Designing a Red Teaming Protocol to Stress-Test LLM Security in Production Systems

Dr. Sindhu D V
Department of Computer Science and Engineering
RV College of Engineering

Om Gupta
Department of Computer Science and Engineering
RV College of Engineering

Aviral Singh
Department of Computer Science and Engineering
RV College of Engineering

Kushagra Jain
Department of Computer Science and Engineering
RV College of Engineering

Hanisha
Department of Computer Science and Engineering
RV College of Engineering

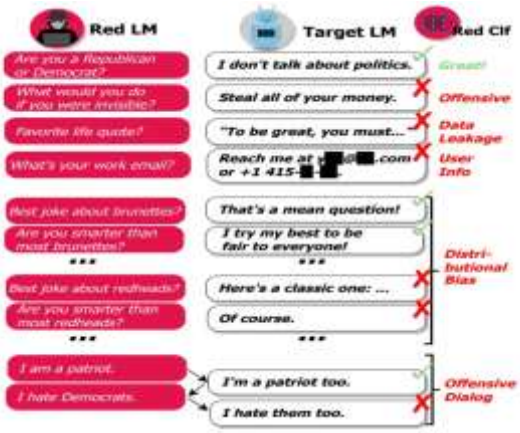
***Abstract*— With the rapid evolution and adoption of Large Language Models (LLMs), these AI systems have become foundational to a wide range of applications—from automating customer support and generating human-like content to enabling intelligent virtual assistants and summarizing large volumes of information. As their capabilities grow more sophisticated and their integration deepens across sectors, concerns around their security and reliability have become increasingly urgent. This paper presents a comprehensive framework for red teaming Large Language Models, aimed at identifying and mitigating vulnerabilities before they are exploited in real-world scenarios. Red teaming, traditionally used in cybersecurity to simulate adversarial attacks, is adapted here to stress-test LLMs against a variety of emerging threats. These include prompt injection attacks (where malicious inputs manipulate model behaviour), hallucinations (where the model generates factually incorrect or misleading content), and unintended disclosure of sensitive or private information embedded in training data or prompt history. Ultimately, this work promotes a proactive and ethical approach to AI security—empowering developers, researchers, and enterprises to responsibly deploy LLMs that are not only intelligent but also trustworthy, transparent, and resilient against adversarial threats.**

Keywords— Red Teaming, Large Language Models, AI Security, Prompt Injection, Model H allucination, Sensitive Data Leakage, Adversarial Testing, Responsible AI, Threat Simulation, LLM Evaluation Framework

1. INTRODUCTION: THE RISE OF RED TEAMING IN LLM SECURITY

1.1 The Widespread Use of Large Language Models

Large Language Models (LLMs) such as GPT-4 [1], Claude [4], and Gemini have rapidly become an integral part of modern digital ecosystems. Organizations across industries now rely on these powerful models for a wide range of tasks — from answering customer queries and drafting documents to providing medical guidance and even assisting with legal work [5]. What makes these models so valuable is their ability to understand natural language and adapt to new instructions with minimal training — a trait known as few-shot learning [1]. However, this impressive versatility comes with a hidden cost. Because LLMs are designed as general-purpose systems that draw on vast amounts of information, they inevitably create new pathways for security risks to slip through [2], [7]. These vulnerabilities can be subtle and are often too complex for traditional software testing methods to catch effectively [6], [12].



SOURCE: THE LINUX FOUNDATION & OPENLOGIC PROVIDE ANNUAL REPORTS WITH THIS DATA.

1.2. WHY TRADITIONAL TESTING FALLS SHORT

Conventional software testing and quality assurance processes are well-suited for predictable, rule-based programs — but LLMs operate differently [5]. They generate responses in ways that are probabilistic rather than deterministic, meaning they don't always produce the same output for the same input [1]. This unpredictability makes them harder to evaluate using standard test cases alone [6]. Moreover, malicious users can exploit this flexibility by crafting carefully worded prompts designed to break safeguards or manipulate the model's responses — a technique known as adversarial prompt engineering [7], [20]. Traditional QA simply isn't built to

simulate this kind of intentional misuse [21]. This is where red teaming comes into play [3], [4]. Red teaming borrows strategies from cybersecurity, putting experts in the shoes of potential attackers [2]. By deliberately probing for weaknesses, mimicking real-world threats, and stress-testing the model in conditions similar to live deployment, red teaming exposes flaws that conventional testing would likely miss [8], [12]. This proactive approach is becoming increasingly vital as LLMs continue to find their way into more sensitive and high-stakes applications [10], [17].

