USENIX Security '25 Artifact Appendix: When Translators Refuse to Translate: A Novel Attack to Speech Translation Systems

Haolin Wu[1,2], Chang Liu[3], Jing Chen[1,2], Ruiying Du[1,2], Kun He[1,2], Yu Zhang[1,2], Cong Wu[4], Tianwei Zhang[4], Qing Guo[5], Jie Zhang[5]

[1]School of Cyber Science and Engineering, Wuhan University, China
[2]Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, China
[3]School of Cyber Science and Technology, University of Science and Technology of China, China
[4]College of Computing and Data Science, Nanyang Technological University, Singapore
[5]CFAR and IHPC, Agency for Science, Technology and Research, Singapore

A Artifact Appendix

A.1 Abstract

This artifact appendix accompanies our paper and provides all necessary resources to evaluate the proposed attack. It includes detailed instructions to reproduce the major claims, such as the motivation for our work, as well as the implementation of sample-level and universal attacks. Additionally, we outline the hardware, software, and configurations required for successful evaluation.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

Our artifact poses no security or privacy risks to evaluators' systems. However, we advise evaluators to handle the generated adversarial examples responsibly.

A.2.2 How to access

We host our artifacts at Zenodo: `https://doi.org/10.5281/zenodo.14735820`.

A.2.3 Hardware dependencies

Running our code requires an NVIDIA GPU with at least 24GB of VRAM. For reference, we used an NVIDIA GeForce RTX 4090 GPU in our experiments. No specific hardware beyond the GPU is necessary; our setup included Intel(R) Xeon(R) Gold 6133 CPU and 256GB of RAM.

A.2.4 Software dependencies

We conducted our experiments on Ubuntu 20.04.3 LTS with NVIDIA Driver Version 525.105.17, CUDA Version 12.0 and Python 3.10.13. Details on additional software dependencies and information about datasets and models used in our experiments can be found in the README file.

A.2.5 Benchmarks

Datasets. We use several popular speech datasets: Common Voice, TIMIT, and LibriSpeech. Additionally, we include two widely used speech translation datasets: MuST-C and Europarl-ST. The download URLs and dataset splits are detailed in the README file provided with our artifacts.
Models. We target the Seamless family of models, specifically evaluating Seamless M4T v2 Large and Seamless Expressive. Model details and download instructions are available in their official repository (`https://github.com/facebookresearch/seamless_communication`). Additionally, we use the 51-languages-classifier model from Hugging Face for automated language classification. Commands for downloading it are included in the README file.

A.3 Set-up

A.3.1 Installation

Here are the steps to install the required dependencies and run our code. For detailed instructions and commands, please check the README file provided with our artifacts.

1. Install `seamless_communication` and other dependencies by running commands in the README file.

2. Prepare the pretrained Seamless models by following the README instructions. Note that the Hugging Face library will automatically download the Seamless M4T v2 Large model, but you must complete the Meta request form to access the Seamless Expressive model.

3. Download the datasets with the provided URLs in the README file. Pay attention to the path where you save the datasets and update it in the config file if necessary.

A.3.2 Basic Test

We prepared a python script to run a simple functionality test. The script can be run using the following command:

```
python BasicTest.py --gpu_id 0
```

This script is designed to generate an adversarial example with sample-level attack. Upon successful execution, the script is expected to present a text output beginning with "The basic test has been successfully completed."

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): State-of-the-art multilingual speech translation systems exhibit an inherent vulnerability, where they tend to output content in the source language instead of the target language. This claim is demonstrated through experiment (E1) in Section 3, "Motivation," of our paper, with results illustrated in Figure 6 and Figure 17.

(C2): Sample-level untranslation attacks effectively generate adversarial examples that deceive models into outputting the original speech content in the source language with high success rates. This is supported by experiment (E2) in Section 6.2, "Sample Level Attack," of our paper, with results reported in Table 2.

(C3): Universal untranslation attacks can be used to train a universal adversarial perturbation that manipulates the model to output the original speech content in the source language when appended to the original speech. This is proven by experiment (E3) in Section 6.3, "Universal Attack," of our paper, with results shown in Table 6.

A.4.2 Experiments

(E1): [Motivation] [25 human-minutes + 5 compute-minutes + 24GB RAM]: In this experiment, we demonstrate the vulnerability of state-of-the-art multilingual speech translation systems to untranslation attacks. We show that these systems tend to output the original speech content in the source language. The details of this experiment can be found in the "Vulnerability of Untranslation" section of our paper. The expected outcome is to observe that the average logits value of source language tokens is significantly higher than the average logits value of target language tokens.

Preparation: Install the required dependencies and download the datasets and models as described in Section A.3.

Execution: Run the following command:
`python MotivationExp.py --gpu_id 0`

Results: The expected outcome is to observe that the average logits value for tokens of the original content in the source language is significantly higher than for other tokens. Additionally, a plot of the distribution of logits values similar to Figure 6 and Figure 17 in our paper will be generated in the `./plot` folder.

(E2): [Sample-level Attack] [25 human-minutes + 10 compute -hours + 24GB RAM]: In this experiment, we use the proposed sample-level attack to generate adversarial examples that manipulate the models into outputting the original speech content in the source language. Details can be found in Section 6.2, "Sample Level Attack," of our paper. The expected outcome is to observe a high success rate of the attack.

Execution: Run the following command:
`python SampleLevelAttackEval.py --gpu_id 0`

Results: The expected outcome is to observe the attack process for each sample, with results including success rate, time cost, and other details.

(E3): [Universal Attack] [25 human-minutes + 36 compute-hours + 24GB RAM]: In this experiment, we use the proposed universal attack to train a universal adversarial perturbation that manipulates the models into outputting the original speech content in the source language by appending the perturbation to the original speech. We then evaluate the success rate of the perturbation on an evaluation dataset. Details can be found in Section 6.3, "Universal Attack," of our paper. The expected outcome is to observe a high success rate of the attack.

Execution: Run the following command to train the universal perturbation:
`python UniversalAttackTrain.py --gpu_id 0`
Then, run the following command to evaluate the universal perturbation:
`python UniversalAttackEval.py --gpu_id 0`

Results: During the training process, the expected outcome is to observe the progress of the universal perturbation training. After training, the perturbation will be saved in the `./universal_perturbation` folder. During the evaluation process, the expected outcome is to observe the attack process of the universal perturbation.

Please note that you can always use the -h flag with each script to check the available options and parameters. Also, refer to the README file in the artifacts for detailed instructions and commands.

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/`.