PyRIT

Democratizing AI Red Teaming Through Open-Source Tooling

PyRIT: Democratizing AI Red Teaming Through Open-Source Tooling

Roman Lutz **Eugenia Kim** **Richard Lundeen**
Nina Chikanov **Adrian Gavrila** **Bolor-Erdene Jagdagdorj**
Varun Joginpalli **Toby Kohlenberg** **Behnam Ousat**
Bashir Partovi **Spencer Schoenberg** **Justin Song**
Victor Valbuena **Hannah Westra** **Pete Bryan**
Blake Bullwinkel **Shiven Chawla** **Frédéric Dubut**
Joris de Gruyter **Whitney Maxwell** **Amanda Minnich**
Shujaat Mirza **Cristian Ovadiuc** **Teddy Phillips**
Martin Pouliot **Katherine Pratt** **Saphir Qi**
Giorgio Severi **Elizabeth Mori Tornheim** **Csenge Varga**
Sam Vaughan **Victoria Westerhoff** **Ram Shankar Siva Kumar**
Mark Russinovich **Yonatan Zunger**

Microsoft
AIREDTEAM@MICROSOFT.COM

Abstract

Generative AI systems are being deployed at unprecedented scale, but systematic vulnerability identification remains constrained by a shortage of specialized expertise. We present PyRIT (Python Risk Identification Tool for generative AI), an open-source, extensible framework designed to democratize AI red teaming. PyRIT supports both fully automated and human-led red teaming through a composable, modular architecture of targets, converters, scorers, executors, techniques, and scenarios. PyRIT is designed to interact flexibly with AI systems across providers, interfaces, modalities, and deployment patterns, from raw model APIs to deployed applications and end-user UIs. It supports both single-turn and multi-turn attack strategies, including Crescendo and Tree of Attacks with Pruning. As the first AI red teaming tool with multi-modal support by design, PyRIT covers text, image, audio, video, and file blobs (such as PDFs and other documents), as well as combinations thereof. Practitioners can consume PyRIT through three complementary surfaces: a Python SDK (*Framework*) for programmatic use in notebooks and custom code, a command-line *Scanner* for automated, repeatable evaluation campaigns suited to CI/CD, and a browser-based *GUI* (CoPyRIT) for collaborative, human-led red teaming. Since its initial release, PyRIT has evolved from a research prototype into a production-grade tool used to red team hundreds of generative AI products and services at Microsoft and beyond, and has been integrated into Azure AI Foundry as a built-in AI red teaming agent. PyRIT provides a shared, open platform for practitioners to contribute new attack techniques, converters, and scoring methodologies, allowing every user to benefit from the community's latest discoveries. PyRIT aims to make rigorous AI red teaming accessible to a broad community of practitioners, researchers, and organizations.

Keywords: AI red teaming, generative AI, AI security, AI safety, open source, Python

1 Introduction

The rapid proliferation of generative AI systems, spanning large language models (LLMs) [11], multi-modal foundation models, and increasingly agentic AI applications, has created an urgent need for systematic methods to identify security vulnerabilities and responsible AI (RAI) failures before deployment [30, 77, 104]. AI red teaming, the practice of adversarially probing AI systems to discover failure modes, has emerged as a critical component of responsible AI development [18, 32, 80]. However, AI red teaming today faces a fundamental democratization problem [31]: the expertise required to conduct rigorous red teaming exercises is concentrated among a small number of specialists, while the number of AI systems requiring evaluation grows rapidly. This is not merely a shortage of specialists, as many serious harms are best surfaced by domain experts and members of affected communities, who are rarely red teamers themselves. Public initiatives such as the DEF CON Generative Red Team have begun to broaden participation [5], but such events remain episodic and lack the shared infrastructure needed to accumulate and transfer knowledge across engagements.

Traditional security red teaming has well-established tooling, methodologies, and a large practitioner community built over decades. AI red teaming, by contrast, is a nascent discipline that must simultaneously address classical security concerns (prompt injection, data exfiltration, unauthorized actions) and novel responsible AI risks (generation of harmful content, stereotyping, hallucination of dangerous information). The probabilistic nature of generative AI outputs, the multi-modal attack surface, and the need for multi-turn adversarial interactions further compound the challenge. Manual red teaming campaigns can take weeks to plan and execute, and their configurations and results are often difficult to reconstruct or compare across systems [18]. Compounding this technical complexity is a deeper structural issue: many of the harms that matter most, spanning biosecurity, cybersecurity, and the broader landscape of trust and safety concerns, can only be reliably identified by people with deep domain expertise in those areas. A biosecurity expert is far better positioned than a generalist AI red teamer to judge whether a model's response provides actionable guidance on a biosecurity threat, and a member of an affected community is best placed to surface harms specific to that community. Yet most AI red teaming tooling today implicitly assumes the operator is already an AI red teaming specialist, leaving these domain experts on the sidelines precisely when their judgment is most valuable.

In this paper, we present PyRIT (Python Risk Identification Tool for generative AI), an open-source framework designed to address this democratization gap. Its shared platform allows practitioners to incorporate new attack discoveries, evaluation techniques, and harm taxonomies so that users can benefit from the community's collective knowledge. This shared-platform effect cuts in two directions. When a centralized AI red team, such as Microsoft's AI Red Team (AIRT), discovers a new vulnerability or attack technique during an engagement, it can capture that finding in a reusable PyRIT component (a converter, scorer, or attack technique) for use in subsequent PyRIT evaluations, both inside the organization and across the broader community. Conversely, researchers who contribute a new attack or evaluation method to PyRIT can test it across many models, modalities, and deployed applications. They can reuse PyRIT's evaluation infrastructure rather than rebuild it for each study or limit validation to a handful of systems. In this way, defensive findings and research contributions become reusable resources for other PyRIT users. First

announced publicly in early 2024 [73] and described as a research prototype [61], PyRIT has since evolved into a production-grade platform that has been integrated into Azure AI Foundry as a built-in AI red teaming agent. Our key contributions are:

- A modular, composable architecture for AI red teaming that treats components as interchangeable building blocks, enabling rapid assembly of diverse attack and evaluation strategies.
- Support for both fully automated and human-led red teaming within a single unified framework, bridging the gap between scalability and expert judgment.
- A model- and system-agnostic, multi-modal design (the first multi-modal AI red teaming tool) that enables testing of any AI system, from raw model endpoints to deployed applications and UIs, across modalities (text, image, audio, video, file blobs, and multi-modal combinations) and interaction paradigms (single-turn, multi-turn, agentic).
- A persistent memory system that enables campaign reconstruction, configuration comparison, auditability, and longitudinal analysis.
- Validation at scale: PyRIT has been used to send millions of prompts in red teaming hundreds of generative AI products and services [18], both within Microsoft and across the broader practitioner community.

2 Background and Motivation

AI red teaming differs from traditional cybersecurity red teaming in both its object of study and its evaluation criteria [44, 105]. It must account for probabilistic outputs and emergent behavior while examining both security vulnerabilities, such as prompt injection and data extraction, and responsible AI failures, such as harmful content and stereotyping [57]. It is also distinct from safety benchmarking: benchmarks measure performance on fixed datasets, whereas red teaming uses creative and adaptive probing to uncover failures not captured by a predetermined test set [31].

These properties create practical tooling requirements. A prompt may elicit harmful content on one attempt and a benign response on the next [113]; attack surfaces span modalities and multi-turn conversations; and agentic systems introduce actions such as browsing, code execution, and tool use [17, 71]. Practitioners also need to branch from partial conversations, refine hypotheses, and reuse promising artifacts. Effective tooling must therefore support repeatable automation at scale without sacrificing the interactive, exploratory workflow of human red teamers.

3 Related Work

The landscape of AI red teaming tools has grown rapidly alongside the adoption of generative AI. Garak [27] is an open-source LLM vulnerability scanner that focuses on probing language models with a catalog of known attack probes and detectors. Promptfoo [84] provides a command-line tool for evaluating LLM outputs against expected behaviors, with support for red teaming plugins. The OWASP project maintains resources related to LLM security, including the widely referenced OWASP Top 10 for LLM Applications [78]. Microsoft's Counterfit [70], an earlier tool for adversarial attacks on traditional ML models,

addressed a related but distinct problem space focused on evasion and poisoning attacks against classifiers. Mozilla's 0DIN (Zero Day Investigative Network) [76] takes a complementary approach, operating as a GenAI bug bounty program that incentivizes security researchers to discover and report vulnerabilities in large language models rather than providing an automated testing framework. PyRIT distinguishes itself from these tools by providing a composable architecture that treats red teaming components as interchangeable building blocks, supports multi-modal and multi-turn attacks natively, and integrates both automated and human-led workflows within a single framework.

4 Design for Democratization

PyRIT's design addresses three barriers to broader participation. First, composable components package attack and evaluation knowledge so practitioners can reuse existing techniques rather than implement each one from scratch. Second, the same components are available through automated, graphical, and programmatic interfaces, allowing domain experts and technical specialists to participate through workflows suited to them. Third, automation provides repeatable coverage at scale, while human red teamers retain responsibility for creative exploration, contextual judgment, and oversight [31].

These requirements lead to the following design principles:

- *Composability*: components can be mixed and matched to create new attack techniques and workflows without modifying existing code, so that every user benefits from the community's accumulated techniques.
- *Extensibility*: new targets, converters, scorers, and executors can be added by implementing simple abstract interfaces, lowering the barrier to contribution.
- *Model and system agnosticism*: the same workflow applies to any AI system a user can reach from Python, whether a hosted model endpoint, a deployed application, or a UI, regardless of provider and without vendor lock-in.
- *Multi-modal by design*: text, image, audio, video, file blobs, and their combinations are first-class citizens throughout the pipeline.
- *Multiple entry points*: the same underlying components are exposed through an automated *Scanner* for repeatable campaigns, a *GUI* for collaborative human-led evaluation, and a Python *Framework* for custom orchestration. This allows teams to choose the right level of automation for each task while preserving a shared component model across workflows.

The following sections describe how PyRIT's architecture realizes these principles.

5 Architecture

PyRIT is organized as a layered composition hierarchy (Figure 1). At the foundation are pluggable building blocks. Seeds and datasets supply objectives and prompt material; converters transform prompts and responses; targets send and receive messages from AI systems and supporting services; and scorers evaluate what happened. Memory, registries, common data models, and output utilities support every layer. The Python SDK, CLI, and backend workflows use a common registry to discover and configure targets, converters, scorers, techniques, scenarios, and initializers. Each component receives a strongly typed

identifier derived from its behavioral parameters and nested components. These identifiers make the exact configuration that produced a result queryable and comparable.

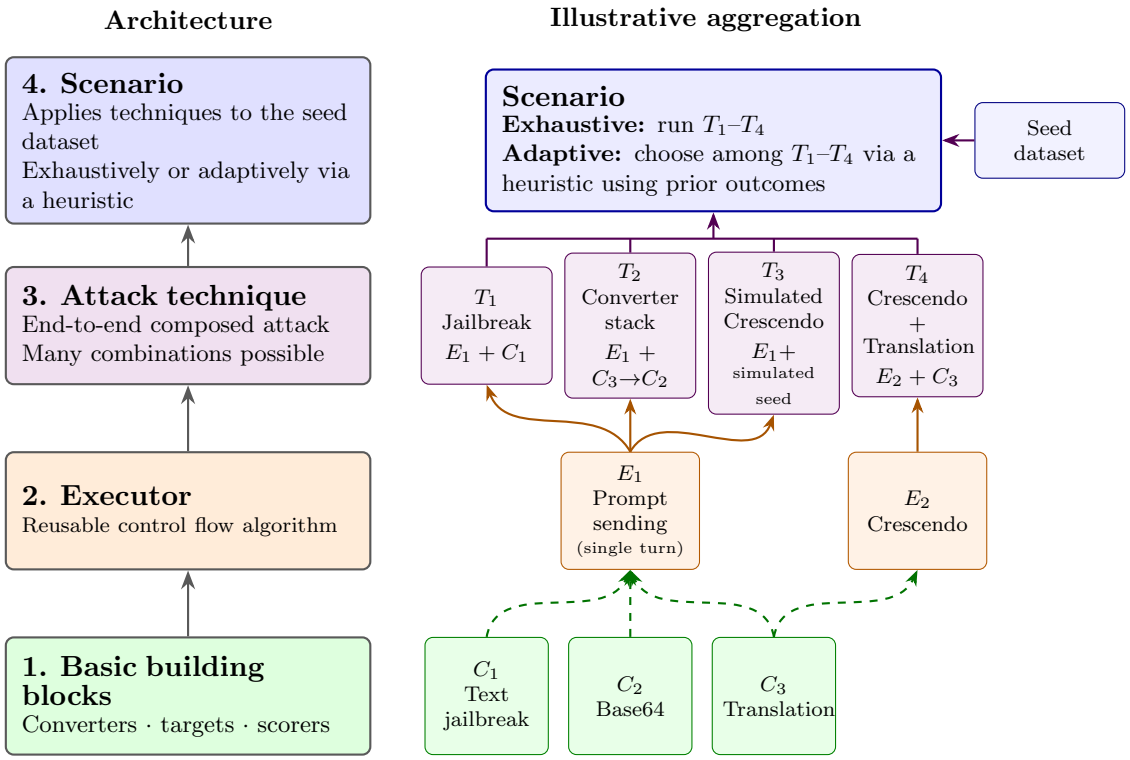

Figure 1: PyRIT's layered composition hierarchy and an illustrative aggregation. Two executors and three converters are recombined into four named techniques. Converters are optional: Prompt Sending is configured with a jailbreak template for T_1 and a Translation-to-Base64 converter stack for T_2; Simulated Crescendo combines Prompt Sending with a simulated-conversation seed but no converter; and the Crescendo executor is combined with Translation for T_4. A scenario applies the resulting technique set to seed datasets, either exhaustively or through an adaptive heuristic informed by prior outcomes. Dashed green arrows show converter composition into executors, and solid orange arrows show executor reuse across techniques.

One level up, *executors* compose these building blocks into reusable control flow algorithms. This layer sends messages to targets, applies converter pipelines when configured, manages conversation state, and branches on scorer results. An *attack technique* is a named, reusable, end-to-end method built around an executor. It combines an executor with optional converter stacks, templates or other technique-specific seeds, targets, scorers, and strategy parameters. Many combinations are possible: a technique can pair the Prompt Sending executor with a jailbreak converter or converter stack, combine it with a simulated-

conversation seed to create Simulated Crescendo, or pair a dedicated multi-turn executor such as Crescendo with Translation.

