BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

DEVELOPMENT OF AN LLM RED-TEAMING TOOLKIT

VÝVOJ TOOLKITU PRO RED-TEAMING VELKÝCH JAZYKOVÝCH MODELŮ (LLM)

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **ADAM VESELÝ**
AUTOR PRÁCE

SUPERVISOR **Ing. JAKUB REŠ**
VEDOUCÍ PRÁCE

BRNO 2026

Abstract

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v anglickém jazyce.

Abstrakt

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v českém (slovenském) jazyce.

Keywords

Sem budou zapsána jednotlivá klíčová slova v anglickém jazyce, oddělená čárkami.

Klíčová slova

Sem budou zapsána jednotlivá klíčová slova v českém (slovenském) jazyce, oddělená čárkami.

Reference

VESELÝ, Adam. *Development of an LLM Red-Teaming Toolkit.* Brno, 2026. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Jakub Reš,

Development of an LLM Red-Teaming Toolkit

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Mr. Ing. Jakub Reš. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis. I have used ChatGPT to correct spelling and other language mistakes. I used Grok when working on the software.

.....
Adam Veselý
December 5, 2025

Acknowledgements

I would like to thank my supervisor Ing. Jakub Reš for his guidance and support throughout the development of this thesis. I also appreciate the assistance provided by my colleagues and friends who contributed their insights and expertise.

Contents

1 Introduction **3**

2 Background and Related Work **4**
2.1 Large Language Models 4
2.2 LLM Safety Risks 5
2.3 Red Teaming: Definitions and Methodology 6
2.4 Attack Taxonomy 7
2.5 Evaluation Metrics and Benchmarks 8
2.6 Open-source Red-Teaming Tools and Frameworks 9
2.7 Synthesis and Research Gap 10
2.8 Conclusion of the Chapter 11

3 Design **14**

4 Implementation **15**

5 Evaluation **16**

6 Conclusion **17**

Bibliography **18**

List of Figures

Chapter 1

Introduction

The rapid advancement and widespread deployment of large language models (LLMs) have transformed natural-language interaction with computers. These models now power chatbots, code assistants, translation systems, and creative tools used daily by millions of users.

However, their remarkable capabilities come with significant safety and ethical risks. LLMs can generate harmful, biased, misleading, or illegal content when subjected to carefully crafted adversarial inputs, a practice commonly known as *jailbreaking* [27, 30].

Real-world incidents such as ChatGPT being tricked into providing bomb-making instructions [5], or Gemini's image-generation controversy [24], have demonstrated that even flagship commercial models remain vulnerable to adversarial prompting. In response, red teaming, a cybersecurity technique involving simulated attacks to expose vulnerabilities, has been adopted by leading AI organisations (OpenAI, Anthropic, Google DeepMind, etc.) as a core component of LLM safety evaluation [15].

With the adoption of the EU AI Act in 2024, systematic risk assessment including red teaming will become a legal requirement for high-risk AI systems deployed in the European Union from 2026 onward [17]. Consequently, efficient, reproducible, and extensible red-teaming tools are no longer a luxury but an essential part of responsible AI development.

Despite significant progress, most existing open-source red-teaming frameworks suffer from limited modularity, poor support for modern systems, query cost and computational requirements, etc., that hinder adoption by smaller research teams and individual developers [19, 21, 1, 14]. This creates a clear need for a new, lightweight, developer-friendly red-teaming toolkit that lowers the barrier to LLM systematic safety testing.

The main goal of this bachelor's thesis is therefore the design, implementation, and evaluation of a modular open-source red-teaming toolkit for large language models that addresses the identified shortcomings of current solutions.

The remainder of this thesis is structured as follows: Chapter 2 surveys the current state of research in LLM red teaming and reviews existing tools. Chapter 3 presents the proposed system architecture. Chapter 4 details the implementation of the red-teaming toolkit. Chapter 5 evaluates the toolkit on selected LLMs. Finally, Chapter 6 summarises the results and outlines directions for future work.

