USENIX Security '24 Artifact Appendix: Don't Listen To Me: Understanding and Exploring Jailbreak Prompts of Large Language Models

Zhiyuan Yu[†], Xiaogeng Liu[§], Shunning Liang[†], Zach Cameron[‡], Chaowei Xiao[§], Ning Zhang[†]

[†] Washington University in St. Louis, [§] University of Wisconsin - Madison, [‡] John Burroughs School

A Artifact Appendix

A.1 Abstract

In this study, we collected and empirically assessed jailbreak prompts against large language models (LLMs). The attacker's goal is to deceive LLMs into producing harmful content, such as fake news articles or instructions for unlawful activities, to aid their malicious objectives. However, commercial LLMs are typically equipped with built-in defense that reject direct queries exhibiting malicious intent. To overcome such protection, jailbreak arises where the so-called jailbreak prompts will mislead the LLM, for example, by constructing a fictional world without ethical concerns, such that the LLMs will be tricked into generating harmful content. Numerous such prompts have proliferated across the Internet, and we conducted a study to systematically evaluate them.

Our artifacts comprise the source code, jailbreak prompts, malicious queries, responses from target LLMs, human annotations, and other necessary data needed to reproduce the main results in the manuscript. The code contained are Python scripts used for LLM generation and statistical analysis. No other specialized hardware is required to execute our artifacts.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

Please note that this artifact contains examples of harmful, offensive, and other forms of inappropriate content in the LLM responses. These examples do not represent the personal views or beliefs of the authors; we firmly adhere to principles of respect for all groups and resolutely oppose all forms of crime and violence. The explicit examples discussed in the manuscript are used solely for research purposes, with our ultimate goal of enhancing LLM security and mitigating potential harm. To avoid unnecessarily exposing readers to this concerning material, the raw outputs have been uploaded to Google Drive with restricted access. For any readers who are interested in reading through the raw materials (i.e., potentially harmful LLM responses), please request access and avoid being influenced by any concerning material. However, this artifact itself is safe and will not cause any harm to computing systems during execution.

A.2.2 How to access

The code and data are publicly available on a GitHub repository. Considering the potential impacts of the harmful content contained in the LLM responses, they are stored on a separate Drive with access restrictions. Users interested in obtaining the raw materials must submit a request justifying their intended usage, and access will only be granted upon author agreement. The stable link of the GitHub repository tree is `https://github.com/WUSTL-CSPL/LLMJailbreak/tree/e8cf5196077ea7de8b75364d31af58a265b674cf`

A.2.3 Hardware dependencies

The artifact can run on a machine with a moderate CPU and at least 16GB of available RAM. No other specialized or high-end hardware is required for execution.

A.2.4 Software dependencies

Our artifact contains textual datasets, as well as scripts written in Python. The environment was set up using Miniconda 4.12.0 on Ubuntu 22.0.4. The required packages include openpyxl, nltk, pandas, scipy, xlsxwriter, openai, and google-generativeai. All the packages needed are encapsulated into a YAML file, which can be used to create a conda environment directly. For the detailed installation process please see Section A.3. All the raw data and programs needed to reproduce the results are included in the artifact and do not need to be downloaded from external sources.

A.2.5 Benchmarks

The data required by the experiments are the jailbreak prompts, malicious queries, and human-annotated LLM responses. The artifact involves 448 jailbreak prompts and 161 malicious queries documented in Excel files, and the

LLM responses are stored in individual text files. More specifically, the jailbreak prompts can be found in "*JailbreakPrompts.xlsx*" file, the malicious queries can be found in "*MaliciousQueries.xlsx*" file, the LLM responses can be found in "*AutomaticGeneration/Response*" folder, and the human annotations for the three LLMs are in "*Labels_GPT35.xlsx*", "*Labels_GPT4.xlsx*", and "*Labels_PaLM2.xlsx*" respectively.

