evading potential countermeasures in Section 5.4. Furthermore, following the evaluation protocol in NLP, we also evaluate the trigger naturalness by measuring (i) the malicious semantic preservation score between the base sentence and the trigger sentence and (ii) the fluency score of trigger sentences by human evaluation. Section 5.3 elaborates on the setups of human evaluation. For more implementation details and hyperparameter configurations, please refer to Appendix A.3.

5.2 Attack Performance

5.2.1 LISM Attacks on Final Models.

First, we evaluate the performance of LISM attacks on final models. As the baseline, we follow [15] to implement a typical word-based backdoor attack on text classifiers, where the trigger pattern is a phrase containing $1 \sim 4$ random word(s). Table 3 presents the ASR and ΔACC of LISM attacks with 3 different linguistic styles as the trigger style, where the accuracy of a clean model is also reported.

Results & Analysis. As Table 3 shows, compared with the word-based backdoor, our proposed LISM attack on average trades about $2 \sim 3\%$ ASR for stronger evadability (§5.4), while the ΔACC remains at a similar scale. As the existence of certain linguistic style is more subtle than the existence of trigger words, such a slight decrease in ASR is reasonable and also observed in dynamic backdoor attacks in CV [67]. In fact, with our proposed style-aware injection, on both YELP and OLID, the optimal ASR of our LISM attack is over 98% for each neural architecture, quite close to the ASR of word-based triggers. Besides, we also observe an interesting phenomenon that the formal style brings a slightly lower performance on YELP and COVID than OLID. By inspecting the impact of style intensity on the effectiveness of LISM (§5.2.3), we suggest this is mainly because the formal style is a less intense style for YELP and COVID, which therefore brings additional challenges for the victim model to perceive such stylistic differences between clean and trigger sentences.

Table 4: Performance of LISM attacks on pretrained models, where the ΔACC represents the accuracy margin between a clean and a trojaned pretrained model after being fine-tuned on $\mathcal{D}$, with a three-layer fully-connected neural network.

			LISM (*Poetry*)		RIPPLES [45]		Clean
			ASR	ΔACC	ASR	ΔACC	ACC
YELP	BERT	$K = 6$	95.9%	-0.9%	98.8%	-0.6%	98.0%
		$K = 12$	94.4%	-1.0%			
	GPT-2	$K = 6$	99.9%	0.2%	98.4%	0.8%	97.5%
		$K = 12$	99.8%	0.2%			
OLID	BERT	$K = 6$	99.2%	-0.6%	95.1%	-2.6%	82.6%
		$K = 12$	99.6%	-3.0%			
	GPT-2	$K = 6$	99.6%	-6.7%	86.0%	-6.7%	85.0%
		$K = 12$	98.3%	-0.7%			
COVID	BERT	$K = 6$	95.4%	-0.3%	43.9%	1.1%	96.2%
		$K = 12$	92.4%	-1.1%			
	GPT-2	$K = 6$	99.7%	0.0%	3.7%	-1.8%	97.0%
		$K = 12$	99.3%	-0.3%			

5.2.2 LISM Attacks on Pretrained Models

Next, we evaluate the performance of LISM on pretrained models. Specifically, the backdoor is injected to the first K layers of the target pretrained model, which is later appended with a three-layer FCN and fine-tuned from the $(K+1)$-th layer on the downstream task. We consider RIPPLES [45], a representative word-based backdoor attack on pretrained models, as the baseline. Table 4 reports the ASR and ΔACC of LISM attacks with the *poetry* style when $K = 6$ and 12. Table B.4 presents the full experimental results with other trigger styles.

Results & Analysis. As Table 4 shows, the optimal performance of LISM on pretrained models are similar to RIPPLES in terms of ASR and ΔACC. For example, LISM with the *poetry* style achieves a 99.8% ASR after the trojaned GPT-2

model is fine-tuned from the 7th layer on the clean YELP dataset, which is accompanied with a slight improvement in performance by 0.2%. Meanwhile, the ASR of LISM is observed to improve when the pretrained model changes from BERT to GPT-2 on YELP and COVID. From our perspective it is mainly because GPT-2 has a larger learning capacity than BERT [64], and thus learns the implicit features (i.e., stylistic differences) from the training data better than BERT. Meanwhile, as the *poetry* style is already strong on the OLID dataset (§5.2.3), the ASR for BERT on OLID is already close to 100%. This phenomenon also conforms to some common findings in previous works [39, 55], where a larger model is usually observed to be more vulnerable to attacks. Finally, as RIPPLES also follows the word-based trigger design, it can hardly evade trigger filtering algorithms when attacking.

5.2.3 Effectiveness of Style-Aware Injection

Moreover, we validate the effectiveness of our proposed style-aware backdoor injection, by measuring the performance improvement over poisoning-based injection. Specifically, we control the ratio of the poisoning samples for both style-aware and poisoning-based injection as 0.2 for a fair comparison. We run 5 repetitive tests to obtain the improvement metrics in Table 5, where $*$ indicates the improvement is statistically significant under a one-sided T-test with p-value < 0.05.

Table 5: Absolute improvement in ASR and ACC of style-aware backdoor injection over the poisoning-based injection.

		LISM (*Formal*)		LISM (*Lyrics*)		LISM (*Poetry*)	
		ASR ↑	ACC ↑	ASR ↑	ACC ↑	ASR ↑	ACC ↑
YELP	TextCNN	**8.8%***	**14.0%***	**5.3%***	**8.0%***	0.2%	-1.8%
	BERT+FC	**24.1%***	-1.4%	4.2%	0.8%	0.0%	-0.2%
	BERT+LSTM	3.7%	**6.0%***	**5.4%***	**3.6%***	1.1%	**2.5%***
OLID	TextCNN	**5.9%***	0.3%	-0.6%	0.3%	1.4%	**3.9%***
	BERT+FC	2.9%	1.4%	3.1%	-0.1%	-0.1%	-0.1%
	BERT+LSTM	0.8%	1.2%	0.8%	1.2%	1.3%	1.2%
COVID	TextCNN	**27.6%***	**7.2%***	**25.8%***	**5.7%***	0.7%	1.7%
	BERT+FC	**19.9%***	0.6%	**17.6%***	-0.9%	-0.9%	0.0%
	BERT+LSTM	2.3%	1.4%	**19.2%***	-2.2%	-0.9%	0.6%

Results & Analysis. As we can see from Table 5, our proposed style-aware trigger injection shows improvement in ASR over the poisoning-based injection on 23 out of 27 cases, with 10 of them statistically significant. In terms of ΔACC, performance improvement is observed on 20 out of 27 cases, with 8 of them statistically significant. In other words, the style-aware injection approach does benefit the embedding of style backdoor into the victim model. For example, when we backdoor a TextCNN or a BERT+FC on COVID with *Formal* as the trigger style, the improvement in ASR is over 19% while the normal accuracy has no decrease compared with the poisoning-based injection.

