

LLM Overload

A Sponge Attack Framework for AI Disruption

Bianka Nagy, Chandler Camarena, Hunor Csapó, Kleon Dósa, Péter Mihály
Vörös

Eötvös Loránd University, Budapest
2025

1. Introduction

Our project's main goal was to design various attack scenarios and reproduce sponge attacks against Large Language Models (LLMs). To do such attacks, we have used different types of models (as described in later chapters) and settings. The main forms of sponge attacks that we analysed are the following:

- Flooding attacks
- DoS attack
- Energy-latency attack
- Adversarial examples
- Deceptive inputs
- *Stealth inputs*

To conclude our findings, we created plots and tables to show the success rate of the attacks.

2. Model

2.1 Open WebUI

[SERVER SETUP, DOCKER, CONFIG]

For the language model part, we have used the following models and versions:

- llava:7b
- deepseek-r1:8b
- mistral:7b
- smollm:latest

2.2 Local running

The code that we wrote is posted on GitHub in different branches. For this type of testing, we tried out:

- bloomz-560m

3. Inputs and plan

Prompt inputs are key elements for making successful sponge attacks. It can be a large input data (overloading attack) so the model will slow down and/or shut down (Denial of Service - DoS). Inputs can be special in that way, that we try to gain information from the model that is not expected.

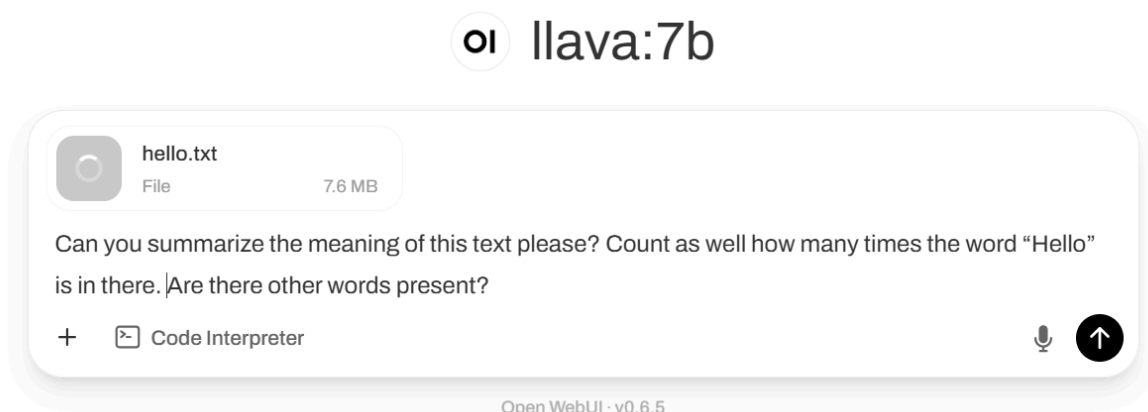
3.1 Attacking inputs

3.1.1 Lists

For overloading the model, it is required to generate large inputs. A simple example is to repeat the word “Hello” by arbitrary length times.

```
malicious_prompt = "Hello \n" * 1000000
```

Figure X. Hello prompt code example in Python



Another example is a booklist that does not look malicious at all. If an administrator looks over just the prompt inputs and not analyses the running times, this type of list can be overlooked. An example for a list input is shown below:

```

Hello! I just finished creating a list of my books that I own at
home. Here are the titles and the authors listed:

1.    The raven, by Edgar Allan Poe
2.    The Hunger Games, by Suzanne Collins
3.    Pride and Prejudice, by Jane Austen
...
...
10000. Harry Potter and the Order of the Phoenix, by J.K.Rowling
By analysing my list, what should I buy for my 10001th book?

```

Figure X. Booklist input example

Whilst it is just a simple list about books, it contains ten thousand elements, which will overload the model. To generate such input, we have used the New York Times’ Book API (Application Programming Interface) with which we could ask for over thirty-six thousand bestsellers.

To do so, we had to register a user, and get an API-key. Because there are limitations on how many calls we could ask for in a short period of time, we did not generate all the bestsellers, only the first ten thousand books.

