International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2024: 7.101

Survey of Public Red-Teaming Frameworks for LLM: Techniques, Coverage, and Gaps

Karthikeyan Thirumalaisamy

Independent Researcher, Washington, USA
Corresponding Author Email: *kathiru11[at]gmail.com*

Abstract: ***With LLMs (Large Language Model) being deployed in an increasing number of high-risk environments, red- teaming is becoming one of the most important methods to expose the potential for unsafe behavior, jailbreaking, and adversarial vulnerability prior to actual exposure via a real-world attack. Recently, a large number of publicly accessible, research-based, and open-source tools have been developed to help automate, or otherwise enhance, the red team's process. Although these tools vary greatly in terms of how they approach the problem, what they cover, and their level of development, there does not exist a single source of information that outlines the current landscape of publicly accessible tools for Red Teaming LLMs. Therefore, this paper will provide a systematic analysis of the various frameworks used for red-teaming of LLMs by examining the methodologies of each framework, the various types of attack strategies employed by each framework, the levels of automation provided with each framework, and the objectives of each framework related to evaluating the safety of the framework. The paper will also provide insight into commonalities, advantages/disadvantages, and operational limitations of each framework and identify areas where red-teaming tools lack sufficient capability such as: performing long-horizon multi-turn attacks, exploiting agent/tool interactions, testing adversarial in multiple languages, and creating dynamic adaptive attack loops. The authors' ultimate goal with this paper is to assist researchers, developers, and users of systems utilizing LLMs to understand the current landscape of publicly accessible red-teaming tools for LLMs and to provide guidance on future directions for developing robust, scalable, and comprehensive adversarial test tools for LLMs.***

Keywords: LLM Security, Red-Teaming, Adversarial Testing, Jailbreak Generation, Automated Attack Frameworks

1. Introduction

Large Language Models (LLMs) have transformed the way we build software systems by enabling us to develop many types of applications such as Search Engines, Conversational Agents, Autonomous Workflows, Safety-Critical Decision Systems & Enterprise Applications etc., are becoming increasingly pervasive in our daily lives. As LLMs evolve both in terms of their capabilities and the number of people using these systems, there is growing interest in identifying and mitigating potential risks from adversarial attacks. Therefore, red-teaming (the systematic adversarial evaluation of a system to identify potential unsafe, unintended, or harmful behavior) has become a key area of focus for ensuring that LLMs are secure and robust.

In contrast to traditional software systems, LLMs have a highly dynamic and context-dependent attack surface, because of their ability to engage with users in natural language, their complex reasoning structures, and their ability to support multi-turn conversations, and also due to the wide variety of ways that attackers can utilize tools to achieve their goals (e.g., inducing harmful responses or evading alignment). To address the challenges associated with this type of attack surface, automated red-teaming frameworks have been developed, which provide repeatable, scalable, and model-specific testing methods for evaluating the vulnerabilities of LLMs. In addition to leveraging techniques such as Adversarial Prompt Generation, Multi-Turn Conversation Strategies, Attack Pattern Libraries, and Autonomous Agents, these frameworks allow for the systematic probing of model vulnerabilities.

Although a number of automated red-teaming frameworks exist for LLMs, the current state of the field is characterized by significant fragmentation. The tools differ significantly in terms of their purpose, methodology, level of maturity, and target use case. For example, some tools are designed to test single-turn "jailbreak" attacks against LLMs, while other tools are designed to test multi-turn conversational attacks against LLMs. Additionally, some tools rely on multi-agent systems to simulate long-term (persistent) adversaries. The existing frameworks for red-teaming LLMs lack a unified evaluation system which assesses their methods and attack strategy classification and automation levels and complete coverage of known vulnerabilities.

Therefore, the objective of this survey is to provide a structured evaluation of the publicly documented red-teaming frameworks for LLMs, to evaluate their methodologies, classify them based upon the attack strategies and automation capabilities employed, and to assess the degree to which each tool provides coverage of the various vulnerability categories that have been previously identified. Furthermore, the survey will identify areas where additional research is needed to advance the field, specifically with regards to developing standard benchmarks for evaluating the effectiveness of red-teaming frameworks, the evaluation of red-teaming frameworks using multi-agent adversarial attack scenarios, and the development of red-teaming frameworks that model real world application environments.

