

THREE TABLETOP SCENARIOS + A FIELD CHECKLIST

Incident Response Playbook for Your Next Supply Chain Attack

Three tabletop scenarios to practice containment, establish the Blast Radius, and decide when work can resume.

How to use this guide

When you learn that a dependency has been compromised, you may need to pause affected work before confirming whether it ran in your environment. The next `npm install` could fetch an affected version; an import or local test run could execute code already downloaded. An unapproved release may already be reaching customers. In each case, you need a fast, trusted way to reach someone who can stop affected jobs or suspend publishing.

This article is a tabletop starting point for AppSec engineers working with platform, identity, and endpoint teams. The three scenarios are fictional; adapt their systems and assumptions to your environment. Read each case through, pause at the decision, and compare your response with the walkthrough. Keep the [two-page incident response checklist](#) beside you to record owners, scope, and supporting evidence.

Before starting, identify a response lead, technical owners, backups, and a trusted communication channel. Agree who can pause development or publishing while the investigation proceeds.

A / SCENARIO

**Compromised
dependency**

B / SCENARIO

**Untrusted
pull request**

C / SCENARIO

**Suspected
insider misuse**

01 / SCENARIO A

A compromised dependency reaches a developer laptop

A researcher publishes an advisory identifying malware in an npm package version. Your project has no committed package-lock.json, and the version range in package.json allows the affected release. A CI build is due to run, and nobody has yet checked recent developer installs.

THE SITUATION

The platform owner pauses the affected CI workflows and cancels queued or running jobs. Developers receive instructions to stop installs, updates, tests, and builds in affected projects and report what they have already run. Confirm which jobs were stopped and when; track developer acknowledgments and environments still being checked.

If you have a malware-blocking proxy in place, such as the one in [Boost Developer Endpoint Protection](#), confirm that it rejects downloads of the affected version. Record where it is enforced and review available download records. Earlier downloads, cached copies, and environments outside its coverage still need investigation.

A developer reports installing the package earlier that morning and running the project's tests. In this scenario, the installation log identifies the affected version, and endpoint records show its code executing during the test run. Malicious code can run in an install script or later, when tests, build tools, or the application import the library. As [Shai-Hulud campaigns](#) demonstrate, local execution can expose credentials that allow further compromise and package publication.

PAUSE HERE

What access would you restrict immediately, and what records would establish whether the attacker went further? Use checklist 03A and 04-10.

THE RESPONSE

The endpoint owner isolates the laptop and preserves available evidence before rebuilding. From a trusted system, the identity owner revokes credibly exposed access, prioritizing publishing, administration, and production. They also check active sessions, refresh tokens, derived credentials, and newly created access; a replacement secret alone would leave those questions unresolved.

A saved query links a source-control token available on the laptop to an unexpected workflow run. Identity records connect that run to a cloud session, so the cloud owner restricts its access while investigators trace subsequent activity. The response record separates observed use from other reachable resources and explicitly lists gaps in audit log coverage.

The team restarts the reviewed CI jobs with verified dependency versions and replacement publishing credentials issued into a clean build environment. The laptop stays isolated until recovery and replacement access are validated; the cloud owner keeps access restricted until the suspect session is invalidated or expires. The status update tells developers which work can resume, what remains suspended, who is investigating, and when they will hear more. Removing the upstream package does not remove cached copies or credentials already taken.

TRACE THE NEXT USE

Laptop — Workflow — Cloud session

02 / SCENARIO B

A pull request crosses a trust boundary

B

THE SITUATION

A researcher reports that an active PR test run can feed attacker-controlled content into a privileged workflow. In an open-source project, the contribution might come from a public fork. In a large organization, the same boundary can exist between an internal contributor and another team's release authority.

For this exercise, a release workflow downloads the project archive produced by the PR job and runs its build script with publishing credentials available. The PR job itself had limited permissions, but its output is now being used by a job with greater access. The response lead contacts the platform owner, who cancels the PR job and pauses the release workflow.

PAUSE HERE

Does stopping the PR's job contain the incident if another workflow already consumed its output? Use checklist 03B, 07, and 10-11.

FOLLOW THE ARTIFACT

PR job

Limited permissions

Project archive

Release job

Publishing credentials available

THE RESPONSE

Preserve the triggering event, approved and executed commit SHAs, job logs, artifact identifiers, and downstream run records. Check the revision actually executed against the one reviewed. In this scenario, the release-job logs confirm that the build script already ran with publishing credentials available. Containment now includes that runner, its credentials, and any outputs it produced.