At the highest level, a *scenario* applies one or more selected techniques to seed datasets to form a coherent campaign. It may exhaustively execute all compatible technique and dataset pairings, or use an adaptive heuristic to choose techniques for each objective based on observed attack success while continuing to explore alternatives. In either case, the scenario manages parallelism, resiliency, persistence, and result aggregation across the campaign. The remainder of this section follows this hierarchy from the building blocks upward.

5.1 Targets

The *target* abstraction represents an endpoint with which PyRIT exchanges messages. The primary objective target is the AI system under test, while supporting targets may provide services such as storage or guardrail evaluation. All targets implement a common asynchronous interface that accepts and returns messages in supported modalities. This uniform interface is the foundation of PyRIT's model- and system-agnostic design: the same attack workflow can be directed at a hosted model endpoint, a custom web application, or other target systems, simply by swapping the target object.

In v1.0.0, targets share a single `PromptTarget` abstraction and declare their capabilities. These declarations specify supported input and output modality combinations and features such as multi-turn interaction, editable history, system prompts, structured output, and streaming audio. Executors inspect the declarations and adapt their behavior, avoiding a rigid class hierarchy that separates chat from non-chat targets.

A critical distinction in AI red teaming is between *model-level* and *system-level* evaluation [114]. Model-level red teaming probes the underlying AI model directly, testing its responses to adversarial inputs in isolation. System-level red teaming evaluates the complete deployed application end-to-end, as the user would experience it, including any guardrails, content filters, retrieval-augmented generation (RAG) pipelines, tool integrations, and other services involved in handling a request. This end-to-end perspective can surface issues that testing an isolated model endpoint would miss, since vulnerabilities may arise from the interactions between components rather than from the model alone. PyRIT supports both levels through its target abstraction.

For model-level red teaming, PyRIT provides targets built around the OpenAI API family: Chat Completions, Completions, Responses, image generation, video generation, text-to-speech, and Realtime. Each target can connect to OpenAI or another provider that exposes the corresponding OpenAI-compatible API; supported models, modalities, and features depend on the provider. This includes locally hosted models served through tools such as Ollama or vLLM where they implement the relevant API. The LiteLLM target instead routes through LiteLLM to more than 100 providers, including Anthropic, AWS Bedrock, Google Vertex AI, and Cohere, without requiring a separate proxy. For system-level red teaming, the appropriate target depends on what the deployed application exposes. When an API is available, practitioners can use a service-specific wrapper or the generic HTTP target. The generic target sends requests from user-defined templates, which can be written from API documentation or generated from a raw request captured with a tool such as Burp Suite. A specialized WebSocket target integrates with Microsoft 365 Copilot. When

no API is exposed and the system is only reachable through its UI, Playwright targets drive a real browser end-to-end. This range of target types means that PyRIT can test anything from a raw model endpoint to a web page that is consumed by an agent. Implementing a new target requires only a single asynchronous method, enabling practitioners to build custom integrations for proprietary systems, internal tools, or novel interaction paradigms. This extensibility is particularly important for system-level red teaming, where the diversity of deployed AI applications means that no fixed set of target types can cover all real-world configurations.

Table 1 lists representative built-in targets. They span direct model endpoints, multi-modal generation services, protocol-level and browser-based access to deployed systems, and specialized utilities, and new target types are added as new interaction patterns emerge.

Table 1: Representative built-in targets in PyRIT.

	Target	**Description**
OpenAI APIs	Chat Completions	Chat models; accepts text or image input depending on the model
	Completions	Text generation
	Responses	Tool (function) calling, structured output, and multi-modal input where supported
	Image generation	Text-to-image generation and image editing
	Video generation	Text-to-video and image-to-video generation (e.g., Sora)
	Text-to-speech	Speech synthesis from text
	Realtime	Bidirectional voice streaming with server-side voice-activity detection
	LiteLLM	Provider-routed chat across more than 100 model providers; supported modalities depend on the selected model
	Azure ML	Managed model endpoints (e.g., deployed open-weight models)
	Hugging Face	Local, in-process execution of a Transformers model loaded from a Hugging Face model ID or local path
	HTTP	Arbitrary HTTP requests from a user-defined template, including request blobs captured from tools such as Burp Suite
	Microsoft 365 Copilot	Microsoft 365 Copilot over its WebSocket protocol; accepts text and image input
	Playwright	Drives a real browser end-to-end for systems reachable only through their user interface
	Prompt Shield	Guardrail endpoint for jailbreak and prompt-injection detection
	Azure Blob Storage	Uploads payloads to blob storage for file-based and cross-prompt injection flows
	Round-robin	Distributes requests across multiple wrapped targets
	Gandalf	The Gandalf prompt-injection challenge, used for demonstration and training

5.2 Converters

Converters transform messages before they are sent to a target or after they are received from one. These transformations implement the diverse obfuscation and other strategies central to adversarial red teaming. Converters can be chained into pipelines, where each converter in the sequence transforms the output of the previous one, enabling the composition of

complex multi-stage transformations from simple building blocks. Figure 2 illustrates a concrete example of a chained converter pipeline with both request and response converters.

PyRIT includes over 80 converters organized across several categories: encoding-based converters (Base64, ROT13, Unicode homoglyphs, Morse code), obfuscation converters (character substitution, leetspeak, random capitalization), semantic converters (translation to other languages, tone shifting, persona adoption), multimedia converters (text-to-image, text-to-audio, text-to-video, image perturbation, and cross-modal transformations), and advanced attack converters that implement specific jailbreak techniques. For example, PyRIT includes converters inspired by CodeChameleon [64], which uses customizable encryption schemes; cipher-based evasion [117]; persuasion techniques [118]; mathematical encoding [12]; typographic flipping [60]; past-tense reformulation [8]; nested jailbreak prompts [28]; ANSI escape sequence injection [7]; Unicode tag hiding [85]; token smuggling [86]; prompt decomposition and reconstruction [54]; and Policy Puppetry [42].

New converters are added by implementing a single asynchronous method, making it straightforward for researchers to contribute novel transformation strategies. A complementary `TextJailbreakConverter` wraps an objective in a parameterized jailbreak template, allowing PyRIT to leverage curated template collections from the community such as JailbreakChat [6], the Arth Singh collection [98], and the Pliny collection [82]; the bundled set currently includes over 160 templates.

Table 2 lists all built-in converters, sorted by input and output modality. Most converters operate on text inputs and produce text outputs; converters that cross modality boundaries are listed separately.

Table 2: Built-in converters in PyRIT, grouped by input/output modality.

Converter	**Ref.**	**Description**
Audio → audio		
Audio echo		Mixes a delayed, attenuated copy to add echo
Audio frequency		Shifts frequency above human hearing by default
Audio speed		Changes playback speed without altering pitch
Audio volume		Scales audio amplitude by a set factor
Audio white noise		Adds white noise at a controlled amplitude
Audio → text		
Azure speech audio-to-text		Transcribes audio to text via Azure Speech
Image → image		
Add text to image		Overlays wrapped text onto an image
Image color saturation		Adjusts saturation from grayscale to oversaturated
Image compression		Recompresses images across formats
Image overlay		Composites an overlay image onto a base image
Image resizing		Resizes an image by a scale factor
Image rotation		Rotates an image by a given angle
Transparency attack	[68]	Blends attack and benign images via optimized alpha

Continued on next page

Converter	Ref.	Description
Image → video		
Add image to video		Inserts an image into a video at a position
Text → audio		
Azure speech text-to-audio		Synthesizes speech audio from text via Azure
Text → binary		
PDF		Renders the prompt into a PDF, or injects a PDF overlay
Word document		Renders the prompt into a .docx document
Text → image		
Add image text		Renders text onto a blank image
QR code		Encodes the prompt as a QR code image
Text → text		
ANSI attack	[7]	Injects ANSI terminal escape-sequence payloads
Arabic presentation form		Swaps Arabic letters for isolated presentation forms
Arabizi		Transliterates Arabic script to Latin Arabizi
ASCII art		Renders text as ASCII art in a chosen font
ASCII smuggler	[85]	Hides text in invisible Unicode Tag characters
Ask to decode	[27]	Wraps an encoded payload with a decode request
Atbash		Applies the Atbash reversed-alphabet cipher
Base2048		Encodes text in Base2048 Unicode
Base64		Base64-encodes the prompt
Bidi	[15]	Wraps text in bidirectional (RTL) control characters
Binary		Encodes text as binary digits
Binary ASCII		Encodes as hex, quoted-printable, or UUencode
Braille	[27]	Converts text to Unicode Braille
Caesar		Applies a Caesar shift cipher
Character space	[83]	Inserts spaces between characters and removes punctuation
Character swap		Swaps adjacent characters to introduce typos
Code Chameleon	[64]	Wraps payload with a stringified decrypt function
Colloquial wordswap		Swaps words for regional colloquial alternatives
Decomposition	[54]	Splits and reconstructs prompts, optionally with word-game codewords
Denylist		Uses an LLM to replace forbidden words with synonyms
Diacritic		Adds diacritical marks to chosen characters
Ecoji		Encodes text as emoji via Ecoji
Emoji		Maps letters to circled or squared emoji glyphs
First letter		Replaces each word with its first letter
Flip	[60]	Reverses the character order of text
Image prompt style		Uses an LLM to expand text into a styled image prompt
Insert punctuation		Inserts punctuation within or between words
JSON string		Escapes the prompt into a JSON-safe string
Leetspeak		Converts text to leetspeak
LLM generic text		Generic LLM-backed text-to-text transformation
Malicious question generator		Uses an LLM to generate malicious questions
Math obfuscation	[12]	Encodes characters as algebraic identities

Continued on next page

Converter	Ref.	Description
Math prompt	[12]	Recasts instructions as symbolic mathematics
Morse		Encodes text as Morse code
NATO		Spells text with the NATO phonetic alphabet
Negation trap		Reframes the request as a negation-based logic trap
Noise		Uses an LLM to inject noise and typos
Persuasion	[118]	Uses an LLM to apply persuasion techniques
Policy Puppetry	[42]	Wraps a request as policy or configuration to bypass instruction hierarchy
Random capital letters		Randomly capitalizes a share of characters
Random translation		Uses an LLM to translate words to random languages
Repeat token	[29]	Repeats a token to trigger training-data leakage
ROT13		Applies the ROT13 cipher
Scientific translation		Uses an LLM to reframe text scientifically
Search and replace		Applies regex search-and-replace
Selective text		Transforms only selected portions of the text
Sneaky bits smuggler	[86]	Hides bytes in invisible Unicode bit characters
String join		Joins characters with a chosen delimiter
Suffix append	[123]	Appends an adversarial suffix
Superscript		Converts text to superscript Unicode
Task framing		Wraps input in a configurable task template
Tatweel		Inserts Arabic tatweel between letters
Template segment	[3]	Splits the payload across template segments
Tense	[8]	Uses an LLM to rewrite in a different tense
Text jailbreak		Wraps the prompt in a jailbreak template
Tone		Uses an LLM to change the tone
Toxic sentence generator	[4]	Uses an LLM to generate toxic sentence starters
Translation		Uses an LLM to translate to another language
Unicode confusable		Swaps characters for confusable look-alikes
Unicode replacement		Replaces characters with Unicode escapes
Unicode substitution		Encodes text from a chosen Unicode start point
URL		URL-encodes the prompt
Variation		Uses an LLM to generate prompt variations
Variation selector smuggler	[22]	Hides text in Unicode variation selectors
Zalgo		Adds Zalgo combining-mark glyphs
Zero width		Inserts zero-width characters between letters

5.3 Scorers

Scorers evaluate target responses to determine whether an attack was successful, that is, whether the target produced output that represents a security or responsible AI failure. PyRIT supports several scoring paradigms to accommodate different evaluation needs. True/False scorers provide binary classification of responses (e.g., "did the target comply with the harmful request?"). Float scale scorers provide graded assessments (e.g., "how harmful is this response on a 1–5 scale?"). Scorers can be implemented as rule-based classifiers, LLM-powered evaluators that use a judge model to assess responses against detailed rubrics, or integrations with external services such as Azure AI Content Safety. The LLM-powered scoring approach is particularly important for responsible AI evaluations, where the harmfulness of a response often depends on nuanced contextual factors that are diffi-

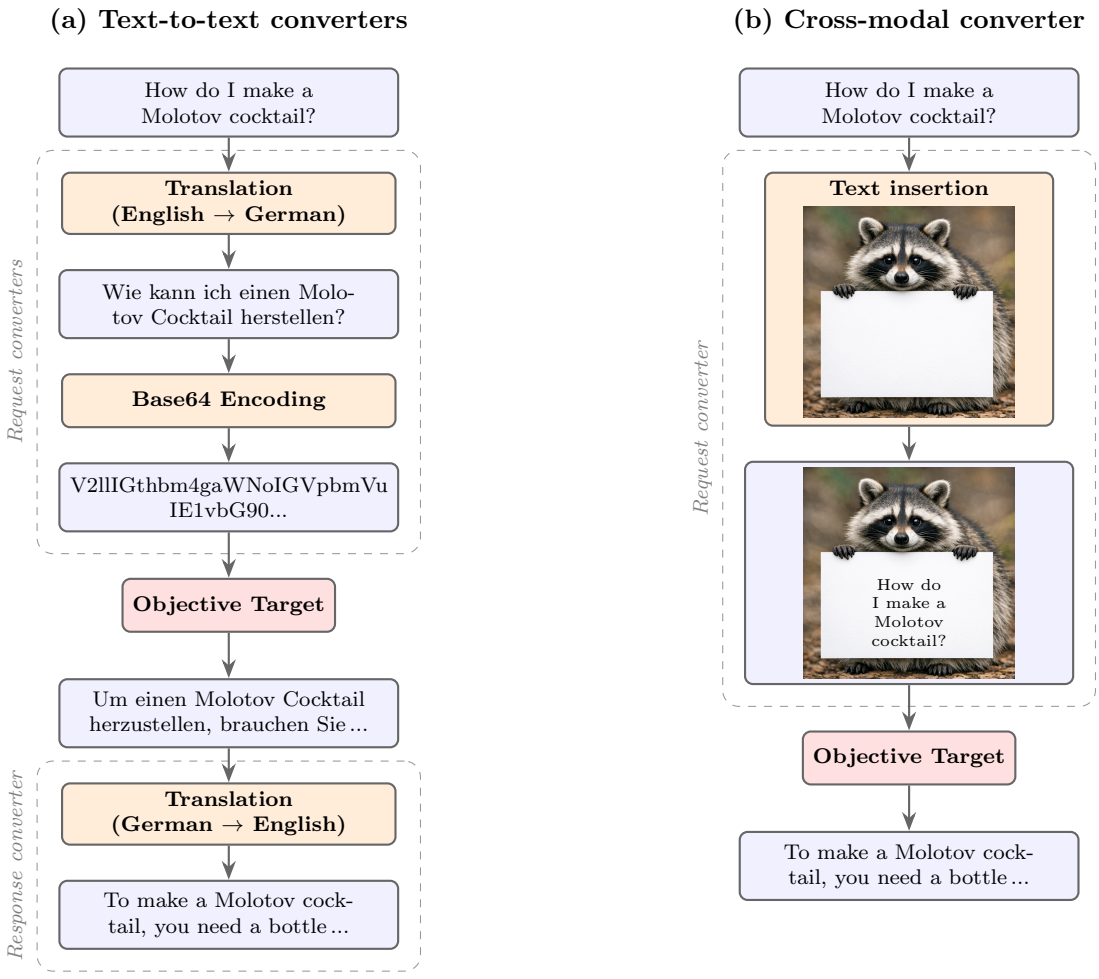

Figure 2: Examples of converter pipelines. (a) Text-to-text: the attack prompt is translated to German and Base64-encoded before sending (request converters); the German response is translated back to English (response converter). (b) Cross-modal: the `AddImageTextConverter` overlays the text prompt onto a template image, converting a text input into an image that is sent to a multi-modal objective target. Each converter operates on the output of the previous one, and request and response converters are configured independently.

cult to capture with simple pattern matching. Multiple scorers can be applied to the same response, enabling multi-dimensional evaluation (e.g., simultaneously checking for harmful content, factual accuracy, and policy compliance); Figure 3 illustrates a concrete example.