Ultimately, the purpose of this survey is to provide a basis for researchers, developers, and security professionals to understand what tools currently exist to support the red-teaming of LLMs, how these tools operate, and where future research efforts should be directed to enhance the safety and reliability of evaluations conducted on large language models.

2. Taxonomy of Red-Teaming Frameworks for Large Language Models

The methodologies used in the large number of public red teaming frameworks for Large Language Models differ in significant ways in terms of how attacks are generated, how deep in terms of automation each framework is, how attackers are modeled, and what kinds of risks the frameworks are designed to reveal. In order to organize the many different methods that have been developed, we develop a taxonomy of the existing public frameworks based on the style in which attacks are generated, the type of interaction models used by the framework and the type of operational goal of the framework. The use of a single classification system will help to make disparate toolsets, including both academic "jail break" generators and more complex multi-turn adversarial dialogue systems and also multi-agent red-teaming environments more comparable.

2.1. Single-Turn Adversarial Prompt Generators

Single-turn adversarial prompt generators create one-time jailbreaks and dangerous queries which attempt to outsmart LLM safety mechanisms through a single dialogue exchange. The frameworks generate one optimized input to achieve the highest probability of obtaining prohibited responses during their first interaction. The systems employ different generation approaches which include heuristic-based transformations and keyword manipulation and gradient-guided optimization and evolutionary search techniques to test model decision boundaries. Single-turn systems operate independently from context development so they enable quick stress testing which makes them appropriate for automated evaluation systems and model version comparison and alignment update testing. These tools serve as essential tools for vulnerability assessment and model-agnostic jailbreak testing because they lack the ability to create persistent attacks.

2.1.1. Garak

Garak is an open-source framework (the largest to date) that has been developed by the AI safety community to test the robustness of Large Language Models with red-team attacks and stress-tests. Garak has a large number of probes that are adversarial in nature; they are designed to take advantage of the vulnerabilities present in systems by using single-turn attacks to create system weaknesses from the adversary's perspective. The modular design of Garak allows the user to choose which "prober" they want to use to target each of the various types of vulnerability classes, such as: content that can be harmful, jailbreaks, weaknesses in prompt injection, unsafe completions, or policy evasion.

The primary mechanism of operation for Garak involves generating one-shot adversarial prompts, which makes it ideal for testing how LLM's behave when directly attacked. Garak also comes with numerous pre-built adversarial templates and mutation strategies; these enable systematic coverage over a variety of safety domains. Finally, because Garak is open source and can be easily installed via pip, it is very simple to incorporate Garak into your continuous integration pipeline, batch evaluations and automated testing workflows.

2.1.2. Rebuff

Rebuff is an open-source tool that identifies and evaluates attacks using single-turn (one-shot) adversarial prompts to test LLMs. While Garak has a large, open-source "red-team" plugin library, Rebuff was built to be as focused as possible on identifying and evaluating all types of prompt-injection attacks in real-time. Using both attack simulation and defensive testing capabilities, Rebuff generates known injection patterns, compares the model response with what would be considered safe behavior, and can identify whether a prompt is malicious via its use of embedding-based similarity comparisons and a classification layer; therefore, Rebuff is useful to evaluate how resistant an LLM will be to one-shot jailbreaks, indirect injections, and hand-crafted, adversarial payloads. Due to its light-weight design and easy-to-use REST style integration, Rebuff can be used easily by developers who are looking for a simple way to check their applications quickly for vulnerabilities without having to set up a more complex multi-turn red-teaming environment. Rebuff is fully open-source and is widely referenced in applied LLM security testing.

2.1.3. Adversarial Prompt

Adversarial Prompt is a utility with a focus on creating one shot adversarial prompts to allow a user to "jailbreak" an LLM filter system. Adversarial Prompt has a primary goal of generating single turn jail break prompts and does this using three strategies; template mutation, lexical substitution and semantic reframing. Adversarial Prompt generates candidate prompts through applying various transformation strategies (e.g., synonym substitution, constraint inversion, persona prompting, goal hijacking) that increase the probability of a harmful or restricted response. Adversarial Prompt is lightweight, can be quickly run and easily integrated into your CI pipeline/evaluation workflow which makes it ideal for rapid stress testing of the safety boundaries of an LLM during the early stages of development. Although the attack strength of the Adversarial Prompt is less than multi-turn/red team or agent based methods, Adversarial Prompt is a good starting point and a useful tool for doing basic jailbreak assessments of an LLM during early stage evaluations.

