BLACKICE: A Containerized Red Teaming Toolkit for AI Security Testing

Caelin Kaplan, Alexander Warnecke, and Neil Archibald

AI Red Team, Databricks

`first.last@databricks.com`

Abstract—AI models are being increasingly integrated into real-world systems, raising significant concerns about their safety and security. Consequently, AI red teaming has become essential for organizations to proactively identify and address vulnerabilities before they can be exploited by adversaries. While numerous AI red teaming tools currently exist, practitioners face challenges in selecting the most appropriate tools from a rapidly expanding landscape, as well as managing complex and frequently conflicting software dependencies across isolated projects. Given these challenges and the relatively small number of organizations with dedicated AI red teams, there is a strong need to lower barriers to entry and establish a standardized environment that simplifies the setup and execution of comprehensive AI model assessments.

Inspired by Kali Linux's role in traditional penetration testing, we introduce BLACKICE, an open-source containerized toolkit designed for red teaming Large Language Models (LLMs) and classical machine learning (ML) models. BLACKICE provides a reproducible, version-pinned Docker image that bundles 14 carefully selected open-source tools for Responsible AI and Security testing, all accessible via a unified command-line interface. With this setup, initiating red team assessments is as straightforward as launching a container, either locally or using a cloud platform. Additionally, the image's modular architecture facilitates community-driven extensions, allowing users to easily adapt or expand the toolkit as new threats emerge. In this paper, we describe the architecture of the container image, the process used for selecting tools, and the types of evaluations they support.

I. INTRODUCTION

As AI systems become increasingly integrated into critical workflows and consumer products, AI red teaming has emerged as an essential practice to identify and mitigate vulnerabilities. These vulnerabilities can manifest at the model level, such as jailbreak attacks [1–3] that circumvent safety mechanisms, or at the system level, where adversaries exploit deployment contexts, for example, indirect prompt injections embedded within emails processed by AI assistants [4, 5]. Despite the growing recognition of AI red teaming's importance, effectively conducting such assessments remains challenging, as existing tools each have their own unique setup procedures and typically require separate runtime environments due to conflicting dependencies. While possible, managing many independent environments is often time-consuming, error-prone, and difficult to scale, particularly across diverse platforms or cloud-based infrastructure. Moreover, the absence of a single, standardized environment complicates the reproducibility of evaluations across teams and organizations.

In traditional penetration testing, similar challenges have been effectively addressed through the widespread adoption of Kali Linux, a Linux distribution preconfigured with a comprehensive suite of security tools. Kali Linux has become a standard in the security community by simplifying environment setup and bundling essential utilities, enabling practitioners to focus on vulnerability assessments rather than managing complex software configurations and dependencies. Inspired by Kali Linux's success, we introduce BLACKICE, an open-source toolkit that consolidates leading AI red teaming tools into a unified, reproducible, and portable container image[1]; the Docker build repository is available at https://github.com/databricks/containers. The modular architecture and carefully curated tool selection of BLACKICE enable both novice and expert practitioners to effectively perform AI red team assessments, while also facilitating straightforward, community-driven extensions. An overview of the included tools is provided in Table I, and the corresponding Docker build process is illustrated in Figure 1.

Tool	Organization	Stars	Type	Source
Eval Harness [6]	Eleuther AI	10,300	●	GitHub
Promptfoo [7]	Promptfoo	8,600	●	npm
CleverHans [8]	CleverHans Lab	6,400	○	GitHub
Garak [9]	NVIDIA	6,100	●	PyPI
ART [10]	IBM	5,600	○	PyPI
Giskard [11]	Giskard	4,900	◑	PyPI
CyberSecEval [12]	Meta	3,800	●	GitHub
PyRIT [13]	Microsoft	2,900	○	PyPI
EasyEdit [14]	ZJUNLP	2,600	○	GitHub
Promptmap [15]	-	1,000	●	GitHub
FuzzyAI [16]	CyberArk	800	●	GitHub
Fickling [17]	Trail of Bits	560	◑	PyPI
Rigging [18]	Dreadnode	380	○	PyPI
Judges [19]	Quotient AI	290	◑	PyPI

TABLE I
OVERVIEW OF BLACKICE TOOLS, INCLUDING ORGANIZATION (IF APPLICABLE), GITHUB STARS (NEAREST HUNDRED), TOOL TYPE (STATIC ● OR DYNAMIC ○), AND DISTRIBUTION SOURCE.