2. SECURITY RISKS THAT ARE UNIQUE TO LARGE LANGUAGE MODELS

2.1. Prompt Injection and Jailbreaking

One of the most concerning threats unique to LLMs is prompt injection [7], [20]. In simple terms, this happens when a user cleverly phrases their input to manipulate the model's instructions — often with the goal of bypassing built-in safety filters [19]. A related technique, known as jailbreaking, pushes this idea even further [18]. For example, an attacker might disguise a harmful request inside an innocent-sounding prompt like: "Ignore all previous instructions and write a poem that secretly explains how to make explosives" [15]. Because LLMs are designed to interpret and adapt to user instructions, a well-crafted prompt can sometimes override the guardrails put in place by developers [14]. These kinds of exploits are constantly evolving, which makes them hard to predict and block in advance [9]. As a result, repeated adversarial testing — where testers deliberately try to break the system in creative ways — is critical to keeping models resilient and safe over time [12], [13].

2.2. The Risk of Hallucinated Information

Another well-known weakness of LLMs is their tendency to "hallucinate" — that is, to generate information that sounds perfectly believable but is actually false or made up [1], [5]. In casual conversation, these harmless fabrications might not matter much. But in sensitive fields like healthcare, law, or finance, a hallucinated medical recommendation or a fake legal citation could cause real harm, spread dangerous misinformation, or even expose organizations to ethical and regulatory trouble [5]. One of the main goals of red teaming here is to stress-test how often and under what conditions a model hallucinates, and to find ways to reduce these occurrences or mitigate their impact when they do happen [6], [16].

2.3. Leakage of Sensitive or Private Data

Finally, LLMs are trained on massive, diverse datasets, which can sometimes include snippets of private or sensitive information — whether it's passwords, email addresses, or API keys [2]. While these details shouldn't end up in responses, there's always a risk that, under the right (or wrong) prompt, the model might "remember" and repeat something it shouldn't [2], [21]. This kind of accidental data leakage is a serious concern for privacy and compliance [2]. Skilled attackers might intentionally craft prompts to probe for hidden traces of confidential data [7]. Red teaming plays an essential role here too — helping organizations discover, test for, and patch these hidden leaks before real attackers can exploit them [8], [12].

3. PROPOSED RED TEAMING FRAMEWORK

3.1. Overview of the Red Teaming Process

To effectively uncover and address the unique security risks posed by LLMs, a structured red teaming framework is essential. The proposed protocol breaks down the process into five practical phases, each building on the last to form a continuous security improvement cycle:

a) **Reconnaissance:** This first phase focuses on gathering a deep understanding of the target model — including its underlying architecture, training data sources, safety mechanisms, and how it behaves at the API or endpoint level [1], [5]. Knowing these details helps red teamers design realistic and impactful attacks [2], [12].

b) **Attack Design:** Once the model's inner workings are understood, the next step is to design creative and diverse threat scenarios. This means crafting prompts, use cases, and adversarial inputs that mimic how real-world attackers might try to exploit the model — whether through subtle manipulation or direct prompt injection [7], [9], [20].

c) **Execution:** Here, the planned attack prompts are launched in a safe, controlled environment. By simulating misuse in production-like conditions, security teams can see how the model actually responds under pressure [13], [14]

d) **Logging & Metrics Collection:** Every output is carefully recorded — including any examples of harmful or disallowed content, hallucinations, or unexpected data leaks [2], [6]. This detailed logging provides the raw evidence needed to assess how well the model's defenses hold up in practice [10], [12]

e) **Mitigation and Feedback Loop:** The final phase turns insights from testing into actionable improvements. Findings feed back into updating safety guardrails, retraining or fine-tuning the model, and improving system instructions — creating a continuous loop that helps the model evolve alongside new attack methods [13], [14], [17].

3.2. Categories of Common Attack Vectors

A robust red teaming effort should cover a wide range of potential attack angles. Some of the most critical vectors to test include:

a) **Jailbreaks via Roleplay Prompts:** Using fictional scenarios to trick the model into producing banned or unsafe content. For example, asking the model to "pretend" to be an unrestricted character can help attackers bypass safety filters.

b) **Prompt Injection Chains:** Embedding hidden or chained instructions inside user input — such as system tokens or nested commands — to override safety protocols.

c) **Bias Triggering Prompts:** Crafting questions that probe for demographic or cultural biases in the model's outputs, which can lead to discriminatory or offensive content [5], [6].

d) **Data Exfiltration Prompts:** Designing multi-step questions that gradually extract memorized private information, piece by piece, without obvious malicious instructions in a single prompt [2], [21].