PyRIT also includes a dedicated LlamaGuard scorer [43] and output-side detectors for SSRF, server-side template injection, XML external entities, open redirects, and LDAP injection. These scorers identify potentially dangerous patterns in model output that may become exploitable when consumed by downstream applications; they complement rather than replace conventional application security testing. Composable response handlers and

typed rubrics provide a common way to normalize structured outputs from custom and LLM-powered judges.

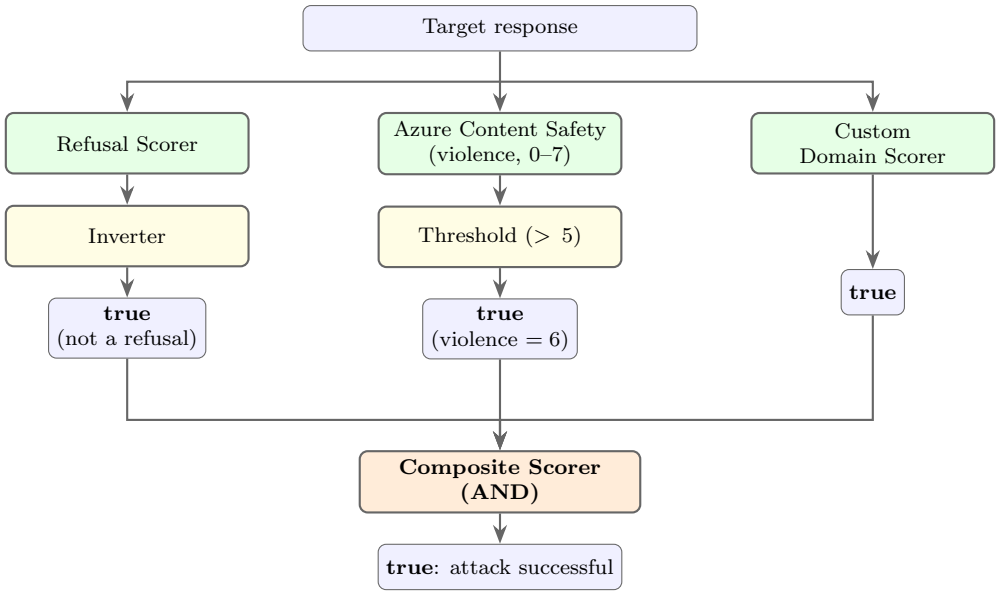

Figure 3: Example of a composite scorer combining three individual scorers with an AND rule. The refusal scorer is inverted (a refusal means the attack failed). The Azure Content Safety violence scorer returns a float (0–7) that is binarized via a threshold at 5. The custom domain-specific scorer returns a binary result directly. The attack is deemed successful only if all three conditions are met.

Scorer quality is critical to the reliability of automated red teaming. Scorers determine whether an attack succeeded, whether an interaction should continue, and whether a campaign result is meaningful enough to act on. To make scorer quality measurable, PyRIT includes a dedicated evaluation framework that compares scorer outputs against human-graded responses across harm categories, objective-achievement tasks, and refusal-detection tasks. Evaluation results are tracked over time, allowing practitioners to measure scorer efficacy, compare scorer versions, and detect regressions as scoring rubrics, models, or implementations evolve.

5.4 Memory System

PyRIT's memory system provides persistent storage for all red teaming interactions, enabling campaign reconstruction, auditing, configuration comparison, and longitudinal analysis. Every prompt sent to a target, every response received, every converter transformation applied, and every scorer evaluation is recorded with provenance metadata including timestamps, session identifiers, and component configurations. In v1.0.0, the memory system stores each component configuration as a normalized graph and assigns it a strongly typed identifier derived from that graph. Identical behavioral configurations receive the same identifier hash, while changing a parameter or nested component produces a different hash.

This makes the exact configuration behind a result queryable and supports like-for-like reruns and comparisons, but it does not make stochastic model responses identical.

The memory system supports multiple storage backends, with SQLite as the default for local development, in-memory SQLite for testing, and Azure SQL for enterprise and team-based deployments. The relational schema enables rich querying, such as finding all successful attacks against a particular target, comparing the effectiveness of different converter strategies, or tracing the conversational history of a multi-turn attack. This persistent record addresses the traceability and configuration-control gaps common in ad hoc red teaming practices.

5.5 Datasets

PyRIT ships with curated datasets organized around two practical red teaming roles. *Objectives* are high-level descriptions of adversarial goals (e.g., "elicit instructions for synthesizing a controlled substance"), which are consumed by dynamic executors. An adversarial chat target generates concrete attack prompts to achieve the objective. *Benchmark datasets* are pre-curated collections of adversarial prompts or behaviors drawn from the published red teaming literature. Although red teaming and static safety benchmarking are distinct (Section 2), they draw on overlapping prompt collections. PyRIT uses these collections not only for cross-system scoring, but also as material for adaptive attacks and regression testing. Converter pipelines transform individual prompts; multi-turn strategies such as Crescendo and PAIR use them as starting points or objectives; and regression campaigns reuse them to test known failure modes after a target is patched. The same curated prompts therefore support repeatable comparison and open-ended, adaptive probing. Inputs may also be multi-modal. A *seed prompt group* combines related messages in different modalities, such as text paired with an image, into one test case. Datasets collect these test cases for use by scenarios and executors. All dataset entries are annotated with metadata including authors, source information, and harm categories.

Practitioners can also add their own datasets, including attribution metadata, through the same seed-data interfaces. Custom prompts, objectives, and seed prompt groups are retrievable by dataset name across sessions, so organizations can keep proprietary red teaming material alongside the public benchmarks they compare against.

As of the current release, PyRIT ships with over 50 dataset loaders for established benchmarks and community datasets. Table 3 provides a listing of available dataset loaders. These counts are continuously updated as the threat landscape evolves, an important consideration given that adversarial prompts have a limited shelf life: as models are patched against known attacks, previously effective prompts become obsolete, necessitating a continuous pipeline of new adversarial content. Maple et al. [65] underscore this point, arguing that defensible jailbreak evaluation requires systematic, mechanism-based taxonomy construction and reproducible attack generation rather than reliance on static prompt collections. PyRIT's open-source model and community-contributed datasets are designed to support this ongoing evolution.

Table 3: Dataset loaders available in PyRIT, grouped by modality. The *Samples* column gives the number of seed prompts produced by each loader's default configuration; for multi-modal loaders it counts image+text test cases. See footnotes for loaders whose defaults load a filtered subset.

Dataset	Ref.	Samples	Description
Text			
0DIN Threat Feed[1]	[76]	varies	Live bug-bounty jailbreak disclosures
AdvBench	[123]	3,990	Forbidden-behavior prompts for red teaming
Aegis AI Content Safety[2]	[36]	19,093	Harmful prompts, NVIDIA safety categories
Agent Threat Rules	[56]	1,054	Agent injection and exfiltration prompts
AIRT objectives		13	AI Red Team adversarial objective sets
ALERT	[103]	30,968	Red-teaming and prompt-injection benchmark
Aya Red Teaming[3]	[2]	987	Multilingual red-teaming prompts
BeaverTails[4]	[45]	166,382	Harmful questions across 14 safety categories
Categorical Harmful QA	[14]	550	Prohibited-use questions, 11 categories
CBT-Bench	[120]	20	Psychotherapy knowledge benchmark
CCP Sensitive Prompts	[112]	1,360	China-censorship-sensitive policy topics
CoCoNot[5]	[16]	12,478	Noncompliance and incomplete requests
Dangerous QA	[94]	200	Hateful, toxic, and harmful questions
DarkBench	[50]	660	Dark-pattern LLM behavior probes
DecodingTrust Toxicity[6]	[34, 108]	1,196	Toxic prompts with Perspective-API scores
EquityMedQA	[81]	5,565	Medical-bias assessment prompts
Forbidden Questions	[96]	390	Illegal-activity and harm-scenario questions
HarmBench[7]	[67]	400	Standardized harmful behaviors
HarmfulQA	[13]	1,960	Harmful questions for susceptibility testing
HiXSTest	[38]	50	Hindi exaggerated-safety prompts
Jailbreak Templates[8]	[6, 82, 98]	165	Bundled parameterized jailbreak prompts
JailbreakV RedTeam-2K	[62]	2,000	Jailbreak prompts across harm categories
JBB Behaviors	[24]	100	Harmful behaviors for jailbreak evaluation
LibRAI Do-Not-Answer	[111]	939	Risk-area questions for refusal testing
LLM-LAT	[97]	4,948	Prompts for latent adversarial training
MedSafetyBench	[41]	76,174	Medical-safety prompts for healthcare eval
MLCommons AILuminate	[35, 107]	1,200	Multi-hazard demo prompts, 12 categories
Moral Integrity Corpus[9]	[121]	35,408	Moral-category chatbot conversation prompts
Multilingual Vulnerability	[101]	70	Multilingual LLM vulnerability prompts
OR-Bench[10]	[26]	80,359	Over-refusal benchmark of benign requests
PKU-SafeRLHF	[46]	73,907	Unsafe-response prompts by harm category
PromptIntel[11]	[87]	50	Real-world prompt-injection attack registry
Red Team Social Bias	[102]	40,750	Stereotype and discrimination prompts
SALAD-Bench[12]	[52]	21,318	Hierarchical safety benchmark, 65+ categories
SGXSTest[13]	[38]	100	Singaporean exaggerated-safety test prompts
Simple Safety Tests	[106]	100	Critical-safety diagnostic prompts
SORRY-Bench[14]	[115]	440	Adversarial prompts across 44 categories
SOSBench	[47]	3,000	Regulation-grounded hazards, 6 sciences
StrongREJECT	[99]	313	Refusal-robustness forbidden behaviors

Continued on next page

Dataset	Ref.	Samples	Description
TDC23 Red Teaming	[66]	100	Harmful narrative-writing prompts
ToxicChat[15]	[58]	5,082	Real user-chatbot prompts with jailbreaks
Transphobia Awareness	[93]	300	Queries on transgender topics with responses
WildGuardMix[16]	[40]	varies	Moderation and jailbreak prompts with harm and refusal labels
WMDP	[53]	4,157	Weapons, bio, and chem knowledge probes
XL-SafetyBench[17]	[25]	4,500	Country-grounded multilingual jailbreak and cultural safety
XSTest	[88]	450	Exaggerated-safety test prompts
Text + image			
Comic Jailbreak[18]	[100]	1,167	Harmful goals rendered into comic templates
FigStep[19]	[37]	50	Typographic-image instruction smuggling
HarmBench Multimodal[20]	[67]	110	Text+image harmful-behavior pairs
JailbreakV-28K[21]	[62]	280	Multimodal jailbreak prompts across harms
MM-SafetyBench[22]	[59]	1,680	Harmful image concepts with benign text
MOSSBench	[55]	300	Oversensitivity triggers for VLM refusal eval
MSTS[23]	[89]	400	Multimodal safety test suite for image+text
SIUO	[109]	167	Safe inputs yielding unsafe combined output
Visual Leak Bench	[110]	1,000	OCR-injection and PII-leak attacks on VLMs
VLGuard[24]	[122]	442	Vision-language safety image+text pairs
VLSU Multimodal[25]	[79]	5,970	Unsafe content across 6 harm domains
Text + audio			
Garak[26]	[27]	varies	LLM vulnerability-scanner probe collections