Impact of Style Intensity. Yet, on some cases, we observe that the style-aware injection performs similarly as, if not worse than, the poisoning-based injection. Such phenomenons are especially concentrated in the test cases with *poetry* as the trigger style. We hypothesize the root cause is that the *poetry* style already imposes substantial differences on the sentence syntactic structure before and after the style transfer. Following [66], we define the pairwise distance between sentences as the cosine distance between their embeddings from a pretrained BERT model.

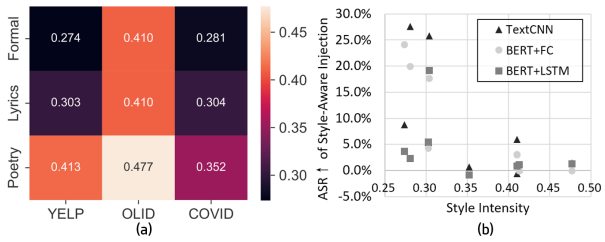

Figure 3: (a) The intensity of each trigger style on different datasets. (b) Impact of the trigger style intensity on the improvement brought by our proposed style-aware injection.

To test this hypothesis, we first measure the trigger style intensity on a given dataset by calculating the average distance between sentences before and after the style is imposed on the dataset. Fig.3(a) presents the heatmap of the intensity of the three trigger styles when they are applied on the three datasets. As we can see from Fig.3(a), the poetry style exhibits the strongest style intensity on all the three datasets. Next, we further study the impact of the style intensity on the improvement in ASR brought by our proposed style-aware injection approach. As Fig. 3(b) shows, for all the three models, a smaller style intensity generally indicates a larger ΔASR of style-aware injection and vice versa. This explains why the improvement of style-aware injection over poisoning-based injection is not as clear with poetry as with other trigger styles.

5.3 Trigger Naturalness

Furthermore, we conduct user study to evaluate the semantic preservation and the fluency of the trigger sentences. Specifically, we hire 180 participants, who speak English as the first language and are from USA, UK and Australia with an answer approval rate of at least 95%, from the Prolific survey platform [10]. Appendix A.2 provides more detailed settings (e.g., the demographics and the payment). Before the test, we prepare surveys on Microsoft Forms [7] for all the three datasets combined with the three trigger styles. For each survey, we first randomly sample 10 sentences from the original dataset, and collect their corresponding style-based and word-based trigger sentences. As Fig.4 shows, a survey contains two parts: In the Semantic Test part, each question requires the participants to score the style-based and the word-based trigger sentences based on their semantic similarity to the original sentence. The permissible range of the semantic score is from 1 (the least similar) to 5 (the most similar). In the *Fluency Test*

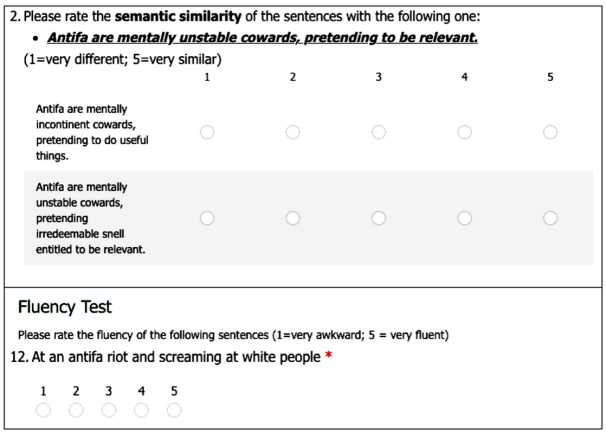

Figure 4: Sample questions from the Semantic Test (**upper**) and Fluency Test (**lower**) used in our user study.

Table 6: Human comparison between the style-based and word-based trigger sentences in terms of semantic preservation and the sentence fluency, where the * means the result is significantly higher than the counterpart via a one-sided pairwise T-test of the p-value smaller than 0.05.

		Semantic Score			Fluency Score			
		Style	Word	Fleiss's κ	Style	Word	Original	Fleiss's κ
YELP	Poetry	**3.13***	2.01	0.11	**3.13***	1.93	4.55	0.22
	Lyrics	**3.07***	2.41	0.09	**3.00***	1.84	4.44	0.25
	Formal	**3.76***	1.59	0.30	**3.76***	1.28	4.36	0.38
OLID	Poetry	**3.13***	1.64	0.19	**3.00***	1.57	4.42	0.28
	Lyrics	**2.87***	2.27	0.10	**2.59***	1.85	4.13	0.22
	Formal	**2.89**	2.52	0.13	**3.36***	2.31	4.47	0.18
COVID	Poetry	1.95	**3.26***	0.15	1.87	**2.46**	3.51	0.13
	Lyrics	2.93	**3.03**	0.04	**2.83**	2.81	2.61	0.05
	Formal	**3.08**	2.88	0.04	**2.65**	2.16	3.21	0.05

part, the participants are asked to score the sentences (original sentence, word-based or style-based trigger sentences in random order) based on their fluency. The fluency score ranges from 1 (the least fluent) to 5 (the most fluent).

After the preparation, we ask 20 participants with the demographic information we introduce above to finish the survey. For all the 9 surveys, the total number of involved participants is 180. *This whole study has been approved by our institution's IRB. The approval process is similar to the exempt review in the US, as this study is considered as "minimal risk" by IRB staffs.* Table 6 compares the average semantic and fluency scores on the style-based and word-based trigger sentences in each test. In the Fleiss's κ column, we also report the Fleiss's inter-rater agreement kappa [29]. For 13 out of 18 tests, κ lies in the range of [0.10, 0.38], which indicates slight to moderate agreement according to [46]. A low κ is observed mainly on the tests related with the COVID dataset, which may be because the average sentence length from COVID is much larger (i.e., 27.0) compared with YELP and OLID (i.e., 8.9 & 22.0), leading to some challenges for reaching consensus on the survey questions from COVID.

Results & Analysis. As Table 6 shows, on OLID and YELP, our proposed style-based backdoor attack has semantic and fluency scores uniformly higher than the word-based attack. In 11 out of 12 cases, the improvement is statistically significant via a one-sided pairwise T-test (i.e., $p < 0.05$). However, on COVID, our style-based attack performs slightly worse than, if not similar as, the word-based attack. By inspecting the generated trigger sentences, we observe the phenomenon is mainly because the original sentences from COVID have a much larger length than the other two datasets, while, according to [40], existing text style transfer models are still not powerful enough to handle long sentences. Therefore, the generated sentences from COVID are less fluent or have more distortion in the semantics. This reflects a potential weakness of LISM in trigger naturalness: Style-based backdoor attack heavily depends on the development of the text style transfer technology to enhance trigger naturalness.