Chapter 2

Background and Related Work

This chapter provides the technical and empirical background necessary to understand red-teaming of large language models (LLMs).

We first cover the architecture and training paradigm of modern LLMs and the alignment methods applied to improve their safety.

Next, we survey the primary categories of safety risks and known vulnerability classes.

We then turn to red-teaming methodology as adapted to LLMs — definitions, goals, and a taxonomy of test methods.

Subsequently, we describe common attack techniques (single-turn, multi-turn, prompt injection, universal triggers, trojans/backdoors, etc.)

Finally, we review evaluation metrics and benchmarks used to measure safety failures, and survey existing open-source tools and frameworks.

The chapter concludes with a synthesis of gaps in current tooling and evaluation practice, motivating the design of the toolkit proposed in this thesis.

2.1 Large Language Models

Large language models (LLMs) are sequence-to-sequence (encoder-decoder) or autoregressive (decoder) models based on the Transformer architecture [26].

They are trained on massive, internet-scale corpora using a self-supervised objective such as next-token prediction. After pretraining, many high-performance models undergo instruction fine-tuning and additional alignment to become more useful and safer in downstream use [29].

By 2025, publicly available and research-grade LLMs often exceed tens to hundreds of billions of parameters [11], and many incorporate advanced architectural and inference-optimization techniques such as mixture-of-experts layers [6], retrieval-augmented generation [2], or quantization-aware training for efficient deployment [4].

Because of the scale of their training data and capacity, LLMs encode a wide variety of linguistic patterns, factual knowledge, biases, and behavioural priors.

This scale gives them impressive generative and reasoning capabilities, but also makes them susceptible to emergent, unintended behaviors that are not trivially predictable — including safety failures, policy evasion, and social-engineering style manipulation [27].

2.1.1 Alignment and Safety Interventions

To mitigate risks from raw pre-trained models, developers commonly apply a pipeline of alignment techniques.

First, supervised fine-tuning (SFT) on curated instruction-following datasets helps the model adhere to desired task formats [29, 25].

Subsequently, reinforcement learning from human feedback (RLHF) is often used: human annotators rate model outputs, a reward model is trained on those ratings, and a proximal policy optimization method (e.g., PPO) updates the model to align it to human preferences [16, 22].

Variants such as Direct Preference Optimization (DPO) or RLAIF (reinforcement learning from AI feedback) aim to improve efficiency and scalability of alignment without sacrificing safety or quality [20, 8].

Despite alignment efforts, evidence demonstrates that safety training is not foolproof. Behavioural failures persist, especially under adversarial or adversary-chosen inputs: prompt-based attacks and jailbreaks remain a major vulnerability class (cf. [27, 30, 18]).

The fact that surface-level filtering and instruction tuning can be bypassed indicates that LLMs continue to rely on shallow heuristics rather than robust semantic safety guarantees.

2.2 LLM Safety Risks

LLM misuse and unintended outputs pose a broad array of risks.

The following taxonomy, commonly adopted in recent red-teaming efforts, captures the primary threat vectors:

- **Malicious usage:** generation of instructions or actionable content facilitating wrongdoing (e.g., bomb-making, illicit substances, hacking).
- **Harassment, hate, and discrimination:** outputs that demean or promote bias against individuals or protected groups.
- **Misinformation and hallucinations:** confidently stated false information, invented facts, or fabricated references.
- **Privacy violations:** unintended leakage of sensitive information, either from memorized training data or through malicious prompts.
- **Self-harm / dangerous content:** generation of content promoting self-harm, suicide, or exploitation (especially of minors).
- **Emergent misuse and behavioural failures:** including instruction-following failures, refusal evasion (the model ignoring safety instructions), social-engineering exploitation, or covert manipulation over multiple turns.

These categories correspond to those used in large-scale safety benchmarks and red-team evaluation suites.

For example, frameworks such as *HarmBench* operationalize a broad set of harm categories and provide standard test sets and evaluation protocols across LLMs and red-teaming methods [12].

