Witty Gerbil Chrome Extension: A Browser-Based Framework for Comprehensive AI System Evaluation

Samuel Cameron
Independent Researcher
United States
samuelrcameron@gmail.com

ABSTRACT

Many AI red-teaming frameworks target Large Language Models (LLMs) and other generative systems via direct access or through Application Programmable Interfaces (API)s. These approaches do not always reflect the complexities of real-world deployments. Production AI applications often incorporate content moderation, guardrails, user interface constraints, and other filtering mechanisms, which can alter both user inputs and system outputs. To capture the full-range AI system flow, we present *Witty Gerbil Chrome Extension*, a browser-based testing solution coupled with a Python orchestration back-end. By automating AI interactions directly in the browser, our framework preserves production safeguards and transforms, providing a more realistic picture of overall system risk. This paper outlines the architecture, operation modes, and limitations of this extension, emphasizing the importance of holistic AI evaluations that include user-facing layers in addition to core model testing.

CCS CONCEPTS

• **Software and its engineering** → **Software creation and management**; • **Security and privacy** → *Domain-specific security and privacy architectures*; **Software security engineering**; • **Computing methodologies** → *Natural language generation*; **Discourse, dialogue and pragmatics**; • **Human-centered computing** → **HCI design and evaluation methods**.

KEYWORDS

AI Red-Teaming, Responsible AI, Browser-Based Testing, LLM Security, Chrome Extension

ACM Reference Format:
Samuel Cameron. 2025. Witty Gerbil Chrome Extension: A Browser-Based Framework for Comprehensive AI System Evaluation. In *Companion Proceedings of the ACM Web Conference 2025 (WWW Companion '25), April 28-May 2, 2025, Sydney, NSW, Australia.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3701716.3717741

1 INTRODUCTION

Red-teaming for Large Language Models (LLMs) and other generative AI systems is crucial to evaluate vulnerabilities such as bias, prompt injection, and the generation of malicious content [1]. However, traditional testing that focuses solely on the raw model or its API endpoints overlooks the real-world controls applied in production. These include content moderation filters, specialized guardrails, and user interface constraints [2].

To address this gap, we introduce the *Witty Gerbil Chrome Extension*, which orchestrates AI interactions directly in the browser. By automating the user journey, this extension preserves the front-end transformations, system prompts, and other production-layer protections that can alter both inputs and outputs. A Python back-end supports the extension by providing dynamic prompts, session control, and data logging. This architecture effectively captures the *full* user experience, reflecting a more realistic risk profile than can be achieved through model- or API-only assessments.

2 BACKGROUND AND RELATED WORK

Existing AI red-teaming solutions typically focus on direct model calls or well-defined APIs. Frameworks such as *Garak* [3], *PyRIT* [4], and *ARTKit* [5] automate adversarial probing without front-end transformations. To move closer to realistic usage scenarios, some systems integrate with web traffic proxies such as Burp Suite (*PyRIT-Ship* [6]), intercepting requests as they traverse the network. Although valuable, these setups can be cumbersome and do not mimic genuine user interaction.

3 HOLISTIC APPROACH TO AI TESTING

To illustrate where this tool adds value, we now place it in the broader context of AI Safety and Security. Holistic AI security evaluations are most effective when embedded in broader Responsible AI (RAI) initiatives that incorporate cross-business unit governance and oversight, secure AI application development, ongoing monitoring of the AI systems in production, evaluation, and testing of the models and associated applications, and more. This work focuses primarily on the testing and evaluation portion of the program.

3.1 Evaluation Phases

We recommend a three-phase evaluation approach:

(1) **Model Benchmarking:** Evaluate model performance, ethics, and safety concerns using curated datasets. This is typically done directly on the raw model or via minimal API wrappers, with much of the process automated as part of a broader model registry and pipeline.

(2) **Red-Teaming:** Investigate and probe the entire AI system, including any guardrails, content moderation, or prompt engineering layers, to uncover vulnerabilities such as prompt injection, jailbreaks, or system prompt manipulations. This phase also considers advanced threats such as embedding inversion [7, 8] and data poisoning [9], broadening the scope of potential security concerns. Much of this phase is automated or carried out using specialized tools.
(3) **Expert Manual Testing:** Conduct in-depth, manual evaluations by expert analysts to detect subtle or context-specific vulnerabilities that automated tools may overlook. Insights from the first two phases guide more targeted and sophisticated attacks during this final stage.

4 ARCHITECTURE AND IMPLEMENTATION

Figure 1 shows the overall design: a Chrome extension injected into the webpage for DOM manipulation, plus a Python back-end for orchestration.

4.1 Chrome Extension (Front-end)

DOM Monitoring: Watches for changes in the chat interface, capturing AI output and user prompts.
Interaction Simulation: Automatically inserts new prompts and sends to the assistant, mimicking an actual user.
UI Overlay: Presents configuration options (e.g., "max turns") and allows testers to upload datasets of adversarial prompts.

4.2 Python back-end

Session Management: Tracks conversation turns and terminates runs after a specified limit.
Dynamic Prompt Generation: Adapts new prompts based on the conversation context, pursuing an objective (e.g., eliciting policy-violating content).
Data Logging: Stores user queries, system responses, and metadata for later review or auditing.

5 EVALUATION MODES

5.1 Guided Red-Teaming

Testers receive suggested prompts designed to push the system toward an adversarial objective. Each suggestion can be edited before sending, balancing automation with human oversight.