2.1.4. Promptfoo

Promptfoo provides open-source support for adversarial testing of prompts as well as for generating jailbreak style prompts (e.g., using role confusion or attempting to override instructions). Although primarily used for testing the behavior of LLM's with respect to changes in the input prompt, Promptfoo now supports single turn adversarial testing of inputs to identify prompt injection and policy bypass vulnerabilities in LLMs.

Users of Promptfoo may use it to create one shot query adversarial prompts based upon defined template and attack pattern definitions, which are then evaluated against the target model by executing them at scale and comparing the response generated from those prompts to a set of pre-defined expectations and/or evaluations using assertions, rules and/or customized evaluation checks to verify whether the model violated one or more specified safety and/or behavioral constraints. Promptfoo is applicable for testing of both hosted API's and self-hosted LLM's since it is a black-box test methodology; therefore, no internal knowledge of the model

being tested is required. Additionally, Promptfoo is easy to reproduce, and is well suited for rapid boundary testing and establishing a baseline for potential vulnerabilities utilizing single-turn adversarial prompts, and may be easily integrated within a CI/CD environment.

2.1.5. LLMFuzzer

Single-turn adversarial prompt generation through open-source implementations based on fuzzing techniques inspired by large language model (LLM) fuzzing techniques provides several open source tools that can be used to create single turn adversarial prompts in a systematic manner. Each tool uses previously defined transformation rules such as paraphrasing, role reversal, semantic obscuration, and/or re-ordering of instructions to generate prompts that will cause an LLM to respond in an unsafe/unintended manner.

Unlike many multi-turn red teaming frameworks that generate independent, one shot prompts to test a model's response to each prompt individually, these tools generate prompts independently and examine responses to identify whether the generated responses violate policies (e.g., policy bypass), contain potentially harmful content, or leak instructions from the original input prompt. Unlike multi-turn frameworks that require sophisticated conversational state tracking, the mutation-based methodology allows users to test a wide range of potential prompts without needing to track the conversational state between prompts.

In particular, these tools have been shown to be highly effective at finding weak points in a model's safety boundaries that are prone to failure when exposed to similar types of prompts repeatedly, as well as identifying patterns of user-provided input that are consistently capable of evading safety features across a number of different models. As they are relatively light weight, they lend themselves well to use in automated scanning/ regression testing pipelines.

2.2. Multi-Turn Red-Teaming Frameworks

Whereas a single-turn adversarial prompt generator is able to generate a single prompt to attempt to elicit an undesirable response from a large language model (LLM), a multi-turn red-teaming framework simulates persistent adversaries that engage in a conversation with a large language model over multiple conversational turns. The purpose of these frameworks is to test if the safety limits of a large language model can be eroded through repeated manipulations of context, escalations of role, re-framing of questions, or repeated refinements of instructional prompts. Many dangerous behaviors are produced in deployed systems by means of incremental conversation, where each turn influences the next. In order to better reflect this type of threat model, multi-turn red-teaming frameworks maintain a record of all previous responses between the model and adversary; they also adapt their strategies as the conversation progresses based on the output of the model.

Characteristics of Red-Teaming Frameworks:

- Stateful interaction with the LLM in a multi-round format
- Adaptive strategy based upon prior outputs from the LLM
- Persistent or strategic adversaries
- More realistic than one-time prompt attacks
- Evaluates conversational agents, assistants and APIs based on conversations.

2.2.1. PyRIT (Python Risk Identification Toolkit)

The PyRIT (Python Risk Identification Toolkit), which was created by Microsoft is an open-source tool that enables large language model red teaming using automatically generated multi-turn adversarial interactions. In comparison to a single turn prompt generator, PyRIT is capable of simulating a persistent attacker who adapts his/her strategy across multiple conversational turns; thus, PyRIT is highly effective at probing safety boundaries, policy evasion, and cumulative alignment failure. PyRIT has a modular architecture that enables security testers to create attack strategies, objectives and success criteria and execute them iteratively against target LLMs. PyRIT also includes a wide variety of adversarial techniques such as: role-play escalation, context manipulation, prompt rewriting, and goal driven exploitation enabling very realistic simulations of human adversaries. The modular architecture of PyRIT also enables researchers and practitioners to easily add new attack patterns, scoring mechanisms, and evaluation metrics.

