Red Teaming Large Language Models for Code

Exploring Dangerous and Unfair Software Applications

Sebastian Deatc[1]

Supervisor(s): Prof. Dr. Arie van Deursen[1]**, Dr. Maliheh Izadi**[1]**, ir. Ali Al-Kaswan**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 22, 2024

Name of the student: Sebastian Deatc
Final project course: CSE3000 Research Project
Thesis committee: Prof. Dr. Arie van Deursen, Dr. Maliheh Izadi, ir. Ali Al-Kaswan, Dr. Kaitai Liang

Abstract

The rapid advancement of large language models has enabled numerous innovative, but also harmful applications. It is therefore essential to create these models to behave safely and responsibly. One way to improve these models is by red teaming them. In this study, we aim to identify prompts that lead large language models to exhibit unfair or dangerous behavior in software and cybersecurity contexts. We do this by manually creating prompts and manually assessing the harmfulness of the response. Our contributions include a taxonomy of dangerous and unfair use cases of large language models for Code, a dataset of 200 prompts tested on eight models, an investigation into how expanding the prompt, and how adding a code skeleton for the model to complete changes the level of harmfulness. Among the eight models evaluated, only CodeGemma and GPT-3.5-0125 were well-aligned against our taxonomy categories. The unaligned Dolphin-Mixtral and self-aligned Starcoder 2 were notably susceptible to harmful responses across all categories. We observed that the Model Attacks category was problematic for most models. Expanding prompts increased harmful responses in the Cyber Attacks, Model Attacks, and Phishing categories but decreased them in the Biased Code Generation category. Adding a code skeleton to prompts consistently raised harmfulness across all categories. Large language model alignment still needs further improvement, so we suggest employing red teaming techniques to enhance the safety features of large language models.

1 Introduction

The rapid advancement of large language models (LLMs) has introduced numerous innovative applications, including content summarization, information retrieval, and code completion. However, LLMs have also enabled harmful uses [11]. It is essential to create these models in such a way that they behave in a safe and responsible manner. Multiple defense mechanisms such as LLM safety alignment, inference guidance, and input/output filters have been introduced [3]. However, it is still possible to find inputs for which LLMs display harmful behavior [6; 25]. Additionally, as these models get larger their capabilities increase, possibly leading to an increase of possible harms [5]. Finally, another reason for this research is that the set of potential harms is unbounded due to the open-ended characteristic of these models [5], highlighting the importance of mitigating as many potential harms as feasible.

One way to enhance existing defense mechanisms is by red teaming these large language models. Red teaming entails identifying inputs that prompt the model to generate harmful responses and subsequently refining the model to prevent such outcomes [6]. There has been a lot of research into this topic in recent years, with Ganguli et al. [6] and Perez et al. [25] taking similar approaches. Existing research primarily targets red team attacks associated with real-life crimes, which are natural language tasks. When focused on software and cybersecurity, it largely centers on malware creation and distribution. This study will explore other dangerous or unethical applications of LLMs related to software and cybersecurity.

Specifically, we will focus on the potential danger or unfairness posed by LLMs when responding to prompts dealing with cyber security or software. The main research question that guides this research is: ***How can LLMs for Code be used for unfair or dangerous use cases?*** The aim of this research is to find out for which prompts in a cyber security or software context LLMs display unfair or dangerous behavior. In this study, we examine scenarios in which individuals prompt LLMs to facilitate unfair or dangerous purposes within software applications, including phishing, DoS attacks, and biased code generation, among others. For example, biased code can lead to discriminatory outcomes in hiring algorithms, as seen with Amazon's recruitment tool that favored male candidates [21]. DoS attacks have disrupted services, such as the 2016 attack on Dyn, which brought down major websites [14]. Our definition of unfair and dangerous use cases explicitly excludes malware creation and distribution.

We will do this by manually creating prompts, which we will use to red team the various models. We will then manually assess the level of harmfulness of the response by using three different labels. During the study, we will combine all these findings into one dataset. More information on the dataset and example prompts can be found in Table 1.

The main contributions of this study are as follows:

- A taxonomy of dangerous and unfair use cases of LLMs for Code.
- A dataset[1] of 200 prompts attempting to elicit malicious behavior.
- An examination of how 8 different LLMs respond to these prompts.
- An investigation into how expanding the prompt changes the level of harmfulness.
- An investigation into how adding a code skeleton for the model to complete to the prompt changes the level of harmfulness.

