SOFTWARE FACTORY · THREAT TAXONOMY · THE GAME'S REFERENCE

THE FACTORY IS THE TARGET

A Threat Taxonomy for the Software Supply Chain

This is the reference for the Software Factory card game: 221 threats at 17 stations of the software factory, from the developer's laptop to the last hop into production, plus 10 structural hazards. Each entry states the threat as a property of a component or a design decision, the principle a sound design follows, design measures, and mappings to STRIDE, CAPEC and CWE.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions.
Licensed under Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0): you may share the whole taxonomy, unchanged and with credit, for non-commercial purposes. In addition, whether or not the use is commercial, you may use it to threat model systems, including for paying clients; use it internally and for training inside any organisation; and quote individual entries, with credit, in paid courses, reports and talks. Shipping it in a product, such as a tool's built-in library, a printed deck or a reprint for sale, needs written permission.

Brett Crawley

ostering.com · Version 1.0, October 2026

HOW TO USE IT

Every entry has a reference, a threat, a mitigation principle and two to four design measures. The reference is the station code plus a playing-card rank, from Ace down to 2, or T1 to T10 for the structural hazards. It is a stable identifier, not a severity: in the card game the ranks follow severity because play is trick-based, but in the taxonomy a reference only names an entry, and it stays the same as the taxonomy grows. Where a threat has been seen in the wild, the entry names the public incident or disclosure. The taxonomy is the reference for the card game Software Factory, and the two share one set of references.

1. Draw the factory: every machine, repository, pipeline, runner, store, registry and cluster a change passes through on its way to production, with the trust boundaries and the credentials marked.
2. Take one station and walk every entry in it against each element it applies to.
3. For each entry, ask whether the design satisfies the principle. Write down a finding, or "no exposure" and the design decision that defeats it.
4. Where there is a finding, start from the measures given and adapt them to your architecture. They are the shape of a fix, not a prescription.
5. Finish with the structural hazards, played against the design as a whole.

STATIONS

STATION	NAME	PREFIX	ENTRIES	SCOPE
01	Workstation and Agents	**WS**	**13**	IDEs, extensions, dev containers, local credentials, AI coding assistants and agents, pre-commit hooks
02	Version Control	**VC**	**13**	Repositories, branches, tags, review, forks, the hosting platform, its apps and bots
03	Pipeline Definition	**PD**	**13**	Workflow files, triggers, expressions, reusable workflows, third-party actions and plugins
04	Runners	**RN**	**13**	Hosted and self-hosted build agents, caches, artifacts passed between jobs, isolation
05	Pipeline Identity and Secrets	**ID**	**13**	Secret stores, OIDC federation, tokens, service accounts, access to signing keys
06	Build Systems	**BS**	**13**	Make, Maven, Gradle, Bazel, compilers, toolchains, build plugins, remote cache and execution
07	Dependency Resolution	**DR**	**13**	Package managers, manifests, lockfiles, install scripts, dependency update bots
08	Scanners and Gates	**SG**	**13**	SAST, SCA, secrets, IaC, container, licence and DAST scanners; quality gates; policy checks
09	Provenance and Attestation	**PV**	**13**	SLSA provenance, in-toto attestations, signing, Sigstore, transparency logs, verification
10	Registries	**RG**	**13**	Public package and image registries, internal artifact repositories, proxies and mirrors

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

11	Firewall and Quarantine	**FW**	**13**	Dependency firewalls, curated proxies, cooldowns, quarantine, allow lists and exceptions
12	SBOM and VEX	**BM**	**13**	SBOM generation, storage and exchange, VEX statements, decisions driven by inventory
13	Infrastructure as Code	**IC**	**13**	Terraform and OpenTofu, Pulumi, CloudFormation, Ansible; state, modules, providers, plan and apply
14	Container Images	**IM**	**13**	Base images, Dockerfiles, layers, image builds, tags and digests
15	Container Runtime	**RT**	**13**	Runtimes, isolation boundaries, the host kernel, sandboxes
16	Orchestration	**OR**	**13**	Kubernetes API, RBAC, admission, operators, Helm, nodes, cluster networking
17	Delivery and Deployment	**DL**	**13**	CD tools, GitOps controllers, promotion, environments, approvals, rollback
T	Structural hazards	**T**	**10**	Hazards of the software factory as a whole

The mapping line under each threat. Every entry is mapped to STRIDE, to CAPEC 3.9 attack patterns and to CWE 4.20 weaknesses, for example `STRIDE E (Loose) · CAPEC Partial: CAPEC-554, CAPEC-36 · CWE Direct: CWE-420, CWE-424`. **STRIDE** gives the categories, best fit first; *Clean* means a STRIDE prompt would plausibly find the threat, *Loose* that STRIDE gives a label but not the question. For **CAPEC** and **CWE**, the rating applies to the list as a whole, and the IDs are in best-fit order: *Direct* means the first ID names the flaw, and any others cover further aspects of it; *Partial* means the IDs cover only part of the flaw, or a broader version of it; *None* means there is no matching pattern or weakness. Look an ID up at capec.mitre.org/data/definitions/<number>.html or cwe.mitre.org/data/definitions/<number>.html. The Excel, SKOS and STIX downloads carry the names as well as the IDs.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

LICENCE

© 2026 Jonathan Brett Crawley. This taxonomy is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International licence (CC BY-NC-ND 4.0), with the additional permissions below.

Licence text: https://creativecommons.org/licenses/by-nc-nd/4.0/

USE	PERMISSION NEEDED?
Sharing the whole taxonomy non-commercially (OWASP, universities, conferences, individuals)	**No**
Threat modeling with it, including for paying clients	**No**
Internal use and training inside any organisation	**No**
Quoting entries, with credit, in paid courses, reports and talks	**No**
Shipping the taxonomy in a product: a tool's built-in library, a printed deck, a reprint for sale	**Yes, ask me**

Additional permissions. In addition to the rights the licence gives, and whether or not the use is commercial, the copyright holder grants you permission to:

- use the taxonomy to threat model systems you build, operate or assess, including for paying clients;
- use it internally, and for training, inside any organisation;
- quote individual entries, with credit, in paid courses, reports, findings and talks.

Any other commercial use, including shipping the taxonomy or a substantial part of it in a product (a tool's built-in library, a printed deck, a reprint for sale), needs written permission. Ask me on LinkedIn.

Credit. “The Factory Is the Target: A Threat Taxonomy for the Software Supply Chain” by Jonathan Brett Crawley, ostering.com, licensed under CC BY-NC-ND 4.0.

These additional permissions are given alongside the licence. They add to what the licence allows and do not take anything away from it.

STATION 01 · LINE 1 · SOURCE TO ARTIFACT

WS WORKSTATION AND AGENTS (WS)

The developer's machine holds credentials to everything downstream and runs more unvetted code than any server. These cards cover IDEs, extensions, dev containers, local tooling, and the AI assistants and agents that now act with a developer's authority.