A major advantage of PyRIT is its focus on reproducibility and automation. Sessions involving attacks are logged, scored and replayable, making PyRIT ideal for conducting continuous security testing and regression analysis. Additionally, PyRIT integrates with multiple LLM vendors through API calls and may be added to CI/CD pipelines for the purpose of continuously assessing a model's or safety policy's defenses with a red team. In conclusion, PyRIT is one of the most mature and practical open source tools available today for multi-turn LLM red teaming. It closes the gap between academia-based red team methodologies and operational security testing, and serves as a solid foundation upon which to test how large language models react when subjected to long-term, adaptable adversarial pressures.

2.2.2. DeepTeam

DeepTeam, an open-source red teaming framework, allows for the evaluation of both the robustness and the safety of large language models (LLMs) through both automated and semi-automated adversarial testing methods. Unlike those tools used to generate single turn prompts, DeepTeam allows for structured testing workflows that can model real world attacker behaviors (i.e., iterative refinement of the prompt, scenario driven probing). With DeepTeam users are able to define their own red teaming objectives (i.e., elicitation of a violation of policy, unsafe content, or an alignment failure) and then perform systematic testing of all LLM responses relative to each objective. DeepTeam has been designed to be integrated with many of the most commonly used LLM application programming interfaces (APIs) and evaluation libraries, thereby providing users with the ability to conduct reproducible security evaluations and utilize DeepTeam in conjunction with continuous integration (CI) style testing pipelines. As an open source, ready to use tool, DeepTeam provides a user-friendly means of conducting multi-turn red teaming and serves as a complementary tool to other adversarial testing frameworks focused on pure research, but which lack usability, extensibility, and structured evaluation capabilities.

2.2.3. Garak (Multi-Turn Red-Teaming Mode)

Garak is an open-source testing platform designed for probing the vulnerabilities of Large Language Models (LLMs). It performs this by testing both single-turn and multi-turn adversarial testing techniques.

In the multi-turn configuration, Garak creates a simulated long-term adversary that interacts with the tested LLM on a conversational basis during the course of multiple conversations or "turns" to identify how an adversary could violate a Safety Boundary of the LLM. Garak does not isolate each prompt; rather, it retains the conversational history of interactions between the adversary and the tested LLM, enabling the adversary's malicious intent to evolve based upon the LLM's prior response(s). As opposed to isolated prompts, Garak's organized testing methodology includes modular probes, which allow the chaining of prompts, adaptation of follow-up query prompts, and escalation of the malicious intent of the probes. As a result, Garak provides a means to test a variety of complex testing scenarios, including but limited to, progressive jailbreak attempts, gradual policy erosion over time, and indirect instruction steering, all of which cannot be captured by single-turn testing methodologies. Specifically, Garak's multi-turn testing capabilities will help detect vulnerabilities associated with the accumulation of context from the conversational history, instructional override by means of conversational framing, and the delayed unsafe output of the LLM.

From a deployment standpoint, Garak has been implemented as a completely open-source project and is easily deployable via command-line interfaces. Garak also provides support for a variety of model backends, including locally hosted open-source LLMs and API-based LLMs. Therefore, Garak may be integrated into automated testing pipelines as part of an organization's overall security testing strategy. Although Garak does not utilize autonomous Reinforcement Learning (RL)–based adversarial optimization techniques to develop adversarial inputs against the tested LLMs, Garak's structured multi-turn probing methodology provides a practical and reproducible means of assessing the conversational robustness of deployed LLM systems.