2.3 Red Teaming: Definitions and Methodology

Originally developed in cybersecurity and military contexts, *red teaming* refers to the practice of simulating adversaries to probe system vulnerabilities before deployment.

In the context of LLMs, red teaming denotes systematic probing of model behaviour via adversarial prompts or inputs designed to circumvent safety and alignment measures, with the goal of discovering previously unknown failure modes, measuring their prevalence, and informing robust defenses.

Red-team campaigns in LLM settings typically have several objectives:

- **Discovery:** reveal novel, previously undocumented failure modes (e.g., new jailbreak styles, multi-turn attack vectors, prompt injection in application contexts).
- **Quantification:** estimate how often a model fails under adversarial conditions, enabling comparison across models and defense strategies.
- **Reproducibility:** produce repeatable test cases and evaluation pipelines so that safety regressions can be detected over time.
- **Defense hardening:** feed findings back into model tuning, safety filters, or deployment guardrails to reduce vulnerability.

Modern red-teaming approaches in LLMs span from fully manual human-driven pentesting to automated pipelines combining generation, execution, and evaluation.

2.3.1 Manual, Rule-Based and Automated Red Teaming

Manual red teaming: human experts write adversarial prompts, simulate realistic misuse scenarios, and attempt to elicit harmful or policy-violating responses.

This approach benefits from human creativity and insight into real-world misuse, including social-engineering, context-aware manipulation, or subtle misuse cases.

However, it is expensive, time-consuming, and often non-reproducible.

Rule-based / template-based testing: uses curated prompt templates (e.g., standard jailbreaks, role-play prompts, obfuscation methods, encoded instructions) or transformation rules to generate adversarial inputs systematically.

This method is reproducible and simple, but often limited to discovering known failure classes, and does not generalize to novel attacks.

Automated red teaming: algorithmic or model-in-the-loop generation of adversarial prompts — e.g., via search, optimization, or by using another LLM as attacker.

Notable work in this space includes *Tree of Attacks (TAP)* which automatically generates jailbreak prompts against black-box LLMs by iteratively refining candidates and pruning unlikely ones [13].

Automated methods scale well, explore large prompt spaces, and can discover novel failures, but also risk producing unrealistic or trivial prompts — and often depend on the quality of the attacker/judge model and search strategy.

Automated red-teaming has recently been re-envisioned as a sequential, multi-turn process rather than isolated single-turn attempts.

For example, recent work models red-teaming as a Markov Decision Process (MDP), using hierarchical reinforcement learning to optimize long-horizon attacks over entire dialogue trajectories [1].

2.4 Attack Taxonomy

Here we outline the main classes of attacks exploited in LLM red-teaming, including prompt-based attacks, injection attacks, multi-turn manipulations, universal triggers, and trojan/backdoor style vulnerabilities.

2.4.1 Single-turn Jailbreaking

Single-turn jailbreaks are adversarial prompts supplied in one-shot (single user message) that instruct the model to ignore its safety filters or system instructions.

Common techniques include role-play ("Pretend you are ..."), direct overrides ("Ignore previous instructions ..."), encoding or cipher-based obfuscation (to hide disallowed instructions), and other prompt-engineering tricks.

Despite alignment training, many models remain vulnerable to such simple attacks [27, 30].

2.4.2 Prompt Injection

Prompt injection refers to attacks where user-supplied input (or external content, in the case of integrated applications) is directly interpreted by the LLM as instructions, potentially overriding or modifying hidden system prompts.

This is a major concern for real-world applications embedding LLMs (chatbots, agents, document processors, pipelines) because attackers can inject malicious instructions that the model treats as legitimate.

Empirical studies have demonstrated that many deployed LLM-based applications are vulnerable to prompt injection — for example via the *HouYi* attack, which compromised dozens of applications in a black-box setting [10].

Other work has shown that even automated, universal prompt injection attacks remain effective under defensive measures [9].