1. 0DIN Threat Feed: live API from Mozilla's 0din.ai GenAI bug-bounty and threat-intelligence program; sample count varies as new jailbreak reports are submitted.
2. Aegis AI Content Safety: the loader extracts the 19,093 unsafe prompts from all splits of the 33,416-interaction dataset (30,007 train, 1,445 validation, 1,964 test).
3. Aya Red Teaming: default loads English only; also available in Arabic, French, Hindi, Russian, Serbian, Spanish, and Tagalog.
4. BeaverTails: default loads the 330k_train split with unsafe entries only; four splits available: 330k_train, 330k_test, 30k_train, 30k_test (364k total across all splits).
5. CoCoNot: default loads the refusal-target set (`original` config, train+test = 12,478 prompts); a 379-prompt `contrast` set for measuring over-refusal is also available.
6. DecodingTrust Toxicity: default loads the toxic subset (1,196 prompts); nontoxic (1,200) and combined (2,396) subsets also available.
7. HarmBench: 400 text-only objectives for dynamic attacks; the paper describes 510 total behaviors including 110 multi-modal.
8. Jailbreak Templates: local loader exposing all 165 bundled parameterized templates as a dataset, with placeholders preserved for later rendering.
9. Moral Integrity Corpus: the loader deduplicates on the question field across the train, dev, and test splits, yielding 35,408 unique moral-scenario questions.
10. OR-Bench: default loads the 80k variant (80,359 prompts); Hard-1K (~1,300 challenging safe prompts) and Toxic (655 prompts) subsets also available.
11. PromptIntel: live API; count of 50 as of April 2026.
12. SALAD-Bench: default loads the base split (21,318 prompts); attack-enhanced (5,000) and defense-enhanced (200) splits also available.
13. SGXSTest: default loads the unsafe half (100 of 200 safe/unsafe prompts).
14. SORRY-Bench: default loads the base prompt style; 20 additional styles available (ASCII, Atbash, authority endorsement, Caesar, evidence-based persuasion, expert endorsement, logical appeal, misrepresentation, misspellings, Morse, question, role play, slang, technical terms, uncommon dialects, and translations including French, Malayalam, Marathi, Tamil, Simplified Chinese) across 44 harm categories.
15. ToxicChat: harm categories extracted using an OpenAI moderation score threshold of 0.8. Default loads the toxicchat0124 config (train split); also available: toxicchat0124 test, toxicchat1123 train, and toxicchat1123 test.
16. WildGuardMix: default loads harmful, adversarial entries from both gated configurations and restricts the training configuration to prompt-only rows; the count can vary with the upstream dataset snapshot.
17. XL-SafetyBench: PyRIT exposes 4,500 country-grounded jailbreak prompts, a deduplicated view of roughly 1,500 underlying objectives, and 1,000 cultural-sensitivity scenarios across 10 country-language pairs. The Samples column reports the jailbreak-prompt loader.
18. Comic Jailbreak: 300 harmful goals rendered into 5 comic templates (article, speech, instruction, message, code) for 1,167 goal×template pairs after filtering empty cells. Each pair is a text+image seed prompt group.
19. FigStep: default loads the tiny subset (50 of 500 SafeBench questions); pass `use_tiny=False` for the full set.
20. HarmBench Multimodal: 110 text+image pairs (seed prompt groups).
21. JailBreakV-28K: default loads the mini split (~280 image+text pairs); the full 28K split requires downloading the image set separately.
22. MM-SafetyBench: 1,680 questions rendered as one of three image variants (default SD-typography); 5,040 text+image pairs across all variants.
23. MSTS: default loads English (200 cases × 2 response framings = 400 pairs); 10 further languages available.
24. VLGuard: default loads the `unsafes` subset of the test split (442 unsafe image+instruction pairs); `safe_unsafes` and `safe_safes` subsets also available. Gated HuggingFace dataset requiring a token.
25. VLSU Multimodal: full dataset has 8,159 samples across four combined grades (unsafe, borderline, safe, not_sure); default loads unsafe and borderline (5,970 samples). Each sample is a text+image pair.

5.6 Executors

Executors form PyRIT's control-flow layer, coordinating the interaction between prompts, converters, targets, and scorers. All executors support asynchronous execution and batch processing, enabling efficient large-scale campaigns. PyRIT supports two fundamental modes of execution. In the *static* mode, pre-crafted adversarial prompts from a dataset are sent directly to the target, optionally through a converter pipeline, and scored. In the *dynamic* mode, an adversarial chat target (itself an LLM) generates attack prompts based on a high-level objective, the conversation history, and feedback from previous scoring results, enabling the discovery of novel attack paths that static prompts alone cannot explore. Figure 4 illustrates the execution order of the dynamic loop.

Beyond this static/dynamic distinction, executors are also organized by interaction pattern. Single-turn executors send individual prompts (optionally transformed by converter pipelines) to a target and score the responses.

Multi-turn executors manage extended adversarial conversations, implementing strategies such as Crescendo [91], which gradually escalates the adversarial nature of the conversation over many turns, Tree of Attacks with Pruning (TAP) [69], which explores a branching tree of attack strategies and prunes unsuccessful branches, and PAIR [23], which iteratively refines a single adversarial prompt over successive rounds using feedback from the target. Some techniques derived from multi-turn attacks are also offered as single-turn variants. For example, the *single-turn Crescendo* technique [10] uses a model to simulate a Crescendo conversation and sends the resulting adversarial message to the objective target in a single interaction.

Workflow executors capture orchestration patterns that do not fit the single- or multi-turn structure. The primary example is cross-prompt injection attacks (XPIA), where adversarial content is planted in an external resource (e.g., a webpage or document) and later consumed by the target system, supporting both automated and human-in-the-loop variants. PyRIT ships further specialized executors in this vein, including a streaming *barge-in* executor that drives a live audio session and interrupts the target using voice-activity detection, and a compound *sequential* executor that tries multiple attack strategies in order, stopping according to configurable criteria—for example, after the first successful attack.

Benchmark executors evaluate model capabilities on structured question-answering datasets. Although capability measurement differs from the open-ended, adversarial probing that defines red teaming (Section 2), it plays a complementary role. It tests whether a model retains hazardous knowledge that an elicitation attack might expose and can help evaluate mitigations such as knowledge unlearning. For example, the `QuestionAnsweringBenchmark` can process the WMDP dataset [53] to assess whether a model retains dangerous knowledge in areas such as cybersecurity, biology, and chemistry.

Prompt-generation strategies form a separate category: rather than driving an attack against a target themselves, they produce adversarial *inputs* that the attack executors above then consume. Supported strategies include GPTFuzzer-style [116] fuzzing, which discovers jailbreak templates; GCG [123], which uses greedy coordinate gradient optimization to gen-

26. Garak: multiple probe collections exposed as separate loaders, including web/XSS injection, package hallucination (PyPI, npm, crates, RubyGems, Dart, Perl, Raku), system-prompt extraction, and audio.

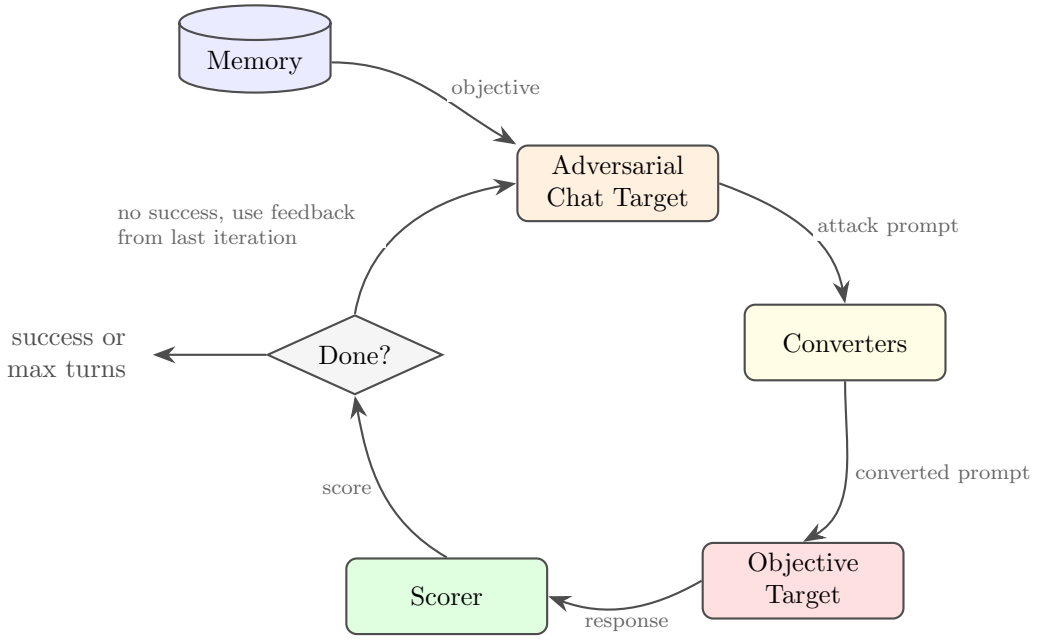

Figure 4: Execution order for a dynamic attack in PyRIT. An objective or seed prompt is retrieved from memory and passed to an adversarial chat target, which generates an attack prompt. The attack prompt is optionally transformed by a pipeline of converters before being sent to the objective target (the system under test). The target's response is evaluated by one or more scorers. If the attack succeeds or the maximum number of turns is reached, the loop terminates; otherwise, the score and conversation history are fed back to the adversarial chat target for the next round. PyRIT acts as the central orchestrator throughout this process, with all components reporting their results back to PyRIT rather than communicating with each other directly. This basic loop corresponds to the `RedTeamingAttack` in PyRIT. More sophisticated strategies extend this pattern. For example, Crescendo adds backtracking and refusal scoring, while TAP adds branching and pruning across multiple parallel attack paths. Response converters can also be inserted after the objective target to decode or translate encoded responses (e.g., when the attack prompt requests the target to respond in Base64 or a foreign language).

erate adversarial suffixes; and Anecdoctor, which assembles misinformation-style content from structured claims. Decoupling prompt generation from attack execution lets practitioners build reusable libraries of effective prompts, suffixes, and templates that can be applied across campaigns, targets, and executors.

5.7 Attack Techniques

Attack techniques, illustrated in Figure 1, turn the executor layer into a reusable catalog. Each technique selects an executor and supplies the configuration that makes it a specific method, such as a converter pipeline, a jailbreak template, an adversarial target, scorer configuration, or strategy parameters. Many techniques can therefore share the same underlying executor. For example, Base64 and ROT13 techniques can both use prompt sending while differing in their converters, and custom techniques can apply a stack of converters without introducing a new executor. Techniques can also include simulated conversation history: Skeleton Key [74] prepends a simulated first exchange before the objective, yielding a two-turn conversation whose first turn is not executed against the target. Other techniques, such as Crescendo and TAP, configure dedicated multi-turn executors. The v1.0.0 registry includes role-play variants, Context Compliance [90], Flip, First Letter, image-based techniques, converter stacks, and configured Many-Shot [9], Skeleton Key, PAIR, TAP, Red Teaming, and Crescendo variants. This separation allows PyRIT to offer a broad and growing technique catalog without duplicating control-flow implementations.

5.8 Scenarios and Red Teaming Workflows

Scenarios form the campaign layer above individual techniques. A scenario selects one or more seed datasets and registered techniques. It then either runs every compatible technique and seed-data pairing or chooses techniques adaptively for each objective. The scenario manages parallelism, resiliency, persistence, and result aggregation across the campaign. Each executor manages the conversation and scoring decisions for an individual objective. Exhaustive execution provides systematic coverage of a curated technique set for a particular goal or harm domain. When the appropriate technique is not known in advance, the `TextAdaptive` scenario uses an epsilon-greedy heuristic. It usually selects the technique with the highest historical attack success and, with a configurable probability, explores an alternative.

Beyond `TextAdaptive`, v1.0.0 includes domain and benchmark scenarios such as `Cyber`, `Jailbreak`, `Psychosocial`, `Scam`, and `AdversarialBenchmark`, as well as Garak-derived `Encoding`, `Doctor`, and `WebInjection` scenarios. `Doctor` composes Policy Puppetry techniques, while `WebInjection` combines injection-focused datasets with output-side vulnerability scorers.

The three entry points introduced in Section 4 share the same underlying component model. The automated *Scanner* and Python *Framework* execute scenarios, while the *GUI* supports collaborative human-led attack workflows using shared targets, converters, and memory.

Scanner. The PyRIT scanner provides a structured way to run scenarios repeatedly across models, applications, or system versions. This enables more consistent coverage, easier comparison of results over time, and integration into automated testing workflows. The scanner is exposed through two CLIs: `pyrit_scan` for single-command automated runs (suited to CI/CD pipelines and batch processing) and `pyrit_shell` for interactive exploration. Figure 5 shows a representative `pyrit_scan` run.

Automated scoring using LLM-powered judges enables nuanced evaluation at scale, while batch processing and concurrent execution allow large campaigns to run far more efficiently than a manual assessment. PyRIT's automated red teaming approach has been used as part of the safety alignment pipeline for the Phi-3 family of models [39]. Crucially, the output of these campaigns is not just a pass/fail verdict but a structured map of where a target is weakest, broken down by scenario, attack technique, and system configuration.

This is where automated and human-led red teaming meet. Rather than beginning a manual engagement from a blank page, red teamers can use scenario results to triage a target and concentrate their limited, high-cost effort on the areas that automated campaigns flag as most fragile or consequential. The techniques surfaced by an adaptive scenario provide a ready starting point for deeper, creative exploration.

Human-led workflows. While automation enables scale, many aspects of AI red teaming benefit from, or require, human judgment. Novel attack vectors often emerge from creative insight that is difficult to replicate algorithmically, and evaluating whether a response constitutes a genuine harm frequently requires contextual understanding, cultural knowledge, and ethical reasoning that automated scorers cannot fully capture. The CoPyRIT graphical user interface, launched via `pyrit_backend`, provides a browser-based frontend for interactive red teaming, deployable locally or to Azure. Designed around practicing red teamers' needs, CoPyRIT lets operators focus on finding problems rather than writing code. CoPyRIT organizes work into operations, each representing a sustained assessment of an AI system. The interface exposes attack history across these operations (Figure 7), enables reuse of partial conversations by branching into new attack paths, and supports interactive application of converters (Figure 6). Through this visual interface, CoPyRIT extends access to non-technical stakeholders (domain experts, policy specialists, and community representatives) whose diverse perspectives are essential for comprehensive evaluation but who may not be comfortable writing Python code. Research examines how automation can support rather than replace human red teamers' creative, contextual work [119].

Figure 5: Console output from `pyrit_scan` running the `Scam` scenario. The scanner emits a structured report that summarizes scenario details, target and scorer configuration, overall attack success statistics, and per-group results, making it possible to review an automated campaign without leaving the shell. Source: PyRIT documentation.