ID	THREAT	MITIGATION
WS·A	**CREDENTIAL HOARD** The workstation holds long-lived tokens for version control, registries, cloud and CI in dotfiles, keychains and CLI configs, and one piece of malware can collect them all. *Seen in: CircleCI session theft (2023); Shai-Hulud (2025).* STRIDE I, S (Clean) · CAPEC Partial: CAPEC-639, CAPEC-560 · CWE Partial: CWE-522, CWE-313	**Credentials on a developer machine are short-lived, bound to the device, and never sufficient on their own to publish or reach production.** 1 Replace static tokens in dotfiles and CLI configs with short-lived credentials issued through single sign-on, refreshed by a credential helper rather than stored as files. 2 Bind developer credentials to the device with hardware-backed keys, such as a platform authenticator or security key, so a copied token file does not work elsewhere. 3 Keep publish, protected-branch push and production rights out of developer credentials entirely, and grant them only to pipeline identities (see ID). 4 Tie every developer credential to the identity provider session, so disabling one user revokes them all, and rehearse that revocation against a time target.
WS·K	**AGENT WITH DEVELOPER AUTHORITY** An AI coding agent runs commands, edits files and calls tools with the developer's credentials, and content it reads (issues, READMEs, dependencies, web pages) can instruct it. *Seen in: Nx s1ngularity (2025); hackerbot-claw CLAUDE.md injection (2026).* STRIDE E, T (Clean) · CAPEC Partial: CAPEC-248 · CWE Direct: CWE-1427, CWE-250, CWE-441	**An agent acts under its own narrowly scoped identity, and any action with effect outside its sandbox needs a person's explicit approval.** 1 Run the agent in a container or VM with its own short-lived, least-privilege credentials, never the developer's session, keychain or SSH agent. 2 Require per-action human approval for shell commands, writes outside the workspace and tool calls that change remote state, with an allow list for read-only actions. 3 Treat repository text, issues, tool output and web content as data that cannot widen the agent's permissions or its approved tool set, whatever it says. 4 Limit the agent's network egress to an allow list of model endpoints, the package proxy and the hosting platform, and log every outbound request.
WS·Q	**MALICIOUS EXTENSION** IDE extensions run with the user's privileges, update automatically from a marketplace and are rarely reviewed. *Seen in: GlassWorm on Open VSX (2025).* STRIDE T, E (Clean) · CAPEC Direct: CAPEC-698 · CWE Direct: CWE-829, CWE-1357	**Only reviewed extensions from approved sources run in the IDE, at versions the organisation chose and can withdraw.** 1 Restrict extension installation to an approved marketplace or an internal mirror, enforced through managed IDE policy on every developer device. 2 Allow-list extensions by publisher and identifier, pin their versions, and turn off automatic updates so each new version is reviewed before rollout. 3 Keep secrets out of files the editor can open, and monitor extension host network traffic for destinations outside the expected set.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
WS·J	**UNTRUSTED WORKSPACE EXECUTION** Opening or cloning a repository runs code: editor tasks, dev container definitions, git hooks, language servers, build files evaluated on open. *Seen in: Contagious Interview repositories (2025–26).* STRIDE E, T (Clean) · CAPEC Direct: CAPEC-511, CAPEC-35 · CWE Direct: CWE-829, CWE-1188	**Opening or cloning a repository runs nothing until a person has decided to trust it.** 1 Enforce restricted mode for untrusted workspaces through managed editor settings, so tasks, debug launchers and workspace-recommended extensions do not run on open. 2 Disable automatic execution of repository-supplied hooks and build file evaluation by language servers until the workspace is explicitly trusted. 3 Pin dev container base images and features by digest, and allow them only from the internal registry.
WS·10	**HALLUCINATED DEPENDENCY** An assistant suggests a package name that does not exist, and an attacker registers it. *Seen in: Package hallucination study (Spracklen et al., 2024).* STRIDE S, T (Loose) · CAPEC Direct: CAPEC-616 · CWE Partial: CWE-829, CWE-1426	**A new dependency enters the code base only after its existence, identity and age have been checked against an approved source.** 1 Fail the build when a manifest names a package that is absent from the approved list or that the internal proxy has never served before. 2 Surface added and changed dependencies as a distinct section of the pull request, and require approval from a dependency owner for each new one. 3 Configure the dependency proxy to hold packages whose first publication is younger than a set number of days (see FW).
WS·9	**LOCAL BYPASS OF CONTROLS** A developer can build, sign or publish from their own machine, skipping every control in the pipeline. STRIDE T, E (Clean) · CAPEC Partial: CAPEC-554, CAPEC-443 · CWE Direct: CWE-424, CWE-266	**Releases can only come out of the pipeline, and a consumer can tell a pipeline artifact from any other.** 1 Issue publishing and signing credentials only to pipeline identities through OIDC trusted publishing and keyless signing, so no person ever holds them. 2 Generate SLSA provenance on an isolated build platform and make every consumer and deploy gate reject artifacts without valid provenance (see PV). 3 Alert on any registry publish or transparency log entry whose identity is not a registered release workflow.
WS·8	**TOOLCHAIN DRIFT** Local compilers, SDKs, CLIs and package managers differ from the pipeline's and are installed from unverified sources, such as piping a script to a shell. STRIDE T (Clean) · CAPEC Direct: CAPEC-670, CAPEC-185 · CWE Partial: CWE-494	**Developer and pipeline toolchains come from the same verified source at the same versions.** 1 Define the toolchain once as code, such as a pinned dev container image or a version manager file, and build both local and pipeline environments from it. 2 Install tools only through a package manager or internal mirror that verifies checksums or signatures, and remove pipe-to-shell installers from setup scripts. 3 Pin each tool installer's checksum in the toolchain definition, and alert the platform team when an upstream download no longer matches it.
WS·7	**PRE-COMMIT BLIND SPOT** Secrets and unsafe changes leave the machine because local checks are optional, skippable or absent. STRIDE I, T (Loose) · CAPEC Direct: CAPEC-207 · CWE Direct: CWE-602, CWE-540	**Anything that must not reach the repository is blocked by the server, and local checks only give earlier feedback.** 1 Enable server-side secret push protection on every repository and reject pushes that contain detected secrets. 2 Mirror every local pre-commit check as a required status check on protected branches, so skipping it locally changes nothing. 3 Distribute local hooks through a managed hook framework installed by default, and measure how often they are skipped.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
WS·6	**AGENT CONTEXT LEAKAGE** Assistants send source, secrets and internal context to third-party models, or retain them in logs. STRIDE I (Clean) · CAPEC None · CWE Direct: CWE-201, CWE-532	**An assistant receives only context the organisation has cleared for it, sends it only to approved destinations, and keeps it under known retention.** 1 Exclude secret files, environment files and credential paths from assistant indexing with an organisation-managed ignore policy. 2 Route assistant traffic only to approved model endpoints with agreed retention terms, and block other model endpoints at the network. 3 Scan prompts and attached context for secrets before they leave the machine, and redact or block any match. 4 Store assistant logs and transcripts on organisation-controlled storage with defined retention and access control.
WS·5	**AGENT-AUTHORED CHANGE** Code an assistant wrote is merged with less scrutiny, carrying vulnerabilities, encumbered snippets or attacker-steered logic. STRIDE T, R (Loose) · CAPEC Partial: CAPEC-444 · CWE Partial: CWE-1426	**Generated code meets at least the review bar of human code, and a named person is accountable for every change that merges.** 1 Mark agent-authored commits and pull requests with a commit trailer or a bot identity, and display the mark in the review view. 2 Record the person who requested each agent change as its accountable owner, and require approval from a second code owner before merge. 3 Run SAST, secret, licence and snippet-matching scans as required checks, with rules tuned to insecure patterns common in generated code.
WS·4	**REMOTE ENVIRONMENT EXPOSURE** Cloud workspaces, port forwards and tunnels expose a development environment, and its credentials, to the network. STRIDE I, E (Clean) · CAPEC Partial: CAPEC-180, CAPEC-121 · CWE Partial: CWE-306, CWE-613, CWE-668	**A remote development environment is reachable only by its owner through authenticated, expiring access, and holds no more credential than a laptop would.** 1 Require identity provider authentication on every forwarded port and tunnel, and forbid public or anonymous port visibility by organisation policy. 2 Expire tunnels and port forwards automatically after a short idle period and when the session ends. 3 Issue cloud workspace credentials per user and per session, and prevent other users from opening, sharing or snapshotting a workspace that holds them.
WS·3	**CLONED SECRETS AT REST** Local clones, caches and shell history hold secrets and private source that leave with a lost device or a synced backup. STRIDE I (Clean) · CAPEC Partial: CAPEC-507, CAPEC-675 · CWE Direct: CWE-313, CWE-530	**Source and credentials on a device stay protected at rest and can be removed when the device is lost or retired.** 1 Enforce full-disk encryption and remote wipe through device management, and allow private repositories to be cloned only from managed devices using conditional access. 2 Store tokens in the operating system keychain rather than plain files, and configure shells to keep commands that carry secrets out of history. 3 Exclude source directories, tool caches and credential files from personal cloud backup and file sync.
WS·2	**CONFIGURATION SPRAWL** User-level config files redirect registries, hooks or credentials and differ from machine to machine. STRIDE T (Clean) · CAPEC Partial: CAPEC-75, CAPEC-154 · CWE Partial: CWE-15, CWE-427	**Where tools fetch from and what they run is set centrally, and neither a user file nor a repository can silently redirect it.** 1 Distribute package manager, registry and git configuration through device management, and lock the registry setting to the internal proxy. 2 Ignore repository-level configuration that changes registries, hooks or credential helpers unless a person approves it for that repository. 3 Report each machine's effective tool configuration to a central inventory, and alert on registries or hooks outside the approved set.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

VC

STATION 02 · LINE 1 · SOURCE TO ARTIFACT

VERSION CONTROL (VC)

The repository is the system of record for intent: every later control assumes that what is in it is what its owners meant. These cards cover repositories, branches, tags, review, identities and the hosting platform itself.