Prompt injection remains among the most significant security threats for LLM-based systems, as acknowledged by security guidance frameworks and cheat sheets (e.g., from OWASP) tailored to LLM applications [23].

2.4.3 Multi-turn and Conversational Attacks

Rather than achieve a jailbreak in a single prompt, adversaries may perform a series of manipulative steps — gradually steering the conversation, exploiting context persistence, memory, and the model's inability to consistently refuse undesirable requests.

Multi-turn attacks may involve context poisoning, social-engineering style dialogue, bait-and-switch tactics, or incremental obfuscation.

This vector is increasingly recognized as one of the most dangerous and under-evaluated.

Recent research recasts automated red-teaming as a multi-turn optimization problem — which better reflects realistic adversarial scenarios — and shows that RL-based red-teaming significantly outperforms single-turn methods in eliciting harmful content [1].

2.4.4 Universal / Transferable Triggers and Trojan-style Attacks

Universal triggers or transferable adversarial prompts aim to find short token sequences (prefixes, suffixes, or embedded instructions) that reliably trigger undesired model behaviour across different inputs and even across different models.

This makes the attack highly reusable and dangerous.

One example is the black-box Trojan prompt attack framework *TrojLLM*, which demonstrates that universal stealthy triggers can be discovered for widely-used LLM APIs such as GPT-3.5 and GPT-4, enabling malicious manipulation across diverse prompt inputs [28].

Such attacks bypass input sanitization heuristics because the trigger is embedded in seemingly benign inputs.

2.4.5 Prompt-based Evasion and Adversarial Perturbations

Beyond explicit instructions or injected content, adversaries may attempt adversarial prompting via subtle token-level perturbations, insertion, deletion, or encoding — i.e., minimal but adversarial changes that remain semantically similar to benign prompts but cause the model to deviate.

Defenses based on sanitization or heuristic filtering often fail to catch these, especially in the presence of context sensitivity or long prompts.

Certified-safety approaches have been proposed to mitigate this class: e.g., erase-and-check, which systematically removes tokens and reruns safety filters to detect adversarial prompt manipulations, providing (under assumptions) a safety guarantee against insertions, suffixes, or adversarial infusions up to a bounded size [7].

While promising, such methods are computationally expensive and may degrade user experience or model utility.

Because these perturbation-based attacks operate at the token level and can be obfuscated, they represent a difficult-to-detect threat, especially when combined with other attack modalities (multi-turn, universal triggers, trojans).

2.5 Evaluation Metrics and Benchmarks

Robust and systematic evaluation is critical for red-teaming.

The following metrics and benchmarks are widely adopted in research and practice.

2.5.1 Quantitative Metrics

- **Attack Success Rate (ASR):** the proportion of adversarial attempts (prompts) that succeed in eliciting harmful or policy-violating outputs. ASR is the most common metric, but it depends heavily on the definition of "harmful" and on the quality of the judge (classifier, LLM-judge, human).
- **Refusal Bypass Rate:** a variant of ASR that measures how often safe-mode refusals or safety filters are bypassed — i.e., the model issues a disallowed output rather than refusing the request.

- **Judge Reliability Metrics:** when using automated judges (regex, classifier, LLM-based), it is crucial to measure false positives / false negatives, calibration error, and inter-annotator agreement (if doing human calibration). Poor judges can substantially distort ASR and other metrics.
- **Robustness Metrics:** measure how stable attacks (or defenses) are under variations — e.g., paraphrase of prompt, different model temperature, different seeds, context shuffling, or small perturbations.
- **Transferability / Generality:** measure how well attacks discovered on one model or configuration transfer to other models, prompts, or deployments.
- **Conversational / Trajectory Metrics:** in multi-turn attack scenarios, metrics may capture success over a dialogue trajectory (e.g., whether harmful content emerges at any point), time-to-failure, or complexity (number of turns needed).

2.5.2 Benchmarks and Standardized Suites

To enable systematic comparison across red-teaming methods and LLMs, standard benchmarks have recently been developed.