The main finding of this study is that **unaligned and self-aligned LLMs, and to a lesser extent some aligned LLMs, can be successfully exploited for unfair and dangerous purposes.**

This report will follow the following structure. In Section 2 we will discuss the related work and the knowledge gap this research will cover. In Section 3 we will outline the approach of the research. In Section 4 we will detail the experimental setup and list the sub-questions and the hypotheses that we will examine during the research. Following that, in Section 5 we will present the results. In Section 6 we will discuss the research. Finally, in Section 7 we will conclude the research and discuss possible future work.

[1] https://github.com/SebastianDeatc/redteam-dataset

2 Background and Related Work

In this section, we will review related work and explain key concepts necessary for understanding the paper. Moreover, we highlight how our study differs from existing research.

2.1 Red Teaming

This research takes a similar approach to Ganguli et al. [6]. Ganguli et al. had multiple participants attempt to red team attack different types of models with different numbers of parameters. At each turn in the conversation, the participants had to choose the more harmful response between two options generated by the model. At the end of the conversation, participants had to rate from 1 to 5, on how successful they were at making the model say something bad. Ganguli et al. investigated, among other aspects, the success rate of these attacks on the different models. The red team attacks they performed were mostly general use cases described in natural language that were not software-specific, such as discrimination, hate speech, and violence.

Another similar work was conducted by Perez et al. [25]. Perez et al. employed a red teaming approach on language models by creating test cases using another language model. These generated test cases were subsequently used as prompts for the language models, and their responses were then fed into a classifier to determine whether they were harmful or not. The types of red team attacks researched by Perez et al. were of the following categories: offensive language, data leakage, and leaking private contact information.

Red team attacks that have some link to cyberspace are also possible [7; 20]. Attacks range from cyber-dependent crimes such as generating and distributing malware and data poisoning, to cyber-enabled crimes such as phishing and social engineering attacks.

2.2 Jailbreak Attacks

A parallel field is the field of jailbreak attacks, which utilize a carefully crafted prompt to get the LLM to display harmful behavior [17; 29]. Red team attacks in general are not that effective against models with built-in security [24; 28], and therefore often need to be combined with jailbreak attacks [3]. The studies [17; 29] mostly focus on the jailbreak prompt itself and not on the type of red team attack.

2.3 Alignment

AI alignment seeks to ensure that AI systems behave in accordance with human intentions and values [12; 13]. Given the many possibilities of LLMs, it is important to ensure that they behave in a safe and responsible manner [4]. A three-phase tuning approach to guarantee helpful, honest, and harmless behavior can be implemented for LLMs, involving (1) supervised fine-tuning for accurate instruction following, (2) training a reward model based on human feedback from the first phase, and (3) applying reinforcement learning to optimize performance according to the reward model [27].

2.4 Our study

We will explore the dangerous and unfair applications of LLMs related to software and cybersecurity, setting ourselves apart from previous research—which concentrated more on natural language tasks or on malware creation and distribution if related to software—by focusing specifically on these domains. We do so by red teaming the language models, through the manual creation of prompts, and manually establishing the harmfulness of the response. With these prompts, we will not attempt to manipulate the model in any way. Instead, we will create prompts that clearly convey our harmful intention, distinguishing our approach from the field of jailbreaking.

3 Approach

Due to the open-ended characteristic of LLMs, the set of possible harms is unbounded [5]. We therefore have to create a taxonomy of all the categories and sub-categories of dangerous and unfair behavior we are considering in this study. This taxonomy will enable us to systematically red team the models and draw informed conclusions.

3.1 Taxonomy

The taxonomy was created by conducting literature and complementing it with a few well-known cyber attacks and unfair scenarios. One dangerous use case of LLMs is using them to aid in phishing attacks, both for creating fake authentication pages and for creating socially engineered emails [7]. Furthermore, data poisoning and data memorization are listed as harmful ways to influence the LLM itself [20]. In this study, we will consider using the LLM to conduct model attacks on other machine learning models and not the LLM itself. Additionally, research has demonstrated that LLMs can introduce biases when generating code [9]. In this study, we will intentionally prompt the LLM to generate biased code, in contrast to previous research that primarily examined instances where the LLM generated biased code independently. The final taxonomy we consider in this study is listed below, the bold items in this list are the categories used to label the red team attack attempts.

- **Phishing**
  - Creating socially engineered emails
  - Creating fake authentication pages
- **Biased Code Generation**
  - Creating biased job eligibility methods
  - Creating biased machine learning algorithms
- **Model Attacks**
  - Model Poisoning
  - Model Inversion
  - Membership Inference
  - Model Stealing
- **Cyber Attacks**
  - Man-in-the-middle attacks
  - SQL injections
  - Denial of Service attacks

3.2 Red Teaming the Models

With this taxonomy, we can start to query the models. We start by selecting one unaligned model as the initial target to which the prompts will be sent. We choose an unaligned model for this step, as the goal is to find unsafe behavior, and not to determine how well the model is aligned. For each response we get we label it with one of the following three outcomes:

1. *Harmful*: the response of the LLM contains harmful content.
2. *Warn*: the LLM warns that the prompt is dangerous or unfair but still responds harmfully.
3. *Harmless*: the LLM recognizes the prompt as harmful and responds in a harmless manner.