ID	THREAT	MITIGATION
VC·A	**PROTECTED BRANCH BYPASS** Admins, bots, apps or rule exceptions can push to a release branch or merge without the required review. *Seen in: Required reviews bypass via GitHub Actions (Cider, 2021).* STRIDE E, T (Clean) · CAPEC Partial: CAPEC-180, CAPEC-554 · CWE Partial: CWE-863, CWE-266, CWE-654	**Review requirements on release branches apply to every actor, and any exception is narrow, owned and visible.** 1 Enforce branch protection through rulesets that also bind administrators, and remove bypass rights from apps and bots that lack a named owner and reason. 2 Require approval from a code owner other than the most recent pusher, so an author cannot approve their own change. 3 Provision exactly one platform account per person from the identity provider, and audit approvals made by bots or service accounts. 4 Alert on every use of a bypass permission and on every change to the ruleset itself.
VC·K	**MUTABLE TAGS AND REFS** Tags and branches that consumers treat as releases can be moved or force-pushed to different commits. *Seen in: tj-actions/changed-files (2025); trivy-action (2026).* STRIDE T (Clean) · CAPEC Partial: CAPEC-445, CAPEC-669 · CWE Direct: CWE-471, CWE-494	**A published release reference always resolves to the same content, and consumers depend on content identity rather than names.** 1 Protect release tags with rulesets that block update, deletion and force-push, and use immutable releases where the platform offers them. 2 Reference dependencies, actions and deploy targets by full commit SHA or content digest instead of tag or branch name. 3 Raise an alert from the audit log stream on any tag move or force-push to a protected ref.
VC·Q	**PLATFORM ACCOUNT TAKEOVER** A maintainer or organisation owner account is taken over through phishing, weak MFA, session theft or a reset flaw, giving control of every repository it owns. *Seen in: Gentoo GitHub organisation (2018).* STRIDE S (Clean) · CAPEC Direct: CAPEC-98, CAPEC-593 · CWE Direct: CWE-1390, CWE-640, CWE-613	**Write and admin access depends on credentials that resist phishing and session theft, and high-impact actions need fresh proof of identity.** 1 Enforce phishing-resistant MFA, such as passkeys or FIDO2 security keys, through SSO for every member with write or admin rights. 2 Keep platform sessions short and bound to managed devices with conditional access at the identity provider. 3 Require re-authentication for sensitive actions, including visibility changes, deploy key creation, repository transfer and ruleset edits. 4 Harden account recovery so that email access alone cannot reset an account, and notify organisation owners of every recovery event.
VC·J	**OVER-PRIVILEGED APPS AND TOKENS** Installed apps, bots, deploy keys and personal access tokens hold broad, long-lived write access across many repositories. *Seen in: Stolen Heroku and Travis CI OAuth tokens (2022).* STRIDE E, S (Clean) · CAPEC Partial: CAPEC-560, CAPEC-21 · CWE Direct: CWE-266, CWE-1220	**Every app and token holds the narrowest repository scope and permissions it needs, for a limited time, with a known owner.** 1 Replace classic personal access tokens and broad deploy keys with fine-grained expiring tokens or app installations scoped to named repositories. 2 Require organisation approval for new app installations and fine-grained tokens, and enforce a maximum token lifetime. 3 Maintain an owned inventory of apps, tokens and deploy keys, review it on a fixed schedule, and revoke anything unused since the last review.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
VC·10	**MALICIOUS MAINTAINER** A contributor builds trust over time, gains maintainer rights, then introduces malicious change. *Seen in: xz Utils, CVE-2024-3094 (2024).* STRIDE T, E (Loose) · CAPEC Direct: CAPEC-443, CAPEC-538 · CWE Partial: CWE-506, CWE-654	**Release power is granted gradually, never held by one person alone, and exercised under review that does not rest on trust in an individual.** 1 Grant maintainer rights through a documented process, with a probation period before the new maintainer can tag or publish releases. 2 Require two-person review for releases and for changes to build, release and pipeline files, enforced by code owners and rulesets. 3 Add a mandatory second reviewer to changes from recently promoted maintainers for a defined period. 4 Review maintainer and release rights on a schedule and remove those held by inactive accounts.
VC·9	**REVIEW SUBVERSION** A change that looks benign in review hides behaviour: invisible or bidirectional Unicode, very large or generated diffs, binary test fixtures. *Seen in: Trojan Source, CVE-2021-42574 (2021); xz Utils test files (2024).* STRIDE T (Loose) · CAPEC Partial: CAPEC-636, CAPEC-267 · CWE Partial: CWE-451, CWE-1007	**What reviewers see is what the build uses, and content that cannot be meaningfully reviewed is regenerated or refused.** 1 Render invisible and bidirectional Unicode visibly in the review view, and fail a required check on unexpected control characters in source. 2 Rebuild generated files and binary fixtures from source in the pipeline, and fail the check when the committed copy differs. 3 Set a maximum diff size for a single approval, and require changes above it to be split before review.
VC·8	**COMMIT IDENTITY SPOOFING** Author and committer fields are unverified, so a commit can claim to be anyone, and signature checks are optional. STRIDE S, R (Clean) · CAPEC Direct: CAPEC-692 · CWE Direct: CWE-807, CWE-347	**Any authorship a decision relies on is cryptographically verified and bound to a person through the identity provider.** 1 Enforce signed commits on protected branches through rulesets. 2 Bind signing keys to accounts by registering them with the platform, or use keyless commit signing with Sigstore tied to the identity provider. 3 Base auto-merge and ownership decisions on the authenticated pusher or verified signer, never on the free-text author field.
VC·7	**REPOJACKING** A renamed or deleted organisation or repository name is re-registered, and anything that references the old name fetches from the new owner. *Seen in: Repojacking research (2020 onward).* STRIDE S, T (Clean) · CAPEC Direct: CAPEC-695 · CWE Direct: CWE-706, CWE-829	**References to external repositories resolve to a verified owner and content, not to whoever holds the name today.** 1 Reference external repositories and actions by full commit SHA, and mirror the critical ones into the organisation. 2 Configure fetch tooling to fail on repository redirects instead of following them silently. 3 Scan build and workflow files for references to renamed or deleted owners, and keep the organisation's own retired names registered.
VC·6	**SECRETS IN HISTORY** A credential committed once survives deletion in history, forks, caches and mirrors. STRIDE I (Clean) · CAPEC Direct: CAPEC-37 · CWE Direct: CWE-540, CWE-798	**A secret that reaches version control is treated as compromised, and pushes that carry secrets are stopped at the server.** 1 Enable server-side secret push protection on every repository, with bypass limited to named reviewers and logged. 2 Revoke and rotate any secret found in history at once, and treat history rewriting as clean-up rather than remediation. 3 Restrict forking of private repositories, and limit read access to history and forks to people who need it.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
VC·5	**VISIBILITY LEAK** Private code becomes reachable through a visibility change, a fork, or data from deleted and private forks that the network still serves. *Seen in: Cross Fork Object Reference (Truffle Security, 2024).* STRIDE I (Clean) · CAPEC Partial: CAPEC-87 · CWE Direct: CWE-527, CWE-459	**Private code stays private unless a deliberate, approved change says otherwise, including every copy held in forks.** 1 Restrict repository visibility changes to organisation owners, and alert on every change. 2 Disable forking of private and internal repositories by organisation policy. 3 Assume any commit ever pushed into a public fork network stays fetchable by hash, and rotate secrets and review content whenever a repository was ever public.
VC·4	**HOSTING PLATFORM VULNERABILITY** A self-hosted VCS server is internet-facing and holds every repository and runner registration token. *Seen in: Bitbucket Server CVE-2022-36804 (2022).* STRIDE E, I (Clean) · CAPEC Partial: CAPEC-88 · CWE Partial: CWE-1395, CWE-668	**A self-hosted platform presents a small attack surface, is patched quickly, and does not hand over every downstream credential when breached.** 1 Apply security releases to the server within a defined window, and report patch compliance to the platform owner. 2 Place the server behind a VPN or identity-aware proxy rather than exposing it directly to the internet. 3 Use ephemeral runners with short-lived registration and federated identity, so the server stores few credentials worth stealing.
VC·3	**AUDIT GAP** Changes to protection rules, webhooks, apps and membership are not logged, retained or reviewed. STRIDE R (Clean) · CAPEC Partial: CAPEC-268, CAPEC-571 · CWE Direct: CWE-778	**Every change to the controls that guard repositories is recorded outside the platform and reviewed.** 1 Stream audit logs to a separate log store that platform administrators cannot modify or delete. 2 Alert on changes to rulesets, webhooks, app installations, membership and visibility. 3 Retain audit logs for at least the organisation's detection and investigation window, and test that a year-old event can still be retrieved.
VC·2	**WEBHOOK EXPOSURE** Webhooks deliver events to endpoints that are unauthenticated or attacker-changeable, or carry private data. STRIDE S, I (Clean) · CAPEC Partial: CAPEC-194, CAPEC-154 · CWE Partial: CWE-923, CWE-940, CWE-201	**Webhook deliveries are authenticated, go only to approved endpoints, and carry no more than the receiver needs.** 1 Limit who can add or change webhooks to repository and organisation administrators, and alert on every change. 2 Sign every delivery with a per-webhook secret and verify the HMAC signature at the receiver before processing. 3 Send minimal payloads over TLS, and let the receiver fetch further detail with its own scoped credential.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

PD

STATION 03 · LINE 1 · SOURCE TO ARTIFACT

PIPELINE DEFINITION (PD)

A pipeline definition is code that runs with more privilege than the people who edit it. These cards cover workflow files, triggers, expressions, reusable templates, and the third-party actions, orbs and plugins they call.