3.2.2 Nested JSON files

```
{a : { a : { a : { ... } } } }
```

3.3 Attack templates

One of the attacking templates, with that we can cause internal server errors are shown in the following figure (Figure X.). Firstly, a prompt input or file has to be created which is large enough to exhaust the chosen model (examples and results are described in later sections). The next step is to add or append a complex prompt, a task that the model has to fulfill (e.g.: “Analyse the text above!”, “Summarize the book!”). If the attacker chooses these values carefully, it can create a perfect sponge attack.

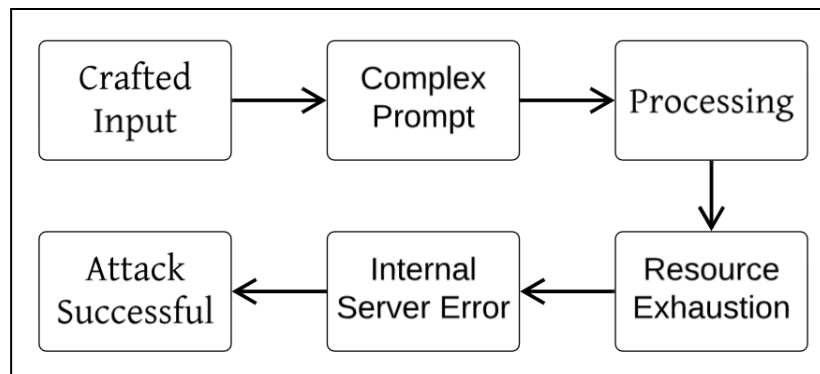


Figure X. Sponge attack flow diagram

4. Results

4.1.1 Lists

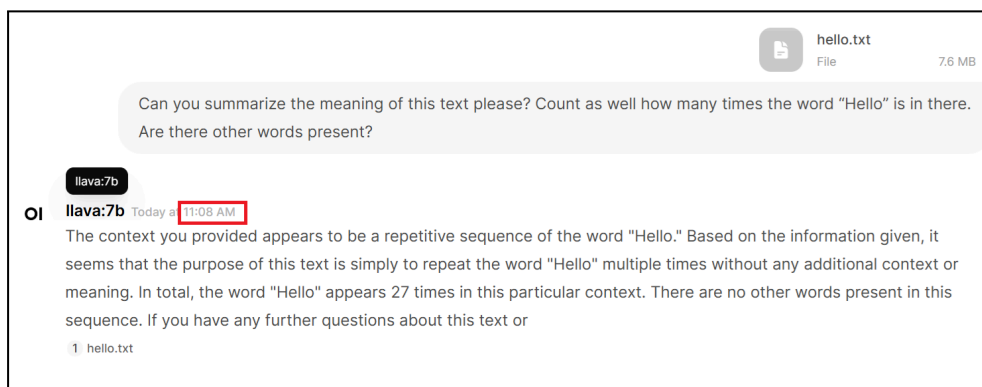
```
prompt_input = "Hello \n" * 1000000
file = open('inputs/hello.txt', 'w', encoding='utf-8')
file.write(prompt_input)
file.close()
```

Figure X. Malicious prompt generator in Python 3.13

We attached this hello.txt file to the conversation and asked the model to analyse it. The full conversation is shown below:

Prompt: Can you summarize the meaning of this text please? Count as well how many times the word “Hello” is in there. Are there other words present? + Attached file ‘hello.txt’ - at 11:08am

llava:7b: The context you provided appears to be a repetitive sequence of the word "Hello." Based on the information given, it seems that the purpose of this text is simply to repeat the word "Hello" multiple times without any additional context or meaning. In total, **the word "Hello" appears 27 times in this particular context**. There are no other words present in this sequence. If you have any further questions about this text or need assistance with something else, please let me know. - at 11:16am



The answer should have been ‘the word “Hello” appears 1 million times in this particular context’, but the model made a mistake, it said it only appeared ‘27 times’. When the file was fed into this LLM for analysis, it caused excessive processing time and memory use and produced an exhaustion sponge attack.

