MINT

Maintainer-INformed Triage

From noisy reports to grounded assessment, reproduction and evidence



Jessy Ayala
University of California, Irvine

Maintainer time is the scarce resource

Illustrative incoming report

“Critical remote code execution in the upload handler.”

Missing version, **unclear** trigger, **unsupported** impact, or **nonexistent** artifacts

Locate the real code

Repository-specific evidence

Reconstruct the behavior

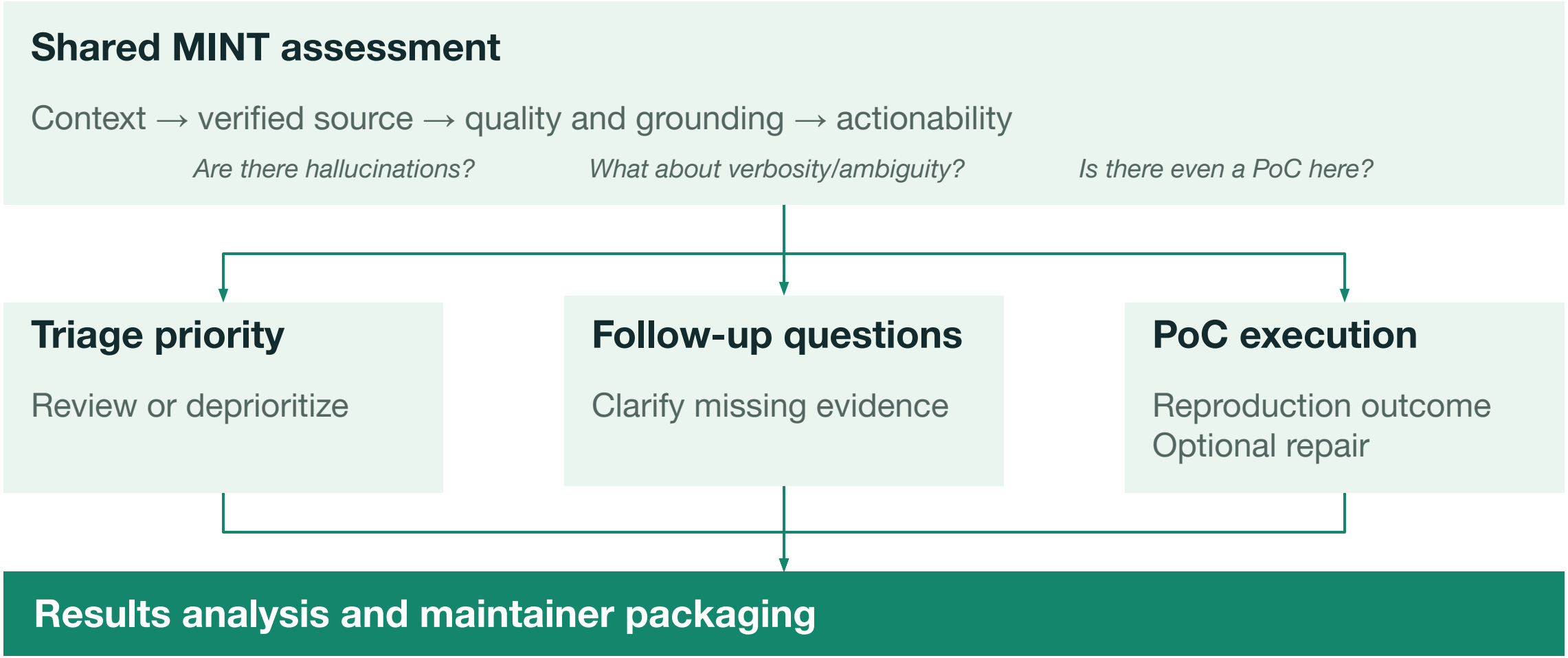
Trigger, environment and controls

Judge the impact

Attacker assumptions and security boundary

Tool design: maintainer survey and expert input inform the checks and outputs

The full pipeline has several valid routes



Establish report and project context

Shared assessment · Nodes 1–5

Node	Purpose
1 · Ingest	Read the vulnerability report and project policy
2 · Metadata	Extract project, vulnerability, revision, reporter and cited evidence
3 · Researcher context	Assess initial signals; keep history separate from claim evidence
4 · Repository	Retrieve the relevant repository revision
5 · Repository tree	Map structure, languages, dependencies and build system

