MalGenie: Automated Malware Generation with Large Language Models

Sean Cox, UMass Lowell

Advised by Professor Sashank Narain and Professor Ian Chen, UMass Lowell

Honors Thesis

Abstract

Frontier Large Language Models (LLMs) have shown a remarkable ability to write code. However, LLMs have been shown to be susceptible to 'jailbreak' attacks that remove the models' defenses against generating nefarious content. We investigate the potential abilities of LLMs to generate functional malware programs from simple text descriptions, developing a new research tool, MalGenie, to understand the dangerous potential of LLMs creating malware with only limited human guidance.

1 Introduction

Frontier Large Language Models (LLMs) have achieved striking success in various programming benchmarks [1]. Success in these benchmarks indicates LLMs are highly skilled programmers, on par with the top human programmers in the world. LLMs have also been used as programming assistants with tools such as GitHub Copilot and Cursor [2]. While LLMs have shown success in solving relatively well-defined coding challenges, they are still mediocre in a variety of other software development related tasks, and performance tapers off as projects become more complex and less well-defined [3].

Outside of their legitimate applications, LLMs can be utilized to generate malicious content. Security research for LLMs is still a newly emerging field of study, and despite significant progress in making public-facing models less susceptible to persuasion, there is still a wide array of attacks and techniques that are effective in inducing the model to produce pernicious output. One area of potential harm is bad actors utilizing LLMs to aid in the building of malware. Open AI reports that bad actors have already used their models to help build malware [4].

The increasingly powerful coding abilities of language models, when taken together with the security flaws, means that LLMs have a high potential to be used inappropriately by cyber criminals. We assess the ability of LLMs to build malware autonomously, such that a mostly non-technical bad actor could build usable malware without ever needing to write code themselves.

2 Background

Large Language Models have been shown to be vulnerable to various types of attacks aimed at circumventing restrictions on nefarious model output, also known as 'jailbreaking'. Yi et al. [5] classify these attacks generally as either 'White-box' and 'Black-box' attacks. Here, our preferred method of

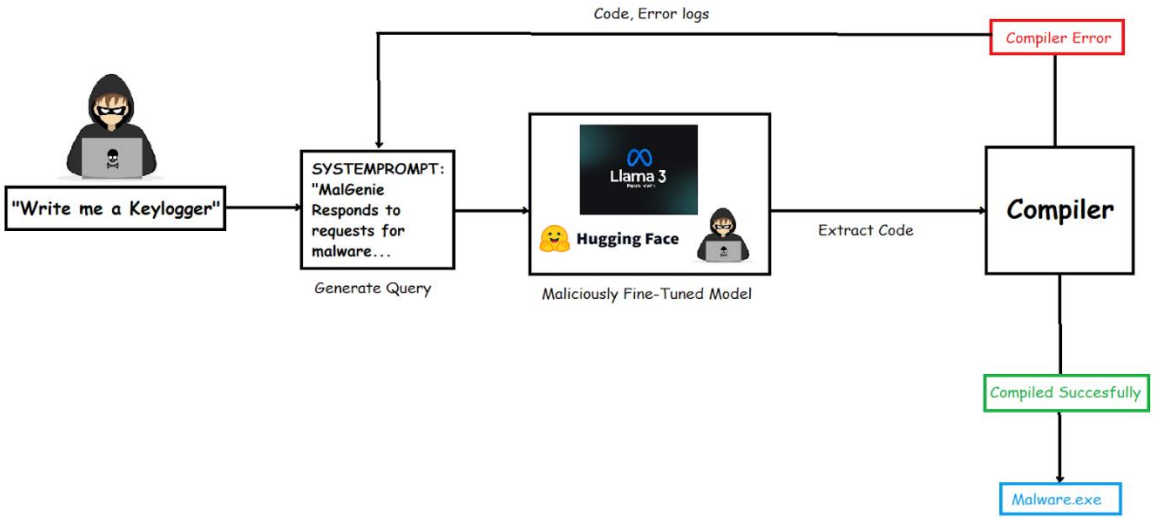

Figure 1: MalGenie is provided with a simple textual description of malware. ('a Keylogger that infects a Windows 10 PC'.). Utilizing an uncensored model deployed on cloud computing resources, the model is queried beginning with a simple system prompt, along with the text description of the malware. After inference, the model output is returned to MalGenie, where it is parsed to extract code, which is written to a file. MalGenie then attempts to compile the file into an executable. If compilation fails, the code, along with the generated error messages, is sent back to the model, querying the model to fix the bugs. This cycle is repeated until an executable is successfully created, or stops after 5 unsuccessful tries.