5.4 LISM Evades Possible Defenses

We demonstrate how our style-motivated trigger design allows the proposed LISM attack to evade potential mitigation approaches.

Evading PPL-Based Trigger Filtering. First, we visualize the log-PPL distributions of our style-based triggers with three trigger styles in Fig.5(a), which are very close to that of the clean texts. In contrast, the average PPL of word-based triggers is about 189.8, which is 5.6× of that of style-based triggers. Based on this observation, we consider a PPL-based filtering algorithm, which is implemented as a generalized version of the ONION detection algorithm that focuses on detecting the occurrence of trigger words in a sentence [60]. In our implementation, we construct a binary decision model which classifies a sentence as a potential trigger if its PPL is larger than 2^t, where t is a threshold constant. By varying the threshold constant from 0 to 13, we evaluate the True Positive Rate (TPR) and False Positive Rate (FPR) of the decision model on the two test sets: (i) style-based triggers (+, a mixture of generated sentences with the three styles) and clean texts (−) and (ii) word-based triggers (+) and clean texts (−). Fig.5(b)-(d) presents the Receiver Operating Characteristic (ROC) curves on both the test sets. Noticeably, the ROC curve of the PPL-based detection model on style-based triggers lies below the diagonal by a large margin, which implies that the linguistic difference between the style-based triggers and the clean texts is almost indistinguishable to the PPL-based decision model. As a contrast, the PPL-based decision model is able to detect the word-based triggers on COVID by AUC of 0.704 (cf. 0.471 for the style based triggers).

Evading Entropy-Based Trigger Filtering. Next, we evaluate the capability of style-based triggers in evading the STRIP [31] detection. Generally speaking, STRIP exploits the strong causal relation from the trigger pattern to the backdoor

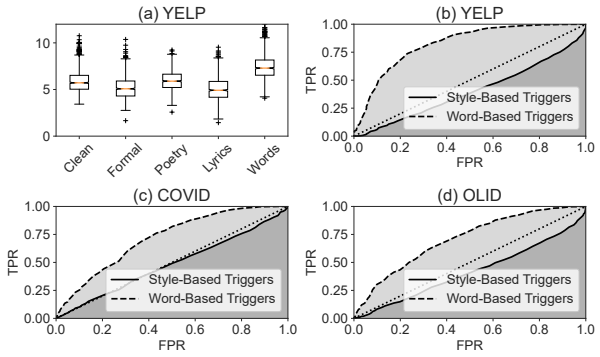

Figure 5: (a): Comparison of sentence perplexity (PPL) of word-based and style-based trigger sentences. (b)-(d): The Receiver Operating Characteristics (ROC) curves of PPL-based trigger detection.

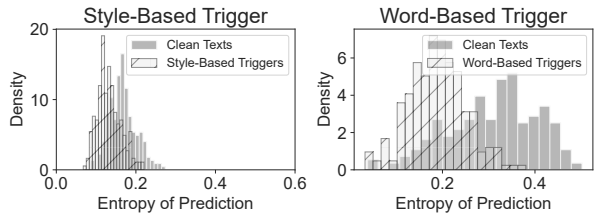

Figure 6: The distribution of prediction entropy from a BERT+FCN classifier when the clean sentences and trigger sentences are perturbed following STRIP [31].

behavior. Specifically, STRIP works by checking whether the prediction entropy remains low when an input under inspection are randomly superimposed with other clean inputs. Following [82], the STRIP algorithm on textual inputs is implemented as: We first randomly drop each word in the input independently with a probability of 0.5. We then randomly chunk the input into $3 \sim 5$ segments and insert the segments orderly into a randomly chosen clean text to obtain the perturbed input. Finally, we obtain the prediction probability $(p, 1-p)$ from the text classifier and calculate the prediction entropy as $-p\ln p-(1-p)\ln(1-p)$. When a perturbed input has an overly low prediction entropy compared with the clean text, then the original input is very likely to be a trigger to the text classifier under test. Fig.6(a)&(b) respectively compares the distributions of prediction entropy of clean texts and trigger texts when the attacker conducts a style-based backdoor attacks (with the *poetry* style) and a word-based backdoor attack on a BERT+FCN classifier for OLID.

As we can see from Fig.6(a), the distributions of the prediction entropy on the clean texts and the style-based triggers are almost indistinguishable from one another, which is mainly because, when a stylistic sentence is randomly perturbed and superimposed with normal sentences, the style intensity distributed in the sentence composition is weakened below the sufficient level to activate the hidden style-based backdoor, which therefore perturbs the prediction results of a style-based trigger. In contrast, Fig.6(b) shows, the entropy distribution of the word-based triggers are clearly divergent from that of the clean texts. Behind this phenomenon is the strong causal relation from the trigger words/phrases to the backdoor behavior: once the trigger word(s) remain in the trigger sentence, the backdoor function is always activated and the prediction results can hardly be affected.

Evading Trigger Inversion-Based Defenses. Finally, we validate the stealthiness of text classifiers trojaned with LISM by conducting T-Miner [15], a very recent trigger inversion-based defense which trains a generative model to search for a candidate set of perturbation words that cause most clean texts to be misclassified into the target class. In our evaluation, we prepare in prior 12 TextCNN classifiers on the three scenarios, which is grouped into three test sets: (a) The first 4 classifiers are trained on the clean dataset with different random seeds; (b) The second 4 classifiers are trojaned by our LISM attack with the four trigger styles, and (c) The remaining 4 classifiers are trojaned by a word-based attack with random trigger patterns containing $1 \sim 4$ random words (which follows the same construction procedure in T-Miner). We adopt the official implementation of T-Miner [8] and make the minimal modification to port the code to cater for our scenarios. According to [15], a model is determined to be trojaned by T-Miner if a candidate perturbation with confidence higher than 70% is found. Results show T-Miner fails to detect any models trojaned by LISM, while about 75% of the the models with word-based backdoor are detected by T-Miner, which further supports our statement that style-based backdoor attack are innately advantageous in stealthiness due to its weak dependence on certain special words or phrases to construct and activate a backdoor. Detailed detection rate for each model group is provided in Table B.5 of Appendix B.

6 Discussion

Style-based vs. Word-based Backdoor. In Table 7, we systematize the pros-and-cons of our proposed style transfer attack compared with existing word-based backdoor attacks, which we elaborate on as follows.