A prominent example is *HarmBench*, which provides an open-source evaluation framework, a large pool of red-teaming methods (18 methods) and a diverse set of target models and defenses (33 LLMs / defense settings) [12].

HarmBench enables reproducible large-scale red-team evaluation and supports both attack and defense-side experiments.

Its release marks a milestone toward standardizing LLM safety evaluation pipelines.

Despite this progress, many prior works still rely on ad-hoc prompt sets, non-public prompt libraries, or private internal red-team pipelines.

In addition, while single-turn attack benchmarks are relatively common, multi-turn / conversational red-teaming remains underrepresented in publicly available benchmark suites.

2.6 Open-source Red-Teaming Tools and Frameworks

Several open-source frameworks and research prototypes aim to support systematic red-teaming.

Their design choices, strengths, and limitations vary.

2.6.1 Frameworks for rule-based or template-based red-teaming

Frameworks such as *PyRIT* (discussed in the introduction) provide a modular architecture for plugging in model backends, defining prompt templates, logging results, and running interactive or batch red-team sessions.

These tools are often lightweight and well-suited for smaller projects or smaller compute budgets, but tend to lack advanced automated search or generation capabilities.

Another such toolkit is *garak*, promoted by a major vendor and widely referenced in the industry.

garak provides "probes, generators, and detectors": probes manage attack logic; generators abstract target models (LLMs, dialog systems, or any component taking text and

returning text); detectors assess whether output indicates a successful attack; and the framework compiles results into human-readable reports (HTML + JSON).

This design allows red-teaming across a variety of model backends and output modalities, but — as with simpler frameworks — may not support automated, learning-based attack generation or multi-turn conversational scenarios out of the box [3].

2.6.2 Automated Red-Teaming Frameworks

More advanced frameworks attempt to integrate attack generation, execution, and evaluation into a unified pipeline.

Notable recent work includes *MAD-MAX*, a modular adversarial red-teaming framework designed to allow multiple attack strategies (template-based, search-based, LLM-driven) in a pluggable architecture [21].

Its modular nature makes it flexible and extensible, but in practice integrating it with diverse model runtimes (local LLaMA variants, API-based models, multi-modal models) and scaling up to large-scale red-teaming remains challenging due to compute cost, model compatibility, and evaluation infrastructure requirements.

The recent work on fully automated, trajectory-based red teaming — recasting red teaming as a sequential decision-making process over entire dialogues — pushes the frontier further.

For example, the automated red-teaming approach by Belaire et al. (2025) formalizes multi-turn red teaming as a Markov Decision Process (MDP), enabling attack policies to optimize over entire conversation trajectories rather than single messages [1].

This methodology captures realistic adversarial behaviour and reveals vulnerabilities that single-turn or template-based attacks may miss.

2.6.3 Defense-oriented and Certified Safety Approaches

In response to the growing sophistication of attacks, some work focuses on hardening LLMs against adversarial prompting.

For example, the framework "erase-and-check" provides a method for certifying safety against adversarial prompt modifications (suffix insertion, insertion at arbitrary positions, adversarial infusions) under bounded adversarial size.

This method recomputes safety classification after systematically removing tokens from the prompt and offers provable safety guarantees under certain assumptions [7].

While promising, such approaches are often computationally expensive and may impair user experience or model usability.

Moreover, the existence of Trojan prompt attacks (as demonstrated by frameworks like *TrojLLM*) indicates that even seemingly benign prompts may embed stealthy triggers that cause harmful behaviour — which complicates defense strategies, necessitating robust input sanitization, runtime monitoring, and possibly dynamic prompt sanitization or adversarial-resistant prompt encoding [28].

2.7 Synthesis and Research Gap

The literature and tools surveyed above show that — while the research community has developed a rich taxonomy of harms, attack strategies, and evaluation metrics — substantial

gaps remain that limit the effectiveness, accessibility, and reliability of red-teaming for large language models.