ID	THREAT	MITIGATION
PD·A	**POISONED PIPELINE EXECUTION** Anyone who can change a workflow file, or a file it runs (scripts, makefiles, test config), in a branch or pull request gets the pipeline to execute their code with its privileges. *Seen in: OWASP CICD-SEC-4; Trivy pull_request_target (2026).* STRIDE E, T (Clean) · CAPEC Partial: CAPEC-678, CAPEC-75 · CWE Direct: CWE-829, CWE-250	**Code from an untrusted change runs only without secrets or write access, and protected runs execute definitions taken from protected branches.** 1 Run pull request builds from forks and unprotected branches with a read-only token and no secrets. 2 Load the workflow definition and the scripts it calls from the protected branch for every protected run, never from the change under test. 3 Assign code owners to workflow files, build scripts and test configuration, so changes to them need review by the pipeline owners.
PD·K	**PRIVILEGED TRIGGER ON UNTRUSTED INPUT** Triggers such as pull_request_target or workflow_run run with secrets and write tokens while checking out or processing a fork's code. *Seen in: Trivy pull_request_target (2026).* STRIDE E (Clean) · CAPEC Partial: CAPEC-234 · CWE Direct: CWE-829, CWE-653	**A run that holds secrets or write tokens never executes or interprets code supplied by an untrusted contributor.** 1 Forbid privileged triggers such as pull_request_target and workflow_run from checking out or running the contributor's commit, enforced by a workflow linter. 2 Split the work into an unprivileged job that builds the change and a privileged job that consumes only inert, validated outputs from it. 3 Require maintainer approval before workflows run for first-time and outside contributors.
PD·Q	**EXPRESSION INJECTION** Untrusted text (pull request titles, branch names, issue bodies, commit messages) is interpolated into shell commands by the pipeline's expression syntax. *Seen in: Ultralytics branch-name injection (2024).* STRIDE E, T (Clean) · CAPEC Direct: CAPEC-88 · CWE Direct: CWE-78	**Event data never becomes executable code, whatever its content.** 1 Pass event fields into steps through environment variables and reference them quoted in the script, never through inline expressions. 2 Make a workflow linter that flags untrusted context fields in run blocks a required check on every repository. 3 Validate fields such as branch names and versions against an allowed pattern before any step uses them.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
PD·J	**UNPINNED THIRD-PARTY STEP** Workflows reference actions, orbs or plugins by a mutable tag or branch, so the publisher, or whoever compromises the publisher, decides what runs. *Seen in: tj-actions (2025); trivy-action (2026).* STRIDE T (Clean) · CAPEC Partial: CAPEC-538, CAPEC-186 · CWE Direct: CWE-829, CWE-494	**Every third-party step runs code the organisation has reviewed, at an identity its publisher cannot change.** 1 Pin actions, orbs and plugins to a full commit SHA, and enforce the requirement through organisation policy. 2 Allow only listed actions and publishers through the organisation's action settings. 3 Review an action's code before first use and on every pin update, with a dependency update bot proposing each update as a reviewable pull request.
PD·10	**TRANSITIVE STEP DEPENDENCY** A pinned action itself pulls unpinned actions, container images or scripts at run time. *Seen in: reviewdog to tj-actions chain, link likely (2025).* STRIDE T (Loose) · CAPEC Partial: CAPEC-446, CAPEC-538 · CWE Direct: CWE-829, CWE-494	**Pinning covers everything a step executes, not only the step itself.** 1 Pin the container images used by actions and composite actions by digest, and reject actions that reference nested actions by tag. 2 Generate an inventory of every action, image and script a workflow executes from the resolved run graph, and diff it on each change. 3 Fork or vendor third-party actions that fetch unpinned content at run time, and pin the internal copy.
PD·9	**REMOTE SCRIPT EXECUTION** Pipeline steps download and run scripts from URLs without checking their integrity. *Seen in: Codecov Bash Uploader (2021).* STRIDE T (Clean) · CAPEC Direct: CAPEC-185 · CWE Direct: CWE-494	**No pipeline step executes fetched code unless its integrity has been checked against a value the organisation controls.** 1 Vendor remote scripts into the repository so the pipeline runs the reviewed copy. 2 Verify a pinned checksum or signature before executing any script that must still be fetched, and fail the step on mismatch. 3 Detect download-and-execute patterns with a workflow linter run as a required check.
PD·8	**INSUFFICIENT FLOW CONTROL** One identity can take a change from commit to production with no independent approval. *Seen in: OWASP CICD-SEC-1; Amazon Q Developer extension (2025).* STRIDE E, R (Clean) · CAPEC Partial: CAPEC-443, CAPEC-122 · CWE Partial: CWE-654, CWE-841	**No single identity can take a change from commit to production, and approval is enforced where deployment happens.** 1 Require approval from someone other than the author before merge, enforced by rulesets. 2 Protect the deployment environment with required reviewers, so the deploy job itself waits for an independent approval. 3 Restrict manual dispatch of deploy workflows to named roles, and record who started each run and with which inputs.
PD·7	**ENVIRONMENT TAMPERING BETWEEN STEPS** A step can change the environment, path or configuration that later steps and jobs inherit. STRIDE T (Clean) · CAPEC Direct: CAPEC-13, CAPEC-38 · CWE Direct: CWE-454, CWE-427	**Each step starts from an environment it can trust, and state passed between steps and jobs is explicit and validated.** 1 Place untrusted steps in a separate job from trusted ones, so they cannot write to the shared environment, path or output files. 2 Pass data between jobs only through declared outputs or artifacts, and validate it in the consuming job. 3 Feed dynamic pipeline generators only from trusted sources, and validate generated definitions against a schema before running them.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
PD·6	**SHARED TEMPLATE BLAST RADIUS** A reusable workflow or pipeline library used by hundreds of repositories is changed with weak review, and the change reaches every pipeline at once. STRIDE T (Loose) · CAPEC Partial: CAPEC-445 · CWE Partial: CWE-829	**A change to a shared template is reviewed and released as a change to every consumer, and consumers adopt it deliberately.** 1 Require platform team code owner review and two approvals for every change to a shared template. 2 Have consumers reference templates by full commit SHA or an immutable version, updated through reviewed pull requests. 3 Roll template changes out in stages, starting with a small group of consumers and watching for failures before wider release.
PD·5	**PARAMETER INJECTION** Runs started by hand or by API accept inputs (versions, targets, script fragments) that change what is built or deployed. STRIDE T, E (Clean) · CAPEC Partial: CAPEC-77 · CWE Direct: CWE-15, CWE-78	**A run accepts inputs only from authorised people, only from a known set of values, and records what it received.** 1 Restrict who can start runs by hand or by API to named roles holding scoped tokens. 2 Declare inputs with types and allowed values, and never interpolate free-text input into a script. 3 Record every input in the run's provenance and audit log (see PV).
PD·4	**COMMENT AND BOT TRIGGERS** Runs start on comments, labels or bot events that an outsider or an automation can produce. *Seen in: hackerbot-claw campaign (2026).* STRIDE S, E (Clean) · CAPEC Partial: CAPEC-1 · CWE Direct: CWE-862, CWE-807	**A comment, label or bot event starts a run only after the actor's permission has been verified, not just the text.** 1 Check the actor's repository permission through the platform API before a workflow acts on a comment or label command. 2 Give comment-triggered runs on outsider changes the same unprivileged context as fork pull requests. 3 Treat bot-authored pull requests as untrusted unless the bot is listed and its change touches only expected files.
PD·3	**CONDITIONAL SKIP** Security steps run only on certain branches, paths or commit messages, which an author can arrange to avoid. STRIDE T, E (Loose) · CAPEC Partial: CAPEC-554 · CWE Direct: CWE-807, CWE-636	**Required security checks run on every change that can reach a protected branch, and a missing result blocks the merge.** 1 Mark security checks as required in rulesets, so a skipped or absent check blocks merge. 2 Remove path, branch and commit-message conditions from security steps, or report an explicit failure whenever a condition skips them. 3 Disable skip markers for required workflows by organisation policy.
PD·2	**DEFINITION DRIFT** Pipelines are copied between repositories and diverge, so insecure patterns persist unseen. STRIDE T (Loose) · CAPEC None · CWE None	**Every pipeline is known, and security policy reaches it centrally rather than through copies.** 1 Build an inventory of every pipeline definition across the organisation from the hosting platform's API, and refresh it on a schedule. 2 Enforce mandatory checks through organisation-level required workflows or rulesets that do not depend on each repository's file. 3 Scan definitions for deprecated and insecure patterns on a schedule, and open an issue for each finding with an owner.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

STATION 04 · LINE 1 · SOURCE TO ARTIFACT

RUNNERS (RN)

The runner is where untrusted code and the factory's secrets share a machine. These cards cover CI controllers, hosted and self-hosted build agents, caches, artifacts passed between jobs, and the isolation between them.

ID	THREAT	MITIGATION
RN·A	**PERSISTENT RUNNER** A self-hosted runner reused across jobs lets one job leave a backdoor for the next, including jobs from other repositories or untrusted pull requests. *Seen in: PyTorch runners (Stawinski and Khan, 2024).* STRIDE T, E (Clean) · CAPEC Partial: CAPEC-124 · CWE Direct: CWE-653, CWE-459	**No state from one job is visible to the next, and untrusted code never runs on hosts that serve trusted jobs.** 1 Use ephemeral just-in-time runners that accept a single job and are destroyed when it ends. 2 Prevent public repositories and fork pull requests from scheduling jobs on self-hosted runners. 3 Rebuild runner hosts from a known image, and verify in a test job that nothing written by an earlier job is present.
RN·K	**CI CONTROLLER COMPROMISE** The CI server is reachable from the internet and holds every pipeline's credentials, so one vulnerability in it exposes them all. *Seen in: TeamCity CVE-2024-27198 (2024); Jenkins CVE-2024-23897 (2024).* STRIDE E, I (Clean) · CAPEC Partial: CAPEC-115, CAPEC-6 · CWE Partial: CWE-1395, CWE-668	**The CI controller is hard to reach, quickly patched, and holds few credentials worth stealing.** 1 Put the controller behind an identity-aware proxy or VPN, exposing only a webhook receiver that verifies signatures. 2 Patch the controller and its plugins within a defined window, and remove plugins that no pipeline uses. 3 Replace credentials stored in the controller with federated short-lived identities issued per job (see ID).
RN·Q	**CACHE POISONING** A low-privilege job writes to a build or dependency cache that a privileged job later restores. *Seen in: Ultralytics cache poisoning (2024).* STRIDE T (Clean) · CAPEC Direct: CAPEC-141 · CWE Partial: CWE-349, CWE-669, CWE-732	**A trusted job restores only cache content written by jobs of equal or higher trust, and verifies it before use.** 1 Scope cache keys by branch and trust level, so untrusted runs cannot write entries that protected branches read. 2 Build releases with an empty cache, or restore only caches written by protected-branch runs. 3 Verify restored dependencies against lockfile hashes before the build uses them.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
RN·J	**MEMORY AND PROCESS SCRAPING** Any step on the runner can read the memory or environment of the runner process and of other steps, and so every secret the job holds. *Seen in: tj-actions (2025); Trivy (2026).* STRIDE I, E (Clean) · CAPEC Partial: CAPEC-545 · CWE Direct: CWE-653, CWE-214, CWE-526	**A secret is present only in the step that needs it, and no step running untrusted code shares a host with it.** 1 Run untrusted steps in separate jobs on separate runners from steps that hold secrets. 2 Fetch each secret inside the step that uses it through a short-lived federated credential, rather than exposing it to the whole job. 3 Start the runner agent as a distinct unprivileged user, and restrict process tracing so steps cannot read its memory or environment.
RN·10	**ARTIFACT POISONING BETWEEN JOBS** An artifact produced by an untrusted run is consumed by a trusted run without verification. *Seen in: Artifact poisoning (Legit Security, 2022).* STRIDE T (Clean) · CAPEC Partial: CAPEC-184 · CWE Partial: CWE-349, CWE-669, CWE-345	**An artifact crosses a trust boundary only with verified origin and integrity, and is unpacked where it cannot overwrite trusted files.** 1 Record a digest and the producing run for each artifact, and verify both in the consuming trusted job. 2 Handle artifacts from untrusted runs as data only, never as code, scripts or configuration for the trusted job. 3 Extract artifacts into a dedicated directory outside the workspace, and reject archives containing absolute paths or traversal entries.
RN·9	**RUNNER NETWORK REACH** Runners sit on networks with routes to production, internal services or the cloud metadata endpoint. STRIDE E, I (Clean) · CAPEC None · CWE Partial: CWE-653	**A runner can reach only what its job needs, and runners for untrusted code are isolated from release and production networks.** 1 Place runners in dedicated network segments with no routes to production or internal services. 2 Block the cloud metadata endpoint from jobs, or require IMDSv2 with a hop limit of one. 3 Separate runners for untrusted pull requests from release runners at the network level, and test the separation from each side.
RN·8	**OPEN EGRESS** Runners have unrestricted internet egress, so stolen data leaves unnoticed. *Seen in: Trivy (2026).* STRIDE I (Clean) · CAPEC None · CWE None	**Runners send traffic only to known destinations, and any other attempt is blocked and visible.** 1 Route runner egress through a proxy that allows only listed registries, mirrors and the hosting platform. 2 Log DNS queries and HTTP egress per job, tagged with the repository and run. 3 Alert on connection attempts to destinations outside the allow list.
RN·7	**ROGUE RUNNER REGISTRATION** A leaked registration token or loose runner group lets an attacker attach their own machine as a runner and receive jobs and secrets. STRIDE S (Clean) · CAPEC Partial: CAPEC-616, CAPEC-560 · CWE Direct: CWE-923, CWE-1390	**Only machines the platform provisioned can join as runners, and every new registration is visible.** 1 Register runners through provisioning automation with short-lived, single-use tokens, and keep no long-lived registration token. 2 Restrict each runner group to named repositories and workflows. 3 Alert on every runner registration from the audit log, and flag any that provisioning did not request.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
RN·6	**CONTAINER SOCKET EXPOSURE** Jobs get the host's container socket or privileged mode to build images, which amounts to root on the runner. `STRIDE E (Clean) · CAPEC Partial: CAPEC-480 · CWE Direct: CWE-250, CWE-653`	**Building images does not give a job control of the host.** 1 Build images rootless, using BuildKit in rootless mode, Buildah or Kaniko. 2 Forbid mounting the container socket and running privileged containers in job configuration, enforced by runner policy. 3 Move any build that truly needs privileged features into a single-use VM or microVM.
RN·5	**SHARED TENANT RUNNER** Runners are shared across teams or customers, so one job can observe or affect another's. `STRIDE I, T (Clean) · CAPEC Partial: CAPEC-124, CAPEC-546 · CWE Direct: CWE-653, CWE-226`	**Jobs from different tenants are separated by a boundary as strong as the trust gap between them.** 1 Isolate jobs from different projects or customers in separate VMs or microVMs, never in containers on a shared kernel. 2 Keep caches, workspaces and temporary storage per tenant. 3 Prevent jobs from listing or inspecting other jobs through the host, the runner agent or the controller API.
RN·4	**SECRETS IN LOGS AND ARTIFACTS** Secrets reach build logs, uploaded artifacts or crash dumps where masking does not apply. *Seen in: ArtiPACKED (Unit 42, 2024); tj-actions log leak (2025).* `STRIDE I (Clean) · CAPEC Partial: CAPEC-37, CAPEC-150 · CWE Direct: CWE-532, CWE-538, CWE-528`	**Secrets are kept out of logs and artifacts by design, with masking as a backstop rather than the control.** 1 Register derived and encoded forms of each secret for masking, and prefer short-lived credentials so a leaked value expires quickly. 2 Exclude credential files and the checkout's git configuration from uploaded artifacts and crash dumps. 3 Scan logs and artifacts for secrets after each run, and restrict log access on public projects to maintainers.
RN·3	**RUNNER IMAGE SUPPLY CHAIN** The runner's base image and preinstalled tools are unverified or out of date. `STRIDE T (Clean) · CAPEC Partial: CAPEC-670 · CWE Direct: CWE-1357, CWE-494, CWE-1395`	**Runner images are built from verified sources, patched on a clock, and traceable to what they contain.** 1 Build runner images in the organisation's own pipeline from base images pinned by digest, and sign the result. 2 Rebuild and roll out runner images on a fixed patch schedule, and retire images older than the window. 3 Verify checksums or signatures of preinstalled tools during the image build, and publish an SBOM for each image.
RN·2	**RESOURCE HIJACKING** Free or shared runners are abused for mining or as attack infrastructure, helped by public triggers. `STRIDE D, E (Clean) · CAPEC Partial: CAPEC-212 · CWE Direct: CWE-770, CWE-862`	**Compute is spent only on jobs that authorised people started, within set limits.** 1 Require approval before workflows run for first-time contributors and forks. 2 Set per-job time, CPU and concurrency limits on every runner pool. 3 Alert on unusual runtime, CPU or spend patterns per repository.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID

STATION 05 · LINE 1 · SOURCE TO ARTIFACT

PIPELINE IDENTITY AND SECRETS (ID)

The pipeline is usually the most privileged identity in the organisation, and most attacks on it end in credential theft. These cards cover secret stores, federated identity, job tokens, service accounts and access to signing keys.

ID	THREAT	MITIGATION
ID·A	**LONG-LIVED STATIC CREDENTIAL** Pipelines reach clouds, registries and clusters with static keys that never expire and work from anywhere. *Seen in: Codecov (2021); CircleCI (2023).* STRIDE S, I (Clean) · CAPEC Partial: CAPEC-560 · CWE Partial: CWE-262	**A pipeline proves its identity at run time and receives credentials that expire with the job and work only from it.** 1 Replace static cloud, registry and cluster keys with OIDC workload identity federation that issues short-lived tokens per job. 2 Restrict any remaining static credential to the runners' network with an IP or conditional access policy, and rotate it on a schedule. 3 Track every remaining static credential in an inventory with an owner and a removal date.
ID·K	**LOOSE FEDERATION CLAIMS** Cloud trust policies accept workload identity tokens whose repository, branch or environment claims are matched too broadly, so another repository, fork or branch can assume the role. *Seen in: OIDC trust policy research (Datadog, Rezonate, 2023).* STRIDE S, E (Clean) · CAPEC Partial: CAPEC-114, CAPEC-180 · CWE Direct: CWE-1220, CWE-863	**A trust policy names exactly which repository, workflow and environment may assume each role.** 1 Match the subject claim exactly on repository and ref or environment, and never use wildcards on organisation or repository. 2 Bind deploy roles to a protected environment claim whose environment requires approval. 3 Test each trust policy by trying to assume the role from a fork, another repository and an unprotected branch, and expect every attempt to fail.
ID·Q	**SECRETS IN UNTRUSTED RUNS** Secrets are available to jobs that run untrusted code: forks, pull requests, third-party steps. *Seen in: tj-actions/changed-files (2025).* STRIDE I, E (Clean) · CAPEC Partial: CAPEC-122 · CWE Direct: CWE-653	**Secrets reach only jobs that run trusted code, and within them only the steps that need each one.** 1 Withhold secrets from fork pull requests and from runs triggered by untrusted actors. 2 Scope secrets to protected environments with required reviewers instead of the whole repository. 3 Expose each secret only to the step that needs it, as an environment variable on that step, never at job or workflow level.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
ID·J	**OVER-SCOPED JOB TOKEN** The token issued to each run has write rights it does not need: pushing code, creating releases, approving pull requests. *Seen in: OWASP CICD-SEC-5.* STRIDE E (Clean) · CAPEC Partial: CAPEC-122 · CWE Direct: CWE-266, CWE-250	**Each run's token carries only the permissions its workflow declares, starting from none.** 1 Set the organisation's default job token permission to read-only. 2 Declare permissions per workflow and per job at least privilege, and enforce declarations with a workflow linter. 3 Disable the setting that lets job tokens create or approve pull requests.
ID·10	**SIGNING KEY EXPOSURE** Code and artifact signing keys sit in pipeline secrets or files, not in a key service with a usage policy. *Seen in: MSI and Intel Boot Guard keys leak (2023).* STRIDE S, I (Clean) · CAPEC Direct: CAPEC-206, CAPEC-474 · CWE Partial: CWE-522, CWE-312	**Signing keys cannot be exported, sign only for authorised pipelines under policy, and every signature is recorded.** 1 Hold signing keys as non-exportable keys in an HSM or cloud KMS, with a usage policy bound to the release pipeline identity. 2 Use Sigstore keyless signing with OIDC identity and transparency logging where it fits, removing stored keys altogether (see PV). 3 Log every signing operation, and alert when a signature comes from an unexpected identity.
ID·9	**PUBLISHING CREDENTIAL REACH** The credential that publishes to a public registry is reachable by many pipelines and people, so one leak lets an attacker publish as the project. *Seen in: Shai-Hulud (2025).* STRIDE S, T (Clean) · CAPEC Partial: CAPEC-560, CAPEC-538 · CWE Partial: CWE-522, CWE-668	**Publishing is possible from one release workflow only, under approval, with no stored token anywhere.** 1 Publish through OIDC trusted publishing bound to a specific repository, workflow and environment. 2 Remove stored registry tokens from pipelines and maintainer accounts once trusted publishing works. 3 Require reviewer approval on the publishing environment before the publish job runs.
ID·8	**CREDENTIALS LEFT IN THE WORKSPACE** Checkout and login steps write credentials to disk, where later steps, caches and artifacts pick them up. *Seen in: ArtiPACKED (Unit 42, 2024).* STRIDE I (Clean) · CAPEC Partial: CAPEC-639, CAPEC-155 · CWE Direct: CWE-459, CWE-538	**Credentials written by setup steps exist only as long as the step that needs them.** 1 Disable credential persistence in checkout steps, for example with persist-credentials set to false. 2 Log out and delete credential files before any untrusted step or upload runs. 3 Exclude credential paths from caches and artifacts, and fail the job if a scan finds them.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
ID·7	**INCOMPLETE ROTATION** After a compromise, rotation is partial or slow, and the attacker keeps a working credential. *Seen in: Trivy (2026).* STRIDE S (Loose) · CAPEC Partial: CAPEC-560 · CWE Partial: CWE-324	**Rotation after a compromise is complete, fast and verified, and old credentials stop working.** 1 Keep a rehearsed, scripted rotation runbook for each class of pipeline credential, and time each rehearsal. 2 Check rotation completeness against the secret inventory, so every copy and consumer is accounted for. 3 Revoke old credentials at the issuer rather than only replacing them in the store, and confirm they are refused.
ID·6	**SECRET STORE COMPROMISE** The vault or CI secret store holds every credential, with broad admin access and weak audit. STRIDE I, E (Clean) · CAPEC Partial: CAPEC-122 · CWE Partial: CWE-732, CWE-778	**Access to the secret store is itself short-lived, narrowly held and audited.** 1 Limit store administrators to a small group with phishing-resistant MFA and just-in-time elevation. 2 Authenticate pipelines to the store with federated short-lived identity and per-path policies. 3 Stream read audit logs off the store, and alert on reads outside a pipeline's normal pattern.
ID·5	**CROSS-ENVIRONMENT REUSE** The same identity serves test and production, so a test pipeline can reach production. STRIDE E (Clean) · CAPEC Partial: CAPEC-180 · CWE Direct: CWE-653	**Each environment has its own identity, and nothing outside production can obtain a production credential.** 1 Give each environment separate identities, roles and preferably separate cloud accounts. 2 Bind production trust policies to the production environment claim, and protect that environment with required reviewers. 3 Test from a non-production job that every attempt to obtain a production credential fails.
ID·4	**SECRET SPRAWL** Copies of one secret live in many repositories, pipelines and vaults, so rotation misses some and nobody owns it. STRIDE I (Loose) · CAPEC None · CWE Partial: CWE-1250	**Each secret has one source of truth and one owner, and every consumer reads from it.** 1 Store each secret once in the central store, and have pipelines reference it there instead of holding copies. 2 Record every consumer of each secret, so all uses can be listed before rotation. 3 Assign each secret an owner and a rotation schedule, and report overdue rotations.
ID·3	**MACHINE IDENTITY SPRAWL** Service accounts, bots and tokens multiply without owners or expiry dates. STRIDE S, E (Loose) · CAPEC None · CWE None	**Every non-human identity has an owner, a purpose and an end date.** 1 Require an owner and an expiry date when any service account, bot or token is created. 2 Disable identities unused for a set period automatically, and delete them after a grace period. 3 Keep a single inventory of non-human identities across the hosting platform, clouds and the secret store.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
ID·2	**DEBUG ACCESS LEAK** Debug logging, re-run-with-debug and interactive sessions into a runner expose secrets to whoever can turn them on. STRIDE I, E (Clean) · CAPEC Partial: CAPEC-121 · CWE Partial: CWE-532, CWE-215	**Turning on debugging does not widen who can see secrets.** 1 Restrict debug logging and interactive runner sessions to named roles. 2 Keep secret values out of debug output by masking them and by never echoing credentials in scripts. 3 Audit every debug session, limit its duration, and alert on sessions opened on release or production jobs.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