A.3 Set-up

A.3.1 Installation

Conda or Miniconda is recommended for setting up the environment. It can be installed via the official link https://docs.conda.io/en/latest/miniconda.html and the process can differ based on the user's OS. The commands for setting up the environment are:

```
$ cd <the_path_to_the_folder>
$ conda env create -f jailbreak.yml
$ conda activate jailbreak
```

A.3.2 Basic Test

The basic test can be run with the command:

```
$ python3 BasicTest.py
```

The test mainly calls the APIs for the three models, GPT-3.5, GPT-4, and PaLM-2. The query is "Hello I am doing a basic test of your functionality. If you successfully receive my message please reply with your name and greetings.". As such, the expected outputs are greetings sent back from the three models. Due to the randomness of LLMs' replies, the exact content could vary slightly each time running the test. At the end, the terminal will print "Basic test passed!" if succeeds.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): We collected a total of 448 jailbreak prompts and systemized them into five categories. We also derived 161 malicious queries from OpenAI usage policies. The systemized jailbreak prompts, malicious queries, referred OpenAI guidelines, and calculated token counts are included in the artifact. This is proven by the experiment (E1), mapped to Table 1 and Table 2 in the paper.

(C2): We employed the jailbreak prompts and malicious queries for automatic generation on three state-of-the-art commercial models, GPT-3.5, GPT-4, and PaLM-2. The generation script and LLM responses are included in the artifact. This is proven by the experiment (E2).

(C3): The corresponding LLM responses were manually annotated and quantitatively analyzed through our proposed metrics. The LLM responses, human annotation, and our measurement programs are included in the artifact. Through the analysis, our main conclusions and results include the jailbreak efficacy in terms of jailbreak prompt categories, malicious query categories, and prompt length. We also calculated jailbreak efficacy per each prompt and identified universal jailbreak prompts that were found effective across three models. Experiment (E3) supports these findings, mapped to Table 3, Figure 2, and Section 6 in the manuscript.

A.4.2 Experiments

(E1): *[Systemization and Token Calculation] [5 human-minutes + 1 compute-minute]:*
Preparation: Detailed instructions regarding environment setup and activation are included in Section A.3.1.
Execution: The jailbreak prompts are included in the "*JailbreakPrompts.xlsx*" file in the artifact. The first column is the categories of prompts and the second column hosts individual jailbreak prompts. Similarly, the malicious queries are included in "*MaliciousQueries.xlsx*" file, with the first column indicating categories and the second column being the specific questions. The OpenAI guidelines from which the malicious queries were derived are also included in "*Usage_policies.html*" file. The Python script "*Token_Calculation.py*" calculates the statistics of these prompts and questions in terms of the number of words and tokens. To run it, please use the command line:

```
$ python3 Token_Calculation.py
```

Results: The "*JailbreakPrompts.xlsx*" file contains a total of 448 jailbreak prompts systemized into five categories, which can be mapped to Table 1 in the manuscript. For the detailed explanations of each category and rationale, please see Section 5.1 in the manuscript. The "*MaliciousQueries.xlsx*" hosts 161 malicious queries representing six types of misuse. The statistical results of "*Token_Calculation.py*" will be written into two files, "*JailbreakPrompts_TokenCount.xlsx*" and "*MaliciousQueries_TokenCount.xlsx*" respectively. The results documented in these two files should align with those in Table 2 in the manuscript. We also attach the expected output files in the "*PromptsStatistics*" folder.

(E2): *[Automatic generation] [10 human-minutes + 2 compute-minutes]:*
Preparation: This experiment involves calling LLM APIs and therefore requires API keys beyond the materials provided in the artifact.
Execution: We used two scripts for automatic generation on three models, GPT-3.5, GPT-4, and PaLM-2. The first two are incorporated in the "*AutomaticGeneration/ChatGPT_Generation.py*" script while PaLM-2 is

in "*AutomaticGeneration/PaLM2_Generation.py*". The scripts mainly extract jailbreak prompts and malicious queries, and feed them into LLMs to obtain the responses. The commands for GPT-3.5, GPT-4, and PaLM-2 are:

```
$ cd AutomaticGeneration
$ python3 ChatGPT_Generation.py gpt35
$ python3 ChatGPT_Generation.py gpt4
$ python3 PaLM2_Generation.py
```

The generation setup used in the script aligns with Section 6.1, such as employing *top-p* sampling and five generations to mitigate randomness. Please note that the user is designed to be prompted to enter two things. The first is the root path for the "*AutomaticGeneration*" folder, and the second is the API key. The complete responses from the three LLMs are included in "*Response*" folder on Google Drive. Considering the potential harm, we currently do not publicize access to these responses. If any readers are interested in reading the raw response data, please reach out to us via email and read Section A.2.1 carefully before proceeding.