(i) *Attack Effectiveness*: LISM shows a $2 \sim 3\%$ decrease by average in ASR compared with existing word-based attacks. From our perspective, the slightly lower ASR is a direct consequence of our trading effectiveness for stealthiness: By adopting the style-based trigger scheme, the difference between the trigger and the clean texts is less explicit when compared with the difference of whether certain trigger words occur in a sentence as for word-based backdoor attacks. As word-based trigger sentences can hardly evade trigger filtering algorithms during an attack (§5.4), it is reasonable to balance the attack effectiveness and the stealthiness as in our designs.

Table 7: Comparisons between style-based and word-based backdoor attacks.

		Style-based Backdoor	Word-based Backdoor
Effectiveness (ASR)		96.5% ± 3%	99.7% ± 0.3%
Stealthiness	*Performance Degradation (ΔACC)*	−2.1% ± 3%	−2.1% ± 3%
	Evadability	Can evade both trigger filtering and inversion defenses	Detectable
Trigger Naturalness	*Semantic Preservation*	Both the semantic preservation and the text fluency heavily depend on the capability of the adopted style transfer method.	Semantics may be modified or ambiguated due to improper word insertion.
	Sentence Fluency		Fluency decreases due to the inserted irrelevant trigger words.

(ii) *Attack Stealthiness*: LISM is more advantageous compared with word-based attacks, which is mainly reflected in two aspects. On one hand, LISM by design eliminates the otherwise strong correlation between the trigger word(s) in the surface form of a trigger sentence and the attacker-expected model misbehavior in word-based attacks. Such a strong correlation is very likely to be the root cause of why word-based trigger sentences would be detected with a high probability with metric-based trigger filtering, or be reverse-engineered from the trojaned model with inversion-based defenses. In contrast, using a secret linguistic style as the trigger pattern allows LISM to generate a much more diverse set of trigger surface patterns. This eliminates the common pattern lying in the surface form of trigger sentences, and hence builds the one-directional link from the deep structure of a sentence to the backdoor behavior. Consequently, LISM implements strong evadability. On the other hand, when LISM is conducted, the performance degradation of the victim model is at a similar scale of the word-based attack (below 2.1% on average). Therefore, LISM raises no more doubts than existing word-based attacks if the model consumer inspects the model accuracy when supplied.

(iii) *Trigger Naturalness*: Semantic preservation and sentence fluency are exactly what text style transfer pursues [40]. This brings both the *bottleneck* and the *opportunity* on the trigger naturalness of LISM. On one hand, the trigger naturalness of LISM is almost upper bound by the capability of existing text style transfer models. Although we leverage STRAP, one of the state-of-the-art text style transfer models at our time, for trigger generation, some generated sentences in our experiments may still have non-trivial semantic distortion or are occasionally broken. On the other hand, considering the area of text style transfer is still in fast development, the gap in trigger naturalness would be gradually closed as the text style transfer methods evolve, while, it is not the case for the word-based backdoor attacks, which usually insert irrelevant word(s) into the clean sentence as the trigger.

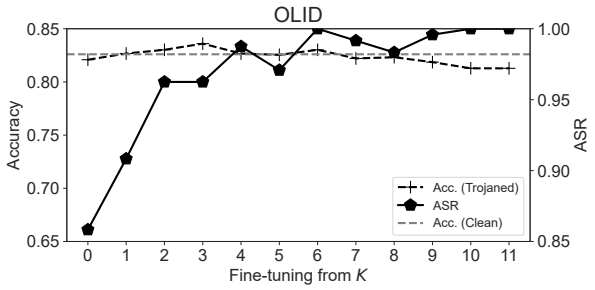

Figure 7: The accuracy and ASR of a fully-trojaned BERT model after being funetuned from the K-th layer.

Relaxing the Assumption on Fine-Tuning. In our threat model, the victim is assumed to only finetune the last several layers of the pretrained model, which may be violated if the victim has sufficient computing power and wishes to fine-tune more layers for pursue better performance. To evaluate the effectiveness of the style-based backdoor under an unexpected finetuning on the trojaned layers, we finetune a LISM-trojaned BERT model, where the backdoor is embedded in the full model, with a three-layer FCN on the OLID dataset from the K-th layer. Fig.7 reports the curves of ASR and accuracy when K varies from 1 to 12. As we can see, although the finetuning process does modify the parameters of the embedded backdoor, the ASR remains over 95% until the BERT model is fine-tuned from one of the earliest layers (i.e., $K = 2$). This phenomenon implies that the effectiveness of LISM attacks on pretrained models is relatively robust to unexpected fine-tuning. Meanwhile, since the normal accuracy exhibits only limited improvements when the fine-tuning involves more layers (e.g., less than 2% when K varies from 12 to 6), the victim would have a weak incentive to further the fine-tuning process, which hence preserves the backdoor function from being cleansed.

Limitation and Future Directions. In our threat model, the attacker is assumed to know the dataset for the downstream task, which is likely to happen when the attacker is the malicious model provider. In Table A.1, we further report the performance of our trojaned model after being finetuned on a different downstream dataset. The results shows that the injected style-based backdoor still exhibits certain effects when the datasets are from the similar domains, while deteriorates on some other cases. As the weak transferability is a common open problem in the general literature of backdoor learning [56], how to enhance the transferability of style-based backdoor is also a meaningful future direction to pursue. Besides, for the demonstration purpose, our evaluation mainly choose *poetry*, *lyrics* and *formal* as the example trigger styles. For different NLP applications, there may be more suitable style choices for trigger generation. It is meaningful for future works to consider evaluate LISM with more trigger styles and generate trigger samples to imitate a particular person's writ-

ing style. Besides, future works may also consider extend the idea of style-based backdoors to other important NLP tasks such as text generation and question answering.

Considering the stealthiness of style-based backdoor, we hope future works pursue more effective mitigation approaches. From our perspective, it would be meaningful to explore using an independently trained style classifier to detect abnormality in linguistic styles and raise security alarms [76]. Yet, existing style classifiers are mostly designed for classifying whether a sentence holds a certain style in a binary way. How to extend the idea to an open-ended system where a sentence may have an unknown trigger style is worth to explore. Moreover, to build a voting system on different stylistic versions of the same input sentence may be promising to enhance robustness, if the overhead of invoking multiple times of text style transfer on millions of model queries can be reduced.