STATION 06 · LINE 1 · SOURCE TO ARTIFACT

BS BUILD SYSTEMS (BS)

A build should turn exactly the reviewed source into exactly one output, and every threat here breaks that link. These cards cover build tools, build scripts, plugins, compilers and toolchains, and remote caches and executors.

ID	THREAT	MITIGATION
BS·A	**BUILD PLATFORM IMPLANT** Compromised build infrastructure injects behaviour into every artifact without any change to the source. *Seen in: SolarWinds SUNSPOT (2020).* STRIDE T (Clean) · CAPEC Direct: CAPEC-673, CAPEC-670 · CWE Partial: CWE-1293, CWE-1357	**The artifact that ships can be shown to follow from the source alone, independently of the platform that built it.** 1 Rebuild each release on an independent rebuilder with separately administered infrastructure, and block publication when its output digest differs from the platform's. 2 Build release artifacts on ephemeral hosts recreated from a verified image for each job, so an implant cannot persist between builds. 3 Limit administrative access to build hosts to a small named group with just-in-time elevation, and log every session to a store the build hosts cannot write.
BS·K	**BUILD FROM UNOFFICIAL SOURCE** The build uses source that does not match the repository: local patches, a different ref, files outside version control. *Seen in: Webmin backdoor, CVE-2019-15107 (2019).* STRIDE T (Clean) · CAPEC Partial: CAPEC-445, CAPEC-444 · CWE Direct: CWE-348, CWE-349	**The build obtains its own source from the canonical repository and records exactly what it obtained.** 1 Have the build platform check out source itself from the canonical repository at a full commit hash, never from a workspace, upload or developer-supplied archive. 2 Fail the build when the working tree contains files not tracked at that commit, other than declared generated outputs. 3 Record the repository URI and commit digest in platform-generated provenance, and verify both against the expected repository at release (see PV).
BS·Q	**RELEASE ARCHIVE DIVERGENCE** Build scripts or release archives contain logic absent from the repository, so what ships differs from what was reviewed. *Seen in: xz Utils release tarball (2024).* STRIDE T (Clean) · CAPEC Partial: CAPEC-538, CAPEC-443 · CWE Direct: CWE-348, CWE-349, CWE-912	**Every shipped archive is produced by the pipeline from a tagged commit, and everything in it traces back to reviewed content.** 1 Generate release archives in the pipeline from the signed tag, and remove maintainers' ability to upload release assets by hand. 2 Compare each archive's contents with the tagged tree, and fail the release on any file that is neither in the tree nor a declared build output. 3 Review build scripts and macros as application code, and regenerate configure scripts and similar files from source in the pipeline rather than shipping pre-generated copies.
BS·J	**MALICIOUS BUILD PLUGIN** Build plugins run arbitrary code at build time with the build's privileges, and are fetched from repositories that get less scrutiny than application dependencies. STRIDE T, E (Clean) · CAPEC Direct: CAPEC-670, CAPEC-446 · CWE Direct: CWE-829, CWE-494	**Build plugins are treated as dependencies with build-time privileges and receive at least the same controls as libraries.** 1 Pin build plugins by version and checksum in a verified lockfile, such as Gradle dependency verification, and resolve them through the same internal proxy and firewall as libraries. 2 Require security review and a named owner before a new plugin or plugin repository is added to any build. 3 Run the build step with network egress blocked, so a plugin cannot fetch further code or send data out (see BS·8).

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
BS·10	**COMPROMISED TOOLCHAIN** A compiler, linker or SDK is trojaned, so output differs from source however clean the source is. *Seen in: XcodeGhost (2015).* STRIDE T (Clean) · CAPEC Direct: CAPEC-670 · CWE Partial: CWE-1357, CWE-494	**Toolchains are verified artifacts with a recorded identity, and their output can be cross-checked with an independently sourced toolchain.** 1 Install compilers, linkers and SDKs from pinned, signature-verified packages or digest-pinned build images, never from a download made during the build. 2 Record every toolchain component by digest in the build provenance. 3 Rebuild critical releases with an independently sourced or bootstrapped toolchain and compare the outputs, following the diverse double-compiling method.
BS·9	**REMOTE CACHE POISONING** A shared remote build cache accepts outputs written by untrusted builds and serves them to release builds. STRIDE T (Clean) · CAPEC Direct: CAPEC-141 · CWE Partial: CWE-349, CWE-669, CWE-732	**A release build consumes only cache entries written by builds trusted at the same level as itself.** 1 Grant write access to the release cache only to isolated release builders, and give developer machines and pull request builds a separate cache. 2 Key cache entries on a hash of every input that affects output, including toolchain, flags, environment variables and target platform. 3 Log each cache write with the writer's identity, and periodically re-execute a sample of cached actions from clean to compare digests with the stored entries.
BS·8	**NON-HERMETIC BUILD** The build fetches from the network while it runs, so its output depends on what the network returned at that moment. STRIDE T (Loose) · CAPEC Partial: CAPEC-185 · CWE Partial: CWE-494, CWE-829	**Every input to a build is declared and fetched by digest before the build runs, and the build itself can reach nothing else.** 1 Split fetching from building, so that a prior step resolves every declared input by digest into a local store. 2 Disable network access for the build step at the sandbox or runner level, and fail the build on any attempted connection. 3 Build the same commit twice on different hosts on a schedule, and alert when the outputs differ.
BS·7	**NON-REPRODUCIBLE OUTPUT** Timestamps, paths and ordering make each build different, so tampering cannot be found by rebuilding. STRIDE T, R (Loose) · CAPEC None · CWE None	**Release builds are deterministic, so an independent party can rebuild them and obtain identical bytes.** 1 Remove sources of non-determinism by setting SOURCE_DATE_EPOCH, normalising file order and permissions, and stripping build paths with path-prefix mapping. 2 Rebuild each release on an independent rebuilder and publish the comparison alongside the artifact. 3 Keep a list of artifacts that are not yet reproducible, each with an owner, and explain every mismatch with a diffing tool such as diffoscope.
BS·6	**GENERATED CODE TRUST** Code generators, schema compilers and macros emit code nobody reviews, from inputs that can be steered. STRIDE T (Loose) · CAPEC Partial: CAPEC-670, CAPEC-678 · CWE Partial: CWE-94, CWE-1357	**Generated code comes from pinned generators and reviewed inputs, and changes to it are visible to reviewers.** 1 Pin code generators, schema compilers and macro packages by version and digest, and resolve them through the same controls as other dependencies. 2 Commit generated output, or attach a diff of it to each pull request, so reviewers see what changes when an input or generator changes. 3 Flag for review any dependency update that changes a schema, template or IDL file feeding code generation.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
BS·5	**REMOTE EXECUTION TRUST** Remote build workers run actions for many clients and return outputs the client accepts unverified. STRIDE T, S (Clean) · CAPEC Partial: CAPEC-124 · CWE Partial: CWE-345, CWE-653	**Outputs from remote executors are bound to the client's inputs and can be checked, and no tenant can affect another's results.** 1 Run release actions on dedicated remote executors isolated from other tenants and developer workloads, with separate content-addressed storage. 2 Authenticate executors with mutual TLS, and record in the build log which worker produced each output. 3 Re-execute a random sample of actions locally or on a second pool, and fail the build when an output digest differs.
BS·4	**HARDENING FLAGS DISABLED** Build flags in a release turn off security features, or include debug and test-only code paths. STRIDE T (Loose) · CAPEC Direct: CAPEC-678, CAPEC-121 · CWE Partial: CWE-489, CWE-1269	**Release build configuration is defined once, has a named security owner, and is checked against the artifact actually produced.** 1 Define release compiler and linker flags in a central, code-owned configuration that projects import rather than copy. 2 Inspect each release binary for the expected hardening (stack protector, PIE, full RELRO, fortify, control-flow protection) and fail the release when any is missing. 3 Exclude debug and test-only code through build configuration, and assert in the pipeline that test endpoints and debug symbols are absent from release output.
BS·3	**SECRETS BAKED INTO OUTPUT** The build embeds environment variables, configuration or source maps holding secrets into shipped artifacts. STRIDE I (Clean) · CAPEC Direct: CAPEC-37, CAPEC-191 · CWE Direct: CWE-538, CWE-212	**Build output can contain only values that were declared safe to publish.** 1 Pass the build an explicit allow list of environment variables rather than the runner's whole environment, and inject runtime configuration at deploy time instead. 2 Scan every artifact, image layer and source map for secrets before publishing, and block publication on a finding. 3 Publish source maps and debug files only to an access-controlled symbol store, never alongside public artifacts.
BS·2	**UNRECORDED BUILD INPUTS** Tool versions, environment and parameters are not recorded, so a shipped artifact cannot be investigated later. STRIDE R (Clean) · CAPEC None · CWE Direct: CWE-223, CWE-778	**Each shipped artifact has a complete, durable build record that the build itself could not alter.** 1 Record toolchain versions, build image digest, parameters, environment and resolved dependencies in platform-generated provenance for every build. 2 Keep build records and logs in append-only storage outside the build's reach, retained at least as long as the artifact is supported. 3 Test the records periodically by rebuilding an old release from its record alone and comparing digests.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

STATION 07 · LINE 1 · SOURCE TO ARTIFACT

DR DEPENDENCY RESOLUTION (DR)

Resolution is the moment the factory decides whose code to run, usually automatically and often by name alone. These cards cover package managers, manifests, lockfiles, install-time execution and dependency update bots.