2.3. Comparison of Single-Turn and Multi-Turn LLM Red-Teaming Frameworks

This section compares both single-turn and multi-turn red-teaming tool architectures as they relate to LLMs by comparing the features of adversarial prompt generators used for probing model safety boundaries and those of multi-turn red-teaming tools that generate persistent attackers that are capable of adapting their attacks across conversational turns. The comparison is intended to illustrate the relative differences among tools in terms of attack depth, level of automation, extensibility and usefulness in simulating real world attacks against LLMs for purposes of security evaluations. The purpose of the comparison is to assist both practitioners and researchers in selecting tools that meet their goals and the characteristics of their respective threat models.

Tool	Turn Type	Primary Focus	Key Capabilities	Use Cases
Garak	Single & Multi-Turn	Broad vulnerability probing	Prompt attacks, jailbreak probes, heuristic tests, plugin-based extensibility	Baseline security assessment, and regression testing
Rebuff	Single-Turn	Prompt injection detection	Injection pattern detection, canary tokens, response validation	Protecting RAG and tool-augmented prompts
Promptfoo	Single-Turn	Prompt robustness testing	Test matrices, prompt mutation, scoring, CI/CD integration	Prompt comparison and stress testing
Adversarial Prompt Generator	Single-Turn	One-shot jailbreak attempts	Automated adversarial prompt creation, heuristic mutation	Quick boundary testing
PyRIT	Multi-Turn	Automated red-teaming	Multi-turn attack chains, scoring, extensible attack strategies	Enterprise-grade red-teaming
DeepTeam	Multi-Turn	Continuous adversarial testing	Agent-based attacks, CI/CD integration, safety regression	Continuous LLM security evaluation
Garak (Multi-Turn Mode)	Multi-Turn	Persistent probing	Stateful attacks, conversation replay, plugin-based probes	Advanced vulnerability discovery

3. Gaps in Current LLM Red-Teaming Tooling

Although there are a number of open source and publicly documented red teaming frameworks for Large Language Models (LLMs) available to the public today, current tools have several significant limitations that impact both their ability to effectively be used and be adopted. The limitations mentioned above demonstrate that while LLM red-teaming is a growing area of study, it has yet to achieve the level of maturity and standardization associated with traditional application security testing.

3.1. Limited Coverage of Realistic Adversaries

The majority of current LLM red-teaming tools focus on exploiting vulnerabilities at the prompt level through single turn interaction or tightly defined multi-turn conversations. These tools have proven effective at detecting violations of safety boundaries immediately (jail breaks, refusal bypasses, policy evasions) but only represent an extremely small portion of actual, real world, adversary behavior. Real world human attackers are frequently capable of planning and executing long term strategies to achieve their objectives - adapting their tactics across extended interactions with the system, learning from past failed attempts, and continuing their pursuit of a malicious goal through multiple sessions.

However, many of the current tools fail to include functionality to model these types of behaviors. The tools do not keep track of the history of the previous attempts made against the system; therefore, there is no mechanism to

dynamically adjust attack strategies based upon partial successes; nor do the tools simulate changes in the attacker's ultimate goals during the course of the interactions. Additionally, the majority of the evaluation of the effectiveness of attacks occur individually rather than as part of a larger campaign that includes multiple prompts, tools, and/or agents interacting with each other in concert. Therefore, the capabilities of these tools to describe sophisticated threat profiles (staged manipulation, gradually poisoned context, etc.) that involve the coordinated use of LLM-powered systems are limited.

Although existing red-teaming tools provide some useful insights into the vulnerabilities of the prompt robustness and alignment of LLMs, they do not effectively portray a sophisticated, real-world adversary. Future tools will need to be developed to address this gap which will likely require the inclusion of persistent attacker models, adaptive strategy development, and coordination among multiple agents to accurately depict the threat landscape facing deployed LLM applications.

3.2. Lack of Standardized Metrics and Benchmarks

The current LLM red-teaming tools face a major restriction because they lack established metrics which would enable researchers to measure attack success and model resistance levels. The current frameworks depend on tool-specific success criteria which include policy violations and heuristic classifier triggers and unsafe content generation but these criteria prevent researchers from making tool and model and study comparisons. Research studies often present success as a simple pass/fail result which fails to show the complex nature of adversary interactions while also disregarding situations where safety protocols fail partially or when risks change based on specific contexts.