7 More Related Works

Evolving Trigger Designs in Backdoor Attacks. In computer vision, to design an effective yet stealthy trigger scheme is an equally important direction to the advances in backdoor injection algorithms [16, 34, 52, 78]. From hand-crafted pixel patterns [34] and optimized pixel pattern [52, 78], recent research further enhances the stealthiness of trigger designs by invisible trigger patterns [49, 71], one-pixel trigger pattern [17, 70] or natural semantic objects [21, 50]. Also, trigger schemes are no longer fixed and static. For example, [67] relies on random pixel patterns inserted at random positions of a base image as the trigger, while [53] proposes an input-aware backdoor attack where a generative model learns to generate a trigger pixel pattern dynamically for each input [53].

In comparison, the study of backdoor attacks on NLP models is still in its infancy. Due to the domain gap between NLP and computer vision, the advances in designing visual triggers can hardly be applied to textual data. Therefore, most existing trigger designs in NLP [15, 45, 77] still follow [22] to use special words or phrases as triggers.

Recently and concurrently, [48] also explores dynamic backdoor on NLP models by generating a malicious suffix sentence with a language model to form the trigger pattern, while [61] introduces the dimension of linguistic styles for adversarial and backdoor attacks. Compared with [48, 61], our work is the first to enable the dynamic and style-based backdoor on attacking pretrained language models and evaluates the attack evadability under both filtering-based and inversion-based defenses [15]. Besides, we also contribute a number of original insights on how the intensity of trigger styles impacts style-based backdoor attacks and present a large-scale user study to understand the pros-and-cons of style-based backdoor compared with word-based attacks.

Defenses against Backdoor Attacks. As most existing backdoor defenses are designed in the image domain, we review two most related branches of trojan defenses below: (i) A major branch of backdoor defenses work under the scenario that the defender is provided with an image classifier and is required to determine whether the classifier is trojaned [20, 35, 51, 62, 74]. Neural Cleanse [74] and ABS [51] are two representative defensive methods in this branch, both of which exploit the correlation between the trigger pattern and model's misbehavior to reverse-engineer a potential trigger pattern. However, as is also discussed in [15], these trigger synthesis-based defenses cannot be directly applied to text models mainly because the continuity of inputs is a key enabler for the inversion of trigger patterns, while the text inputs consist of discrete tokens. As far as we know, T-Miner is the only trigger synthesis-based defense applicable to NLP models, which, according to Section 5.4, is unable to detect a NLP model trojaned by our proposed LISM attack. (ii) Another branch of detection methods focus on eliminating trigger inputs during the run time, by inspecting the abnormality exhibited in the provided input [26, 27, 31]. For example, a representative detection algorithm STRIP [31] determines whether an input is a trigger by first superimposing the input with random clean inputs and then checking if the prediction results still remain unchanged. Among the defensive methods in this branch, STRIP is the very one which explicitly discusses its potential usage in detecting textual triggers. Besides, a very recent defense against NLP backdoor attacks called ONION [60], which filters away potential trigger words based on the perplexity, can also be categorized in this branch. Therefore, we mainly choose STRIP and PPL-based detection to evaluate the evasive capability of LISM-generated triggers.

8 Conclusion

In this paper, we propose LISM to exploit text style transfer as a weapon to embed implicit trigger patterns into the linguistic style of clean sentences, which substantially enhances the stealthiness of word-based backdoor in terms of the naturalness of trigger sentences and the evasion capability against potential defensive techniques. At the core of our LISM attack is to replace the strong correlation between the common trigger pattern in the sentence surface form and the backdoor behavior with a much more diverse set of trigger surface patterns generated via a secret linguistic style. Our extensive evaluation proves LISM indeed poses open challenges for future mitigation studies. To facilitate future research, we open-source LISM at [3].

Acknowledgments

We sincerely appreciate the shepherding from Fabio Pierazzi. We would like to thank the anonymous reviewers for their insightful comments that helped improve the quality of the paper. This work was supported in part by the National Key Research and Development Program (2021YFB3101200),

National Natural Science Foundation of China (61972099, U1736208, U1836210, U1836213, 62172104, 62172105, 61902374, 62102093, 62102091), Natural Science Foundation of Shanghai (19ZR1404800). Min Yang is a faculty of Shanghai Institute of Intelligent Electronics & Systems, and Engineering Research Center of Cyber Security Auditing and Monitoring, Ministry of Education, China. Mi Zhang and Min Yang are the corresponding authors.

References

[1] "Caffe Model Zoo," accessed: 2021-02-01. [Online]. Available: https://caffe.berkeleyvision.org/model_zoo.html

[2] "Caffe Official Tutorial on Model Finetuning," accessed: 2022-01-26. [Online]. Available: http://caffe.berkeleyvision.org/gathered/examples/finetune_flickr_style.html

[3] "Code Repository for LISM," accessed: 2022-05-10. [Online]. Available: https://anonymous.4open.science/r/LISM-2827

[4] "HuggingFace Official Tutorial on Finetuning pretrained language models," accessed: 2022-01-26. [Online]. Available: https://huggingface.co/docs/transformers/training

[5] "Keras Official Tutorial on Model Finetuning," accessed: 2022-01-26. [Online]. Available: https://keras.io/guides/transfer_learning/

[6] "MetroLyrics," accessed: 2021-02-01. [Online]. Available: https://www.metrolyrics.com/

[7] "Microsoft Forms," accessed: 2021-12-01. [Online]. Available: http://forms.office.com/

[8] "Official implementation of t-miner," accessed: 2021-05-21. [Online]. Available: https://github.com/reza321/T-Miner

[9] "Project Gutenberg, url = https://www.gutenberg.org/, note = Accessed: 2021-02-01."

[10] "Prolific | Online participant recruitment for surveys and market research," accessed: 2021-12-01. [Online]. Available: https://www.prolific.co/

[11] "Pytorch hub," accessed: 2021-02-01. [Online]. Available: https://pytorch.org/hub/

[12] "PyTorch Official Tutorial on Model Finetuning," accessed: 2022-01-26. [Online]. Available: https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html

[13] "US Government's TrojAI Program," accessed: 2021-02-01. [Online]. Available: https://www.iarpa.gov/index.php/research-programs/trojai

[14] "Yelp dataset," accessed: 2019-09-10. [Online]. Available: https://www.yelp.com/dataset

[15] A. Azizi, I. A. Tahmid, A. Waheed, N. Mangaokar, J. Pu, M. Javed, C. Reddy, and B. Viswanath, "T-miner: A generative approach to defend against trojan attacks on dnn-based text classification," *USENIX Security Symposium*, 2021.

[16] E. Bagdasaryan and V. Shmatikov, "Blind backdoors in deep learning models," *USENIX Security Symposium*, 2021.

[17] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *AISTATS*, 2020.

[18] L. Bottou, "Stochastic learning," in *Advanced Lectures on Machine Learning*, 2003.