ID	THREAT	MITIGATION
DR·A	**INSTALL-TIME CODE EXECUTION** Package managers run lifecycle scripts on install, so resolving a malicious version executes it on developer machines and runners. *Seen in: ua-parser-js (2021); Shai-Hulud (2025).* STRIDE E, T (Clean) · CAPEC Direct: CAPEC-538, CAPEC-185 · CWE Direct: CWE-829, CWE-1188	**Resolving and installing a package does not, by itself, run that package's code.** 1 Disable install scripts by default, using settings such as npm ignore-scripts or the pnpm onlyBuiltDependencies allow list, and permit them only for named packages after review. 2 Run dependency installation in a job that holds no secrets, tokens or cloud credentials. 3 Fetch packages in one step and run the install step with network egress blocked, so a script cannot download a second stage or send data out.
DR·K	**DEPENDENCY CONFUSION** The resolver prefers a public package over a private one of the same name, or searches several indexes and takes the highest version. *Seen in: Birsan research (2021); PyTorch torchtriton (2022).* STRIDE S (Clean) · CAPEC Direct: CAPEC-159, CAPEC-616 · CWE Direct: CWE-427, CWE-348	**Each package name maps to exactly one authoritative source, and publishing a package elsewhere cannot change that mapping.** 1 Publish internal packages under a reserved scope or namespace, and register that scope or placeholder names on the public registries. 2 Configure each scope to resolve from exactly one index, and remove extra-index-url and similar fallback settings. 3 Route resolution through an internal proxy that refuses to serve a public package whose name matches an internal one.
DR·Q	**FLOATING VERSION RANGES** Manifests accept ranges, so a new malicious release is picked up with no change anyone reviewed. *Seen in: colors and faker (2022).* STRIDE T (Clean) · CAPEC Partial: CAPEC-186, CAPEC-538 · CWE Direct: CWE-829, CWE-494	**The exact set of dependencies a build uses changes only through a reviewed change to the repository.** 1 Commit a lockfile with integrity hashes and install in frozen mode in CI, for example npm ci, pnpm install --frozen-lockfile or pip install --require-hashes. 2 Fail any CI job that modifies the lockfile or installs a version not recorded in it. 3 Propose dependency updates as pull requests that change the lockfile, so each new version passes review and the usual checks (see DR·7).
DR·J	**TYPOSQUATTING AND LOOKALIKES** Packages named to be mistaken for popular ones (typos, separators, scopes, combined words) are installed by mistake. *Seen in: crossenv (2017).* STRIDE S (Clean) · CAPEC Direct: CAPEC-630, CAPEC-616 · CWE Partial: CWE-829, CWE-1007	**Adding a dependency is a deliberate, reviewed event, and names close to well-known packages are challenged.** 1 Detect each new direct dependency in a pull request, and require approval from a code owner of the dependency manifests. 2 Compare new package names with a list of popular packages using edit distance and separator and scope normalisation, and flag close matches for review. 3 Block known lookalike names and very recently published packages at the dependency firewall unless they are explicitly approved (see FW).

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
DR·10	**OWNERSHIP TRANSFER** A package passes to a new owner through handover, an expired maintainer domain or a re-registered deleted name, and the new owner ships malicious releases. *Seen in: event-stream (2018); PyPI Revival Hijack (JFrog, 2024).* STRIDE S, T (Clean) · CAPEC Direct: CAPEC-695, CAPEC-538 · CWE Partial: CWE-1357, CWE-1104	**A change in who controls a dependency is noticed and treated as a new trust decision.** 1 Monitor dependencies for changes of maintainer, publishing account, source repository or maintainer email domain, and raise each change for review. 2 Pin dependencies by version and hash, so a release from a new owner or a re-registered name is not installed until someone approves it. 3 Track dependencies that are unmaintained or rely on expired maintainer domains, and assign owners to replace or fork them.
DR·9	**LOCKFILE TAMPERING** Lockfiles are long and unreviewed, so a change can point a package at a different source or hash. *Seen in: Lockfile injection (Liran Tal, 2019).* STRIDE T (Clean) · CAPEC Partial: CAPEC-154, CAPEC-75 · CWE Partial: CWE-346, CWE-494	**A lockfile change is checked by machine against allowed sources and hashes, so its length cannot hide a redirection.** 1 Lint every lockfile in CI with a tool such as lockfile-lint, and reject entries that resolve outside the approved registry hosts or use insecure schemes. 2 Verify that each entry carries an integrity hash and that the hash matches the artifact served by the internal mirror. 3 Summarise lockfile diffs in the pull request as added, removed and re-sourced packages, so reviewers see the substance rather than the raw file.
DR·8	**MAINTAINER SABOTAGE** A legitimate maintainer deliberately ships destructive or political behaviour in a release. *Seen in: node-ipc, CVE-2022-23812 (2022).* STRIDE T (Loose) · CAPEC Direct: CAPEC-443, CAPEC-538 · CWE Partial: CWE-506, CWE-1357	**No release from a trusted maintainer reaches production before it has been observed, and any release can be withdrawn quickly.** 1 Apply a cooldown so that newly published versions are not proposed or installed until they have been public for a set number of days. 2 Compare each update's install-time behaviour, network access and file access with the previous version, and hold updates whose behaviour changes for review. 3 Keep known-good versions in the internal mirror, and be able to pin one across every repository with a single policy change.
DR·7	**AUTOMATED UPDATE MERGE** Dependency bots open and auto-merge updates, so a malicious release reaches production on the bot's authority. STRIDE T, E (Clean) · CAPEC Partial: CAPEC-186 · CWE Partial: CWE-266, CWE-829	**An update bot can propose a new version but cannot, by itself, decide that it ships.** 1 Restrict auto-merge to updates that meet an explicit policy, such as patch releases of allowed packages after a cooldown, and require human review for the rest. 2 Configure the bot to wait a set number of days after a release before proposing it. 3 Run bot pull requests with the same untrusted triggers and secret-free permissions as external contributions (see PD). 4 Apply branch protection to the bot's identity, so it cannot bypass required checks or review.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
DR·6	**MANIFEST CONFUSION** The metadata a registry shows differs from what the package manager installs. *Seen in: npm manifest confusion (Darcy Clarke, 2023).* STRIDE T, S (Loose) · CAPEC Direct: CAPEC-691, CAPEC-690 · CWE Direct: CWE-348	**Decisions about a package rest on the contents of the artifact that is installed, not on metadata served separately.** 1 Read scripts, dependencies and licence from the manifest inside the downloaded archive, not from the registry API. 2 Compare registry metadata with the archive's manifest at the proxy or firewall, and quarantine packages where they differ. 3 Generate the SBOM from the installed tree after installation (see BM).
DR·5	**TRANSITIVE BLIND SPOT** Most of the tree arrives transitively and is never chosen or reviewed by anyone. STRIDE T (Loose) · CAPEC Partial: CAPEC-446 · CWE Partial: CWE-1357	**The full resolved tree is known for every build, and changes to it are as visible as changes to direct dependencies.** 1 Generate the complete dependency tree from the lockfile for every build, and store it with the build record. 2 Show added and removed transitive packages in each pull request, and require review when a change introduces new ones. 3 Set a limit on the number of new transitive packages a single change may add, and require written justification above it.
DR·4	**RESOLVER DIFFERENTIAL** The package manager, the SCA scanner and the SBOM generator resolve the same manifest to different trees. STRIDE T (Loose) · CAPEC None · CWE Direct: CWE-436	**Every tool that reasons about dependencies works from the same resolved tree that the build installs.** 1 Make the lockfile the single input for the installer, the scanner and the SBOM generator, and fail when a tool resolves the manifest itself. 2 Produce scanner and SBOM input from the installed tree in the build environment, including platform-specific and optional dependencies. 3 Compare the package sets reported by the installer, the scanner and the SBOM for each build, and alert on any difference.
DR·3	**VENDORED COPY DRIFT** Third-party code copied or vendored into the repository is invisible to SCA and never updated. STRIDE T (Loose) · CAPEC None · CWE Partial: CWE-1104	**Vendored and copied third-party code is inventoried, attributed and updated like any other dependency.** 1 Record every vendored directory with its upstream source, version and licence in a machine-readable manifest that the SBOM generator reads. 2 Detect copied third-party code with snippet or file-hash matching during scanning, and require that it be declared. 3 Assign each vendored component an owner, and include it in the normal update and vulnerability processes.
DR·2	**UNPUBLISHED DEPENDENCY** A dependency or version disappears from the registry and builds fail or fall back to another source. *Seen in: left-pad (2016).* STRIDE D, S (Clean) · CAPEC Partial: CAPEC-603, CAPEC-616 · CWE Partial: CWE-636, CWE-348	**Builds depend on an internal copy of every dependency, and a missing version fails the build rather than changing its source.** 1 Mirror every resolved dependency into an internal repository, and resolve only from that repository. 2 Fail resolution when a pinned version or hash is unavailable, with no fallback to another index or to a different version. 3 Retain mirrored versions after upstream deletion, and alert owners when an upstream package they use is removed.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

SG

STATION 08 · LINE 1 · SOURCE TO ARTIFACT

SCANNERS AND GATES (SG)

A scanner is privileged code that reads everything and decides what ships, so it is both a control and a target. These cards cover SAST, SCA, secrets, IaC, container, licence and DAST scanners, AI reviewers, and the gates that act on their results.