The investigation follows the artifact into the release run and checks for credential issuance, registry activity, and further access. A valid signature or provenance record can identify the producing workflow without establishing that its inputs were authorized or safe. Record suspect package versions and digests, known consumers, and distribution that remains unknown.

Before restarting, the platform team changes the release workflow so it no longer runs build scripts supplied by PR artifacts, reviews its permissions, and tests the revised workflow with harmless inputs. The incident lead checks that queued jobs, reruns, alternate triggers, and delegated automation cannot restore the same route. Suspect outputs are replaced through a trusted build and publishing path.

The release owner authorizes resumption for the validated workflow and artifacts, with unresolved outputs still quarantined. Developers receive precise instructions about which jobs may run; downstream recipients receive affected identifiers and mitigation guidance through the designated communication owner.

A release has no recorded approval

THE SITUATION

An engineer notices a new package version published overnight, when no release was scheduled. Publication records associate it with a familiar employee account, but the release owner finds no matching approval. The employee describes it as maintenance. The team checks for an authorized exception while restricting further publication.

An independent administrator restricts the implicated account's publishing access and protects evidence outside its administrative reach. The response lead involves designated insider-response, HR, and legal contacts through a restricted channel. The technical investigation separates the account's recorded actions, the authority granted to it, and the question of who operated it.

PAUSE HERE

Could another credential or service account restore the access you just removed? Use checklist 03C, 05-07, and 11.

REVOKE EVERY ROUTE

Implicated account

- **Original token**
Removed
- **New publishing credential**
Investigate and invalidate

THE RESPONSE

Audit records show that the account created another publishing credential before the release. Removing its original token therefore leaves a second route to investigate and contain. The identity owner inventories sessions, keys, applications, and delegated grants, then records invalidation results and any access that remains usable until expiry. Permissions and automation configurations show who could reverse those restrictions.

Investigators compare repository access, workflow changes, and publication events with approved roles and change records. These records establish actions and authorization; they do not, by themselves, establish an employee's intent. Preserve that distinction in updates and limit attribution to the people handling the investigation.

The release owner quarantines the suspect version, traces known downloads and deployments, and coordinates recipient guidance. Recovery includes reviewing the release inputs, replacing affected outputs, and restoring only the access required for the approved publication process. An independent owner verifies that the implicated identity and its delegated access cannot bypass or undo the restored controls.

The lesson applies wherever a contributor can exercise more authority than the task permits. Our research on [supply chain insider threats](#) explores how repository permissions and release controls can create that gap.

Establish the capabilities all three responses need

The first containment action differs across these cases, but the response record serves the same purpose: show what was stopped, what remains exposed, what the evidence supports, and who can authorize recovery. Checklist 12-16 keeps communication, trusted restoration, scoped resumption, and follow-up connected to those findings.

P1 / EXERCISE

Test your Blast Radius prediction

Choose a starting point from your threat model and trace direct and delegated access before an incident. Include the credentials, sessions, and automation available from a developer endpoint, as well as CI and publishing identities. Use simulation to identify where removing a permission or long-lived credential changes what an attacker could reach. Keep the map's assumptions visible and revisit it when access changes.

That predicted Blast Radius guides containment and investigation. During an incident, audit evidence establishes which paths were actually used. Reducing unnecessary access beforehand also reduces the number of paths responders must investigate.

BLAST RADIUS

Prediction

What could this identity reach?

Investigation

Which paths were actually used?

P2 / EXERCISE

Test your ability to investigate

Identify the records required for each path, their owners, retention, and collection health. Protect the archive from the identities it records, then verify that responders can retrieve and query it. An archive that takes hours to access delays every decision depending on it. Preserve queries, event identifiers, timelines, and coverage gaps so another responder can assess the conclusions.

During the exercise, identify who reviews dependency advisories, workflow activity, and publication records, and how a concern reaches someone authorized to act. Verify external updates through trusted sources; our [LiteLLM investigation](#) documented a compromised account closing a public disclosure. Record detection and effective-containment times separately, preserving unknowns when calculating mean time to detect (MTTD).

P3 / EXERCISE

Test containment and recovery decisions

Confirm who can act, including backups, and what evidence would justify resuming each system. Assign follow-up actions and verify procedures through safe technical tests. Exercise the ability to revoke access and restore the delivery process without relying on the implicated identity.