5.2 Fully Automated Conversations

Allows long-running interactions without human intervention. The extension repeatedly sends newly generated prompts, captures replies, and updates the conversation state until a maximum turn limit is reached.

5.3 Automated Benchmarking

A fully automated mode that iterates over a dataset provided by the user. A dataset of the resulting conversation is available for download afterward.

6 LIMITATIONS

- **Browser Lock-In:** Only supports Chromium-based browsers at present.
- **Complex Web Interfaces:** Highly dynamic pages may need manual CSS or JavaScript selectors.
- **Performance:** Browser automation incurs more overhead than direct API calls which may result in longer benchmark run times.
- **Single Generator Model:** Currently relies on OpenAI GPT-4o for prompt generation.
- **No Early Stopping:** If the adversarial goal is met early, the system still proceeds until the specified turn count is reached.

7 FUTURE WORK

Planned enhancements include Firefox and Safari compatibility, multi-model support in the Python back-end, intelligent stopping criteria (e.g., halting once an attack objective is fulfilled), and integrated analytics for real-time scoring of red-team success.

8 CONCLUSION

The *Witty Gerbil Chrome Extension* represents a significant advancement in AI system evaluation by enabling testing that more accurately reflects real-world usage conditions and potential risks. By operating directly within the browser environment that end users experience, it captures the full stack of content filtering, guardrails, and user interface elements that may influence AI system behavior. This approach enables security teams to uncover the true risk profile of deployed AI systems by evaluating not just the core model, but the entire user-facing system including front-end protections, rate limiting, content filtering, and other production safeguards.

The extension's architecture, combining a lightweight browser-based front-end with a powerful Python orchestration back-end, provides a flexible framework for both manual and automated testing scenarios. This design enables systematic discovery of potential vulnerabilities while maintaining the authentic context of actual user interactions. Beyond its immediate applications in security testing, Witty Gerbil Chrome Extension establishes a foundation for developing more sophisticated automated testing approaches that can comprehensively assess AI systems in their production environments, advancing the field of AI red-teaming and security evaluation.

AVAILABILITY

The Chrome extension and Python orchestration script are available at:

https://github.com/Witty-Gerbil/witty_chrome_extension

The repository includes instructions for installing the extension, running the Python script, and configuring both automated and manual tests. Contributions are welcomed and encouraged. This work is released under the MIT License (see the LICENSE file in the repository for details).

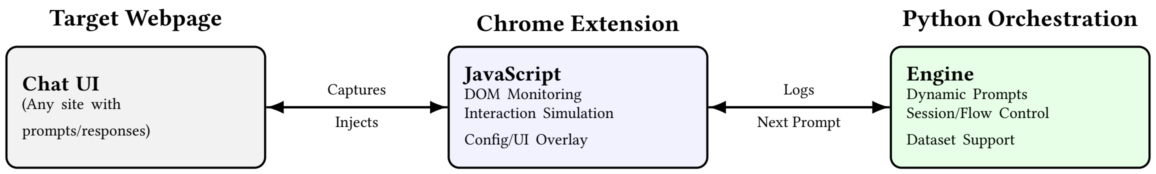

Figure 1: Architecture of the Witty Gerbil Extension: JavaScript injection for capturing and generating user-like input, with Python orchestration managing session flow and data logging.

ACKNOWLEDGMENTS

We acknowledge prior work on AI red-teaming solutions, particularly Garak [3], PyRIT [4], and ARTKit [5], which paved the way for AI red-teaming as a whole.

REFERENCES

[1] Raney, A.; Bendelac, S.; Manville, K.; Tan, M.; Yamaguchi, K. An AI red team playbook. In *Assurance and Security for AI-enabled Systems*, editors: Harguess, J.D., Bastian, N.D., and Pace, T.L., volume 13054, pages 130540B, SPIE, 2024. https://doi.org/10.1117/12.3021906.

[2] Ayyamperumal, S.G. and Ge, L. Current State of LLM Risks and AI Guardrails. *arXiv preprint arXiv:2406.12934*, 2024. https://arxiv.org/abs/2406.12934.

[3] Derczynski, L. et al. garak: A Framework for Security Probing Large Language Models. *arXiv preprint arXiv:2406.11036*, 2024. https://arxiv.org/abs/2406.11036.

[4] Lopez Munoz, G.D. et al. PyRIT: A Framework for Security Risk Identification and Red Teaming in Generative AI Systems. *arXiv preprint arXiv:2410.02828*, 2024. https://arxiv.org/abs/2410.02828.

[5] ARTKit. https://github.com/BCG-X-Official/artkit. Accessed 2025.

[6] Microsoft Open Source et al. PyRIT-Ship. https://github.com/microsoft/PyRIT-Ship. Accessed 2025.

[7] Morris, J.X.; Kuleshov, V.; Shmatikov, V.; Rush, A.M. Text Embeddings Reveal (Almost) As Much As Text. *arXiv preprint arXiv:2310.06816*, 2023. https://arxiv.org/abs/2310.06816.

[8] Morris, J.X.; Zhao, W.; Chiu, J.T.; Shmatikov, V.; Rush, A.M. Language Model Inversion. *arXiv preprint arXiv:2311.13647*, 2023. https://arxiv.org/abs/2311.13647.

[9] Chen, Z.; Xiang, Z.; Xiao, C.; Song, D.; Li, B. AgentPoison: Red-teaming LLM Agents via Poisoning Memory or Knowledge Bases. *arXiv preprint arXiv:2407.12784*, 2024. https://arxiv.org/abs/2407.12784.