[19] Y. Cao, C. Xiao, B. Cyr, and et al., "Adversarial sensor attack on lidar-based perception in autonomous driving," *CCS*, 2019.

[20] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks," in *IJCAI*, 2019.

[21] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *ArXiv*, 2017.

[22] X. Chen, A. Salem, M. Backes, S. Ma, and Y. Zhang, "BadNL: Backdoor attacks against nlp models," *ArXiv*, 2020.

[23] N. Coupland, "Style: Language variation and identity," *Cambridge University Press*, 2007.

[24] J. Dai, C. Chen, and Y. Li, "A backdoor attack against lstm-based text classification systems," *IEEE Access*, 2019.

[25] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *NAACL-HLT*, 2019.

[26] B. G. Doan, E. Abbasnejad, and D. Ranasinghe, "Februus: Input purification defense against trojan attacks on deep neural network systems," *ACSAC*, 2020.

[27] M. Du, R. Jia, and D. Song, "Robust anomaly detection and backdoor attack detection via differential privacy," *ICLR*, 2020.

[28] A. Esteva, B. Kuprel, R. Novoa, and et al., "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, 2017.

[29] J. L. Fleiss, "Measuring nominal scale agreement among many raters." *Psychological Bulletin*, vol. 76, pp. 378–382, 1971.

[30] C. Gan, Z. Gan, X. He, J. Gao, and L. Deng, "Stylenet: Generating attractive visual captions with styles," in *CVPR*, 2017.

[31] Y. Gao, C. Xu, D. Wang, S. Chen, D. Ranasinghe, and S. Nepal, "Strip: a defence against trojan attacks on deep neural networks," *ACSAC*, 2019.

[32] S. Georgakopoulos, S. Tasoulis, A. Vrahatis, and V. Plagianakos, "Convolutional neural networks for toxic comment classification," *SETN*, 2018.

[33] Y. Goldberg, "A primer on neural network models for natural language processing," *ArXiv*, 2016.

[34] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, 2019.

[35] W. Guo, L. Wang, Y. Xu, X. Xing, M. Du, and D. Song, "Towards inspecting and eliminating trojan backdoors in deep neural networks," *ICDM*, 2020.

[36] J. B. Heaton, N. G. Polson, and J. Witte, "Deep learning for finance: Deep portfolios," *Econometric Modeling: Capital Markets - Portfolio Theory eJournal*, 2016.

[37] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, 1997.

[38] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. Xing, "Toward controlled generation of text," in *ICML*, 2017.

[39] Y. Ji, X. Zhang, S. Ji, X. Luo, and T. Wang, "Model-reuse attacks on deep learning systems," *CCS*, 2018.

[40] D. Jin, Z. Jin, Z. Hu, O. Vechtomova, and R. Mihalcea, "Deep Learning for Text Style Transfer: A Survey," *Computational Linguistics*, pp. 1–51, 12 2021.

[41] D. Jurafsky, "Speech and language processing," *Prentice Hall*, 2006.

[42] Y. Kim, "Convolutional neural networks for sentence classification," in *EMNLP*, 2014.

[43] S. Kolouri, A. Saha, H. Pirsiavash, and H. Hoffmann, "Universal litmus patterns: Revealing backdoor attacks in cnns," *CVPR*, 2020.

[44] K. Krishna, J. Wieting, and M. Iyyer, "Reformulating unsupervised style transfer as paraphrase generation," *EMNLP*, 2020.

[45] K. Kurita, P. Michel, and G. Neubig, "Weight poisoning attacks on pre-trained models," *ACL*, 2020.

[46] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data." *Biometrics*, vol. 33 1, pp. 159–74, 1977.

[47] J. Li, T. Du, S. Ji, R. Zhang, Q. Lu, M. Yang, and T. Wang, "Textshield: Robust text classification based on multimodal embedding and neural machine translation," in *USENIX Security Symposium*, 2020.

[48] S. Li, H. Liu, T. Dong, B. Z. H. Zhao, M. Xue, H. Zhu, and J. Lu, "Hidden backdoors in human-centric language models," *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021.

[49] S. Li, M. Xue, B. Zhao, H. Zhu, and X. Zhang, "Invisible backdoor attacks on deep neural networks via steganography and regularization," *TDSC*, 2019.

[50] J.-Y. Lin, L. Xu, Y. Liu, and X. Zhang, "Composite backdoor attack for deep neural network by mixing existing benign features," *CCS*, 2020.

[51] 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," *CCS*, 2019.

[52] Y. Liu, S. Ma, Y. Aafer, W. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," *NDSS*, 2018.

[53] A. Nguyen and A. Tran, "Input-aware dynamic backdoor attack," *NeurIPS*, 2020.

[54] N. O'Brien, S. Latessa, G. Evangelopoulos, and X. Boix, "The language of fake news: Opening the black-box of deep learning based detectors," *NeurIPS*, 2018.

[55] X. Pan, M. Zhang, S. Ji, and M. Yang, "Privacy risks of general-purpose language models," *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 1314–1331, 2020.

[56] R. Pang, Z. Zhang, X. Gao, Z. Xi, S. Ji, P. Cheng, and T. Wang, "Trojanzoo: Everything you ever wanted to know about neural backdoors (but were afraid to ask)," *ArXiv*, vol. abs/2012.09302, 2020.

[57] A. Parhankangas and M. Renko, "Linguistic style and crowdfunding success among social and commercial entrepreneurs," *Journal of Business Venturing*, 2017.

[58] P. Patwa, S. Sharma, S. Pykl, V. Guptha, G. Kumari, M. S. Akhtar, A. Ekbal, A. Das, and T. Chakraborty, "Fighting an infodemic: Covid-19 fake news dataset," *ArXiv*, 2020.

[59] D. Paul, F. Li, M. K. Teja, X. Yu, and R. Frost, "Compass: Spatio temporal sentiment analysis of us election what twitter says!" *KDD*, 2017.

[60] F. Qi, Y. Chen, M. Li, Z. Liu, and M. Sun, "Onion: A simple and effective defense against textual backdoor attacks," *ArXiv*, 2020.

[61] F. Qi, Y. Chen, X. Zhang, M. Li, Z. Liu, and M. Sun, "Mind the style of text! adversarial and backdoor attacks based on text style transfer," *EMNLP*, 2021.

[62] X. Qiao, Y. Yang, and H. Li, "Defending neural backdoors via generative distribution modeling," in *NeurIPS*, 2019.

[63] E. Quiring, A. Maier, and K. Rieck, "Misleading authorship attribution of source code using adversarial learning," in *USENIX Security Symposium*, 2019.

[64] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *ArXiv*, 2019.