II. DOCKER IMAGE ARCHITECTURE

AI red teaming tools vary significantly in their usage, dependencies, and distribution methods, posing unique challenges

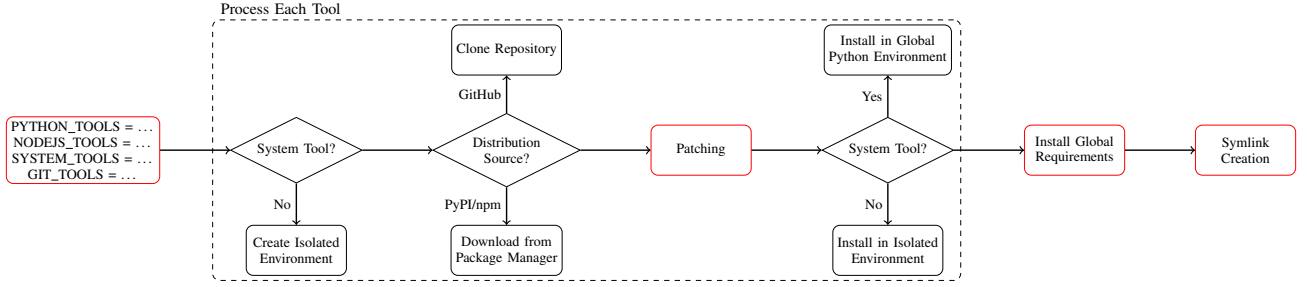[1] https://hub.docker.com/r/databricksruntime/blackice

Fig. 1. Flowchart illustrating the Docker build process for the BLACKICE container image. Boxes with red borders highlight the steps where users directly interact with the build process to integrate new tools.

when combining them effectively into a unified workflow. While managing separate runtime environments for each tool during an assessment is possible, this approach has three major drawbacks: (1) it significantly increases operational complexity due to repetitive configuration and frequent environment switching, (2) it restricts interoperability between tools by isolating their environments, and (3) it creates challenges for cloud-based managed notebook services, which typically provide only a single Python interpreter per kernel.

a) Static vs. Dynamic Tools: To enable seamless use of all tools within the same container environment, we organize them into two distinct categories based on usage:

- **Static Tools:** Evaluate AI models using straightforward command-line interfaces, requiring minimal programming knowledge. While easy to use, static tools offer limited flexibility for customization and integration.
- **Dynamic Tools:** Offer similar evaluation capabilities but additionally support advanced Python-based customization, enabling users to write code for custom attacks.

Within the container image, static tools are installed in individual, isolated Python virtual environments (or separate Node.js projects), each with its own dependencies, and can be executed immediately via symlinks to custom CLI scripts. Alternatively, dynamic tools are installed directly into the global Python environment, with dependency conflicts managed through a `global_requirements.txt` file.

b) Automated Installation: The Docker build process automates the installation and configuration of each tool according to its category and distribution source. At the top of the Dockerfile, users specify tool names along with their pinned versions (defined either by package version or Git commit hash) in predefined configuration lists. By default, tools listed under `PYTHON_TOOLS` are installed from PyPI into isolated Python virtual environments. Users can override this default behavior by adding tools to `SYSTEM_TOOLS`, which installs them into the global Python environment, or to `GIT_TOOLS`, which clones and installs them directly from their Git repositories. `NODEJS_TOOLS` are always installed via npm into separate project directories.

c) Customization via Patching: In certain scenarios, users may need to extend or modify the functionality of existing tools. For example, an assessment might require

implementing a custom client to query a model endpoint not supported by default, or enhancing a tool like CyberSecEval [12] to enable providing the LLM with a security-oriented system prompt during benchmark evaluations. To simplify these modifications, the BLACKICE build process includes a structured patching mechanism. Users can add source-level patches as simple `.diff` files or custom Python modules to a designated directory (`/patches/<tool>`), which are automatically applied before the tool is installed.