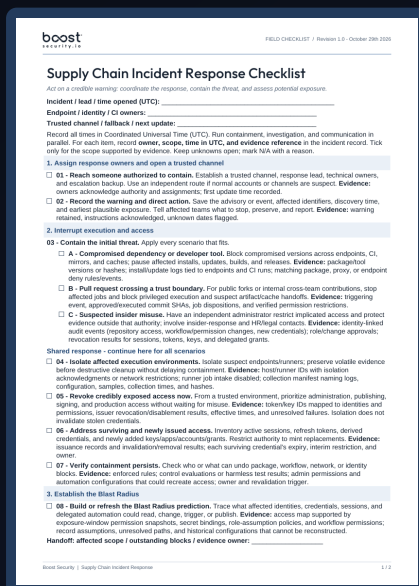
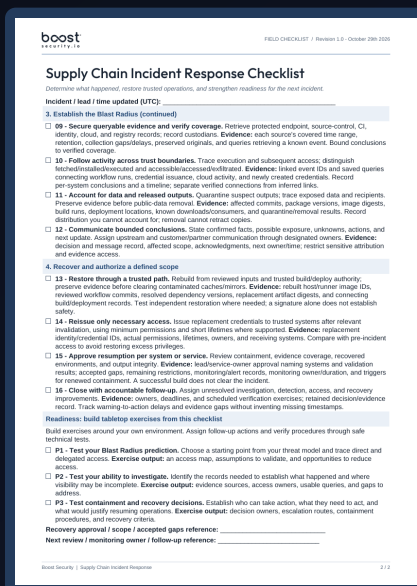
Use the findings to improve the starting conditions: enforce reviewed dependency selections, separate untrusted execution from privileged operations, constrain release authority, and minimize standing access. GitHub Actions [workflow lockfiles](#) became [generally available on October 28, 2026](#). Boost Security contributed to their [design and testing](#). Review what you lock and account separately for runtime downloads; locking a dependency does not establish its safety.

An upcoming Boost Security Labs article will examine how GitHub Actions hardening controls work together and how to verify their boundaries.

Use P1-P3 in the checklist to build exercises around your own environment. Keep the completed decisions and follow-up work with your runbooks, and keep the printable checklist accessible to the people who will respond when the next warning arrives.

ANNEX / TWO PRINTABLE PAGES

Keep the checklist at hand.



Supply Chain Incident Response Checklist

Act on a credible warning: coordinate the response, contain the threat, and assess potential exposure.

Incident / lead / time opened (UTC): _____

Endpoint / identity / CI owners: _____

Trusted channel / fallback / next update: _____

Record all times in Coordinated Universal Time (UTC). Run containment, investigation, and communication in parallel. For each item, record **owner**, **scope**, **time in UTC**, and **evidence reference** in the incident record. Tick only for the scope supported by evidence. Keep unknowns open; mark N/A with a reason.

1. Assign response owners and open a trusted channel

- ☐ **01 - Reach someone authorized to contain.** Establish a trusted channel, response lead, technical owners, and escalation backup. Use an independent route if normal accounts or channels are suspect. **Evidence:** owners acknowledge authority and assignments; first update time recorded.
- ☐ **02 - Record the warning and direct action.** Save the advisory or event, affected identifiers, discovery time, and earliest plausible exposure. Tell affected teams what to stop, preserve, and report. **Evidence:** warning retained, instructions acknowledged, unknown dates flagged.

2. Interrupt execution and access

03 - Contain the initial threat. Apply every scenario that fits.

- ☐ **A - Compromised dependency or developer tool.** Block compromised versions across endpoints, CI, mirrors, and caches; pause affected installs, updates, builds, and releases. **Evidence:** package/tool versions or hashes; install/update logs tied to endpoints and CI runs; matching package, proxy, or endpoint deny rules/events.
- ☐ **B - Pull request crossing a trust boundary.** For public forks or internal cross-team contributions, stop affected jobs and block privileged execution and suspect artifact/cache handoffs. **Evidence:** triggering event, approved/executed commit SHAs, job dispositions, and verified permission restrictions.
- ☐ **C - Suspected insider misuse.** Have an independent administrator restrict implicated access and protect evidence outside that authority; involve insider-response and HR/legal contacts. **Evidence:** identity-linked audit events (repository access, workflow/permission changes, new credentials); role/change approvals; revocation results for sessions, tokens, keys, and delegated grants.

Shared response - continue here for all scenarios

- ☐ **04 - Isolate affected execution environments.** Isolate suspect endpoints/runners; preserve volatile evidence before destructive cleanup without delaying containment. **Evidence:** host/runner IDs with isolation acknowledgments or network restrictions; runner job intake disabled; collection manifest naming logs, configuration, samples, collection times, and hashes.
- ☐ **05 - Revoke credibly exposed access now.** From a trusted environment, prioritize administration, publishing, signing, and production access without waiting for misuse. **Evidence:** token/key IDs mapped to identities and permissions, issuer revocation/disablement results, effective times, and unresolved failures. Isolation does not invalidate stolen credentials.
- ☐ **06 - Address surviving and newly issued access.** Inventory active sessions, refresh tokens, derived credentials, and newly added keys/apps/accounts/grants. Restrict authority to mint replacements. **Evidence:** issuance records and invalidation/removal results; each surviving credential's expiry, interim restriction, and owner.
- ☐ **07 - Verify containment persists.** Check who or what can undo package, workflow, network, or identity blocks. **Evidence:** enforced rules; control evaluations or harmless test results; admin permissions and automation configurations that could recreate access; owner and revalidation trigger.