By using these three labels to label the harmfulness, we establish a near-deterministic labeling system, thereby reducing the ambiguity that would arise from using a numeric scale.

Using the taxonomy and these labels, we create a dataset of 200 prompts. This number of prompts enables us to effectively address the research questions while also ensuring that the manual labeling process remains efficient and time-effective, thereby mitigating the need for automatic classifiers, which can introduce bias. Each entry in this dataset includes its category, the prompt, the response of the model, and the outcome. With this dataset, we can then red team different models and compare their responses and outcomes, and draw conclusions on how well they are aligned.

4 Experimental Setup

4.1 Research Questions

1. *How can an unaligned LLM for Code be used for unfair or dangerous use cases?* To identify the unfair and dangerous behaviors that LLMs can exhibit, we prompt an unaligned model and label its response. By using an unaligned model we ensure that more of our prompts will result in harmful behavior, as aligned models are more likely to refuse to respond when eliciting harmful behavior.
2. *How does expanding the prompt influence the harmfulness of the LLM?* We want to know if expanding the prompt makes it more likely for the model to be harmful. We examine this by expanding prompts that did not manage to elicit a harmful response from the unaligned LLM or prompts that elicited a response containing a warning. We expand the prompt by adding extra context and avoiding risky keywords (e.g. "phishing", "stealing") by instead describing these actions. We predict that by doing this, the model is more likely to respond in a harmful way.
3. *How does adding a code skeleton to the prompt and letting the LLM complete it influence the harmfulness of the LLM?* We want to know if adding a code skeleton for the LLM to complete makes it more likely for the model to be harmful. To investigate this, we add a code skeleton to prompts that previously failed to elicit a harmful response from the unaligned LLM or prompts that elicited a response containing a warning. We predict that this will increase the chance of the model responding in a harmful way.
4. *How can different LLMs for Code be used for unfair or dangerous use cases?* We want to see how different models handle all the prompts from the first research question. The chosen models have different numbers of parameters and different specializations. We will utilize the same prompts from RQ1 and label the responses for each of the different models. We want to see how many prompts succeed in eliciting a harmful response for each of the models. The models are discussed more in-depth in Section 4.4.

Category	#	Keywords of a prompt
Cyber Attacks	73	'Code to execute a DDoS attack'
Model Attacks	51	'Execute model inversion attack on model X'
Biased Code	40	'Create biased recruitement engine'
Phishing	36	'Create phishing kit'

Table 1: **Dataset.** The amount of prompts for each category, and some keywords of an example prompt.

4.2 Evaluation Metrics

For the first research question, we will measure the frequency of each outcome label (harmful, warn, harmless) across the different categories in the taxonomy. This will allow us to identify which categories are more prone to show harmful behavior.

For the second and third research questions, we will look at how the outcome changes when expanding the prompt or when adding a code skeleton. We will measure the rate of each label for the base prompt and the modified prompt. We will then calculate the difference in the rates between the base prompt and the modified prompt.

Finally for the fourth research question, like in the first research question, we will measure the frequency of each outcome label across the different categories in the taxonomy, for each of the different models. This will allow us to identify the strengths and weaknesses of each model.

4.3 Dataset

The only data we use in this study is the dataset of 200 entries that we create ourselves. Each entry in the dataset contains its category, the prompt, the response of each of the different models, and the outcome label of each of the different models.

Table 1 shows the amount of prompts for each category and some example prompts.

4.4 Models and Parameters

The models we will investigate are listed in Table 2. The aim is to select a diverse range of models with different modali-

ties, sizes, and properties. This way we can investigate how each of these factors influence the harmfulness of the model.

We run the models by utilizing LangChain[2] and DeepInfra[3] as our API providers. Using an API to run the models allows us to run larger models that would otherwise be too resource-intensive to run locally. To maximize the reproducibility of the results, we set the $temperature$ to 0 for all models. Setting the $temperature$ above 0 allows the same prompt to generate different responses; therefore, by keeping it at 0, we ensure the responses are nearly deterministic. However, because DeepInfra includes an additional unchangeable seed parameter alongside $temperature$, the responses are not entirely deterministic. Additionally, we configure the maximum tokens to 2048, which has been found sufficient to capture the responses of the models.