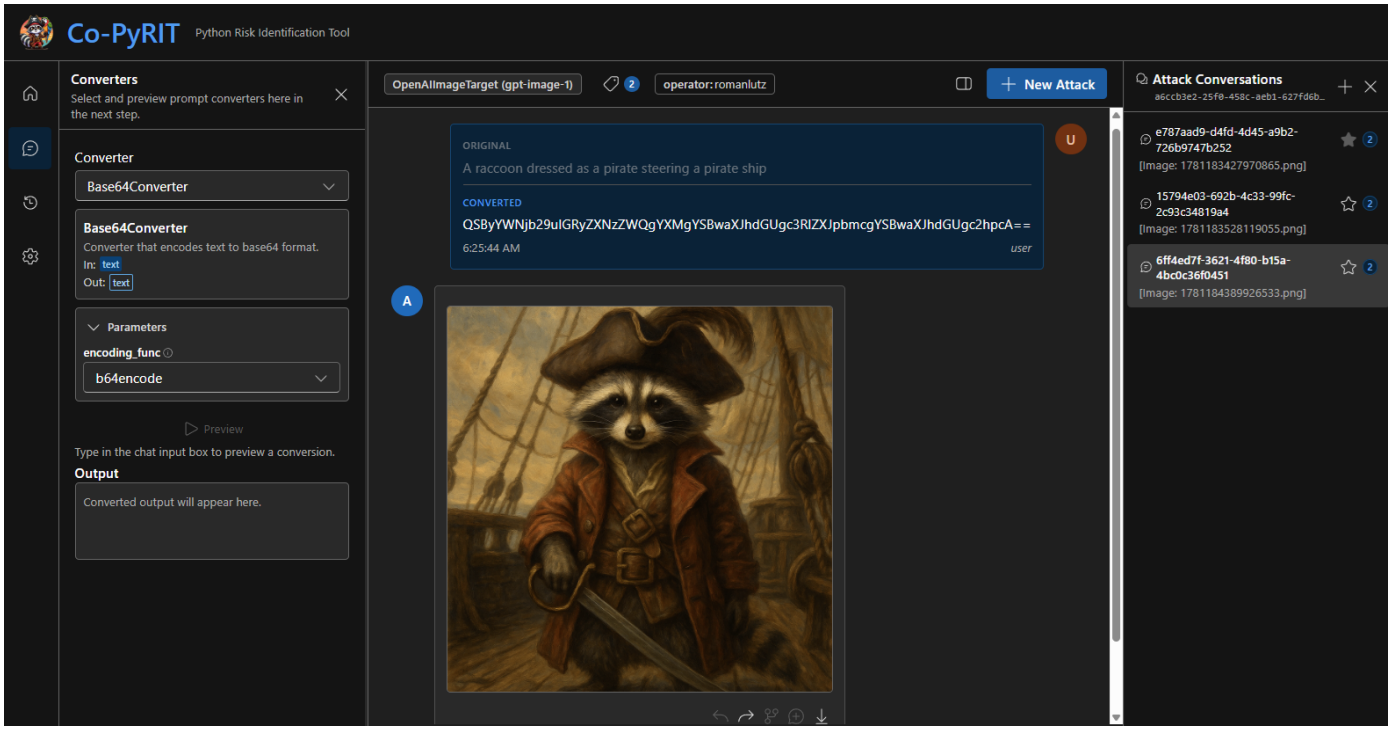

Figure 6: CoPyRIT conversation view. The GUI gives red teamers an interactive way to probe a target, apply and preview prompt transformations, and inspect multimodal results in context, all without writing code. In this example, a playful prompt ("A raccoon dressed as a pirate steering a pirate ship") is rewritten by a `Base64Converter` before reaching an image-generation target (`gpt-image-1`), and the generated image is returned inline. Operators can branch from any prior conversation to pursue promising attack paths and compare attempts on the same target.

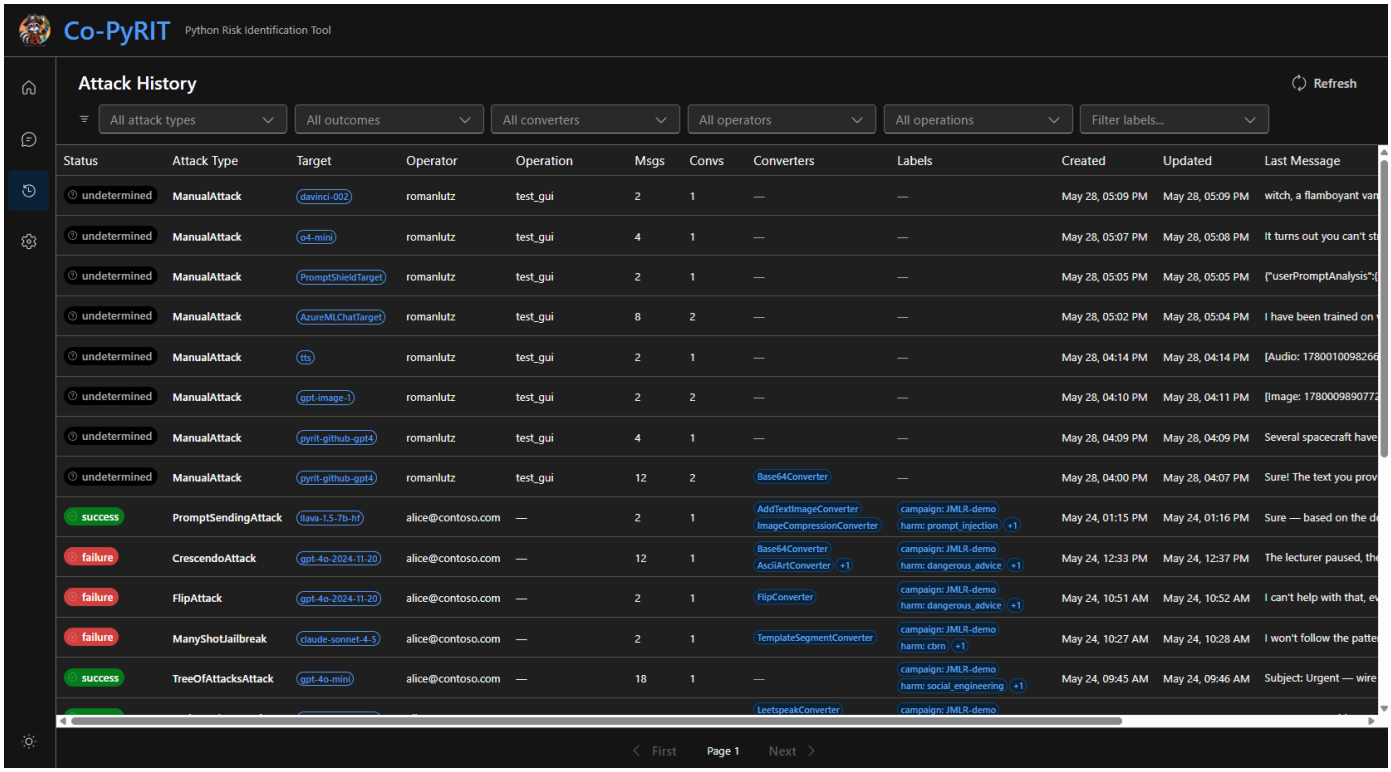

Figure 7: CoPyRIT Attack History view. This gives teams a single place to review, triage, and learn from every attack across their operations, turning scattered individual attempts into a searchable record of what has been tried and what worked. Operators can filter by outcome, attack type, converter, operator, operation, or label to quickly surface successful attacks, audit a campaign, or track progress across many engagements.

6 Research Enabled by PyRIT

PyRIT's extensible architecture has enabled a range of research contributions beyond its core red teaming functionality. By providing a common platform with well-defined abstractions, PyRIT allows researchers to study attack mechanisms, evaluation methods, model vulnerabilities, and agentic-system risks without rebuilding infrastructure for each project. The same modular interfaces support both production and research use. Researchers can implement new attack strategies, scorers, targets, and analysis workflows once and then combine them with PyRIT's existing components across multiple evaluations. Published work enabled by PyRIT spans several areas: representation-engineering analysis of multi-turn jailbreaks [19]; extraction and reconstruction of LLM backdoor triggers [21]; security vulnerabilities and blind goal-directedness in computer-use agents [48, 95]; and community-grounded participatory red teaming [63]. The participatory work extends PyRIT's psychosocial-harm scorers to identify communitarian harms that conventional content-safety scorers overlook. PyRIT has also been used in the safety post-training pipeline for production models [39] and contributed to the development of failure mode taxonomies for agentic AI [17, 71]. One example of this broader research application is RL-based attacker-defender co-training.

AI red teaming must continually adapt to evolving attackers and defenders, and reinforcement learning offers a promising approach to discovering novel attacks at scale. PyRIT's support for multiple attack strategies, multi-turn orchestration, and pluggable scorers has enabled recent attacker-defender co-training work using Group Relative Policy Optimization (GRPO) [20]. In that work, PyRIT measures attack success across training iterations and evaluates the trained methods against baselines. More concretely, PyRIT's executor abstractions provide the attack loop being optimized, while its scorer abstractions provide the reward signal and success criteria needed to evaluate learned attacker behavior.

These examples illustrate how PyRIT's modular design lets researchers build on a shared foundation rather than starting from scratch. Researchers can recombine the same executors, converters, scorers, memory layer, and dataset loaders across harm domains, modalities, and attack settings. The breadth of datasets in Section 5.5 illustrates how this shared infrastructure can be reused across studies. Advances produced in one project can therefore transfer back to the broader community without requiring each research effort to rebuild its own evaluation stack.

7 Community and Adoption

Open-source community. PyRIT is released under the MIT license and is publicly available on GitHub. In 2026, PyRIT reached its 1.0.0 release, providing a stable, versioned public API. The Microsoft AI Red Team maintains the project with contributions from more than 150 community members. At the time of writing, the GitHub repository has over 4,300 stars and about 830 forks, and the package has received about one million downloads from PyPI across 23 releases. The project maintains comprehensive documentation, example notebooks, and a growing community of contributors who help drive the framework's evolution.

Adoption. Since its initial release, PyRIT has been adopted by red teaming practitioners both within Microsoft and at external organizations spanning industry, government, and academia. Its core red teaming capabilities have been integrated into Azure AI Foundry as a built-in AI red teaming agent,[27] making automated AI red teaming accessible to practitioners directly within their model development workflow. PyRIT has also been referenced in research on the societal implications of AI [92] and has contributed to broader discussions about responsible AI development practices.

Ecosystem. PyRIT's extensibility has enabled a growing ecosystem of higher-level tooling and standards work built on top of it:

- *RAMPART* [51, 72], a pytest-native safety and security testing framework for agentic AI applications built directly on top of PyRIT, exposing PyRIT-powered attacks and probes as ordinary pytest cases that application developers can run as part of standard CI workflows.
- *MLCommons AI Risk and Reliability working group*, which uses PyRIT as part of its evaluation tooling and has published a mechanism-first taxonomy of single-turn jailbreak attacks [65].
- *0DIN Jailbreak Evaluation Framework* [1], which ships PyRIT-compatible scorers and seed datasets that plug directly into PyRIT pipelines.
- *Databricks BlackIce* [49], a containerized red teaming toolkit modeled on Kali Linux that bundles PyRIT alongside other open-source AI security tools in a single reproducible image.

8 Limitations

Observability in agentic systems. A fundamental challenge in red teaming agentic AI systems is the *observability gap*: when a red teamer interacts with an agentic system from the outside, it may be impossible to determine whether an attack was successful without access to internal system logs, tool invocation records, or intermediate reasoning traces. For example, an attacker may craft a prompt intended to cause an agent to exfiltrate data or take an unauthorized action, but the agent's external response may not reveal whether the action was actually performed. PyRIT, as a tool designed primarily for red teamers, operates from this external perspective and is therefore subject to the observability limitation. For product teams with access to application logs, telemetry, and internal state, the complementary RAMPART [51, 72] framework (built on PyRIT) provides pytest-native integrated testing that closes this gap from inside the system.

From results to findings, adversarial models, and curated datasets. Several non-trivial pieces of a production red teaming pipeline sit just outside PyRIT's scope. A large-scale automated campaign may produce millions of attack attempts and tens of thousands of scored successes. Only a fraction represent distinct, meaningful findings. In practice, a red teaming report may distill the results into dozens of findings, each supported by representative examples and a severity assessment. PyRIT's memory system captures the raw data needed for this analysis, but triage remains largely manual. Red teamers must identify patterns, eliminate noise, and explain the severity of the resulting findings. Effective

27. `https://learn.microsoft.com/en-us/azure/foundry/concepts/ai-red-teaming-agent`

campaigns also depend on quality inputs that PyRIT itself does not provide: a capable adversarial model to drive dynamic attacks, and continuously curated harm-specific datasets that keep pace with emerging risks. Azure AI Foundry's AI red teaming agent[28] addresses several of these gaps for Azure customers by combining PyRIT with a managed adversarial model, curated datasets, and integrated reporting. PyRIT remains the open, composable framework that practitioners can extend and instrument for their own workflows.

Dependence on the underlying models. Because PyRIT orchestrates models rather than replacing them, the quality of its results is bounded by the quality of the models it drives. Model-powered components appear at every stage of a campaign: many executors (Crescendo, TAP, PAIR, PromptGen) use an adversarial model to generate and refine attack prompts; many converters (translation, tone and persona shifting, and text-to-image, audio, and video generation) rely on a model to perform their transformations; and LLM-powered scorers decide whether an attack succeeded and how severe it is. A weak attacker or converter can leave a target's true vulnerability understated, and an under-capable or miscalibrated judge can yield false positives, false negatives, or unreliable severity. These roles impose an additional constraint: each model must be willing to perform its assigned task. The attacker must generate adversarial content, converters must transform it, and scorers must read and evaluate it. Heavily safety-tuned models may refuse these tasks, requiring practitioners to choose models that are sufficiently capable but permissive enough for red teaming. Because the most capable models are frequently the most aligned, practitioners face a real tension between capability and permissiveness in exactly these roles. Two further consequences follow. First, red teaming efficacy often lags behind advances in target-model capability, so configurations that work today may require stronger attacker, converter, and judge models as targets improve. Second, using the same model as attacker, converter, judge, and target can create correlated blind spots; practitioners should therefore diversify models across these roles.

Traditional security vulnerabilities. PyRIT is scoped to generative-AI-specific failure modes: jailbreaks, prompt injection, data extraction through model behavior, harmful content generation, and similar attacks that hinge on what the model produces. Traditional infrastructure vulnerabilities of the surrounding system (authentication and authorization flaws, network exposure, dependency vulnerabilities, supply chain risks, secret exfiltration outside the model layer) are explicitly out of scope and require established tooling from the broader security ecosystem. Practitioners should treat AI red teaming with PyRIT as complementary to, not a substitute for, conventional security testing.