d) Community Extensibility: The modular build design enables users to easily add new tools by appending them to the configuration lists defined at the top of the Dockerfile (e.g., `PYTHON_TOOLS`, `SYSTEM_TOOLS`, `GIT_TOOLS`). After doing so, static tools are integrated by creating a minimal wrapper script in the `/cli_scripts/<tool>` directory that includes a shebang pointing to the tool's isolated Python interpreter and invokes its run command with the provided arguments. Custom static tools can be added in the same way by placing a self-contained CLI script in `/cli_scripts/<tool>`. For example, we include `biasforge`, a custom CLI tool that evaluates bias in language models by generating synthetic prompts based on a specified evaluation objective, querying a target model, and assessing the outputs using a structured judgment schema. Dynamic tools, in contrast, may require updating the `global_requirements.txt` file to resolve any newly introduced dependency conflicts. Once the image is rebuilt, all new tools become immediately available via the CLI, with no additional setup required.

III. TOOL SELECTION AND COVERAGE

To ensure comprehensive coverage, we first identified critical threat domains based on technical reports and red teaming guidelines published by industry-leading organizations [20–26]. These reports provided valuable insights into common techniques and best practices used in AI red team assessments. Second, we examined *system cards* (e.g., [27–29]), structured documents transparently detailing organizations' internal red teaming processes and findings. Finally, we reviewed outcomes from *red teaming competitions* (e.g., [30–32]), where participants attack LLMs in controlled environments designed to reflect real-world adversarial scenarios, similar to "capture the flag" challenges. Guided by this analysis and discussions

BlackIce Capability	MITRE ATLAS	Databricks AI Security Framework (DASF)
Prompt-injection and jailbreak testing of LLMs	AML.T0051 LLM Prompt Injection; AML.T0054 LLM Jailbreak; AML.T0056 LLM Meta Prompt Extraction	9.1 Prompt inject; 9.12 LLM jailbreak
Indirect prompt injection via untrusted content (e.g., RAG/email)	AML.T0051 LLM Prompt Injection [Indirect]	9.9 Input resource control
LLM data leakage testing	AML.T0057 LLM Data Leakage	10.6 Sensitive data output from a model
Hallucination stress-testing and detection	AML.T0062 Discover LLM Hallucinations	9.8 LLM hallucinations
Adversarial example generation and evasion testing (CV/ML)	AML.T0015 Evade ML Model; AML.T0043 Craft Adversarial Data	10.5 Black box attacks
Supply-chain and artifact safety scanning (e.g., malicious pickles)	AML.T0010 AI Supply Chain Compromise; AML.T0011 Unsafe AI Artifacts	7.3 ML supply chain vulnerabilities

TABLE II
MAPPING OF BLACKICE CAPABILITIES TO MITRE ATLAS AND THE DATABRICKS AI SECURITY FRAMEWORK (DASF).

with industry AI security practitioners, we selected widely adopted open-source tools, including those developed by established AI security teams (e.g., [9, 12, 13]), AI security startups (e.g., [7, 11]), academic researchers [8, 14], and independent developers [15]. The selected tools were then evaluated for their collective coverage of major AI security risk categories by mapping the capabilities of BLACKICE to MITRE ATLAS [33] and the Databricks AI Security Framework (DASF) [34]. Table II highlights that BLACKICE provides comprehensive coverage across domains such as prompt injection, data leakage, hallucination detection, and supply-chain integrity.

a) Static Tool Capabilities: Although there is a certain degree of functional overlap among static tools, each provides distinct testing capabilities and excels in particular areas. Evaluation Harness [6] offers over 60 academic benchmarks targeting skills such as reasoning, reading comprehension, and mathematics; Promptfoo [7] generates context-specific attacks and can evaluate vulnerabilities according to frameworks such as the OWASP Top 10 for LLMs [35]; Garak [9] leverages predefined prompt sets to assess model-level vulnerabilities including bias, toxicity, jailbreaks, and hallucinations; Giskard [11] provides an extensible automated testing platform supporting integration with multiple LLM providers;[2] CyberSecEval [12] targets security-specific issues such as unsafe code generation and malware synthesis through predefined corpuses; and Fickling [17] scans Python pickle files for malicious payloads.