Use the relevant revision and submission-time context

Preserve submission-time evidence

Record provenance for the assessment and for later observations

At filing	Report, policy and then-available prior-report context
Target revision	Exact inspected commit, selection method and uncertainty
Later observations	Execution and patch results, recorded as new evidence

Verify the artifacts before reasoning about them

Shared assessment · Node 6 · LLM-enhanced verification and source retrieval

Locate → Recover → Retrieve

Locate

Check paths, symbols, revisions and cited artifacts

Recover

Resolve likely misreferences against the inspected project

Retrieve

Provide verified source for downstream assessment

A candidate match must be checked; a missing path needs ownership and revision context

Assess the evidence, then recommend a route

Shared assessment · Nodes 7–10

7 · Prior reports	Relevant historical reports available at submission time
8 · Report quality	Completeness, clarity, reproduction evidence and consistency
9 · Grounding	Whether verified code supports the technical claims
10 · Actionability	Review, clarification, deprioritization or execution

Keep report quality, technical support, scope and severity distinct

The next route depends on evidence and readiness

Illustrative route choices after actionability assessment

Available evidence and readiness	Next route
Enough triage evidence; no execution selected	Triage only → review or deprioritize
Missing evidence, unsupported target, or environment unavailable	Follow-up suggestions for review or clarification
Usable trigger, supported target, and reviewed environment ready	Conditional MINT+ execution

A supplied PoC receives technical assessment even after deprioritization

Investigations add different kinds of evidence

Separate evidence drivers after assessment

Static plausibility

Deterministic vulnerability-pattern checks with targeted LLM analysis

Screenshot grounding

Extract visible claims and check them against the repository revision

These routes can inform investigation; they do not establish reproduction

Execution starts with a reviewed test contract

Conditional execution (setup) · Node 11

Target

Exact historical commit
and supported adapter

Trigger and oracle

Reviewed PoC and a
claim-specific observation

Environment

Isolated runner with
recorded dependencies

Controls and limits

Functionality checks,
resource and time bounds

Usable trigger + supported target + ready environment → execution

A reproduction test needs an oracle and controls

Illustrative native-crash test; the oracle defines the observation to check

Reported behavior **A supplied input causes the reported heap-buffer-overflow**

Trigger test

Check the expected failure signature and record the observed effect

Control and scope

Check appropriate normal behavior; record which boundary was tested

A different crash does not establish the reported claim

Reproduction outcomes carry different meanings

Conditional execution (run & produce result) · Node 12

Execution outcome	Interpretation
Behavior **reproduced**	Claim-specific behavior observed in the recorded test scope
Behavior **not observed**	Completed test did not observe the claimed effect
Different behavior observed	An observation occurred, but it did not match the claim
Execution **unresolved**	Setup, dependencies or infrastructure were inconclusive

Candidate-patch validation is a separate test

Optional repair route after supported behavior is observed

Apply candidate

Fresh checkout of the same historical commit

Rerun the trigger

Rebuild and repeat the reviewed vulnerability test

Check functionality

Run available controls and preserve test scope

Patch outcomes retain the limits of the test

Interpret the PoC rerun together with functionality controls

Observed result	Interpretation
Effect stops; controls pass	Supported under the recorded test contract
Reported failure persists	Candidate rejected
Effect stops; functionality fails	Symptom suppression
Narrower observed effect stops; controls pass	Scope-limited evidence
Required build, rerun or control missing	Candidate remains unvalidated

Passing the available contract does not establish general correctness

Every route produces a maintainer evidence package

Results analysis and maintainer packaging · Node 13

Triage	Technical behavior	Security significance	Patch result

Keep these judgments separate, including their uncertainty

Maintainer brief

- Recommendation and reasons
- Uncertainty and next action

Evidence package

- Logs and screenshots
- Environment and test artifacts

Pipeline metrics

- Time, cost and resource use
- Route and completion status

When a report reaches reproduction and validation

Stored orchestration demonstration · supplied candidate patch

Prototype triage input	Review / high supplied to the orchestration fixture
Reproduction	Expected sanitizer failure reproduced at the affected revision
Candidate validation	Submitted PoC no longer triggers the failure; functionality control passes
Maintainer action	Inspect the diff and run broader project CI before considering a merge

A maintainer brief turns findings into an action

Condensed from an example reports · supplied candidate

Observed behavior	Expected sanitizer failure reproduced at the affected revision
Patch evidence	Same PoC no longer triggers the failure; functionality control passes
Limits	Prepared triage input; evidence is bounded by the test contract
Next action	Inspect the diff and run project CI before considering a merge

