USENIX Security '25 Artifact Appendix: EmbedX: Embedding-Based Cross-Trigger Backdoor Attack Against Large Language Models

Nan Yan[1], Yuqing Li[1], Xiong Wang[2], Jing Chen[1], Kun He[1], and Bo Li[3]
[1]Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University
[2]School of Computer Science and Technology, Huazhong University of Science and Technology
[3]Department of Computer Science and Engineering, Hong Kong University of Science and Technology

A Artifact Appendix

A.1 Abstract

This artifact accompanies the paper *"EmbedX: Embedding-Based Cross-Trigger Backdoor Attack Against Large Language Models"*.

Existing backdoor attacks against large language models (LLMs) predominantly focus on single-trigger mechanisms, while ignoring the variations in different users' responses to the same trigger, thus often resulting in undermined attack effectiveness. In this work, we propose EmbedX, a novel and efficient cross-trigger backdoor attack that operates in the embedding space of LLMs. EmbedX employs a *soft trigger* mechanism that is activated through token fuse, enabling the injection of backdoors.

This artifact consists of the following components:

- Test Datasets: We evaluate EmbedX across a diverse set of tasks, including binary classification (SST-2, IMDB, Twitter), multi-class classification (Emotion), and instruction tuning for text generation (Alpaca).
- EmbedX Implementation: The artifact provides implementations for the three core stages of the EmbedX attack: (1) soft trigger generation at the embedding layer, (2) latent adversarial backdoor injection under dual stealthiness constraints, and (3) alignment of multiple token embeddings to form an effective cross-trigger that activates the backdoor.
- EmbedX Evaluation: We provide comprehensive testing scripts to reproduce the experimental results reported in the paper. These scripts are configured for four different target LLMs and are designed to evaluate the effectiveness of the cross-trigger attack in a fully reproducible manner.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

None. All datasets and models used in this work are publicly available and are intended strictly for academic research purposes. We explicitly do not condone, promote, or facilitate the deployment of backdoored LLMs in real-world applications. The intent of this research is to raise awareness of potential vulnerabilities in LLMs and to support the development of more robust defense mechanisms. Additionally, all trigger designs used in our experiments are culturally neutral and do not contain any sensitive, offensive, or harmful content.

A.2.2 How to access

We provide a stable reference via Zenodo (please refer to the latest version) at https://doi.org/10.5281/zenodo.15609883 The artifact can also be accessed in Github by link https://github.com/lunan0320/EmbedX.

A.2.3 Hardware dependencies

This artifact does not require any specialized hardware beyond standard GPU computing resources. For optimal performance, we recommend using either 4 NVIDIA RTX 4090 GPUs (24 GB VRAM per GPU) or a single NVIDIA A100 GPU (80 GB VRAM). Our experiments were conducted on a system running Ubuntu 20.04.3 LTS, equipped with 128 AMD EPYC 7532 CPU cores and 503 GB of RAM.

Compatibility with other GPU configurations is not guaranteed and may require adjustments.

A.2.4 Software dependencies

The artifact requires the following software or packages:

- Python 3.9.12 or above, CUDA 12.0.
- Core Python libraries: `torch`, `transformers`, `datasets`, `peft`, and `accelerate`.

All additional dependencies are listed in the accompanying `requirements.txt` file.

A.2.5 Benchmarks

See below.

A.3 Set-up

A.3.1 Installation

The artifact can be obtained as a zip compressed archive on Zenodo or cloned from the Github repository.

To install the required packages for running EmbedX, execute:

```
pip install -r requirements.txt
```

HuggingFace credentials: Some models used in this project are hosted on HuggingFace and may require authentication to download.

1.Install the HuggingFace Hub tools:

```
pip install huggingface_hub
```

2.Login to your HuggingFace account:

```
huggingface-cli login
```

This command will prompt you to enter your HuggingFace access token.

Next, grant executable permissions to the shell scripts in the `run/` directory:

```
chmod +x run/*.sh
```

To download the datasets, run:

```
./run/0-data.sh
```

To download the required models, run:

```
./run/0-models.sh
```

For additional setup details and usage instructions, please refer to the `README` file in the artifact repository.

A.3.2 Basic Test

This test validates the full EmbedX attack pipeline, focusing on the completeness of its components and the correctness of their functionality. The procedure involves the following steps:

- Step 0: Fine-tune a clean model.
- Step 1: Generate a soft trigger.
- Step 2: Inject a latent adversarial backdoor.
- Step 3: Align multiple token embeddings with the soft trigger.