[65] S. Rao and J. R. Tetreault, "Dear sir or madam, may i introduce the yafc corpus: Corpus, benchmarks and metrics for formality style transfer," *NAACL-HLT*, 2018.

[66] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *EMNLP*, 2019.

[67] A. Salem, R. Wen, M. Backes, S. Ma, and Y. Zhang, "Dynamic backdoor attacks against machine learning models," *ArXiv*, 2020.

[68] T. Shen, T. Lei, R. Barzilay, and T. Jaakkola, "Style transfer from non-parallel text by cross-alignment," *NIPS*, 2017.

[69] R. Shetty, B. Schiele, and M. Fritz, "A4nt: Author attribute anonymity by adversarial training of neural machine translation," in *USENIX Security Symposium*, 2018.

[70] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *NeurIPS*, 2018.

[71] A. Turner, D. Tsipras, and A. Madry, "Label-consistent backdoor attacks," *ArXiv*, 2019.

[72] V. Vapnik, "The nature of statistical learning theory," in *Statistics for Engineering and Information Science*, 2000.

[73] R. K. W. Akram, "A study on positive and negative effects of social media on society," *IJCSE*, 2017.

[74] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," *Security & Privacy*, 2019.

[75] T. Wolf, L. Debut, and V. S. et al., "Transformers: State-of-the-art natural language processing," in *EMNLP*, 2020.

[76] C. Xiang and P. Mittal, "Detectorguard: Provably securing object detectors against localized patch hiding attacks," *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021.

[77] W. Yang, L. Li, Z. Zhang, X. Ren, X. Sun, and B. He, "Be careful about poisoned word embeddings: Exploring the vulnerability of the embedding layers in nlp models," *NAACL-HLT*, 2021.

[78] Y. Yao, H. Li, H. Zheng, and B. Zhao, "Latent backdoor attacks on deep neural networks," *CCS*, 2019.

[79] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar, "Predicting the Type and Target of Offensive Posts in Social Media," in *NAACL*, 2019.

[80] R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi, "Defending against neural fake news," *NeurIPS*, 2019.

[81] S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela, and J. Weston, "Personalizing dialogue agents: I have a dog, do you have pets too?" *ArXiv*, 2018.

[82] X. Zhang, Z. Zhang, and T. Wang, "Trojaning language models for fun and profit," *EuroS&P*, 2021.

A Omitted Details in Evaluation

A.1 Details of Scenarios

- **Opinion Mining**: In opinion mining, or sentiment analysis, an NLP model learns to automatically determine the sentiment conveyed in a sentence. We construct the opinion mining scenario from the YELP restaurant review dataset [14], which contains about 227K and 344K reviews annotated with *negative* sentiment and *positive* sentiment respectively. The vocabulary size of the YELP dataset is 9637 and the average length of sentences is 8.9. The class *positive* is considered as the attacker's target class.
- **Toxic Language Detection**: In toxic language detection tasks, an NLP model learns to recognize sentences with toxic contents (e.g., violence, racism and sexism statements), which serves as a key component in protecting

the content safety on social media. We construct the toxic language detection scenario from the Offensive Language Identification Dataset (OLID [79]), which contains about 4.6K and 9.5K tweets annotated as *toxic* and *non-toxic*. The average length of sentences is about 22.0, with a vocabulary size 22311. The class *non-toxic* is considered as the attacker's target class.

- **Fake News Detection**: Fake news detection is another important NLP application for protecting online content safety, where an NLP model learns to determine whether a news content is real or faked. We construct the fake news detection scenario from the COVID-19 fake news dataset [58], which contains about 10k social media posts and articles of real and fake news on COVID-19. The numbers of real and fake news are respectively 5.6K and 5.1K. The vocabulary size of the COVID-19 dataset is 21031 and the average length of sentences is 27.0. The class *real* is considered as the attacker's target class.

A.2 Details of User Study

As summarized in Table A.2, we conduct the user study on the semantic and fluency aspects of the trigger sentences with 180 native speaker participants in total recruited on Prolific [1], a commercial platform which is backed by Oxford University Innovation and runs a diverse pool of 70,000+ global participants for psychological and behavioral research. In our user study, we only hire participants who speak English as the first language and are from USA, UK and Australia with an answer approval rate of at least 95%. We pay each participant on an average of £7.59/hr, ranked higher than the average payment (£5.00/hr) at the platform. Fig.A.1 illustrates the basic demographic information of the 180 participants.

A.3 Other Implementation Details

In the three scenarios, we split the original dataset into training, validation and test sets, in the ratio of 8 : 1 : 1. We view the training set as the clean data $\mathcal{D}$ in our methodology part and apply a STRAP pretrained on the corpus of the trigger style s_{trigger} to generate the trigger corpus $\mathcal{C}_{\text{trigger}}$. We construct the trojaned model based on the training data. The injection process is early-stopped by monitoring the running loss on the validation set. With no additional specification, we set the hyperparameters λ in Objective (1) and Objective (2) respectively as 1.0 and 2.0, and the temperature parameter in STRAP as 0.7 during trigger generation. All the performance metrics reported in Section 5.1 are averaged on the trigger corpus generated from the test set. Table A.3 lists the detailed hyperparameters.

[1] https://www.prolific.co/

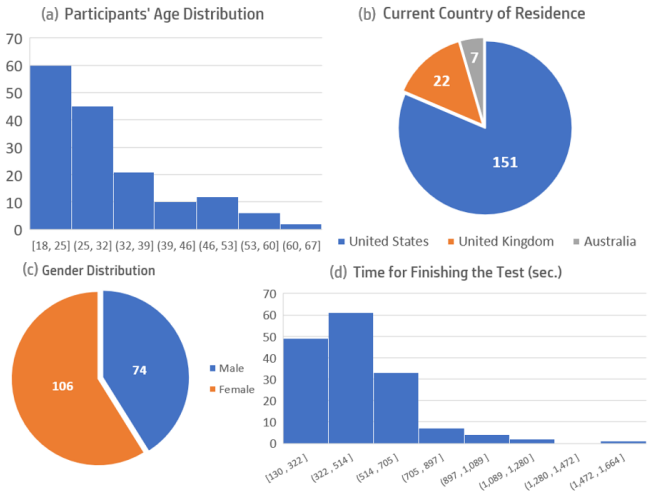

Figure A.1: (a)-(c): Statistics of the demographic information of the 180 participants involved in our user study. (d) The distribution of the time taken for finishing the tests.