The maintainer can inspect the evidence and revise the recommendation

Evaluate evidence quality and maintainer benefit

Assessment

- Correct grounding
- Reports ability to be prioritized/deprioritized
- Uncertainty preserved

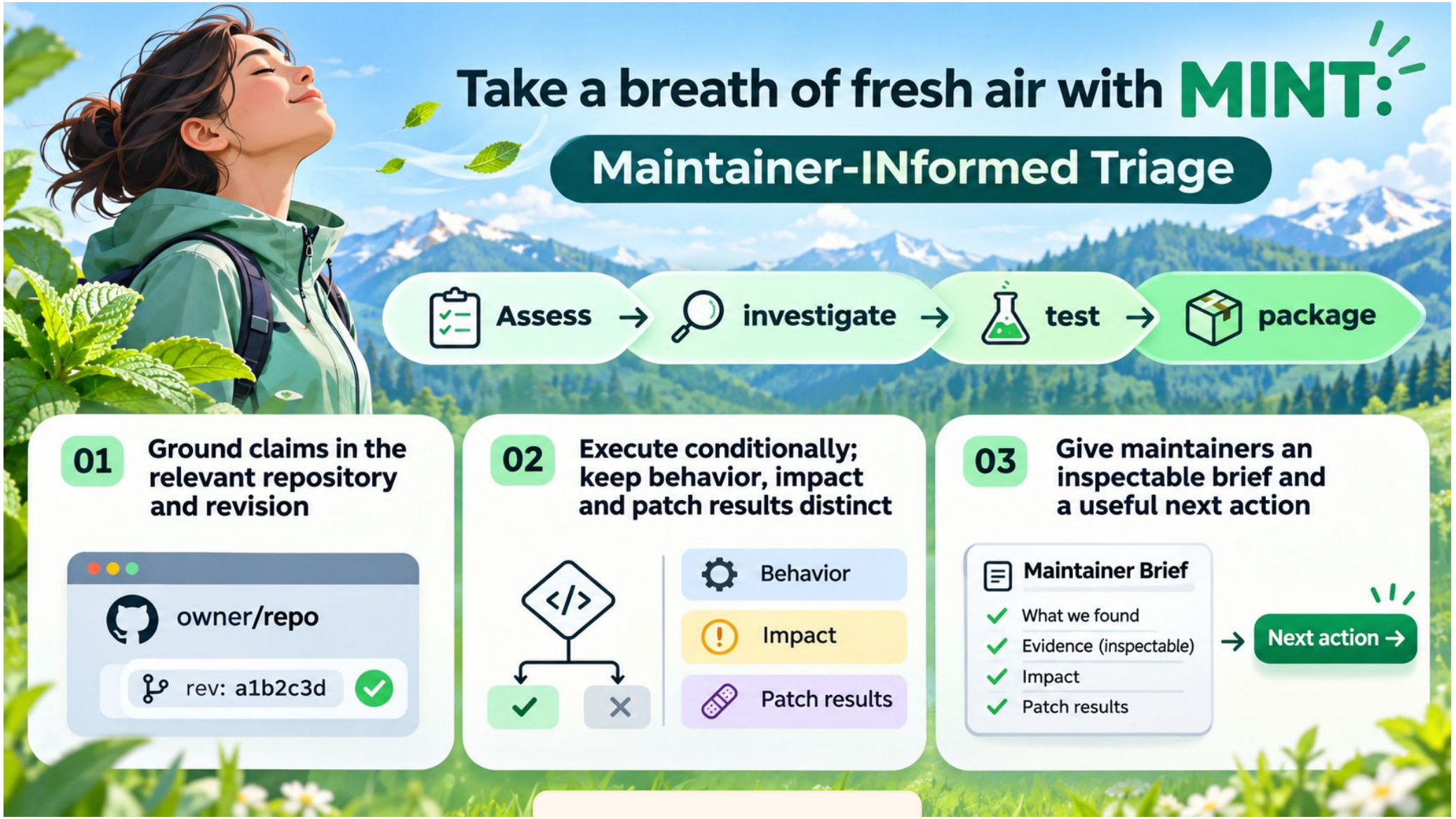
Execution and repair

- Claim-specific accuracy
- Coverage and completion
- Patch and control outcomes

Maintainer use

- Decision quality and effort
- Useful follow-up actions
- Cost and latency

Evaluation in-progress: report route coverage and unresolved cases alongside successful tests



✉️ **Reach me at:**
jessya1@uci.edu