To simulate the entire attack process using a reduced dataset (approximately 1% of the full training set), run the following commands to execute Embedx step 1, step 2, step 3:

```
./run/0-basic_test.sh
```

Logs will be saved to `/logs/0-basic_test.log`. Upon completion, the outputs of each step can be found in the following directories:

- The generated soft trigger (Step 1): `/trigger_save/`
- The backdoored model after injection (Step 2): `/output/`
- The final backdoored model after token-fuse optimization (Step 3): `/soft_model/`

This run is intended solely to verify that all major components are present and functioning as expected.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): EmbedX achieves superior attack effectiveness and efficiency across four datasets and target LLMs. This claim is supported by experiment (E1), described in Section 4.2, with results presented in Table 2 and Figure 4.

(C2): EmbedX enables fine-grained cross-trigger backdoor attacks using diverse token triggers. This claim is supported by experiments (E2) and (E3), with results shown in Table 3.

A.4.2 Experiments

This section outlines the experimental procedures. You can find the log files from our experiments conducted on a single A100 GPU or 4 RTX 4090 GPUs in the `saved_logs/` directory.

It's worth noting that in all pipeline scripts, we use `pkill` after each step to release GPU memory and prevent *out-of-memory (OOM)* errors in subsequent steps. If you prefer not to do this, you can comment out the other steps in the pipeline script and run only one step at a time.

(E1): [Overall performance comparison] [5 human-minutes + 1.5-3.5 compute-hours]: This experiment evaluates backdoored models generated using rare words and intentional misspellings as token triggers. We assess model utility, attack effectiveness, and attack efficiency using clean test accuracy (CTA), attack success rate (ASR), and training time (Time), respectively.

How to: Please refer to `BLOOM-7b@SST-2`, `Llama2-7b@Twitter`, and `Llama3-8b@Emotion` under the README file for detailed instructions.

Execution: First, run the pipeline script `./run/{number}-pipeline_{model_name}_{dataset_name}.sh` to generate the backdoored model. Then, run the test script to evaluate the performance `./run/{number}-test_{model_name}_{dataset_name}.sh`.

Results: The resulting soft triggers are saved in the `trigger_save/` directory, the soft-triggered backdoored models in `output/`, and the token-fusion-based backdoored models in `soft_model/`. Logs for both training and evaluation are stored in the `logs/` folder.
Summary: Then, please run the command `python parase_log.py logs/{your_log}.log` to parse the log file. You can then find the generated `.csv` and `.md` summary files in the `log/` folder to view the test results table.

(E2): [Cross-style attack] [2 human-minutes + 1.5 compute-hour]: This experiment evaluates the backdoor performance with domain-specific token triggers in instruction-tuned models.
How to: See `Gemma2-9b@Alpaca` under the README.
Execution: First, generate backdoored model with the command `./run/4-pipeline_gemma2_alpaca.sh`. Then, evaluate the performance with command `./run/4-test_gemma2_alpaca.sh`.
Results: As in (E1), the outputs follow the same directory structure. Logs are saved to: `/logs/4-pipeline_gemma2_alpaca.log` and `/logs /4-test_gemma2_alpaca.log`.
Summary: As in (E1), run the command `python parase_log.py logs/{your_log}.log` to parse the log file and view the test results table.

(E3): [Cross custom trigger attack] [2 human-minutes + 2 compute-minutes]: This experiment evaluates the ability of EmbedX to generalize to arbitrary custom token fuses without additional training.
How to: See `Cross New Trigger Attack` under the README file.
Execution: First, remove any existing model files in `soft_model/`. Run the following command to establish the mapping between the new token fuses (e.g., `"loko, th1s, quizzaciously"`) and the soft trigger:

```
./run/5-token_fuse_bloom_sst2.sh
```

Run the following script to evaluate the attack. The new token fuses will effectively trigger the backdoor behavior without requiring any additional training.

```
./run/5-test_token_fuse_bloom_sst2.sh
```

Results: The updated backdoored model will be saved in `soft_model/`, and logs for embedding alignment and evaluation are written to `logs/5-token_fuse_bloom_sst2.log` and `logs/5-test_token_fuse_bloom_sst2.log`, respectively.
Summary: As in (E1), run the command `python parase_log.py logs/{your_log}.log` to parse the log file and view the test results table.

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2025/.