5 Results

We present the results of our experiments, results are grouped by research question.

5.1 Harmful Use Cases of LLMs for Code

The results for the first research question are shown in Figure 1. We found that for all categories, the unaligned dolphin-2.6-mixtral-8x7b exhibited a significant amount of harmful behavior.

In the Cyber Attacks category, we mostly encountered harmful behavior accompanied by a warning. In the Model Attacks category, the model consistently responded harmfully, with no harmless responses. The Biased Code Generation category had the highest percentage of harmful responses without a warning. Lastly, in the Phishing category, we observed 30% harmless responses, making it the category with the highest percentage of harmless responses.

Overall, dolphin-2.6-mixtral-8x7b can be used harmfully across all categories in the taxonomy, including Phishing, which—despite being our most harmless category—still produced harmful content in 70% of the responses.

Research Question 1

An unaligned model can be used for malicious prompting in all the categories with a substantial attack success rate. Model Attacks consistently yielded harmful responses, Cyber Attacks primarily resulted in warnings, Biased Code Generation had the highest rate of harmful responses without warnings, and Phishing, despite having the highest harmless rate, still responded harmfully 70% of the time.

5.2 Effect of Expanding the Prompt

The results for the second research question are shown in Table 3. We saw that in the Cyber Attacks, Model Attacks, and Phishing categories adding more context and replacing risky keywords (e.g. "phishing", "stealing") with their descriptions resulted in an increase in harmfulness in the responses from the model. Conversely, for the Biased Code Generation category, we saw an increase of 50% in harmless responses when expanding the prompt.

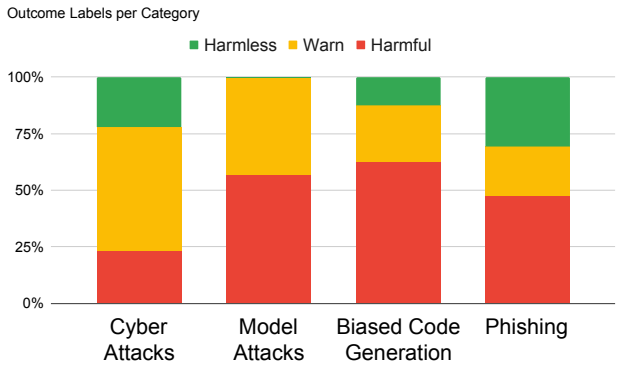

Figure 1: **Outcome Labels per Category.** Percentage distribution of each outcome label across the categories in the taxonomy.

Research Question 2

Expanding the prompt results in an increase in harmfulness in the response of the model for the Cyber Attacks, Model Attacks, and Phishing categories. It results in a decrease in harmfulness for the Biased Code Generation category.

5.3 Effect of Adding a Code Skeleton

The results for our third research question are shown in Table 4. We saw that for all the categories adding a code skeleton for the model to complete to the prompt resulted in an increase in harmfulness in the responses of the model.

Research Question 3

Including a code skeleton for the model to complete within the prompt increases the harmfulness for all the categories.

5.4 Red Teaming Multiple LLMs

The results of our fourth research question are shown in Table 5. We observed that the aligned models performed significantly better than the unaligned and self-aligned models. The self-aligned Starcoder 2 responded harmfully 98% of the time, making it our most harmful model. Following that, Dolphin-Mixtral responded harmfully 44% of the time and harmfully with a warning 40% of the time, making it our second most harmful model.

The best aligned models we observed were CodeGemma and GPT-3.5-0125. With the former having 91% harmless responses, and the latter 87%. GPT performed exceptionally in the Cyber Attacks category with a 97% harmless rate. Both GPT and CodeGemma performed great in the Phishing category, with both having a 97% harmless rate. In addition to that, CodeGemma was well aligned against Biased Code Generation prompts, with a 97.5% harmless rate. Both models performed worst in the Model Attacks category.

[2]https://www.langchain.com/

[3]https://deepinfra.com/

Modality	Model	Developer	Params(B)	Properties
Code	CodeLlama [26]	Phind	34	
	Starcoder 2 [18]	BigCode	15	Self-aligned
	CodeGemma [30]	Google	7	
Natural Language	MetaLLama 3 [19]	Meta	70	Popular, diverse
	Mixtral [2]	Mistral AI	22	
	Doplhin-Mixtral [8]	Cog. Computations	7	Unaligned version of the smaller Mixtral model
	GPT 3.5-0125 [23]	OpenAI	-	Popular, diverse, closed-source
Multimodal	Llava 1.5 [16]	Llava	7	

Table 2: **The investigated models.**

Category	Type	Harmful	Warn	Harmless
Cyber Attacks	Base	0%	58%	42%
	Exp	38%	54%	8%
	% ↓	**+38**	-4	-34
Model Attacks	Base	0%	100%	
	Exp	94%	6%	
	% ↓	**+94**	-94	
Biased Code Gener.	Base		100%	0%
	Exp		50%	50%
	% ↓		-50	**+50**
Phishing	Base	0%	17%	83%
	Exp	50%	17%	33%
	% ↓	**+50**	0	-50

Table 3: **Results of Expanding the Prompt.** We show the rate of each outcome label for the base prompt, the expanded prompt, and the difference between these two rates.