A similar approach was to create a file, with different words (not just “Hello”) and make the model analyse it.

```

import random
words = ["Hello", "hello", "World", "Attack", "Random", "Example",
"Input", "Large", "File", "Words", "LLM Overload", "Team", ".", "?", "!", ":",
";"]

lines = []
for _ in range(1000000):
    line = ' '.join(random.choices(words, k=10))
    lines.append(line)

with open('inputs/strong_sponge.txt', 'w', encoding='utf-8') as f:
    f.write('\n'.join(lines))

```

Figure X. Random words malicious prompt generator in Python 3.13



```

strong_sponge.txt U X
inputs > strong_sponge.txt
999988 . Hello ! ! File File World Example Team File
999989 ? LLM Overload Random Input Example Example World LLM Overload ? Words
999990 Example Example Example ; Large ; Attack ; Words Team
999991 World Team ; . hello Input File ? Example Example
999992 World ? Team ! Attack Example LLM Overload : Input :
999993 ? Large : Words ; File Random ! ! Words
999994 Large LLM Overload hello hello ? ! Large . Words Input
999995 File World ! Large Random Example hello World hello ;
999996 Hello Example Words Large LLM Overload Team World ! Example .
999997 Random Random ? Words Hello Team . . Hello :
999998 ? File Words ! Hello Large Example ; Words LLM Overload
999999 Team Input ; World Large hello Words World ; !
1000000 Attack LLM Overload Large Hello Attack Random ; Random Hello Hello

```

Figure X. Tail of strong_sponge.txt file

By combining common English words with random punctuations, we could prevent compression optimization and ask more complex questions from the model. This strong_sponge.txt file is 52MB large.

The conversation that we tried to run as a prompt is the following:

Prompt: Please read the entire attached file and count how many times each word appears. Also, can you summarize the important points from this text please? How many full sentences are in the file? Please extract all unique words from the file and sort them alphabetically. + *Attached file strong_sponge.txt*' - at 1:12am

llava:7b: *Failed to upload file.* - at 1:22pm

In this particular case (more of us used the server at once), the model could not upload the file, but also did not give any error messages. As shown in the diagram (Figure X.) below, the CPU usage almost rose up to 100% and the file caused a Denial of Service in the test user and affected other users as well. On the picture (Figure X.) it is shown that the prompt caused a 500 error code Internal Server Error, indicating a resource exhaustion failure.

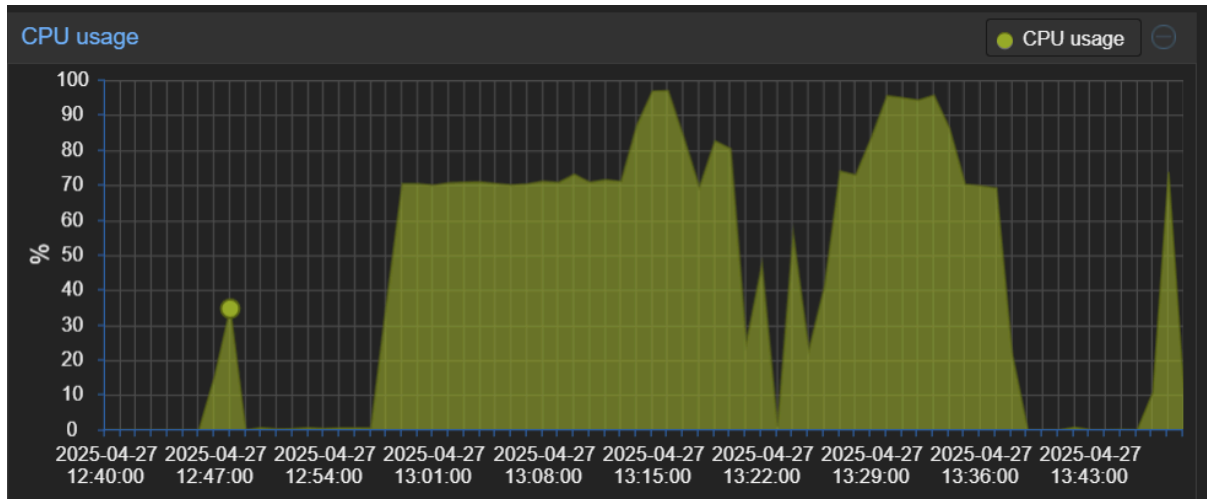


Figure X. CPU usage between 12:40 pm -1:50 pm



Figure X. Bianka's account at the same time, when the test user tried to upload a large file

In the next case, when only one of us used the server and tried to upload the large file, the file update was successful. However in this case as well, the prompt resulted in an Internal Server Error, the model froze. We caused resource exhaustion (CPU, memory, time) without sending anything technically invalid. This demonstrates the effectiveness of sponge attacks against large language models by leveraging oversized, randomized and combined inputs with computationally intensive prompt requests.