In particular:

- **Lack of lightweight, modular, and extensible tooling:** existing frameworks either focus on small scale (template-based, rule-based, lightweight) or on large-scale automated red-teaming — but rarely both. There is a lack of toolkits that are accessible to small research teams or individual developers yet support modern LLM backends, extensible attack/evaluator plugins, and reproducible logging/benchmarks.
- **Judge reliability and evaluation consistency:** many red-teaming efforts rely on heuristic or automated judges (regex filters, simple classifiers), which often lack calibration, robustness, or reproducibility. As a result, reported Attack Success Rates (ASR) may misrepresent true safety risk. While certified-safety methods (e.g., erase-and-check) provide stronger guarantees, they are computationally expensive and may be impractical for everyday red-teaming.
- **Underrepresentation of multi-turn and real-world attack vectors:** most prior work and benchmarks focus on single-shot prompts; multi-turn conversational attacks — which more accurately model real-world adversaries — remain under-evaluated. Recently proposed trajectory-based red-teaming frameworks help, but are not yet part of standard open-source toolkits.
- **Difficulty integrating across diverse model backends and deployment contexts:** LLMs are deployed in variable settings (local open-source models, API-based proprietary models, multi-modal agents, applications with external tools). Existing tools often lack abstractions or adapters covering this diversity.
- **Limited transparency and reproducibility of prompt libraries and experiment artefacts:** many studies do not release their full prompt sets, seeds, or evaluation logs, making independent replication or longitudinal safety regression testing difficult.

These gaps motivate the design goals for the toolkit developed in this thesis: namely, modularity and pluggability (attack generators, model adapters, evaluators), reproducible experiment manifests and logging, hybrid judging (to balance cost and fidelity), and explicit support for multi-turn conversational testing.

By addressing these gaps, the proposed toolkit aims to lower the barrier to entry for systematic LLM red-teaming for researchers, students, and small teams.

2.8 Conclusion of the Chapter

In this chapter we surveyed the technical foundations of large language models — their architecture, alignment techniques, and associated safety risks — and we reviewed a broad spectrum of attack vectors: from simple single-shot jailbreaks to prompt injection, multi-turn manipulations, universal triggers, and Trojan-style prompt attacks.

We examined the diversity of red-teaming methodologies (manual, rule-based, and automated), and we discussed evaluation metrics and recent benchmark frameworks such as *HarmBench*.

We also surveyed existing open-source tools and frameworks, highlighting trade-offs between accessibility, scalability, and coverage.

Finally, we identified core gaps in tooling, evaluation, and reproducibility — gaps that motivate the design and implementation of the red-teaming toolkit proposed in the next chapter.

With this foundation in place, the next chapter will provide a detailed survey of existing open-source red-teaming toolkits, followed by a requirements analysis and the design of the proposed modular, extensible toolkit.

Framework	Attack Types Supported	Model Backend Support	Multi-turn Capability	Evaluation / Judge Type	Key Strengths / Limitations
PyRIT [14]	Rule-based prompts, template attacks, basic prompt injection, simple adversarial transforms.	Supports API-based models (OpenAI, Anthropic), simple REST adapters; limited support for local LLMs.	Partial - supports manual conversational sessions but lacks automated multi-turn logic.	Basic keyword/regex checks; optional external classifiers.	+ Lightweight, easy to use. – Limited automation, no search-based generators, weak judge reliability.
garak [3]	Template attacks, perturbation probes, obfuscation, prompt rewriting via "generators".	API models + HuggingFace backends; modular "generators" allow extension.	Partial - supports sequential probing, but not full conversational trajectories.	Detectors: regex, classifiers; HTML/JSON reporting.	+ Mature, widely used, structured probes. – Limited advanced automated attack generation; multi-turn support minimal.
MAD-MAX [21]	Modular system: rule-based, template-based, LLM-generated attacks, search-based attacks.	Designed for flexible backends; supports API and local models; plugin-based adapters.	**Yes** - modular conversation orchestrator; supports multi-turn flows.	LLM-as-a-judge, rule-based detectors, modular evaluation pipeline.	+ Highly modular, extensible. – High compute requirements; complex to deploy; best suited for labs.
Automatic LLM Red Teaming (Belaire et al. 2025) [1]	LLM-as-attacker, RL/HF-based optimisation, trajectory-level red teaming, multi-turn adversarial search.	API and local models; depends on model-in-the-loop capabilities.	**Yes** - multi-turn attacks optimized using RL / MDP formulation.	LLM judges (safety classifiers, preference models).	+ State-of-the-art automated discovery; finds subtle failures. – Expensive; requires careful calibration; not lightweight.
TAP (Tree-of-Attacks) [13]	Search-based automated jailbreaks, adversarial refinement tree, black-box optimization.	Black-box API models (OpenAI, Anthropic, etc.). Local models possible with wrappers.	No - primarily single-turn; generates one-shot optimized attacks.	Simple heuristic judges; optional LLM-as-a-judge extension.	+ Extremely query-efficient; discovers novel jailbreaks. – Limited to single-turn; no