The current state of red-teaming benchmark datasets and evaluation suites lacks both comprehensive coverage and standardized assessment methods. The current benchmarks for evaluation focus on particular domains which include harmful content creation and jailbreak command generation but they fail to measure how models handle complex dialog sequences and how attackers adapt their strategies and how systems perform when used in actual deployment scenarios with retrieval-augmented generation and tool-enabled agents. The results from red-teaming activities become impossible to duplicate between different systems and new tool developments fail to produce quantifiable results when tested in various assessment scenarios. The absence of established measurement criteria and performance indicators makes it impossible to conduct fair assessments which hampers scientific advancement and restricts practitioners from determining actual LLM system security levels.

3.3. Limited Integration with Real LLM Deployment Pipelines

Most of the current red-teaming tools for Large Language Models (LLMs) function independently as testing applications rather than as part of real-world deployment pipelines. Most often, they run separately from actual production environments like Retrieval-Augmented Generation (RAG), agent orchestration platforms, or Continuous Integration / Continuous Deployment (CI/CD) environments. Therefore, most of these red-teaming tools do not take into consideration all of the contextual elements that have a significant impact on real-world risk exposure including; dynamically constructed prompts, external data retrieved during runtime, chain invocations of tools, session memory, or runtime policy enforcement.

Additionally, most of the current red-teaming tools also do not natively integrate with logging, monitoring, or alerting systems commonly found within the production environments where large language models are deployed. This greatly limits the utility of the red-teaming tools for continuous security validation, regression testing of LLMs post-updates, or discovering new patterns of attacks as they emerge over time. Because most red-teaming frameworks lack native support for integrating into pipelines, the majority of red-teaming efforts are manual and episodic, whereas they need to be continuous and automated. To bridge this gap will require greater alignment between red-teaming frameworks and the architecture of modern LLM deployment pipelines, allowing security testing to become an integral part of operating AI systems, versus being treated as an episodic pre-deployment process.

3.4. Limited Coverage of Agent-Based and Tool-Invocation Attacks

Red Teaming research has seen some success in testing standalone LLM (Large Language Model) behavior; however, the vast majority of existing red teaming tools have either very little or no capability to test agent-based systems with external Tools, APIs, plugins, workflows etc. Most modern LLM deployments are using autonomous or semi-autonomous agents which may take action such as file access, code execution, database queries, web browsing and API calls. The new capabilities create a larger attack vector (unauthorized tool invocation, privilege escalation, command injection, indirect prompt injection via tool response). Most of today's red team frameworks view the LLM as an isolated text generation component and do not simulate the security implications of permissions and execution contexts of tools being used by agents or cross-agent interactions. Therefore, many critical vulnerabilities (e.g., over-powered agents, unsafe tool chaining and malicious tool feedback loops) remain untested. This demonstrates the need for red teaming tools which explicitly account for agent architectures and tool-augmented execution

4. Conclusion

The focus of this paper was the review of the publicly available and open-source red teaming tools for large language models (LLM) with an emphasis on those tools which are practical, replicable, and ready for use by researchers and practitioners. Through the examination of both single-turn adversarial prompt generation tools and multi-turn red teaming frameworks, the authors were able to demonstrate how existing red teaming tools can test the limits of alignment boundaries, safety filtering and policy

enforcement mechanisms through prompt-based attack vectors and controlled conversational adversaries.

The results from our analysis indicate that the red teaming ecosystem for LLMs has advanced from the purely theoretical realm to an applied one; however, it is still not equally developed or comprehensive in terms of its overall scope and depth. Tools currently exist to identify vulnerabilities at the prompt level and repeated "jailbreak" methods, but they do not effectively model a realistic adversary, integrate well into production LLM workflows, nor address newer risks including agent driven attacks and improper usage of tools. Additionally, the lack of established standards and metrics for evaluating the performance of red teaming tools complicate objective comparisons between tools and make reproduction of results difficult.

Ultimately, current red teaming frameworks should be regarded as important but unfinished components of LLM security testing. While they provide useful insight into the vulnerabilities of LLMs, they need to be combined with additional system-wide assessments, deployment aware testing, and ongoing monitoring approaches. As LLMs move forward to become autonomous, tool utilizing entities integrated into real-world applications, future red teaming tools will need to expand their ability to manipulate prompts into a more holistic approach to security testing across the entire life cycle of an LLM. This paper is intended to serve as a foundational resource for future work in developing more comprehensive and lifecycle aware LLM red teaming toolsets by defining the current state of red teaming tools, their limitations, and the challenges remaining in the development of these tools.