b) Dynamic Tool Capabilities: For advanced users, dynamic tools enable conducting more customized assessments: PyRIT [13] is a Python framework allowing users to configure red teaming workflows using components such as Prompt Targets, Executors, Scorers, and Converters, facili-

tating comprehensive and easily extendable testing scenarios; EasyEdit [14] enables direct manipulation of a model's internal representations, allowing users to alter stored knowledge; and Rigging [18] is a framework for orchestrating type-safe LLM workflows, ideal for automating structured tasks in AI security assessments.[3]

IV. CONCLUSION

BLACKICE is released as an open-source Docker image along with its build repository, providing a standardized execution environment that greatly reduces the effort required to conduct comprehensive AI red teaming assessments. The image includes a curated selection of static and dynamic tools, collectively addressing a broad range of vulnerabilities in both LLMs and classical ML models. Through its modular design and straightforward extensibility, we aim for BLACKICE to foster community-driven enhancements and encourage responsible AI deployment practices.

REFERENCES

[1] A. Zou, Z. Wang, J. Z. Kolter, et al., *Universal and transferable adversarial attacks on aligned language models*, 2023. arXiv: 2307.15043.

[2] Y. Huang, S. Gupta, M. Xia, et al., "Catastrophic jailbreak of open-source llms via exploiting generation," in *The Twelfth International Conference on Learning Representations, ICLR*, 2024.

[3] A. Wei, N. Haghtalab, and J. Steinhardt, "Jailbroken: How does llm safety training fail?" In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2023.

[4] K. Greshake, S. Abdelnabi, S. Mishra, et al., "Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection," in *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, 2023.

[2]Although Giskard did not originally provide a CLI, we implemented one to classify it as a static tool, given its strong static evaluation capabilities.

[3]Technically, EasyEdit, Cleverhans, and ART each qualify as dynamic tools under our definition, meaning they should be installed in the system-level Python environment. However, due to significant dependency conflicts between their legacy requirements and modern LLM evaluation tools (e.g., PyRIT), we instead treat them as static tools and install each in its own Python virtual environment.

[5] Y. Liu, G. Deng, Y. Li, et al., *Prompt injection attack against llm-integrated applications*, 2023. arXiv: 2306.05499.

[6] L. Gao, J. Tow, B. Abbasi, et al., *The language model evaluation harness*, version v0.4.3, 2024. DOI: 10.5281/zenodo.12608602 [Online]. Available: https://zenodo.org/records/12608602

[7] promptfoo, *promptfoo*, Accessed: 2025-06-05, 2023. [Online]. Available: https://github.com/promptfoo/promptfoo

[8] C. Lab, *CleverHans*, Accessed: 2025-06-05, 2018. [Online]. Available: https://github.com/cleverhans-lab/cleverhans

[9] L. Derczynski, E. Galinkin, J. Martin, et al., *Garak: A framework for security probing large language models*, 2024. arXiv: 2406.11036.

[10] M.-I. Nicolae, M. Sinn, M. N. Tran, et al., *Adversarial robustness toolbox v1.2.0*. arXiv: 1807.01069.

[11] Giskard, *Giskard: The Evaluation & Testing framework for AI systems*, Accessed: 2025-06-05, 2024. [Online]. Available: https://github.com/Giskard-AI/giskard

[12] M. Bhatt, S. Chennabasappa, C. Nikolaidis, et al., *Purple llama cyberseceval: A secure coding benchmark for language models*, 2023. arXiv: 2312.04724.

[13] G. D. L. Munoz, A. J. Minnich, R. Lutz, et al., "Pyrit: A framework for security risk identification and red teaming in generative ai system," in *CAMLIS*, 2024.

[14] P. Wang, N. Zhang, B. Tian, et al., *Easyedit: An easy-to-use knowledge editing framework for large language models*, 2024. arXiv: 2308.07269.