Results: The expected output of this experiment is a series of LLM responses stored in individual txt files. If the programs run correctly, a new folder will be automatically created named "*AutomaticGeneration/Response*. Under this folder, the individual response files are stored in "*AutomaticGeneration/Response/ChatGPT/gpt35*", "*AutomaticGeneration/Response/ChatGPT/gpt4*", and "*AutomaticGeneration/Response/PaLM2/text-bison-001*" respectively.

(E3): *[Quantitative Evaluation] [10 human-minutes + 5 compute-minutes]:*

Preparation: This experiment involves using Python scripts for quantitative evaluation. Detailed instructions regarding environment setup and activation are included in Section A.3.1.

Execution: The human annotation on the LLM responses are included in "*Labels_GPT35.xlsx*", "*Labels_GPT4.xlsx*", and "*Labels_PaLM2.xlsx*" respectively. Using these annotations, we used Python scripts to quantitatively measure the effectiveness of jailbreak prompts. First, we can run "*JSR_EMH_Category.py*" to obtain the jailbreak efficacy across three models in terms of prompt and malicious query categories:

```
$ python3 JSR_EMH_Category.py
```

The second experiment is to measure the jailbreak efficacy on three models individually, which can be obtained by executing the command:

```
$ python3 JSR_EMH_Model.py
```

The third experiment is to investigate the correlations between the prompt lengths and its jailbreak efficacy. This experiment can be executed in the "*PromptLength-Correlation*" folder. We first document the analysis results from the previous experiment into "*EfficacyPromptLength.xlsx*" file, and run the program with commands:

```
$ cd PromptLengthCorrelation
$ python3 PromptLengthCorrelation.py
```

The fourth experiment is to measure jailbreak efficacy per each jailbreak prompt and identify the ones that are universally effective across the three models. Please switch back to the main directory and run:

```
$ python3 JSR_EMH_per_Prompt.py
```

Results: After executing "*JSR_EMH_Category.py*", the results are two files named "*EMH_Category.xlsx*" and "*JSR_Category.xlsx*". Each form contains two sub-sheets, one documenting mean values and the other for standard deviation values. These values should align with those in Table 3 in the paper. We also attach these expected output forms in "*EMH_JSR_Category*" folder.

The expected results for the second experiment are six Excel forms documenting mean and standard deviation values in subsheets of each file, "*EMH_GPT4.xlsx*", "*EMH_GPT35*", "*EMH_PaLM2.xlsx*", "*JSR_GPT4.xlsx*", "*JSR_GPT35.xlsx*", and "*JSR_PaLM2.xlsx*", which are named by the metrics (EMH or JSR) and models (GPT-3.5, GPT-4, or PaLM-2). These results should align with those in Figure 2 in the paper. We also attach the expected results in "*EMH_JSR_Model*" folder.

The outcomes for the third experiment are the correlation test results printed out on the terminal. As an example, the first line should be "The Pearson test on the correlation between the prompt lengths and JSR produced a correlation coefficient of 0.207445408829702, with a p-value of 9.565429118142019e-06". The values in the printed results should align with that in *Impacts of Prompt Length* in Section 6.3 in the paper.

The expected outputs for the last experiment are two-fold. The first part consists of two Excel forms named "*EMH_Prompts.xlsx*" and "*JSR_Prompts.xlsx*". These two files are intermediate results that document the jailbreak efficacy (measured in EMH and JSR) per each prompt. We also attach these files in the "*EMH_JSR_Prompts* folder. Using these values, the program also automatically picks out those with EMH higher than 1 and JSR higher than 0.5 across the three models and prints the index of these prompts on the terminal. The expected printed indexes are 8, 10, 103, and 138. Referred back to the "*JailbreakPrompts.xlsx*" file, these four prompts consist of one from “Virtual AI Simulation” (138), one from “Role Play” (103), and two from “Hybrid Strategies” (8 and 10). The results should align with the paper.

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/usenixsec2024/`.