References

[1] Github, Garak. [Online]. Available: https://github.com/NVIDIA/garak
[2] L. Derczynski, E. Galinkin, J. Martin, S. Majumdar, and N. Inie, "garak: A framework for security probing large language models," arXiv, vol. 2406.11036, 2024. doi:10.48550/arXiv.2406.11036
[3] G. D. Lopez Munoz, A. J. Minnich, R. Lutz, R. Lundeen, R. S. R. Dheekonda, N. Chikanov, B.-E. Jagdagdorj, M. Pouliot, S. Chawla, W. Maxwell, B. Bullwinkel, K. Pratt, J. de Gruyter, C. Siska, P. Bryan, T. Westerhoff, C. Kawaguchi, C. Seifert, R. S. S. Kumar, and Y. Zunger, "PyRIT: A framework for security risk identification and red teaming in generative AI system," arXiv, vol. 2410.02828, 2024. doi:10.48550/arXiv.2410.02828
[4] Promptfoo, Intro. [Online]. Available: https://www.promptfoo.dev/docs/intro/
[5] Rebuff, Github. [Online]. Available: https://github.com/protectai/rebuff
[6] Azure, PyRIT. [Online]. Available: https://azure.github.io/PyRIT/
[7] Ram Shankar Siva Kumar, Announcing Microsoft's open automation framework to red team generative AI Systems, 2024. [Online]. Available: https://odsc.medium.com/pyrit-the-python-risk-identification-tool-enhancing-generative-ai-security-33c89cd29516
[8] ODSC - Open Data Science, PyRIT: The Python Risk Identification Tool Enhancing Generative AI Security, 2024. [Online]. Available: https://learn.microsoft.com/en-us/azure/aks/confidential-containers-overview
[9] J. Hayase, E. Borevkovic, N. Carlini, F. Tramèr, and M. Nasr, "Query-based adversarial prompt generation," arXiv, vol. 2402.12329, 2024. doi:10.48550/arXiv.2402.12329
[10] Natalie Maus, Patrick Chao, Eric Wong, Jake Gardner, Adversarial Prompting, 2023. [Online]. Available: https://debugml.github.io/adversarial-prompts/
[11] DeepTeam, Quick Introduction. [Online]. Available: https://www.trydeepteam.com/docs/getting-started
[12] Github, DeepTeam. [Online]. Available: https://github.com/confident-ai/deepteam
[13] Karthick Nagarajan, Securing AI with DeepTeam — LLM red teaming framework, 2025. [Online]. Available: https://dev.to/karthick965938/securing-ai-with-deepteam-llm-red-teaming-framework-5fbb
[14] Kevin Hernandez, Red Teaming with the DeepTeam Framework, 2025. [Online]. Available: https://www.krasamo.com/red-teaming/
[15] Qrious Kamal, Self-Hardening Prompt Injection Detector-Rebuff: Anti-Prompt Injection Service Using LLMs, 2023. [Online]. Available: https://medium.com/@kamaljp/self-hardening-prompt-injection-detector-rebuff-anti-prompt-injection-service-using-llms-6ab766c1c444
[16] Ian Webster, Promptfoo vs Garak: Choosing the Right LLM Red Teaming Tool, 2025. [Online]. Available: https://www.promptfoo.dev/blog/promptfoo-vs-garak/
[17] Trevor Carstensen, Interrogating Intelligence: Red Teaming LLMs with Garak, 2025. [Online]. Available: https://trevorcarstensencybersecurity.com/2025/03/17/interrogating-intelligence-red-teaming-llms-with-garak/
[18] Ayush kumar, Promptfoo vs Deepteam vs PyRIT vs Garak: The Ultimate Red Teaming Showdown for LLMs, 2025. [Online]. Available: https://dev.to/ayush7614/promptfoo-vs-deepteam-vs-pyrit-vs-garak-the-ultimate-red-teaming-showdown-for-llms-48if
[19] Github, LLMFuzzer. [Online]. Available: https://github.com/mnns/LLMFuzzer