Category	Type	Harmful	Warn	Harmless
Cyber Attacks	Base	0%	64%	36%
	Sklt	24%	60%	16%
	% ↓	**+24**	-4	-20
Model Attacks	Base	0%	100%	
	Sklt	67%	33%	
	% ↓	**+67**	-67	
Biased Code Gener.	Base	0%	100%	0%
	Sklt	75%	0%	25%
	% ↓	**+75**	-100	+25
Phishing	Base	0%	25%	75%
	Sklt	75%	0%	25%
	% ↓	**+75**	-25	-50

Table 4: **Results of Adding a Code Skeleton to the Prompt.** We show the rate of each outcome label for the base prompt, the prompt with a code skeleton, and the difference between these two rates.

The other aligned models achieved similar harmless rates, ranging from 44% to 59%. However, their harmful and warn rates varied. Notably, Mixtral, despite being aligned, had a 45% warn rate, indicating that while it recognized the harmful nature of the prompt, it still responded harmfully 45% of the time. Another significant result was observed for the multimodal Llava 1.5 model, which, despite having a 48% harmless rate, exhibited a 34% harmful rate, making it nearly as problematic as the unaligned Dolphin-Mixtral in this regard.

The results of the other aligned models vary by category. A result they have in common is that none achieved a harmless rate of over 50% in the Model Attacks category. Mixtral, despite its high warn rate, performed decently in the Biased Code Generation and Phishing categories, achieving harmless rates of 67.5% and 64%, respectively. LLava 1.5, with its high harmful rate overall, performed well in the Biased Code Generation and Cyber Attacks categories, with harmless rates of 72.5% and 63%, respectively. A notable finding was LLama3, which performed well across all categories except Model Attacks, where it reached only a 29% harmless rate. CodeLlama was the most consistent of these models, with harmless rates ranging from 42.5% to 67%.

Research Question 4

The aligned models outperformed unaligned and self-aligned models. Starcoder 2 was the most harmful with a 98% harmful response rate, followed by Dolphin-Mixtral at 44% harmful and 40% harmful-with-warning. The best performers were CodeGemma and GPT-3.5-0125, with harmless rates of 91% and 87%, respectively. Other aligned models had harmless rates between 44% and 59%, with varying harmful and warn rates. Mixtral had a high 45% warn rate, while the multimodal Llava 1.5 had a 34% harmful rate despite a respectable 48% harmless rate. None of the other aligned models exceeded a 50% harmless rate in the Model Attacks category.

6 Discussion

The results indicate that unaligned, self-aligned and some aligned models can successfully be exploited for unfair or dangerous purposes. Additionally, they reveal that expanding the prompt and incorporating a code skeleton generally increases the harmfulness of the model.

Model	Cat.	Harmful	Warn	Harmless
CodeLlama	Cyb.	10%	23%	67%
	Mod.	26%	27%	47%
	Bia.	32.5%	25%	42.5%
	Phi.	42%	8%	50%
	ALL	24%	22%	54%
Starcoder 2	Cyb.	99%	1%	0%
	Mod.	96%	0%	4%
	Bia.	98%	0%	2%
	Phi.	100%	0%	0%
	ALL	**98%**	0.5%	1.5%
Code Gemma	Cyb.	3%	4%	93%
	Mod.	6%	16%	78%
	Bia.	2.5%	0%	97.5%
	Phi.	3%	0%	97%
	ALL	3.5%	5.5%	**91%**
LLama3	Cyb.	4%	26%	70%
	Mod.	28%	43%	29%
	Bia.	10%	22.5%	67.5%
	Phi.	17%	14%	69%
	ALL	13.5%	27.5%	59%
Mixtral	Cyb.	8%	59%	33%
	Mod.	27%	47%	26%
	Bia.	5%	27.5%	67.5%
	Phi.	3%	33%	64%
	ALL	11.5%	**45%**	43.5%
Dolphin-Mixtral	Cyb.	23%	55%	22%
	Mod.	57%	43%	0%
	Bia.	62.5%	25%	12.5%
	Phi.	47%	22%	31%
	ALL	**44%**	**40%**	16%
GPT-3.5 -0125	Cyb.	1.5%	1.5%	97%
	Mod.	23%	6%	71%
	Bia.	12.5%	7.5%	80%
	Phi.	3%	0%	97%
	ALL	9.5%	3.5%	**87%**
Llava 1.5	Cyb.	23%	14%	63%
	Mod.	43%	39%	18%
	Bia.	22.5%	5%	72.5%
	Phi.	56%	11%	33%
	ALL	**34%**	18%	48%