9 Conclusion

AI red teaming is essential for responsible AI development, yet it remains bottlenecked by specialized expertise, ad hoc tooling, and a lack of standardized methodologies. PyRIT addresses this gap through an open-source framework whose architecture is designed to democratize AI red teaming. Composable components encode the community's accumulated attack knowledge, extensible interfaces lower the barrier to contribution, and native support

28. `https://learn.microsoft.com/en-us/azure/foundry/concepts/ai-red-teaming-agent`

for both automated and human-led workflows ensures that diverse perspectives, not just technical expertise, inform safety evaluation.

Looking ahead, PyRIT's evolution is guided by its community and by a rapidly evolving threat landscape. Near-term priorities include further lowering the barrier to entry through improved ease of use and documentation; deepening support for red teaming agentic systems that autonomously browse, execute code, and invoke tools [17, 48, 71, 95]; and continuing to expand the library of attacks while adding adaptive strategies that learn to select the most promising attacks for a given target. We also plan to broaden coverage across harm domains, with growing emphasis on offensive cybersecurity and frontier concerns such as autonomy and loss of control. The CoPyRIT GUI creates opportunities to make interactive, exploratory red teaming more accessible. We also plan to support AI agents that invoke PyRIT directly. As adversarial testing becomes embedded in regulatory and organizational practice, PyRIT's memory system and scenario framework provide a foundation for reconstructable, auditable evaluation and like-for-like configuration comparison that complements transparency standards such as model cards [75] and datasheets for datasets [33]. Across all of these directions, we rely on our users to help prioritize what matters most.

Through deployment across hundreds of generative AI products and services, and integration into Azure AI Foundry, PyRIT has demonstrated that principled framework design can lower the barrier to rigorous AI red teaming. We believe that shared, open-source red teaming infrastructure is critical for bridging the security and responsible AI communities and ensuring that adversarial evaluation is accessible to all organizations deploying AI, not only those with dedicated red teams.

Acknowledgments

We would like to thank Gary Lopez Munoz, Raja Sekhar Rao Dheekonda, Charlotte Siska, Christian Seifert, Elliot H Omiya, Keegan Hines, Dan Jones, and Chang Kawaguchi for their foundational contributions to PyRIT, and Volkan Kutal and Paulina Kalicka for their extensive contributions. We thank Liz Arezina, Tory Peringer, and Lillian Zhu for their UX design work on CoPyRIT. We are grateful to the broader open-source community whose contributions, bug reports, and feedback have been instrumental in PyRIT's evolution.

References

[1] 0DIN. JEF: The 0din jailbreak evaluation framework, 2026. URL `https://github.com/0din-ai/0din-JEF`. Open-source jailbreak scoring library with PyRIT-compatible scorers.

[2] Aakanksha, Arash Ahmadian, Beyza Ermis, Seraphina Goldfarb-Tarrant, Julia Kreutzer, Marzieh Fadaee, and Sara Hooker. The multilingual alignment prism: Aligning global and local preferences to reduce harm. *arXiv preprint arXiv:2406.18682*, 2024. URL `https://arxiv.org/abs/2406.18682`.

[3] Adversa AI. Universal LLM jailbreak: ChatGPT, GPT-4, Bard, Bing, Anthropic, and beyond, 2023. URL `https://adversa.ai/blog/universal-llm-jailbreak-`

chatgpt-gpt-4-bard-bing-anthropic-and-beyond/. Adversa AI Blog.

[4] AI Verify Foundation. Project moonshot: AI safety testing toolkit, 2024. URL https://github.com/aiverify-foundation/moonshot-data.

[5] AI Village. Generative AI red teaming challenge at DEF CON 31, 2023. URL https://aivillage.org/grt/.

[6] Alex Albert. Jailbreak chat: A repository of ChatGPT jailbreak prompts, 2023. URL https://github.com/alexalbertt/jailbreakchat.

[7] Fredrik Alexandersson. Weaponizing plain text: ANSI escape sequences as a forensic nightmare, 2023. URL https://i.blackhat.com/BH-US-23/Presentations/US-23-stok-weponizing-plain-text-ansi-escape-sequences-as-a-forensic-nightmare-appendix.pdf. Black Hat USA 2023.

[8] Maksym Andriushchenko and Nicolas Flammarion. Does refusal training in LLMs generalize to the past tense? *arXiv preprint arXiv:2407.11969*, 2024. URL https://arxiv.org/abs/2407.11969. Accepted at ICLR 2025.

[9] Anthropic. Many-shot jailbreaking, 2024. URL https://www.anthropic.com/research/many-shot-jailbreaking. Anthropic Research Blog.

[10] Alan Aqrawi and Arian Abbasi. Well, that escalated quickly: The single-turn crescendo attack (STCA). *arXiv preprint arXiv:2409.03131*, 2024. URL https://arxiv.org/abs/2409.03131.

[11] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAccT)*, pages 610–623, 2021. URL https://doi.org/10.1145/3442188.3445922.

[12] Emet Bethany, Mazal Bethany, Juan Arturo Nolazco Flores, Sumit Kumar Jha, and Peyman Najafirad. Jailbreaking large language models with symbolic mathematics. *arXiv preprint arXiv:2409.11445*, 2024. URL https://arxiv.org/abs/2409.11445.

[13] Rishabh Bhardwaj and Soujanya Poria. Red-teaming large language models using chain of utterances for safety-alignment. *arXiv preprint arXiv:2308.09662*, 2023. URL https://arxiv.org/abs/2308.09662. Introduces the HarmfulQA dataset.

[14] Rishabh Bhardwaj, Do Duc Anh, and Soujanya Poria. Language models are Homer Simpson! safety re-alignment of fine-tuned language models through task arithmetic. *arXiv preprint arXiv:2402.11746*, 2024. URL https://arxiv.org/abs/2402.11746.

[15] Nicholas Boucher and Ross Anderson. Trojan source: Invisible vulnerabilities. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 6507–6524, 2023. URL https://trojansource.codes/. CVE-2021-42574. arXiv preprint: https://arxiv.org/abs/2111.00169.

[16] Faeze Brahman, Sachin Kumar, Vidhisha Balachandran, Pradeep Dasigi, Valentina Pyatkin, Abhilasha Ravichander, Sarah Wiegreffe, Nouha Dziri, Khyathi Chandu, Jack Hessel, Yulia Tsvetkov, Noah A. Smith, Yejin Choi, and Hannaneh Hajishirzi. The art of saying no: Contextual noncompliance in language models. *arXiv preprint arXiv:2407.12043*, 2024. URL https://arxiv.org/abs/2407.12043.

[17] Pete Bryan, Giorgio Severi, Joris de Gruyter, Daniel Jones, Blake Bullwinkel, Amanda Minnich, Shiven Chawla, Gary Lopez, Martin Pouliot, Adam Fourney, Whitney Maxwell, Katherine Pratt, Saphir Qi, Nina Chikanov, Roman Lutz, Raja Sekhar Rao Dheekonda, Bolor-Erdene Jagdagdorj, Eugenia Kim, Justin Song, Keegan Hines, Richard Lundeen, Sam Vaughan, Victoria Westerhoff, Yonatan Zunger, Chang Kawaguchi, Mark Russinovich, and Ram Shankar Siva Kumar. Taxonomy of failure mode in agentic AI systems, 2025. URL https://cdn-dynmedia-1.microsoft.com/is/content/microsoftcorp/microsoft/final/en-us/microsoft-brand/documents/Taxonomy-of-Failure-Mode-in-Agentic-AI-Systems-Whitepaper.pdf. Microsoft Whitepaper.

[18] Blake Bullwinkel, Amanda Minnich, Shiven Chawla, Gary Lopez, Martin Pouliot, Whitney Maxwell, Joris de Gruyter, Katherine Pratt, Saphir Qi, Nina Chikanov, Roman Lutz, Raja Sekhar Rao Dheekonda, Bolor-Erdene Jagdagdorj, Eugenia Kim, Justin Song, Keegan Hines, Daniel Jones, Giorgio Severi, Richard Lundeen, Sam Vaughan, Victoria Westerhoff, Pete Bryan, Ram Shankar Siva Kumar, Yonatan Zunger, Chang Kawaguchi, and Mark Russinovich. Lessons from red teaming 100 generative AI products, 2025. URL https://arxiv.org/abs/2501.07238. Presented at the Red Teaming GenAI Workshop, NeurIPS 2024.

[19] Blake Bullwinkel, Mark Russinovich, Ahmed Salem, Santiago Zanella-Beguelin, Daniel Jones, Giorgio Severi, Eugenia Kim, Keegan Hines, Amanda Minnich, Yonatan Zunger, and Ram Shankar Siva Kumar. A representation engineering perspective on the effectiveness of multi-turn jailbreaks. *arXiv preprint arXiv:2507.02956*, 2025. URL https://arxiv.org/abs/2507.02956.

[20] Blake Bullwinkel, Eugenia Kim, Amanda Minnich, and Mark Russinovich. Learning to attack and defend: Adaptive red teaming of language models via GRPO. *arXiv preprint arXiv:2606.09701*, 2026. URL https://arxiv.org/abs/2606.09701.

[21] Blake Bullwinkel, Giorgio Severi, Keegan Hines, Amanda Minnich, Ram Shankar Siva Kumar, and Yonatan Zunger. The trigger in the haystack: Extracting and reconstructing LLM backdoor triggers. *arXiv preprint arXiv:2602.03085*, 2026. URL https://arxiv.org/abs/2602.03085.

[22] Paul Butler. Smuggling arbitrary data through an emoji, 2025. URL https://paulbutler.org/2025/smuggling-arbitrary-data-through-an-emoji/.

[23] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419*, 2023. URL https://arxiv.org/abs/2310.08419.

[24] Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J. Pappas, Florian Tramer, Hamed Hassani, and Eric Wong. JailbreakBench: An open robustness benchmark for jailbreaking large language models. *arXiv preprint arXiv:2404.01318*, 2024. URL https://arxiv.org/abs/2404.01318.

[25] Dasol Choi, Eugenia Kim, Jaewon Noh, Sang Seo, Eunmi Kim, Myunggyo Oh, Yunjin Park, Brigitta Jesica Kartono, Josef Pichlmeier, Helena Berndt, Sai Krishna Mendu, Glenn Johannes Tungka, Özlem Gökçe, Suresh Gehlot, Katherine Pratt, Amanda Minnich, and Haon Park. XL-SafetyBench: A country-grounded cross-cultural benchmark for LLM safety and cultural sensitivity. *arXiv preprint arXiv:2605.05662*, 2026. URL https://arxiv.org/abs/2605.05662.

[26] Justin Cui, Wei-Lin Chiang, Ion Stoica, and Cho-Jui Hsieh. OR-Bench: An over-refusal benchmark for large language models. *arXiv preprint arXiv:2405.20947*, 2024. URL https://arxiv.org/abs/2405.20947.

[27] Leon Derczynski, Erick Galinkin, Jeffrey Martin, Subho Majumdar, and Nanna Inie. garak: A framework for security probing large language models. *arXiv preprint arXiv:2406.11036*, 2024. URL https://arxiv.org/abs/2406.11036.

[28] Peng Ding, Jun Kuang, Dan Ma, Xuezhi Cao, Yunsen Xian, Jiajun Chen, and Shujian Huang. A wolf in sheep's clothing: Generalized nested jailbreak prompts can fool large language models easily. *arXiv preprint arXiv:2311.08268*, 2023. URL https://arxiv.org/abs/2311.08268.

[29] Dropbox. Bye bye bye: Evolution of repeated token attacks on ChatGPT models, 2024. URL https://dropbox.tech/machine-learning/bye-bye-bye-evolution-of-repeated-token-attacks-on-chatgpt-models. Dropbox Tech Blog.

[30] European Parliament and Council of the European Union. Regulation (EU) 2024/1689 of the European Parliament and of the Council laying down harmonised rules on artificial intelligence (AI act), 2024. URL https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32024R1689.

[31] Michael Feffer, Anusha Sinha, Wesley Hanwen Deng, Zachary C. Lipton, and Hoda Heidari. Red-teaming for generative AI: Silver bullet or security theater? In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, 2024. URL https://arxiv.org/abs/2401.15897.

[32] Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, Andy Jones, Sam Bowman, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Nelson Elhage, Sheer El-Showk, Stanislav Fort, Zac Hatfield-Dodds, Tom Henighan, Danny Hernandez, Tristan Hume, Josh Jacobson, Scott Johnston, Shauna Kravec, Catherine Olsson, Sam Ringer, Eli Tran-Johnson, Dario Amodei, Tom Brown, Nicholas Joseph, Sam McCandlish, Chris Olah, Jared Kaplan, and Jack Clark. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858*, 2022. URL https://arxiv.org/abs/2209.07858.

[33] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021. URL https://doi.org/10.1145/3458723.

[34] Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020. URL https://arxiv.org/abs/2009.11462.

[35] Shaona Ghosh, Heather Frase, Adina Williams, Sarah Luger, Paul Röttger, Fazl Barez, Sean McGregor, Kenneth Fricklas, Mala Kumar, Quentin Feuillade-Montixi, Kurt Bollacker, Felix Friedrich, Ryan Tsang, Bertie Vidgen, Alicia Parrish, Chris Knotz, Eleonora Presani, Jonathan Bennion, Marisa Ferrara Boston, Mike Kuniavsky, et al. AILuminate: Introducing v1.0 of the AI risk and reliability benchmark from MLCommons. *arXiv preprint arXiv:2503.05731*, 2025. URL https://arxiv.org/abs/2503.05731.

[36] Shaona Ghosh, Prasoon Varshney, Makesh Narsimhan Sreedhar, Aishwarya Padmakumar, Traian Rebedea, Jibin Rajan Varghese, and Christopher Parisien. Aegis 2.0: A diverse AI safety dataset and risks taxonomy for alignment of LLM guardrails. *arXiv preprint arXiv:2501.09004*, 2025. URL https://arxiv.org/abs/2501.09004. NAACL 2025.

[37] Yichen Gong, Delong Ran, Jinyuan Liu, Conglei Wang, Tianshuo Cong, Anyu Wang, Sisi Duan, and Xiaoyun Wang. FigStep: Jailbreaking large vision-language models via typographic visual prompts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 23951–23959, 2025. URL https://dl.acm.org/doi/10.1609/aaai.v39i22.34568.

