Project Moonshot for Generative AI: Turn Red-Team Findings Into Release Gates

Description

A Project Moonshot report is evidence from a defined test run, not a safety certificate. The product team still has to translate findings into release decisions: which failure matters for this use case, who owns the control, what blocks launch and what must be reproduced after the fix.

This guide is for a Singapore product or risk team testing a generative-AI feature before public release. The decision is to use baseline, benchmarking and red-team findings to define owners, severity and stop-release conditions rather than treating a tool run as certification.

Moonshot joins two testing modes

The AI Verify Foundation describes Project Moonshot as an open-source LLM evaluation toolkit bringing benchmarking and red teaming together. Benchmarks measure defined competencies or safety dimensions, while red teaming deliberately probes for behaviour outside the intended design. One does not replace the other. AI Verify Foundation Project Moonshot guide.

Record the test boundary

Save the application version, model and provider, system prompt, retrieval corpus, tools, permissions, sampling configuration, guardrails and evaluation dataset. A result without those inputs cannot be reproduced after the model, prompt or data changes. Treat external model updates as a reason to retest material risks.

Map findings to consequences

For every failure, state the affected user, plausible action and harm. Prompt injection that exposes an internal instruction, a harmful output in a wellness assistant and a factual error in a low-stakes brainstormer deserve different severity. Severity belongs to the use case, not merely the benchmark score.

Use stop-release conditions

Define a blocker before running the final test: for example, unauthorised tool action, personal-data leakage, repeatable high-severity harmful advice or bypass of a required human approval. Assign an owner, mitigation, verification test and residual-risk approver. A dashboard with red cells but no decision owner is not governance.

The programme supports pre and post-deployment testing

IMDA launched Project Moonshot as an open toolkit for evaluating LLM applications and the Foundation describes integration and reporting capabilities. Use a pre-release baseline, a release-candidate rerun and monitored post-deployment tests. Passing once does not cover new prompts, tools, data or attack methods. IMDA Project Moonshot launch.

The two working tools

The first original unit is a finding-to-gate matrix with columns for scenario, reproducibility, affected user, harm, severity, owner, control, blocker and retest. The second is a reproducibility card containing every model and application setting. Together they make a later â??passâ?• comparable with the run that originally failed.

Finding	Possible gate	Required retest evidence
Prompt injection changes tool instruction	Block release if unauthorised action occurs	Same attack suite plus permission-boundary test
Personal data appears in output	Block and investigate data path	Corpus audit, access fix and canary tests
Harmful high-stakes advice	Block affected use case	Domain review and adversarial scenario rerun
Benign formatting failure	May ship with owned defect if impact low	Regression test and documented acceptance

Keep the decision usable after today

A first check can go stale before the task is finished. Put moonshot joins two testing modes, record the test boundary and map findings to consequences on separate dated lines instead of combining them into one â??doneâ?• box. Attach the authority page or document beside the line it supports, record the person who checked it, and write the exact event that will force another check. That event may be a changed account, amended filing, new appointment, revised timetable, altered access route, later test run or updated dataset. The format matters because a future reader must be able to see which fact changed without repeating every part of the exercise.

Next, give the two original tools different owners. The person maintaining a finding-to-release-gate matrix for prompt injection, data leakage, harmful output and system boundary failures should preserve the inputs and arithmetic or branch logic. The person maintaining a reproducibility card recording

model, configuration, prompts, tool access and retest evidence should confirm that the final action followed the chosen route. One person may perform both roles, but the evidence should still distinguish calculation from execution. This prevents a correct plan from being mistaken for proof that the payment, filing, trip, report, repair, training or release actually happened.

Before relying on the result, ask a second reader to reproduce the conclusion from the saved material without being told the preferred answer. They should be able to match the right person, entity, account, property, route, service or software version; identify the controlling date; and explain the strongest stop condition. If they reach another branch, do not average the two answers. Reopen the disputed source, definition or input. A decision that cannot be reproduced is not ready for a consequential step.

Worked example

A support assistant resists general jailbreak prompts but follows an injected instruction embedded in a retrieved document and exposes an internal workflow. The team marks the retrieval-tool path as a release blocker, adds content isolation and tool authorisation, then reruns both the original attack and a broader injection set. A different benchmark score alone cannot close the defect.

The example is a calculation or decision illustration, not a report of an interview, purchase, visit, transaction, taste test or personal outcome. Replace its inputs with the readerâ??s own current evidence.

Where this can go wrong

- Calling a toolkit run certification or proof that the application is safe.
- Saving scores without model, prompt, tool and dataset versions.
- Ranking severity by benchmark colour instead of user consequence.
- Closing a finding after a code change without reproducing the original failure and rerunning adjacent attacks.

Before acting

1. Define the intended users, tasks and prohibited outcomes.
2. Freeze the test configuration and reproducibility card.
3. Run relevant benchmarks and manual or automated red-team scenarios.
4. Assign severity, owner, blocker and residual-risk approver.
5. Retest the original exploit and adjacent paths before release.

Limits and useful next reading

Project Moonshot is a testing toolkit, not a complete governance system or guarantee against unknown failures. High-stakes applications need domain expertise, privacy and security review, operational monitoring and accountable human decisions.

For the next related decision, understand the limits of another assurance label. It is also useful to see an adversarial prompt and phishing decision pattern.

Date Created
31/08/2026
Author
vanessakoh