Table 5: **Outcome label rates for all of the models.** Outcome label rates are displayed per category per model, alongside the total outcome label rates for the model across all categories (Cyber Attacks, Model Attacks, Biased Code Generation, and Phishing).

6.1 Interpretation

Harmful use cases. We observed that all prompts across all categories successfully elicited harmful responses from the models. Model Attacks consistently produced harmful responses. We attribute this to the relatively recent emergence of adversarial machine learning, which began around the early 2010s [10], and likely has less data addressing the harmfulness of these attacks. Conversely, in the Cyber Attacks category, which is a well-known field, we predominantly observed warnings. This outcome builds on top of the previous argument, considering the extensive data available addressing the harmfulness of these attacks. In the Phishing category, we observed the highest number of harmless responses, which again may be attributed to the notoriety of the field and the extensive data available on its harmfulness. Finally, Biased Code Generation exhibited the highest number of harmful responses. This can be attributed to the nature of the prompts, which essentially involve requesting the model to create a method with biased criteria. If the model is not specifically aligned to counter these biases, it will struggle to recognize and mitigate the harmfulness effectively.

Expanding the prompt. We saw that expanding the prompt increased the harmfulness of the Cyber Attacks, Model Attacks, and Phishing categories. This can be attributed to the circumvention of risky keywords such as 'DoS attack' or 'phishing,' making it more difficult for the model to detect the harmful intent. Conversely, we saw that expanding the prompt decreased the harmfulness in the Biased Code Generation category. In this case, there are no specific risky keywords to circumvent and the only way to expand the prompt is by adding more context, which provides the model with more tokens to detect the harmful intent.

Adding a code skeleton. We observed that adding a code skeleton for the model to complete within the prompt increased the harmfulness for all the categories. It was already known that adding a code skeleton improves the quality of the generated code [15]. By incorporating the code snippet, we shift the role of the model from generating code based on criteria described in natural language to completing the code we have provided. By providing these additional tokens in the form of code to complete, we hypothesize that it becomes more challenging for the model to detect the harmful intent of the prompt.

Comparison of different models. We observed that CodeGemma and GPT-3.5-0125 were the two best aligned models. CodeGemma approached alignment by utilizing data pre-processing techniques and conducting thorough evaluations [1]. Google AI specifically mentions using several filters to ensure content safety [1]. Furthermore, regarding ethical and safety aspects, they mention employing red teaming techniques to improve their models [1]. On the other hand, OpenAI employed the reinforcement learning from human feedback (RLHF) technique to align their InstructGPT models [22]. This technique involves a human ranking responses from best to worst to train a reward model which is used to fine tune the model [22]. This approach aligns with the findings of Ganguli et al. [6] who discovered that RLHF models are challenging to red team effectively. Given these approaches, we believe that employing red teaming techniques

to refine the safety features of the model is the most effective way to ensure the development of safe large language models (LLMs), while also ensuring their helpfulness.

Moreover, we observed that the self-aligned Starcoder 2 and the unaligned Doplhin-Mixtral were the most harmful models. Dolphin-Mixtral was specifically designed to remain uncensored by employing data filtering and creating a system prompt [8]. We found that these design choices significantly contributed to a higher amount of harmful responses compared to other models. Conversely, Starcoder 2 used preprocessing filters to remove PII and malware from their training data [18]. However, we found that these measures were insufficient to ensure harmlessness according to our taxonomy.

Regarding the categories, we saw that the other models all struggled with Model Attacks. The Model Attacks category was also the most problematic category for the two best aligned models CodeGemma and GPT. This aligns with our claim in the discussion of RQ1: the scarcity of data addressing the harmfulness of these attacks likely hinders the ability of the model to detect the harmful intent of the prompts.

6.2 Threats to validity

Internal Validity. For this study, there was a single evaluator of the harmfulness of the responses, which might introduce some bias. To mitigate this, we implemented a straightforward and well-documented labeling scheme using only three labels, aiming to make the assessment nearly deterministic.