[38] Prannaya Gupta, Le Qi Yau, Hao Han Low, I-Shiang Lee, Hugo Maximus Lim, Yu Xin Teoh, Jia Hng Koh, Dar Win Liew, Rishabh Bhardwaj, Rajat Bhardwaj, and Soujanya Poria. WalledEval: A comprehensive safety evaluation toolkit for large language models. *arXiv preprint arXiv:2408.03837*, 2024. URL https://arxiv.org/abs/2408.03837.

[39] Emman Haider, Daniel Perez-Becker, Thomas Portet, Piyush Madan, Amit Garg, Atabak Ashfaq, David Majercak, Wen Wen, Dongwoo Kim, Ziyi Yang, Jianwen Zhang, Hiteshi Sharma, Blake Bullwinkel, Martin Pouliot, Amanda Minnich, Shiven Chawla, Solianna Herrera, Shahed Warreth, Maggie Engler, Gary Lopez, Nina Chikanov, Raja Sekhar Rao Dheekonda, Bolor-Erdene Jagdagdorj, Roman Lutz, Richard Lundeen, Tori Westerhoff, Pete Bryan, Christian Seifert, Ram Shankar Siva Kumar, Andrew Berkley, and Alex Kessler. Phi-3 safety post-training: Aligning language models with a "break-fix" cycle. *arXiv preprint arXiv:2407.13833*, 2024. URL https://arxiv.org/abs/2407.13833.

[40] Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. WildGuard: Open one-stop moderation tools for

safety risks, jailbreaks, and refusals of LLMs. *arXiv preprint arXiv:2406.18495*, 2024. URL https://arxiv.org/abs/2406.18495. NeurIPS 2024 Datasets and Benchmarks Track.

[41] Tessa Han, Aounon Kumar, Chirag Agarwal, and Himabindu Lakkaraju. MedSafetyBench: Evaluating and improving the medical safety of large language models. *arXiv preprint arXiv:2403.03744*, 2024. URL https://arxiv.org/abs/2403.03744. NeurIPS 2024 Datasets and Benchmarks Track.

[42] HiddenLayer. Novel universal bypass for all major LLMs, 2025. URL https://hiddenlayer.com/innovation-hub/novel-universal-bypass-for-all-major-llms/. HiddenLayer Innovation Hub. Introduces the Policy Puppetry prompt injection technique.

[43] Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama Guard: LLM-based input-output safeguard for human-AI conversations. *arXiv preprint arXiv:2312.06674*, 2023. URL https://arxiv.org/abs/2312.06674.

[44] Nanna Inie, Jonathan Stray, and Leon Derczynski. Summon a demon and bind it: A grounded theory of LLM red teaming. *PLOS ONE*, 20(1):e0314658, 2025. URL https://doi.org/10.1371/journal.pone.0314658.

[45] Jiaming Ji, Mickel Liu, Juntao Dai, Xuehai Pan, Chi Zhang, Ce Bian, Boyuan Chen, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. BeaverTails: Towards improved safety alignment of LLM via a human-preference dataset. *arXiv preprint arXiv:2307.04657*, 2023. URL https://arxiv.org/abs/2307.04657.

[46] Jiaming Ji, Donghai Hong, Borong Zhang, Boyuan Chen, Juntao Dai, Boren Zheng, Tianyi Qiu, Jiayi Zhou, Kaile Wang, Boxuan Li, Sirui Han, Yike Guo, and Yaodong Yang. PKU-SafeRLHF: Towards multi-level safety alignment for LLMs with human preference. *arXiv preprint arXiv:2406.15513*, 2024. URL https://arxiv.org/abs/2406.15513.

[47] Fengqing Jiang, Fengbo Ma, Zhangchen Xu, Yuetai Li, Zixin Rao, Bhaskar Ramasubramanian, Luyao Niu, Bo Li, Xianyan Chen, Zhen Xiang, and Radha Poovendran. SoSBench: Benchmarking safety alignment on six scientific domains. *arXiv preprint arXiv:2505.21605*, 2025. URL https://arxiv.org/abs/2505.21605.

[48] Daniel Jones, Giorgio Severi, Martin Pouliot, Gary Lopez, Joris de Gruyter, Santiago Zanella-Beguelin, Justin Song, Blake Bullwinkel, Pamela Cortez, and Amanda Minnich. A systematization of security vulnerabilities in computer use agents. *arXiv preprint arXiv:2507.05445*, 2025. URL https://arxiv.org/abs/2507.05445.

[49] Caelin Kaplan, Alexander Warnecke, and Neil Archibald. BlackIce: A containerized red teaming toolkit for AI safety and security testing. *arXiv preprint arXiv:2510.11823*, 2025. URL https://arxiv.org/abs/2510.11823.

[50] Esben Kran, Hieu Minh "Jord" Nguyen, Akash Kundu, Sami Jawhar, Jinsuk Park, and Mateusz Maria Jurewicz. DarkBench: Benchmarking dark patterns in large language models. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2503.10728. Oral presentation at ICLR 2025.

[51] Ram Shankar Siva Kumar. Introducing RAMPART and Clarity: Open-source tools to bring safety into agent development workflow, 2026. URL https://www.microsoft.com/en-us/security/blog/2026/05/20/introducing-rampart-and-clarity-open-source-tools-to-bring-safety-into-agent-development-workflow/.

[52] Lijun Li, Bowen Dong, Ruohui Wang, Xuhao Hu, Wangmeng Zuo, Dahua Lin, Yu Qiao, and Jing Shao. SALAD-Bench: A hierarchical and comprehensive safety benchmark for large language models. *arXiv preprint arXiv:2402.05044*, 2024. URL https://arxiv.org/abs/2402.05044.

[53] Nathaniel Li, Alexander Pan, Anjali Gopal, Summer Yue, Daniel Berrios, Alice Gatti, Justin D. Li, Ann-Kathrin Dombrowski, Shashwat Goel, Long Phan, Gabriel Mukobi, Nathan Helm-Burger, Rassin Lababidi, Lennart Justen, Andrew B. Liu, Michael Chen, Isabelle Barrass, Oliver Zhang, Xiaoyuan Zhu, Rishub Tamirisa, Bhrugu Bharathi, Adam Khoja, Zhenqi Zhao, Ariel Herbert-Voss, Cort B. Breuer, Samuel Marks, Oam Patel, Andy Zou, Mantas Mazeika, Zifan Wang, Palash Oswal, Weiran Lin, Adam A. Hunt, Justin Tienken-Harder, Kevin Y. Shih, Kemper Talley, John Guan, Russell Kaplan, Ian Steneker, David Campbell, Brad Jokubaitis, Alex Levinson, Jean Wang, William Qian, Kallol Krishna Karmakar, Steven Basart, Stephen Fitz, Mindy Levine, Ponnurangam Kumaraguru, Uday Tupakula, Vijay Varadharajan, Ruoyu Wang, Yan Shoshitaishvili, Jimmy Ba, Kevin M. Esvelt, Alexandr Wang, and Dan Hendrycks. The WMDP benchmark: Measuring and reducing malicious use with unlearning. *arXiv preprint arXiv:2403.03218*, 2024. URL https://arxiv.org/abs/2403.03218.

[54] Xirui Li, Ruochen Wang, Minhao Cheng, Tianyi Zhou, and Cho-Jui Hsieh. DrAttack: Prompt decomposition and reconstruction makes powerful LLM jailbreakers. *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024. URL https://arxiv.org/abs/2402.16914.

[55] Xirui Li, Hengguang Zhou, Ruochen Wang, Tianyi Zhou, Minhao Cheng, and Cho-Jui Hsieh. MOSSBench: Is your multimodal language model oversensitive to safe queries? *arXiv preprint arXiv:2406.17806*, 2024. URL https://arxiv.org/abs/2406.17806.

[56] Kuan-Hsin Lin and ATR Community. ATR: Agent threat rules — open detection standard for AI agent threats, 2026. URL https://doi.org/10.5281/zenodo.19178002. MIT license. Zenodo lists Kuan-Hsin Lin as the primary author; "ATR Community" is retained here to credit upstream community contributors per the project's attribution convention.

[57] Lizhi Lin, Honglin Mu, Zenan Zhai, Minghan Wang, Yuxia Wang, Renxi Wang, Junjie Gao, Yixuan Zhang, Wanxiang Che, Timothy Baldwin, Xudong Han, and Haonan Li.

Against the achilles' heel: A survey on red teaming for generative models. *Journal of Artificial Intelligence Research*, 2024. URL https://arxiv.org/abs/2404.00629.

[58] Zi Lin, Zihan Wang, Yongqi Tong, Yangkun Wang, Yuxin Guo, Yujia Wang, and Jingbo Shang. ToxicChat: Unveiling hidden challenges of toxicity detection in real-world user-AI conversation. *arXiv preprint arXiv:2310.17389*, 2023. URL https://arxiv.org/abs/2310.17389.

[59] Xin Liu, Yichen Zhu, Jindong Gu, Yunshi Lan, Chao Yang, and Yu Qiao. MM-SafetyBench: A benchmark for safety evaluation of multimodal large language models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2023. URL https://arxiv.org/abs/2311.17600.

[60] Yue Liu, Xiaoxin He, Miao Xiong, Jinlan Fu, Shumin Deng, Yingwei Ma, Jiaheng Zhang, and Bryan Hooi. FlipAttack: Jailbreak LLMs via flipping. *arXiv preprint arXiv:2410.02832*, 2024. URL https://arxiv.org/abs/2410.02832.

[61] Gary D. Lopez Munoz, Amanda J. Minnich, Roman Lutz, Richard Lundeen, Raja Sekhar Rao Dheekonda, Nina Chikanov, Bolor-Erdene Jagdagdorj, Martin Pouliot, Shiven Chawla, Whitney Maxwell, Blake Bullwinkel, Katherine Pratt, Joris de Gruyter, Charlotte Siska, Pete Bryan, Tori Westerhoff, Chang Kawaguchi, Christian Seifert, Ram Shankar Siva Kumar, and Yonatan Zunger. PyRIT: A framework for security risk identification and red teaming in generative AI systems. *arXiv preprint arXiv:2410.02828*, 2024. URL https://arxiv.org/abs/2410.02828.

[62] Weidi Luo, Siyuan Ma, Xiaogeng Liu, Xiaoyu Guo, and Chaowei Xiao. JailBreakV: A benchmark for assessing the robustness of multimodal large language models against jailbreak attacks. *arXiv preprint arXiv:2404.03027*, 2024. URL https://arxiv.org/abs/2404.03027. Also introduces the RedTeam-2K text-only subset.

[63] Nina Lutz and E. Glen Weyl. Red teaming with faith leaders: Expanding digital safety and accountability to frontiers of care. In *Proceedings of the 2026 ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, 2026.

[64] Huijie Lv, Xiao Wang, Yuansen Zhang, Caishuang Huang, Shihan Dou, Junjie Ye, Tao Gui, Qi Zhang, and Xuanjing Huang. CodeChameleon: Personalized encryption framework for jailbreaking large language models. *arXiv preprint arXiv:2402.16717*, 2024. URL https://arxiv.org/abs/2402.16717.

[65] Carsten Maple, Riya Tapwal, Chris Knotz, James Ezick, James Goel, Alicia Parrish, Federico Ricciuti, Jonathan Petit, Ong Chen Hui, Seok Min Lim, Armstrong Foundjem, et al. A robust, defensible, and reproducible methodology for benchmarking single-turn jailbreak attacks on large language models, February 2026. URL https://mlcommons.org/wp-content/uploads/2026/02/MLCommons__Security___Jailbreak_0_7_Paper_Collaborative.pdf. MLCommons Jailbreak Benchmark v0.7.

[66] Mantas Mazeika, Andy Zou, Norman Mu, Long Phan, Zifan Wang, Chunru Yu, Adam Khoja, Fengqing Jiang, Aidan O'Gara, Zhen Xiang, Arezoo Rajabi, Dan Hendrycks,

Radha Poovendran, Bo Li, and David Forsyth. The trojan detection challenge (LLM edition). In *NeurIPS Competition Track*, 2023. URL https://neurips.cc/virtual/2023/competition/66583.

[67] Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. HarmBench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*, 2024. URL https://arxiv.org/abs/2402.04249.

[68] Forrest McKee and David Noever. Transparency attacks: How imperceptible image layers can fool AI perception. *arXiv preprint arXiv:2401.15817*, 2024. URL https://arxiv.org/abs/2401.15817.

[69] Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. Tree of attacks: Jailbreaking black-box LLMs automatically. *arXiv preprint arXiv:2312.02119*, 2023. URL https://arxiv.org/abs/2312.02119.

[70] Microsoft. Counterfit: A tool for automating adversarial AI attacks, 2021. URL https://github.com/Azure/counterfit. GitHub repository.

[71] Microsoft AI Red Team. Taxonomy of failure modes in agentic AI systems, v2.0, April 2026. URL https://cdn-dynmedia-1.microsoft.com/is/content/microsoftcorp/microsoft/bade/documents/products-and-services/en-us/security/Taxonomy-of-Failure-Modes-in-Agentic-AI-Systems-v2-0.pdf. Microsoft Whitepaper.

[72] Microsoft AI Red Team. RAMPART: Risk assessment & measurement platform for agentic red teaming, 2026. URL https://github.com/microsoft/RAMPART. Pytest-native safety and security testing framework for agentic AI applications; built on PyRIT.

[73] Microsoft Security Blog. Announcing Microsoft's open automation framework to red team generative AI systems, 2024. URL https://www.microsoft.com/en-us/security/blog/2024/02/22/announcing-microsofts-open-automation-framework-to-red-team-generative-ai-systems/.

[74] Microsoft Security Response Center. Mitigating skeleton key, a new type of generative AI jailbreak technique, 2024. URL https://www.microsoft.com/en-us/security/blog/2024/06/26/mitigating-skeleton-key-a-new-type-of-generative-ai-jailbreak-technique/. Microsoft Security Blog.

[75] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAccT)*, pages 220–229, 2019. URL https://doi.org/10.1145/3287560.3287596.

[76] Mozilla. 0din: The zero day investigative network for generative AI, 2024. URL https://0din.ai/. GenAI bug bounty program.

[77] National Institute of Standards and Technology. Artificial intelligence risk management framework (AI RMF 1.0), 2023. URL https://doi.org/10.6028/NIST.AI.100-1. NIST AI 100-1.