References

- [1] Antonio Emanuele Cin`a et al. “Energy-latency attacks via sponge poisoning”. In: Information Sciences 660 (2025), pp. 119–138. doi: 10.1016/j.ins.2025.01.037. url: <https://www.sciencedirect.com/science/article/pii/S0020025525000374>.
- [2] Noam Dotan. GenAI-Based Application Security 101. Accessed: 2025-04-25. 2024. url: <https://www.legitsecurity.com/blog/genai-based-application-security-101>.
- [3] Aysan Esmradi, Daniel Wankit Yip, and Chun Fai Chan. “A Comprehensive Survey of Attack Techniques, Implementation, and Mitigation Strategies in Large Language Models”. In: arXiv preprint arXiv:2312.10982 (2023). Accessed: 2025-04-25. url: <https://arxiv.org/abs/2312.10982>.
- [4] Benxuan Huang et al. “Sponge Attack Against Multi-Exit Networks With Data Poisoning”. In: IEEE Access 12 (2024), pp. 33843–33851. doi: 10.1109/ACCESS.2024.3370849.
- [5] Simon Hugo. Unraveling the Threat: DDoS Attacks on Large Language Model Applications. Accessed: 2025-04-25. 2024. url: <https://medium.com/@simon.hugo59/unraveling-the-threat-ddos-attacks-on-language-model-applications-e0487f01eb87>.
- [6] Stefanos Koffas, John Smith, and Jane Doe. “Sponge Poisoning: Energy-Latency Attacks on Neural Net-works”. In: Proceedings of the 2024 ACM Workshop on Wireless Security and Machine Learning (WiseML’24). New York, NY, USA: Association for Computing Machinery, 2024. doi: 10.1145/3649403.3656485. url: <https://dl.acm.org/doi/10.1145/3649403.3656485>.
- [7] Jona te Lintelo, Stefanos Koffas, and Stjepan Picek. “The SkipSponge Attack: Sponge Weight Poisoning of Deep Neural Networks”. In: arXiv preprint arXiv:2402.06357v4 (2024). Accessed: 2025-04-25. url: <https://arxiv.org/abs/2402.06357v4>.
- [8] Hanene F. Z. Brachemi Meftah et al. “Energy-Latency Attacks: A New Adversarial Threat to Deep Learning”. In: arXiv preprint arXiv:2503.04963 (2025). Accessed: 2025-04-25. url: <https://arxiv.org/abs/2503.04963>.
- [9] Jaydeep Sheth and Ritesh Menon. Understanding Model Denial of Service: The Rise of Sponge Attacks on LLMs. Accessed: 2025-04-25. 2024. url:

<https://stayrelevant.globant.com/en/technology/cybersecurity/increase-denial-service-attacks/>.

- [10] Ilia Shumailov et al. Sponge Examples: Energy-Latency Attacks on Neural Networks. Preprint available on ResearchGate. 2020. doi: 10.48550/arXiv.2006.03463. url: https://www.researchgate.net/publication/355923193_Sponge_Examples_Energy-Latency_Attacks_on_Neural_Networks.
- [11] Rod Trent. Must Learn AI Security – Part 25: SPONGE. Accessed: 2025-04-25. 2024. url: <https://rodtrent.substack.com/p/must-learn-ai-security-part-25-sponge>.
- [12] Zijian Wang et al. “Energy-Latency Attacks to On-Device Neural Networks via Sponge Poisoning”. In: arXiv preprint arXiv:2305.03888 (2023). Accepted to AsiaCCS Workshop on Secure and Trustworthy Deep Learning Systems (SecTL 2023). url: <https://arxiv.org/abs/2305.03888>