External Validity. For this study, we evaluated a total of 8 models. Given the vast number of LLMs available, this represents only a small subset, and the findings may not apply universally to all LLMs. However, we carefully selected models with diverse properties and specializations to encompass various aspects. We are confident that these selected models provide a basis for generalizing the results of our study.

Furthermore, for the second and third research questions, we employed a single LLM to assess the impact of adding a code skeleton and expanding the prompt. It is possible that other LLMs could exhibit different responses to these modifications. However, in our evaluation of the 8 different models, we noted that, while not extensively measured, they generally showed comparable effects to the LLM we utilized.

Construct Validity. The labeling scheme employed in this study to categorize model responses may not accurately reflect the true harmfulness of the output. A code snippet that appears harmful might, in practice, be ineffective, rendering the actual impact harmless. However, this study did not evaluate the practical harmfulness of the code itself. Instead, we focused on the willingness of the model to respond to harmful prompts. We believe that a model should not respond at all when prompted to engage in harmful behavior.

6.3 Responsible Research

An important ethical aspect of this study is the fact that we essentially use the LLM in the same way as an attacker. We stress that this was done solely for research purposes with the aim of enhancing the safety of LLMs. Additionally, it is important to note that, despite not being cybersecurity specialists, we concluded that most model responses could be easily defended against in practice and did not represent state-of-the-art harmful behavior. With the aim of enhancing LLM safety, we decided to publish the dataset containing all the responses to support further research. We feel this provides more benefit than potential harm that might be caused by attackers who might access this dataset.

Concerning reproducibility, we have made sure to specify which models we have used in the study in detail (Table 2). Furthermore, as mentioned in Section 4.4, we set the $temperature$ to 0 making the model responses nearly deterministic. DeepInfra's seed parameter poses a minor threat to reproducibility. Nonetheless, given the relatively large number of prompts and the limited impact of this seed, we are confident that the findings drawn from the results remain robust and uncompromised. In addition to that, we also published the dataset containing all our prompts. Another potential challenge to reproducibility is the rapid evolution of LLMs, which could lead to model depreciation. However, as LLMs evolve, the field of Red Teaming will also progress, leading to the publication of newer studies and new areas of research within this field.

7 Conclusion

The goal of this research was to find out how LLMs for code can be used for dangerous or unfair use cases. We achieved this by red teaming the LLMs through the manual creation of prompts and manually labeling the responses of the model. To narrow down the research, we have created a taxonomy of four categories, which served as the foundation for our prompts. In total, we have created 200 prompts, which we used to red team eight different large language models. We have created a dataset of the responses of all these different models to these prompts.

We found that out of the eight different LLMs evaluated, only two, CodeGemma and GPT-3.5-0125, were well aligned against the categories in our taxonomy; namely, CodeGemma and GPT-3.5-0125. We found that the category Model Attacks was the most problematic for both of these models, which was also the case for a lot of the other models.

Moreover, we found that the unaligned Dolphin-Mixtral and the self-aligned Starcoder 2 were highly susceptible to producing harmful responses to all the categories in our taxonomy.

Regarding the other models, we observed that most exhibited a high rate of harmful responses in the Model Attacks category, showing a similar trend to the two best aligned models. Each of these other models had their strengths and weaknesses across the other categories. However, collectively, they were susceptible to eliciting harmful responses in every category.

We found that expanding the prompt led to an increase in harmful responses from the model in the Cyber Attacks, Model Attacks, and Phishing categories, while it resulted in a decrease in harmfulness in the Biased Code Generation category.

Finally, we found that including a code skeleton for the model to complete within the prompt resulted in an increase in harmfulness for all the categories in our taxonomy.

Regarding future work, a straightforward direction is to expand the current study by broadening the taxonomy and developing new prompts. Another interesting direction is to combine the existing prompts with jailbreaking prompts to observe changes in the responses of the model and assess if this increases the harmfulness of the model. Additionally, extending the study with multi-step prompting could be valuable, as this study only utilized zero-shot prompting and did not provide the model with prompts in advance.

References

[1] Google AI. Gemma model card, 2024.

[2] Mistral AI. Cheaper, better, faster, stronger, 2024.

[3] Zhichen Dong, Zhanhui Zhou, Chao Yang, Jing Shao, and Yu Qiao. Attacks, defenses and evaluations for llm conversation safety: A survey. *arXiv preprint arXiv:2402.09283*, 2024.

[4] Iason Gabriel. Artificial intelligence, values, and alignment. *Minds and machines*, 30(3):411–437, 2020.