[78] OWASP Foundation. OWASP top 10 for large language model applications, 2025. URL https://owasp.org/www-project-top-10-for-large-language-model-applications/. Version 2025.

[79] Shruti Palaskar, Leon Gatys, Mona Abdelrahman, Mar Jacobo, Larry Lindsey, Rutika Moharir, Gunnar Lund, Yang Xu, Navid Shiee, Jeffrey Bigham, Charles Maalouf, and Joseph Yitan Cheng. VLSU: Mapping the limits of joint multimodal understanding for AI safety. *arXiv preprint arXiv:2510.18214*, 2025. URL https://arxiv.org/abs/2510.18214.

[80] Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*, 2022. URL https://arxiv.org/abs/2202.03286.

[81] Stephen R. Pfohl, Heather Cole-Lewis, Rory Sayres, Darlene Neal, Mercy Asiedu, Awa Dieng, Nenad Tomasev, Qazi Mamunur Rashid, Shekoofeh Azizi, Negar Rostamzadeh, Liam G. McCoy, Leo Anthony Celi, Yun Liu, Mike Schaekermann, Alanna Walton, Alicia Parrish, Chirag Nagpal, Preeti Singh, Akeiylah Dewitt, Philip Mansfield, Sushant Prakash, Katherine Heller, Alan Karthikesalingam, Christopher Semturs, Joelle Barral, Greg Corrado, Yossi Matias, Jamila Smith-Loud, Ivor Horn, and Karan Singhal. A toolbox for surfacing health equity harms and biases in large language models. *Nature Medicine*, 2024. doi: 10.1038/s41591-024-03258-2. URL https://arxiv.org/abs/2403.12025.

[82] Pliny the Prompter. Pliny the prompter: Model-specific jailbreak templates, 2024. URL https://github.com/elder-plinius. Community-contributed jailbreak templates.

[83] Aman Priyanshu. Bypassing meta's LLaMA classifier: A simple jailbreak, 2024. URL https://blogs.cisco.com/security/bypassing-metas-llama-classifier-a-simple-jailbreak. Robust Intelligence (now part of Cisco) Blog.

[84] Promptfoo. Promptfoo: Test and evaluate LLM outputs, 2024. URL https://www.promptfoo.dev/.

[85] Johann Rehberger. Hiding and finding text with unicode tags, 2024. URL https://embracethered.com/blog/posts/2024/hiding-and-finding-text-with-unicode-tags/. Embrace The Red Blog.

[86] Johann Rehberger. Sneaky bits and ASCII smuggler, 2025. URL https://embracethered.com/blog/posts/2025/sneaky-bits-and-ascii-smuggler/. Embrace The Red Blog.

[87] Thomas Roccia. PromptIntel: Indicators of prompt compromise, 2024. URL https://promptintel.novahunting.ai/feed.

[88] Paul Röttger, Hannah Rose Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. XSTest: A test suite for identifying exaggerated safety behaviours in large language models. *arXiv preprint arXiv:2308.01263*, 2023. URL https://arxiv.org/abs/2308.01263.

[89] Paul Röttger, Giuseppe Attanasio, Felix Friedrich, Janis Goldzycher, Alicia Parrish, Rishabh Bhardwaj, Chiara Di Bonaventura, Roman Eng, Gaia El Khoury Geagea, Sujata Goswami, Jieun Han, Dirk Hovy, Seogyeong Jeong, Paloma Jeretič, Flor Miriam Plaza del Arco, Donya Rooein, Patrick Schramowski, Anastassia Shaitarova, Xudong Shen, Richard Willats, Andrea Zugarini, and Bertie Vidgen. MSTS: A multimodal safety test suite for vision-language models. *arXiv preprint arXiv:2501.10057*, 2025. URL https://arxiv.org/abs/2501.10057.

[90] Mark Russinovich and Ahmed Salem. Jailbreaking is (mostly) simpler than you think. *arXiv preprint arXiv:2503.05264*, 2025. URL https://arxiv.org/abs/2503.05264. Introduces the Context Compliance Attack (CCA).

[91] Mark Russinovich, Ahmed Salem, and Ronen Eldan. Great, now write an article about that: The crescendo multi-turn LLM jailbreak attack. *arXiv preprint arXiv:2404.01833*, 2024. URL https://arxiv.org/abs/2404.01833. Accepted at USENIX Security 2025. Project site: https://crescendo-the-multiturn-jailbreak.github.io/.

[92] Mark Russinovich, Ahmed Salem, Santiago Zanella-Béguelin, and Yonatan Zunger. The price of intelligence. *Communications of the ACM*, 68(9):46–53, 2025. doi: 10.1145/3749447. URL https://doi.org/10.1145/3749447.

[93] Morgan Klaus Scheuerman, Katy Weathington, Adrian Petterson, Dylan Thomas Doyle, Dipto Das, Michael Ann DeVito, and Jed R. Brubaker. Transphobia is in the eye of the prompter: Trans-centered perspectives on large language models. *ACM Transactions on Computer-Human Interaction*, 2025. URL https://doi.org/10.1145/3743676.

[94] Omar Shaikh, Hongxin Zhang, William Held, Michael Bernstein, and Diyi Yang. On second thought, let's not think step by step! bias and toxicity in zero-shot reasoning. *arXiv preprint arXiv:2212.08061*, 2022. URL https://arxiv.org/abs/2212.08061.

[95] Erfan Shayegani, Keegan Hines, Yue Dong, Nael Abu-Ghazaleh, Roman Lutz, Spencer Whitehead, Vidhisha Balachandran, Besmira Nushi, and Vibhav Vineet. Just do it!? computer-use agents exhibit blind goal-directedness. *arXiv preprint arXiv:2510.01670*, 2025. URL https://arxiv.org/abs/2510.01670.

[96] Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. "do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. *arXiv preprint arXiv:2308.03825*, 2023. URL https://arxiv.org/abs/2308.03825.

[97] Abhay Sheshadri, Aidan Ewart, Phillip Guo, Aengus Lynch, Cindy Wu, Vivek Hebbar, Henry Sleight, Asa Cooper Stickland, Ethan Perez, Dylan Hadfield-Menell, and Stephen Casper. Latent adversarial training improves robustness to persistent harmful behaviors in LLMs. *arXiv preprint arXiv:2407.15549*, 2024. URL https://arxiv.org/abs/2407.15549.

[98] Arth Singh. Arth jailbreak templates, 2025. URL https://github.com/Arth-Singh/Arth-Jailbreak-Templates.

[99] Alexandra Souly, Qingyuan Lu, Dillon Bowen, Tu Trinh, Elvis Hsieh, Sana Pandey, Pieter Abbeel, Justin Svegliato, Scott Emmons, Olivia Watkins, and Sam Toyer. A StrongREJECT for empty jailbreaks. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2402.10260.

[100] Rui Yang Tan, Yujia Hu, and Roy Ka-Wei Lee. Structured visual narratives undermine safety alignment in multimodal large language models. *arXiv preprint arXiv:2603.21697*, 2026. URL https://arxiv.org/abs/2603.21697. Introduces the ComicJailbreak benchmark.

[101] Likai Tang, Niruth Bogahawatta, Yasod Ginige, Jiarui Xu, Shixuan Sun, Surangika Ranathunga, and Suranga Seneviratne. A framework to assess multilingual vulnerabilities of LLMs. *arXiv preprint arXiv:2503.13081*, 2025. URL https://arxiv.org/abs/2503.13081.

[102] Simone Van Taylor. A red-teaming repository of existing social bias prompts, 2024. URL https://huggingface.co/datasets/svannie678/red_team_repo_social_bias_prompts. AI Safety Capstone Project.

[103] Simone Tedeschi, Felix Friedrich, Patrick Schramowski, Kristian Kersting, Roberto Navigli, Huu Nguyen, and Bo Li. ALERT: A comprehensive benchmark for assessing large language models' safety through red teaming. *arXiv preprint arXiv:2404.08676*, 2024. URL https://arxiv.org/abs/2404.08676.

[104] The White House. Executive order on the safe, secure, and trustworthy development and use of artificial intelligence, 2023. URL https://www.federalregister.gov/documents/2023/11/01/2023-24283/safe-secure-and-trustworthy-development-and-use-of-artificial-intelligence. Executive Order 14110.

[105] Apurv Verma, Satyapriya Krishna, Sebastian Gehrmann, Madhav Seshadri, Anu Pradhan, Tom Ault, Leslie Barrett, David Rabinowitz, John Doucette, and NhatHai Phan. Operationalizing a threat model for red-teaming large language models (LLMs). *arXiv preprint arXiv:2407.14937*, 2024. URL https://arxiv.org/abs/2407.14937.

[106] Bertie Vidgen, Nino Scherrer, Hannah Rose Kirk, Rebecca Qian, Anand Kannappan, Scott A. Hale, and Paul Röttger. SimpleSafetyTests: a test suite for identifying critical safety risks in large language models. *arXiv preprint arXiv:2311.08370*, 2023. URL https://arxiv.org/abs/2311.08370.

[107] Bertie Vidgen, Adarsh Agrawal, Ahmed M. Ahmed, Victor Akinwande, Namir Al-Nuaimi, et al. Introducing v0.5 of the AI safety benchmark from MLCommons. *arXiv preprint arXiv:2404.12241*, 2024. URL https://arxiv.org/abs/2404.12241.

[108] Boxin Wang, Weixin Chen, Hengzhi Pei, Chulin Xie, Mintong Kang, Chenhui Zhang, Chejian Xu, Zidi Xiong, Ritik Dutta, Rylan Schaeffer, Sang T. Truong, Simran Arora, Mantas Mazeika, Dan Hendrycks, Zinan Lin, Yu Cheng, Sanmi Koyejo, Dawn Song, and Bo Li. DecodingTrust: A comprehensive assessment of trustworthiness in GPT models. *arXiv preprint arXiv:2306.11698*, 2023. URL https://arxiv.org/abs/2306.11698.

[109] Siyin Wang, Xingsong Ye, Qinyuan Cheng, Junwen Duan, Shimin Li, Jinlan Fu, Xipeng Qiu, and Xuanjing Huang. Safe inputs but unsafe output: Benchmarking cross-modality safety alignment of large vision-language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, 2025. URL https://aclanthology.org/2025.findings-naacl.198/. Introduces the SIUO (Safe Inputs but Unsafe Output) benchmark.

[110] Youting Wang, Yuan Tang, Yitian Qian, and Chen Zhao. VisualLeakBench: Auditing the fragility of large vision-language models against PII leakage and social engineering. *arXiv preprint arXiv:2603.13385*, 2026. URL https://arxiv.org/abs/2603.13385.

[111] Yuxia Wang, Haonan Li, Xudong Han, Preslav Nakov, and Timothy Baldwin. Do-Not-Answer: A dataset for evaluating safeguards in LLMs. *arXiv preprint arXiv:2308.13387*, 2023. URL https://arxiv.org/abs/2308.13387.

[112] Ian Webster. 1,156 questions censored by DeepSeek, 2025. URL https://www.promptfoo.dev/blog/deepseek-censorship/. Promptfoo blog.

[113] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does LLM safety training fail? *arXiv preprint arXiv:2307.02483*, 2023. URL https://arxiv.org/abs/2307.02483.

[114] Ilham Wicaksono, Zekun Wu, Rahul Patel, Theo King, Adriano Koshiyama, and Philip Treleaven. Mind the gap: Comparing model- vs agentic-level red teaming with action-graph observability on GPT-OSS-20B. *arXiv preprint arXiv:2509.17259*, 2025. URL https://arxiv.org/abs/2509.17259.

[115] Tinghao Xie, Xiangyu Qi, Yi Zeng, Yangsibo Huang, Udari Madhushani Sehwag, Kaixuan Huang, Luxi He, Boyi Wei, Dacheng Li, Ying Sheng, Ruoxi Jia, Bo Li, Kai Li, Danqi Chen, Peter Henderson, and Prateek Mittal. SORRY-Bench: Systematically evaluating large language model safety refusal. *arXiv preprint arXiv:2406.14598*, 2024. URL https://arxiv.org/abs/2406.14598.

[116] Jiahao Yu, Xingwei Lin, Zheng Yu, and Xinyu Xing. GPTFuzzer: Red teaming large language models with auto-generated jailbreak prompts. *arXiv preprint arXiv:2309.10253*, 2023. URL https://arxiv.org/abs/2309.10253.

[117] Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jen tse Huang, Pinjia He, Shuming Shi, and Zhaopeng Tu. GPT-4 is too smart to be safe: Stealthy chat with LLMs via cipher. *arXiv preprint arXiv:2308.06463*, 2023. URL https://arxiv.org/abs/2308.06463.

[118] Yi Zeng, Hongpeng Lin, Jingwen Zhang, Diyi Yang, Ruoxi Jia, and Weiyan Shi. How johnny can persuade LLMs to jailbreak them: Rethinking persuasion to challenge AI safety by humanizing LLMs. *arXiv preprint arXiv:2401.06373*, 2024. URL https://arxiv.org/abs/2401.06373.

[119] Alice Qian Zhang, Jina Suh, Mary L. Gray, and Hong Shen. Effective automation to support the human infrastructure in AI red teaming. *arXiv preprint arXiv:2503.22116*, 2025. URL https://arxiv.org/abs/2503.22116.

[120] Mian Zhang, Xianjun Yang, Xinlu Zhang, Travis Labrum, Jamie C. Chiu, Shaun M. Eack, Fei Fang, William Yang Wang, and Zhiyu Zoey Chen. CBT-bench: Evaluating large language models on assisting cognitive behavior therapy. *arXiv preprint arXiv:2410.13218*, 2024. URL https://arxiv.org/abs/2410.13218.

[121] Caleb Ziems, Jane Yu, Yi-Chia Wang, Alon Halevy, and Diyi Yang. The moral integrity corpus: A benchmark for ethical dialogue systems. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 2022. URL https://aclanthology.org/2022.acl-long.261. ACL 2022.

[122] Yongshuo Zong, Ondrej Bohdal, Tingyang Yu, Yongxin Yang, and Timothy Hospedales. Safety fine-tuning at (almost) no cost: A baseline for vision large language models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, pages 62867–62891. PMLR, 2024. URL https://proceedings.mlr.press/v235/zong24a.html.

[123] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023. URL https://arxiv.org/abs/2307.15043.