3. Establish the Blast Radius

- ☐ **08 - Build or refresh the Blast Radius prediction.** Trace what affected identities, credentials, sessions, and delegated automation could read, change, trigger, or publish. **Evidence:** access map supported by exposure-window permission snapshots, secret bindings, role-assumption policies, and workflow permissions; record assumptions, unresolved paths, and historical configurations that cannot be reconstructed.

Handoff: affected scope / outstanding blocks / evidence owner: _____

Supply Chain Incident Response Checklist

Determine what happened, restore trusted operations, and strengthen readiness for the next incident.

Incident / lead / time updated (UTC): _____

3. Establish the Blast Radius (continued)

- ☐ **09 - Secure queryable evidence and verify coverage.** Retrieve protected endpoint, source-control, CI, identity, cloud, and registry records; record custodians. **Evidence:** each source's covered time range, retention, collection gaps/delays, preserved originals, and queries retrieving a known event. Bound conclusions to verified coverage.
- ☐ **10 - Follow activity across trust boundaries.** Trace execution and subsequent access; distinguish fetched/installed/executed and accessible/accessed/exfiltrated. **Evidence:** linked event IDs and saved queries connecting workflow runs, credential issuance, cloud activity, and newly created credentials. Record per-system conclusions and a timeline; separate verified connections from inferred links.
- ☐ **11 - Account for data and released outputs.** Quarantine suspect outputs; trace exposed data and recipients. Preserve evidence before public-data removal. **Evidence:** affected commits, package versions, image digests, build runs, deployment locations, known downloads/consumers, and quarantine/removal results. Record distribution you cannot account for; removal cannot retract copies.
- ☐ **12 - Communicate bounded conclusions.** State confirmed facts, possible exposure, unknowns, actions, and next update. Assign upstream and customer/partner communication through designated owners. **Evidence:** decision and message record, affected scope, acknowledgments, next owner/time; restrict sensitive attribution and evidence access.

4. Recover and authorize a defined scope

- ☐ **13 - Restore through a trusted path.** Rebuild from reviewed inputs and trusted build/deploy authority; preserve evidence before clearing contaminated caches/mirrors. **Evidence:** rebuilt host/runner image IDs, reviewed workflow commits, resolved dependency versions, replacement artifact digests, and connecting build/deployment records. Test independent restoration where needed; a signature alone does not establish safety.
- ☐ **14 - Reissue only necessary access.** Issue replacement credentials to trusted systems after relevant invalidation, using minimum permissions and short lifetimes where supported. **Evidence:** replacement identity/credential IDs, actual permissions, lifetimes, owners, and receiving systems. Compare with pre-incident access to avoid restoring excess privileges.
- ☐ **15 - Approve resumption per system or service.** Review containment, evidence coverage, recovered environments, and output integrity. **Evidence:** lead/service-owner approval naming systems and validation results; accepted gaps, remaining restrictions, monitoring/alert records, monitoring owner/duration, and triggers for renewed containment. A successful build does not clear the incident.
- ☐ **16 - Close with accountable follow-up.** Assign unresolved investigation, detection, access, and recovery improvements. **Evidence:** owners, deadlines, and scheduled verification exercises; retained decision/evidence record. Track warning-to-action delays and evidence gaps without inventing missing timestamps.

Readiness: build tabletop exercises from this checklist

Build exercises around your own environment. Assign follow-up actions and verify procedures through safe technical tests.

- ☐ **P1 - Test your Blast Radius prediction.** Choose a starting point from your threat model and trace direct and delegated access. **Exercise output:** an access map, assumptions to validate, and opportunities to reduce access.
- ☐ **P2 - Test your ability to investigate.** Identify the records needed to establish what happened and where visibility may be incomplete. **Exercise output:** evidence sources, access owners, usable queries, and gaps to address.
- ☐ **P3 - Test containment and recovery decisions.** Establish who can take action, what they need to act, and what would justify resuming operations. **Exercise output:** decision owners, escalation routes, containment procedures, and recovery criteria.

Recovery approval / scope / accepted gaps reference: _____

Next review / monitoring owner / follow-up reference: _____