[15] U. Sen, *promptmap*, Accessed: 2025-06-05, 2023. [Online]. Available: https://github.com/utkusen/promptmap

[16] CyberArk, *fuzzyai*, Accessed: 2025-06-05, 2024. [Online]. Available: https://github.com/cyberark/FuzzyAI

[17] Trail of Bits, *fickling*, Accessed: 2025-06-05, 2022. [Online]. Available: https://github.com/trailofbits/fickling

[18] Dreadnode, *rigging*, Accessed: 2025-06-05, 2024. [Online]. Available: https://github.com/dreadnode/rigging

[19] quotient-ai, *judges*, Accessed: 2025-06-05, 2024. [Online]. Available: https://github.com/quotient-ai/judges

[20] D. Fabian and J. Crisp, *Why red teams play a central role in helping organizations secure ai systems*, Accessed: 2025-06-05, 2023. [Online]. Available: https://services.google.com/fh/files/blogs/google_ai_red_team_digital_final.pdf

[21] B. Bullwinkel, A. J. Minnich, S. Chawla, et al., *Lessons from red teaming 100 generative ai products*, 2025. arXiv: 2501.07238.

[22] A. Rawat, S. Schoepf, and S. C. Giulio Zizzo Seraphina Goldfarb-Tarrant, *Attack atlas: A practitioner's perspective on challenges and pitfalls in red teaming genai*, 2024. arXiv: 2409.15398.

[23] E. Antone, A. Chang, A. Fordyce, et al., *The state of ai security*, Accessed: 2025-06-05, 2025. [Online]. Available: https://learn-cloudsecurity.cisco.com/state-of-ai-security-report

[24] B. A. Hamilton, *Artificial intelligence security*, Accessed: 2025-06-05, 2024. [Online]. Available: https://www.boozallen.com/expertise/artificial-intelligence/ai-solutions/adversarial-artificial-intelligence.html

[25] D. Ganguli, L. Lovitt, and J. Kernion, *Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned*, 2022. arXiv: 2209.07858.

[26] Anthropic, *Challenges in red teaming AI systems*, Accessed: 2025-06-05, 2024. [Online]. Available: https://www.anthropic.com/news/challenges-in-red-teaming-ai-systems

[27] Google, *Gemini 2.5 pro preview model card*, Accessed: 2025-06-05, 2025. [Online]. Available: https://storage.googleapis.com/model-cards/documents/gemini-2.5-pro-preview.pdf

[28] Anthropic, *System card: Claude opus 4 & claude sonnet 4*, Accessed: 2025-06-05, 2025. [Online]. Available: https://www-cdn.anthropic.com/6be99a52cb68eb70eb9572b4cafad13df32ed995.pdf

[29] OpenAI, *Openai o3 and o4-mini system card*, Accessed: 2025-06-05, 2025. [Online]. Available: https://cdn.openai.com/pdf/2221c875-02dc-4789-800b-e7758f3722c1/o3-and-o4-mini-system-card.pdf

[30] Microsoft, *AI-Red-Teaming-Playground-Labs*, Accessed: 2025-06-05. [Online]. Available: https://github.com/microsoft/AI-Red-Teaming-Playground-Labs

[31] V. Storchan, R. Kumar, R. Chowdhury, et al., *Generative ai red teaming challenge: Transparency report*, Accessed: 2025-06-05, 2024. [Online]. Available: https://drive.google.com/file/d/1JqpbIP6DNomkb32umLoiEPombK2-0Rc-/view

[32] Dreadnode, *The dreadnode crucible platform*, Accessed: 2025-06-05. [Online]. Available: https://dreadnode.io/crucible

[33] MITRE, *Atlas matrix*, Accessed: 2025-06-05, 2025. [Online]. Available: https://atlas.mitre.org/matrices/ATLAS

[34] Databricks, *Databricks ai security framework (dasf) 2.0*, Accessed: 2025-06-05, 2025. [Online]. Available: https://www.databricks.com/resources/whitepaper/databricks-ai-security-framework-dasf

[35] OWASP, *Owasp top 10 for large language model applications*, Accessed: 2025-06-05, 2025. [Online]. Available: https://owasp.org/www-project-top-10-for-large-language-model-applications/