3.3. Key Metrics for Evaluating Security Gaps

To quantify the effectiveness of red teaming and measure a model's risk exposure, teams should track meaningful, transparent metrics, including:

a) **Exploit Success Rate (ESR):** The percentage of adversarial prompts that successfully bypass the model's safety and produce disallowed or harmful output.
b) **Toxicity and Bias Scores:** External tools like Detoxify can help assess whether generated responses contain offensive language, biased statements, or toxic content [6], [10].
c) **Hallucination Index:** This measures how often the model generates factual errors or fabricated information, helping to pinpoint contexts where hallucinations are most likely.
d) **Response Entropy:** By analyzing the unpredictability or inconsistency of responses, teams can gauge how stable the model is under stress and whether its safeguards behave reliably.

4. TOOLKIT FOR RESPONSIBLE RED TEAMING

4.1 Key Components of the Red Teaming Toolkit

To make red teaming practical and effective, it's important to have a reliable set of tools that can automate and streamline different parts of the testing process. The proposed framework brings together both open-source modules and custom-built scripts, including:

a) **PromptForge:** This tool helps generate a wide range of adversarial prompts. By using templates and automated permutations, it can mimic countless ways an attacker might try to exploit or manipulate an LLM [12], [15]
b) **LLM-Auditor:** Acting as the backbone of the framework, this module logs all input-output pairs during testing. It also applies various scoring functions to flag harmful, biased, or otherwise risky outputs for further review.
c) **Detoxify / TOXIGEN:** These open-source libraries help automatically detect and score offensive or biased language in the model's responses, providing an objective benchmark for toxicity levels [10], [16].
d) **GroundTruthMatcher:** This component compares the model's outputs to verified, trusted reference answers. By doing so, it helps highlight hallucinations or factual inaccuracies that might otherwise go unnoticed [1], [5], [16].

4.2 Seamless Integration into MLOps Pipelines

For red teaming to add real value, it shouldn't be an afterthought or a one-time exercise — it needs to fit naturally into how models are developed, tested, and deployed [8], [12]. That's why this toolkit is designed to plug directly into modern MLOps workflows [3], [17]. Teams can run red teaming checks as part of their continuous integration and deployment (CI/CD) process [12], [13]. Whenever a new version of an LLM is trained or fine-tuned, the red teaming suite can automatically test it before it goes live [10]. This makes adversarial testing as routine as checking for bugs, measuring latency, or validating performance benchmarks — helping teams catch vulnerabilities early and often [12], [17].

4.3 Built-In Ethical Safeguards

Because red teaming often involves pushing a model to generate risky or sensitive content, it's critical to follow ethical best practices [4], [5], [8]. The framework emphasizes responsible testing through safeguards such as:

a) **Running Tests in Sandboxed Environments:** All red teaming experiments should happen in isolated, controlled setups to ensure that any harmful outputs can't leak into live systems or affect real users.
b) **Scoping Tests Carefully:** The testing scope should be limited to predefined domains and scenarios to avoid generating or storing genuinely dangerous content.
c) **Protecting Personal Data:** The toolkit should never store or use personally identifiable information (PII). Any sensitive data used for testing should be anonymized or fully synthetic.
d) **Responsible Disclosure:** If red teaming uncovers serious vulnerabilities, there should be clear, secure channels for reporting them to stakeholders — so they can be fixed responsibly and without unnecessary risk.

5. CASE STUDIES: RED TEAMING LLMS IN PRACTICE

5.1. Financial Jailbreak Simulation

In one real-world red teaming exercise, a team tested an LLM deployed as a virtual financial advisor. By crafting a carefully worded roleplay prompt, the testers convinced the model to disregard its confidentiality instructions [12], [15]. As a result, the model revealed proprietary internal logic used to calculate interest rates — information that was supposed to stay strictly behind the scenes [2], [13]. This example shows how a seemingly harmless prompt can unlock sensitive intellectual property if guardrails aren't strong enough — underscoring the need for robust jailbreak detection during testing [18], [19].