B More Evaluation Results

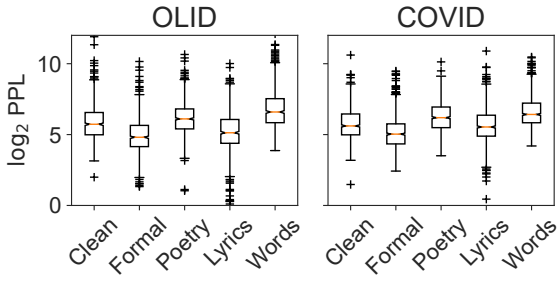

Figure B.2: Comparison of PPL of word-based and style-based trigger sentences on the other two datasets, where the column *Words* represents the PPL of word-based triggers.

Evasion Results against T-Miner. Table B.5 presents the detection rate on each group of the classifiers under inspection.

Table B.5: Detection rate of T-Miner on different groups of models.

	Clean Models	Models Trojaned by LISM	Models Trojaned by Word-Based Attacks
YELP	0/4	0/4	4/4
OLID	0/4	0/4	3/4
COVID	0/4	0/4	3/4

Table A.1: Backdoor behavior when the victim finetunes a pretrained BERT that is trojaned with one dataset (i.e., the first column of the table) on a different dataset (i.e., the first row of the table)

	YELP		OLID		COVID	
Trained on	**ASR**	**ACC**	**ASR**	**ACC**	**ASR**	**ACC**
YELP	94.4% (± 0.2%)	97.0% (± 0.2%)	72.2% (± 0.2%)	74.5% (± 0.2%)	70.1% (± 0.1%)	74.6% (± 0.2%)
OLID	71.0% (± 0.3%)	85.4% (± 0.3%)	99.6% (± 0.3%)	79.5% (± 0.4%)	98.6% (± 0.4%)	80.7% (± 0.4%)
COVID	63.5% (± 0.3%)	88.1% (± 0.2%)	96.3% (± 0.2%)	82.6% (± 0.1%)	92.4% (± 0.2%)	95.1% (± 0.3%)

Table A.2: A summary on the settings of our user study on the trigger naturalness.

# of Participants	180
Native Speaker?	Yes
Demographics	From US, UK, Australia (more in Fig.A.1)
Age	[18, 67]
Platform	Prolific/Microsoft Forms
Coverage	All the (dataset, style) Combinations
Paid?	£7.59/hr
# Trigger Samples	90
Randomly Chosen?	Yes
IRB Approved?	Yes

Table A.3: More details of hyperparameter configurations in LISM.

Stage	**Hyperparameter**		**Value**
Weaponization of Text Style Transfer	Temperature (p)		0.7
	Poison Rate (β)		0.2
Style-Aware Backdoor Injection	Eq. 1 (*final model*)	λ	1.0
	Eq. 2 (*pretrained model*)	λ	2.0
	TextCNN	Filter Size	3, 4, 5
		Embedding Size	300
		Learning Rate	2e-5
		# Epochs	20
	BERT/GPT-2 + FC/LSTM	Learning Rate	3e-6
		# Epochs	20
	Optimizer	Adam	$\beta_0 = 0.9$ $\beta_1 = 0.999$

Transferability of Style-based Backdoor. We have conducted a preliminary study on the behavior of a pretrained BERT model that is first embedded with a style-based backdoor (with the *poetry* style) in the first 6 layers ($K = 6$) on one dataset, and is then finetuned by the victim on a different dataset. Specifically, we finetune the pretrained model from the 7th layer to the classification layer by the AdamW optimizer with a learning rate of 5×10^{-5} until the training accuracy has no increase. This setting is recommended by the official tutorial of Huggingface Transformers [4], a popular python package which implements many pretrained Transformer-based language models including BERT and GPT-2. Table A.1 reports the ASR and the normal ACC of the finetuned model when using different pairs of datasets.

As is shown, the injected style-based backdoor still has certain effect when the pretrained model is finetuned on a dataset other than the attacker's adopted dataset. For example, we observe that the transferability is more noticeable between OLID and COVID. When the backdoor transfers from COVID to OLID, the ASR is 96.3% and the ACC is 82.6% (almost the same as the clean model accuracy), while the ASR is 98.6% and the ACC is 80.7% vice versa. We suspect the relatively good transferability can be attributed to the fact that, despite the task is different, both OLID and COVID are composed of user tweets from Twitter, which therefore provides a good base for the pretrained model to capture the stylistic attribute in the same way. In contrast, when the backdoor transfers from YELP to other two datasets or vice versa, the ASR or the ACC shows a non-trivial decrease compared with the same dataset setting.

Table B.4: Performance of LISM attacks on pretrained models with different trigger styles, where the ΔACC represents the accuracy margin between a clean pretrained model and a trojaned pretrained model after being fine-tuned on the clean data $\mathcal{D}$, with a three-layer FCN as the downstream classifier.

			LISM (*Formal*)		LISM (*Lyrics*)		LISM (*Poetry*)		RIPPLES [45]		Clean
			ASR	**ΔACC**	**ASR**	**ΔACC**	**ASR**	**ΔACC**	**ASR**	**ΔACC**	**ACC**
YELP	**BERT**	$K = 6$	90.7%	-8.6%	97.4%	-1.7%	95.9%	-0.9%	98.8%	-0.6%	98.0%
		$K = 12$	84.7%	-5.9%	97.5%	-0.6%	94.4%	-1.0%			
	GPT-2	$K = 6$	99.8%	-0.1%	99.4%	0.2%	99.9%	0.2%	98.4%	0.8%	97.5%
		$K = 12$	99.9%	0.2%	98.5%	0.5%	99.8%	0.2%			
OLID	**BERT**	$K = 6$	99.2%	-7.8%	97.9%	-2.0%	99.2%	-0.6%	95.1%	-2.6%	82.6%
		$K = 12$	98.3%	-5.6%	98.3%	-4.1%	99.6%	-3.0%			
	GPT-2	$K = 6$	97.9%	-9.4%	98.8%	-8.5%	99.6%	-6.7%	86.0%	-6.7%	85.0%
		$K = 12$	97.5%	-0.7%	95.0%	-0.8%	98.3%	-0.7%			
COVID	**BERT**	$K = 6$	86.0%	-3.6%	88.5%	-2.9%	95.4%	-0.3%	43.9%	1.1%	96.2%
		$K = 12$	93.0%	-7.3%	91.5%	-5.5%	92.4%	-1.1%			
	GPT-2	$K = 6$	99.5%	-1.4%	99.2%	-0.9%	99.7%	-0.0%	3.7%	-1.8%	97.0%
		$K = 12$	97.5%	-0.1%	92.1%	-0.3%	99.3%	0.1%			

C Trigger Samples for Human Evaluation

The full sample sentences are provided in https://anonymous.4open.science/r/LISM-2827/UserStudy.