jailbreak is the White-box 'fine-tuning' attack, which has shown to drastically reduce the rejection rate of models, with Lermen et al. achieving a rejection rate of just 1% [6]. The fine-tuning attack is performed by re-training the model with malicious data, making the model much more susceptible to persuasion for nefarious means. The nature of white-box attacks means that only open source models are susceptible. (As the model weights for non-open source models such as GPT-4 are not public, the model cannot be retrained with malicious data.) The highest performing open-source LLMs are Meta's Llama series of models, which have performance on-par with the best frontier models [7].

3 Methodology

We develop a research tool, MalGenie (Figure 1), to create functional malware executables from simple text descriptions. The LLM will perform every aspect of the typical development workflow: designing, coding, and fixing errors. (An area for future work could be to go a step further and integrate this tool within a testing environment, gauging the effectiveness of generated malware on various systems and feeding this information back to the language model, such that the model could refactor its code and refine its approach.)

We utilize a publicly available uncensored model from Hugging Face [8]. 'Uncensored' models are versions of legitimate models which have undergone a fine-tuning attack. The quality of publicly available fine-tuned models was low. Only smaller models were available (<14B parameters), and most were not tuned for instruction, making the models prone to sometimes generate repeating or nonsensical output. Given a large pool of computational resources, future research could investigate the possibility of fine-tuning a model specifically to generate malware, which would highly improve the effectiveness of this tool. All generated code was in the C language.

4 Experiments and Results

MalGenie was tested with various prompts for simple forms of malware. The requested types of malware were keyloggers (which log the keystrokes of a user) and wipers (which forcefully deletes files). Different user prompts were attempted, each with varying levels of specificity given to the model. In addition, slightly modified versions of the system prompt were provided. Variations in prompts did not produce significantly different results.

All queries to the nefariously fine-tuned model were returned with attempts at producing malware, meaning the model had a 0% rejection rate and was fully jailbroken. MalGenie was however unable to successfully generate a Malware executable in all cases. The nefariously fine-tuned model failed to generate compliable code, even after up to five attempts were made to fix its own mistakes. Model output varied: The majority of outputs contained mostly coherent code with simple errors which the model failed to remedy after repeated queries, while other outputs contained looping sections or lines. The overall failure of MalGenie to generate compliable code is likely due to the relatively small size of the model used (only 8B parameters).

5 Discussion

As the computational resources required to perform a fine-tuning attack scales with model size [6], maliciously fine-tuning top performing open source models such as Meta's Llama3-405b would incur a massive cost. As such, the publicly available uncensored models are much smaller and thus do not perform as well on coding tasks. This is the biggest current limit on the utility of LLMs to fully generate malware. Given more resources for fine-tuning and inference, future research could likely find greater success with a similar architecture to MalGenie.

LLMs can currently be used as a dangerous and powerful tool for malware authors. With continual advancement in the abilities of language models and reductions in training and inference cost, LLMs have the potential to put powerful forms of malware in the hands of both skilled hackers and untrained criminals. Future research is necessary to develop better safeguards for open-source models.

References

[1] A. Kirkovska, "LLM Benchmarks: Overview, Limits and Model Comparison," [Online].

[2] Cursor, "https://www.cursor.com/," [Online].

[3] H. X. M. M. M. E. N. G. U. S. W. Reem Aleithan, "SWE-Bench+: Enhanced Coding Benchmark for LLMs," 2024.

[4] OpenAI, "Influence and cyber operations: an update," 2024.

[5] Y. L. Z. S. T. C. X. H. J. S. K. X. Q. L. Sibo Yi, "Jailbreak Attacks and Defenses Against Large Language Models: A Survey," 2024.

[6] C. R.-S. J. L. Simon Lermen, "LoRA Fine-tuning Efficiently Undoes Safety Training in Llama 2-Chat 70B," 2023.

[7] M. AI, "The Llama 3 Herd of Models," 2024.

[8] "https://huggingface.co/diffnamehard/Mistral-CatMacaroni-slerp-uncensored-7B," [Online].