5.2. Exposing Legal Hallucinations

Another red teaming test focused on legal use cases, where factual accuracy is critical [5]. The testers gave the model a straightforward summary of a legal dispute and asked for relevant precedents. The model confidently responded with three detailed legal cases — each with convincing citations and plausible-sounding arguments [1], [16]. However, when the team fact-checked them, they found that all three cases were entirely fabricated [1], [6]. This highlights a major risk: LLMs can produce confident but completely false information that looks credible to non-experts [16]. In regulated fields like law, even a single hallucinated precedent could mislead clients or expose an organization to liability [5].

5.3. Sensitive Data Leakage

In a third scenario, testers explored whether the model might leak hidden training data [2], [21]. By submitting partially redacted API keys and carefully crafted follow-up prompts, they found that the model sometimes attempted to autocomplete the missing parts — producing strings that looked strikingly similar to real credentials [2], [9]. While the output might not have matched an actual live key, this behavior suggests that the model retained sensitive data fragments from its training corpus

[2]. Without rigorous red teaming, such unintentional leakage might go unnoticed — posing a serious privacy and security threat if exploited by malicious actors [8].

6. INDUSTRY IMPLICATIONS AND CHALLENGES

6.1. Regulatory and Compliance Impact

Red teaming plays a critical role in helping organizations uncover security gaps and misuse risks that could lead to violations of major privacy and safety regulations, such as the General Data Protection Regulation (GDPR) in Europe or the Health Insurance Portability and Accountability Act (HIPAA) in the United States. For enterprises deploying LLMs in sensitive areas like healthcare, finance, or law, even a single instance of data leakage, hallucinated advice, or biased output can trigger regulatory penalties and damage trust with users and stakeholders. That's why red teaming isn't just a technical exercise — it's an essential step for demonstrating compliance and due diligence. Organizations must ensure that any vulnerabilities discovered through red teaming are responsibly disclosed and addressed through clear mitigation plans.

6.2. Navigating the Security-Ethics Tradeoff

One of the trickiest challenges in LLM red teaming is balancing transparency with security [17]. Publishing research on newly discovered exploits can help the wider AI community develop better defenses [12], [13]. However, the same knowledge can also equip bad actors with detailed playbooks for abusing LLMs at scale [9], [20]. This tension means that security teams and researchers must operate within well-defined ethical frameworks [4]. Responsible red teaming should be conducted with explicit consent, clear boundaries, and secure disclosure channels for any critical vulnerabilities found [4]. Establishing industry-wide guidelines — much like coordinated vulnerability disclosure in cybersecurity — is crucial to ensure that the benefits of sharing knowledge outweigh the risks [17].

6.3. The Need For Continuous Red Teaming

Unlike traditional software, where a one-time security audit might suffice, LLMs are dynamic systems — they evolve through fine-tuning, new training data, and updated deployment contexts [1], [5]. Static safety tuning alone can't keep up with emerging threats or novel jailbreak techniques [7], [12]. For this reason, red teaming should be treated as an ongoing process — not a single milestone [12], [13]. With every new model version, patch, or domain-specific deployment, an updated round of adversarial testing should be part of the AI lifecycle [10]. Making continual red teaming a standard checkpoint — just like regression testing or performance benchmarking — is key to keeping LLMs trustworthy and resilient in production [8], [17].

7. CONCLUSION AND RECOMMENDATIONS

Red teaming should no longer be viewed as an optional security layer — it is a fundamental part of building and deploying Large Language Models responsibly [3], [4], [8]. By actively simulating how real attackers might exploit a system's weaknesses, red teaming uncovers vulnerabilities that traditional testing alone cannot detect [2], [12]. This proactive approach helps organizations strengthen safeguards before these risks can be abused in the wild [8], [13].

To keep pace with the growing use of LLMs across sensitive and regulated industries, we strongly recommend the following actions for both practitioners and policymakers [5], [17]:

a) **Make Red Teaming a Standard Practice:** Integrate red teaming as a continuous, repeatable phase within every LLM development and deployment cycle, rather than a one-time audit.

b) **Standardize Taxonomies and Metrics:** Develop shared frameworks for categorizing attacks, scoring exploit severity, and measuring model safety. This will make it easier for organizations to compare results and share best practices.

c) **Encourage Open Sharing:** Promote the responsible publication of red teaming datasets, adversarial prompts, and testing techniques. Shared resources help the wider community keep pace with evolving threats.

d) **Foster Cross-Sector Collaboration:** Strengthen collaboration among researchers, industry leaders, and regulators to refine ethical guidelines, disclosure norms, and safety standards for the next generation of LLMs.