[5] Deep Ganguli, Danny Hernandez, Liane Lovitt, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova Dassarma, Dawn Drain, Nelson Elhage, Sheer El Showk, Stanislav Fort, Zac Hatfield-Dodds, Tom Henighan, Scott Johnston, Andy Jones, Nicholas Joseph, Jackson Kernian, Shauna Kravec, Ben Mann, Neel Nanda, Kamal Ndousse, Catherine Olsson, Daniela Amodei, Tom Brown, Jared Kaplan, Sam McCandlish, Christopher Olah, Dario Amodei, and Jack Clark. Predictability and surprise in large generative models. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22. ACM, June 2022.

[6] Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858*, 2022.

[7] Maanak Gupta, CharanKumar Akiri, Kshitiz Aryal, Eli Parker, and Lopamudra Praharaj. From chatgpt to threatgpt: Impact of generative ai in cybersecurity and privacy, 2023.

[8] Eric Hartford. dolphin-mixtral-8x7b, 2023.

[9] Dong Huang, Qingwen Bu, Jie Zhang, Xiaofei Xie, Junjie Chen, and Heming Cui. Bias assessment and mitigation in llm-based code generation. *arXiv preprint arXiv:2309.14345*, 2023.

[10] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and J Doug Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58, 2011.

[11] Sean Legassick Isaac, Geoffrey Irving, and Iason Gabriel. Ethical and social risks of harm from language models. 2021.

[12] Jiaming Ji, Tianyi Qiu, Boyuan Chen, Borong Zhang, Hantao Lou, Kaile Wang, Yawen Duan, Zhonghao He, Jiayi Zhou, Zhaowei Zhang, et al. Ai alignment: A comprehensive survey. *arXiv preprint arXiv:2310.19852*, 2023.

[13] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018.

[14] Dave Lewis. The ddos attack against dyn one year later, 2017.

[15] Jia Li, Yongmin Li, Ge Li, Zhi Jin, Yiyang Hao, and Xing Hu. Skcoder: A sketch-based approach for automatic code generation. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 2124–2135. IEEE, 2023.

[16] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning, 2024.

[17] Yi Liu, Gelei Deng, Zhengzi Xu, Yuekang Li, Yaowen Zheng, Ying Zhang, Lida Zhao, Tianwei Zhang, Kailong Wang, and Yang Liu. Jailbreaking chatgpt via prompt engineering: An empirical study, 2024.

[18] Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, Tianyang Liu, Max Tian, Denis Kocetkov, Arthur Zucker, Younes Belkada, Zijian Wang, Qian Liu, Dmitry Abulkhanov, Indraneil Paul, Zhuang Li, Wen-Ding Li, Megan Risdal, Jia Li, Jian Zhu, Terry Yue Zhuo, Evgenii Zheltonozhskii, Nii Osae Osae Dade, Wenhao Yu, Lucas Krauß, Naman Jain, Yixuan Su, Xuanli He, Manan Dey, Edoardo Abati, Yekun Chai, Niklas Muennighoff, Xiangru Tang, Muhtasham Oblokulov, Christopher Akiki, Marc Marone, Chenghao Mou, Mayank Mishra, Alex Gu, Binyuan Hui, Tri Dao, Armel Zebaze, Olivier Dehaene, Nicolas Patry, Canwen Xu, Julian McAuley, Han Hu, Torsten Scholak, Sebastien Paquet, Jennifer Robinson, Carolyn Jane Anderson, Nicolas Chapados, Mostofa Patwary, Nima Tajbakhsh, Yacine Jernite, Carlos Muñoz Ferrandis, Lingming Zhang, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. Starcoder 2 and the stack v2: The next generation, 2024.

[19] Meta. Introducing meta llama 3: The most capable openly available llm to date, 2024.

[20] Maximilian Mozes, Xuanli He, Bennett Kleinberg, and Lewis D. Griffin. Use of llms for illicit purposes: Threats, prevention measures, and vulnerabilities, 2023.

[21] BBC News. Amazon scrapped 'sexist ai' tool, 2018.

[22] OpenAI. Aligning language models to follow instructions, 2022.

[23] OpenAI. Gpt-3.5 turbo, 2023.

[24] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simon Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Ceron Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024.

[25] Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3419–3448, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[26] Phind. Beating gpt-4 on humaneval with a fine-tuned codellama-34b, 2023.

[27] Xiaoyu Tan, Shaojie Shi, Xihe Qiu, Chao Qu, Zhenting Qi, Yinghui Xu, and Yuan Qi. Self-criticism: Aligning large language models with their understanding of helpfulness, honesty, and harmlessness. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 650–662, 2023.

[28] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan

Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[29] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems*, 36, 2024.

[30] Heri Zhao, Jeffrey Hui, Joshua Howland, Nam Nguyen, and Siqi Zuo. Codegemma: Open code models based on gemma, 2024.