Chapter 3

Design

Chapter 4

Implementation

Chapter 5

Evaluation

Chapter 6

Conclusion

Bibliography

[1] Belaire, R.; Sinha, A. and Varakantham, P. *Automatic LLM Red Teaming*. 2025. Available at: https://arxiv.org/abs/2508.04451.

[2] Borgeaud, S.; Mensch, A.; Hoffmann, J.; Cai, T.; Rutherford, E. et al. *Improving language models by retrieving from trillions of tokens*. 2022. Available at: https://arxiv.org/abs/2112.04426.

[3] Derczynski, L.; Galinkin, E.; Martin, J.; Majumdar, S. and Inie, N. *Garak: A Framework for Security Probing Large Language Models*. 2024. Available at: https://arxiv.org/abs/2406.11036.

[4] Dettmers, T.; Pagnoni, A.; Holtzman, A. and Zettlemoyer, L. *QLoRA: Efficient Finetuning of Quantized LLMs*. 2023. Available at: https://arxiv.org/abs/2305.14314.

[5] Esmailzadeh, Y. *Potential Risks of ChatGPT: Implications for Counterterrorism and International Security*. 2023. Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4461195.

[6] Fedus, W.; Zoph, B. and Shazeer, N. *Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity*. 2022. Available at: https://arxiv.org/abs/2101.03961.

[7] Kumar, A.; Agarwal, C.; Srinivas, S.; Li, A. J.; Feizi, S. et al. *Certifying LLM Safety against Adversarial Prompting*. 2025. Available at: https://arxiv.org/abs/2309.02705.

[8] Lee, H.; Phatale, S.; Mansoor, H.; Mesnard, T.; Ferret, J. et al. *RLAIF vs. RLHF: Scaling Reinforcement Learning from Human Feedback with AI Feedback*. 2024. Available at: https://arxiv.org/abs/2309.00267.

[9] Liu, X.; Yu, Z.; Zhang, Y.; Zhang, N. and Xiao, C. *Automatic and Universal Prompt Injection Attacks against Large Language Models*. 2024. Available at: https://arxiv.org/abs/2403.04957.

[10] Liu, Y.; Deng, G.; Li, Y.; Wang, K.; Wang, Z. et al. *Prompt Injection attack against LLM-integrated Applications*. 2024. Available at: https://arxiv.org/abs/2306.05499.

[11] Lu, X.; Liu, Z.; Liusie, A.; Raina, V.; Mudupalli, V. et al. *Blending Is All You Need: Cheaper, Better Alternative to Trillion-Parameters LLM*. 2024. Available at: https://arxiv.org/abs/2401.02994.

[12] Mazeika, M.; Phan, L.; Yin, X.; Zou, A.; Wang, Z. et al. *HarmBench: A Standardized Evaluation Framework for Automated Red Teaming and Robust Refusal*. 2024. Available at: https://arxiv.org/abs/2402.04249.