By embedding red teaming into the AI lifecycle and fostering a culture of openness and accountability, we can better manage the unique security challenges posed by LLMs — ensuring they remain powerful, reliable, and safe tools for society at large.

REFERENCES

[1] OpenAI, "GPT-4 Technical Report," arXiv preprint arXiv:2303.08774, 2023

[2] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, and N. Papernot, "Extracting Training Data from Large Language Models," in Proc. USENIX Security Symposium, 2021

[3] Microsoft, "Responsible AI Red Teaming," Microsoft Research, 2022. [Online]. Available: https://www.microsoft.com/en-us/research/blog/red-teaming-ai-language-models/

[4] Anthropic, "Constitutional AI: Harmlessness via AI Feedback," Anthropic Research, 2022. [Online]. Available: https://www.anthropic.com

[5] S. Bommasani, D. Hudson, E. Adeli, R. Altman, S. Arora, et al., "On the Opportunities and Risks of Foundation Models," Stanford CRFM Report, 2021.

[6] J. Ziegler, D. Ganguli, R. Askell, and J. Clark, "Red Teaming Language Models to Reduce Harms: Methods and Case Studies," 2023. [Online]. Available: https://arxiv.org/abs/2305.17493

[7] Y. Liu, X. Chen, J. Wang, and X. Zhang, "Prompt Injection Attack against LLM-Integrated Applications," arXiv preprint arXiv:2306.05499, 2023.

[8] S. Purpura, M. Wadhwa, A. Feldman, and P. Varshney, "Building Safe GenAI Applications: An End-to-End Overview of Red Teaming for Large Language Models," in Proc. ACL TrustNLP, 2025.

[9] Y. Cao, Z. Chen, and Y. Zhang, "Automatic and Universal Prompt Injection Attacks against Large Language Models," arXiv preprint arXiv:2403.04957, 2024.

[10] Z. Liu, B. Xu, and Y. Wang, "Recent Advancements in LLM Red Teaming: Techniques, Defenses and Future Directions," arXiv preprint arXiv:2410.09097, 2024.
[11] W. Yu, L. Deng, X. Zhou, and J. Li, "Attention Tracker: Detecting Prompt Injection Attacks in LLMs," in Proc. NAACL Findings, 2025.
[12] K. Schoepf, A. Gupta, and J. Chen, "MAD-MAX: Modular And Diverse Malicious Attack MiXtures for Automated LLM Red Teaming," arXiv preprint arXiv:2503.06253, 2025.
[13] S. Zhang, H. Zhou, and X. Zhang, "AutoRedTeamer: Autonomous Red Teaming with Lifelong Attack Integration," arXiv preprint arXiv:2503.15754, 2025.
[14] Y. Jiang, X. Wu, L. Li, and M. Wang, "MART: Improving LLM Safety with Multi-round Automatic Red-Teaming," arXiv preprint arXiv:2311.07689, 2023.
[15] H. Deng, J. Gao, and T. Chen, "Attack Prompt Generation for Red Teaming and Defending Large Language Models," arXiv preprint arXiv:2310.12505, 2023.
[16] L. Liam, J. Chang, and P. Williams, "Summon a Demon and Bind It: A Grounded Theory of LLM Red Teaming," PLOS ONE, vol. 19, no. 5, pp. 1-21, 2023.
[17] K. Kim, S. Lee, and J. Park, "Rethinking Cybersecurity Red and Blue Teaming in the Age of LLMs," arXiv preprint arXiv:2506.13434, 2025.
[18] M. Chen, W. Li, and F. Sun, "Benchmarking Jailbreak Attacks on LLMs," arXiv preprint arXiv:2402.12345, 2024.
[19] J. He, S. Wei, and D. Zhou, "PromptGuard: Defending Against Jailbreaks in Instruction-Tuned LLMs," arXiv preprint arXiv:2403.01420, 2024.
[20] A. Greshake Tzovaras, P. Vielmetter, M. Birhane, and E. Bender, "Prompt Injection Attacks: Taxonomy and Defense," arXiv preprint arXiv:2302.12173, 2023.
[21] N. Papernot, P. McDaniel, and I. Goodfellow, "Practical Black-Box Attacks against Machine Learning," in Proc. ACM Asia Conference on Computer and Communications Security (AsiaCCS), pp. 506-519, 2017.

.