[13] Mehrotra, A.; Zampetakis, M.; Kassianik, P.; Nelson, B.; Anderson, H. et al. *Tree of Attacks: Jailbreaking Black-Box LLMs Automatically*. 2024. Available at: https://arxiv.org/abs/2312.02119.

[14] Munoz, G. D. L.; Minnich, A. J.; Lutz, R.; Lundeen, R.; Dheekonda, R. S. R. et al. *PyRIT: A Framework for Security Risk Identification and Red Teaming in Generative AI System*. 2024. Available at: https://arxiv.org/abs/2410.02828.

[15] OpenAI. *Red Teaming Network*. 2023. Available at: https://openai.com/blog/red-teaming-network.

[16] Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C. L. et al. *Training language models to follow instructions with human feedback*. 2022. Available at: https://arxiv.org/abs/2203.02155.

[17] Parliament, E. and Council. *Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act)*. 2024. Available at: https://eur-lex.europa.eu/eli/reg/2024/1689/oj.

[18] Perez, F. and Ribeiro, I. *Ignore Previous Prompt: Attack Techniques For Language Models*. 2022. Available at: https://arxiv.org/abs/2211.09527.

[19] Purpura, A.; Wadhwa, S.; Zymet, J.; Gupta, A.; Luo, A. et al. *Building Safe GenAI Applications: An End-to-End Overview of Red Teaming for Large Language Models*. 2025. Available at: https://arxiv.org/abs/2503.01742.

[20] Rafailov, R.; Sharma, A.; Mitchell, E.; Ermon, S.; Manning, C. D. et al. *Direct Preference Optimization: Your Language Model is Secretly a Reward Model*. 2024. Available at: https://arxiv.org/abs/2305.18290.

[21] Schoepf, S.; Hameed, M. Z.; Rawat, A.; Fraser, K.; Zizzo, G. et al. *MAD-MAX: Modular And Diverse Malicious Attack MiXtures for Automated LLM Red Teaming*. 2025. Available at: https://arxiv.org/abs/2503.06253.

[22] Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A. and Klimov, O. *Proximal Policy Optimization Algorithms*. 2017. Available at: https://arxiv.org/abs/1707.06347.

[23] Series, O. C. S. *LLM Prompt Injection Prevention Cheat Sheet*. 2025. Available at: https://cheatsheetseries.owasp.org/cheatsheets/LLM_Prompt_Injection_Prevention_Cheat_Sheet.html.

[24] Shaw, A.; Ye, A.; Krishna, R. and Zhang, A. X. *Agonistic Image Generation: Unsettling the Hegemony of Intention*. 2025. Available at: https://arxiv.org/abs/2502.15242.

[25] Stiennon, N.; Ouyang, L.; Wu, J.; Ziegler, D. M.; Lowe, R. et al. *Learning to summarize from human feedback*. 2022. Available at: https://arxiv.org/abs/2009.01325.

[26] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L. et al. Attention Is All You Need. *CoRR*, 2017, abs/1706.03762. Available at: http://arxiv.org/abs/1706.03762.

[27] Wei, A.; Haghtalab, N. and Steinhardt, J. *Jailbroken: How Does LLM Safety Training Fail?* 2023. Available at: https://arxiv.org/abs/2307.02483.

[28] Xue, J.; Zheng, M.; Hua, T.; Shen, Y.; Liu, Y. et al. *TrojLLM: A Black-box Trojan Prompt Attack on Large Language Models*. 2023. Available at: https://arxiv.org/abs/2306.06815.

[29] Zhang, B.; Liu, Z.; Cherry, C. and Firat, O. *When Scaling Meets LLM Finetuning: The Effect of Data, Model and Finetuning Method*. 2024. Available at: https://arxiv.org/abs/2402.17193.

[30] Zou, A.; Wang, Z.; Carlini, N.; Nasr, M.; Kolter, J. Z. et al. *Universal and Transferable Adversarial Attacks on Aligned Language Models*. 2023. Available at: https://llm-attacks.org.