ID	THREAT	MITIGATION
SG·A	**WEAPONISED SCANNER** The scanner, or the action that installs it, is compromised and runs with the pipeline's secrets in every repository that uses it. *Seen in: Trivy and KICS GitHub Actions (2026).* STRIDE E, I (Clean) · CAPEC Direct: CAPEC-670, CAPEC-538 · CWE Direct: CWE-829, CWE-250	**A scanner runs as untrusted code with only the access it needs to read the code and report a result.** 1 Pin scanner actions and images by full commit hash or digest, and verify their signature or provenance before the job runs. 2 Run scanners in a separate job with a read-only token and no publish, deploy or cloud secrets. 3 Block network egress from the scanner job, except to an internal advisory mirror where the scanner needs one.
SG·K	**FAIL-OPEN GATE** A scanner error, timeout, lapsed licence or unsupported language passes the gate instead of failing it. STRIDE D, E (Clean) · CAPEC Partial: CAPEC-554, CAPEC-655 · CWE Direct: CWE-636	**A gate passes only on a positive result, and "not scanned" is a distinct outcome with its own policy.** 1 Return a distinct result for scanner error, timeout, licence failure and unsupported file, and map each to a failed gate by default. 2 Record the fail-open or fail-closed choice in the gate's policy-as-code with a named owner, and log every fail-open decision. 3 Test the gate by making the scanner crash, time out and lose its licence, and assert that each case blocks the merge.
SG·Q	**GATE OVERRIDE** Developers or admins can make a required check optional, override a failure, or merge before checks finish. STRIDE E, R (Clean) · CAPEC Partial: CAPEC-554, CAPEC-122 · CWE Partial: CWE-841, CWE-424	**Required checks are enforced by the platform, and an override is a recorded, authorised exception rather than a routine path.** 1 Enforce gates as required status checks through branch rules that also bind administrators, so nothing merges while checks are pending or failed. 2 Restrict overrides to a break-glass role that the change's author cannot use, and require a recorded reason for each one. 3 Alert security owners on every override and on any change to branch rules or required checks.
SG·J	**SUPPRESSION ABUSE** The author of a change can add the inline suppression, ignore entry or baseline update that silences its findings. STRIDE E, T (Clean) · CAPEC Partial: CAPEC-578, CAPEC-554 · CWE Direct: CWE-807	**The author of a change cannot, alone, silence the findings on that change.** 1 Place ignore files, baselines and suppression configuration under code ownership that requires approval from a security owner. 2 Detect new inline suppressions in a diff, and require review from someone other than the author. 3 Give every suppression a reason and an expiry date, and reopen the finding when it expires.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
SG·10	**POLICY TAMPERING** The policy-as-code that defines a gate lives where the teams it governs can change it. STRIDE T (Clean) · CAPEC Partial: CAPEC-75 · CWE Direct: CWE-15, CWE-807	**Gate policy is governed by its owners separately from the teams it constrains, and each result shows which policy produced it.** 1 Keep gate policies and thresholds in a separate repository owned by security, with required review from its owners. 2 Load policy into the gate from that repository by version or digest, so a governed repository cannot override it locally. 3 Record the policy version with each gate result and in the artifact's attestations.
SG·9	**SCANNED IS NOT SHIPPED** The scanner analyses a different tree, branch, image or build from the one that ships. STRIDE T (Clean) · CAPEC Partial: CAPEC-29 · CWE Partial: CWE-367	**A scan result applies to one artifact digest, and only that digest can be released on the strength of it.** 1 Scan the built artifact by digest after the final build step, rather than scanning a branch or tag. 2 Emit each result as a signed attestation whose subject is the artifact digest (see PV). 3 Push artifacts to immutable repositories, and verify at release and admission that the digest has a passing scan attestation.
SG·8	**SCANNER EVASION** Code is written to defeat analysis: obfuscation, dynamic loading, encoded payloads, constructs the scanner does not model. STRIDE T (Loose) · CAPEC Partial: CAPEC-636, CAPEC-655 · CWE Partial: CWE-506, CWE-912	**Detection of deliberately hidden code is a separate control from vulnerability scanning, and assumes an adversary who knows the scanner.** 1 Flag dynamic code loading, evaluation of computed strings, encoded or compressed blobs and binary files in source for human review. 2 Run behavioural malicious-code detection, such as sandboxed install and test steps with network and file monitoring, separately from SAST. 3 Document which constructs each scanner does not model, and route changes that use them to manual review.
SG·7	**REVIEWER PROMPT INJECTION** An LLM reviewer or triage agent reads code, comments or pull request text that tells it to approve or suppress. STRIDE E, T (Clean) · CAPEC Partial: CAPEC-248 · CWE Direct: CWE-1427	**An LLM reviewer's output is advisory input to a gate whose decision rests on controls the reviewed content cannot influence.** 1 Require a deterministic check or a human for every approval, merge or suppression, so an LLM verdict alone never decides one. 2 Pass code, comments and pull request text to the model as delimited data, and report any instructions found in them as a finding. 3 Give the reviewer agent read-only permissions limited to returning a verdict, with no tools that can approve, label, merge or change configuration.
SG·6	**HOSTILE INPUT TO THE SCANNER** The code under scan exploits the scanner: parser bugs, config files the scanner executes, crafted paths in reports. STRIDE E, T (Clean) · CAPEC Partial: CAPEC-35, CAPEC-153 · CWE Partial: CWE-829, CWE-22, CWE-116	**The scanner treats the code it analyses as hostile input and is contained so that compromising it gains nothing.** 1 Run scanners in a sandbox with a read-only checkout, no secrets and no network egress. 2 Disable scanner features that execute repository content, such as build files, plugins or configuration hooks, or allow them only inside that sandbox. 3 Validate scanner reports before other tools consume them, rejecting paths outside the repository and escaping content rendered in comments. 4 Track scanner versions across all pipelines, and roll out security fixes within a set time.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
SG·5	**STALE ADVISORY DATA** Advisory databases lag or cached results are reused, so new vulnerabilities are missed. STRIDE D (Loose) · CAPEC None · CWE Partial: CWE-184	**Vulnerability decisions use current advisory data, and released artifacts are re-evaluated whenever that data changes.** 1 Record the advisory database version and age with each scan result, and fail the gate when the data is older than a set limit. 2 Re-scan the SBOMs of released and deployed artifacts when new advisories arrive, and alert owners on new matches. 3 Alert on a failed advisory feed update rather than continuing silently with cached data.
SG·4	**COVERAGE GAP** Some repositories, languages, IaC, images or generated code are never enrolled in scanning. STRIDE E (Loose) · CAPEC Partial: CAPEC-554 · CWE None	**Scanning coverage is measured against an authoritative inventory, and new assets are enrolled by default.** 1 Build the inventory of repositories, images and IaC from the hosting platform and registries, and report each scanner's coverage against it. 2 Enrol new repositories automatically through organisation-level required workflows or rules. 3 Publish the list of unsupported languages and artifact types, with an owner and a compensating control for each.
SG·3	**FINDINGS LEAKAGE** Scan results, a map of every weakness, go to third-party services, public pull request comments or logs. STRIDE I (Clean) · CAPEC Partial: CAPEC-150 · CWE Direct: CWE-201, CWE-532	**Findings are handled as sensitive data, with access and retention no wider than for the source they describe.** 1 Store findings in a system that inherits the repository's access controls, and restrict security findings to the owning team and security. 2 Keep security findings out of comments and logs on public repositories and forks, and link instead to an access-controlled view. 3 Configure external scanning services with no source retention, and confirm the setting against their data processing terms before enrolment.
SG·2	**ALERT FATIGUE** Finding volume buries the real issues, and teams stop reading results. STRIDE D (Loose) · CAPEC None · CWE Partial: CWE-655, CWE-779	**The volume of findings is managed so that what reaches a person is likely to matter and gets acted on.** 1 Prioritise findings by reachability, exploitation data such as EPSS and KEV, and exposure, and gate only on the highest tier. 2 Measure triage and fix rates per rule, and retire or tune rules whose findings are mostly dismissed. 3 Deduplicate findings across scanners and branches before they reach anyone's queue.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

PV

STATION 09 · LINE 1 · SOURCE TO ARTIFACT

PROVENANCE AND ATTESTATION (PV)

Provenance is only worth what its verifier enforces and only as honest as the platform that wrote it. These cards cover SLSA provenance, in-toto attestations, artifact signing, Sigstore and transparency logs, and the verification that is meant to act on them.

ID	THREAT	MITIGATION
PV·A	**FORGEABLE PROVENANCE** Build steps can write or alter the provenance, so a compromised build attests to whatever it likes. STRIDE T, S (Clean) · CAPEC Partial: CAPEC-690, CAPEC-673 · CWE Direct: CWE-653, CWE-522	**Provenance is written and signed by the build platform in a place no tenant build step can reach.** 1 Generate provenance in a platform-controlled step isolated from user-defined build steps, as SLSA Build L3 requires. 2 Sign provenance with keys or a workload identity the build steps cannot access, such as Sigstore keyless signing bound to the platform's identity. 3 Test the isolation with a build step that tries to read the signing material or write the provenance, and assert that both attempts fail.
PV·K	**VERIFICATION NOT ENFORCED** Provenance and signatures are produced, but nothing rejects an artifact that lacks them or fails verification. STRIDE T, S (Clean) · CAPEC Partial: CAPEC-68 · CWE Direct: CWE-347, CWE-494	**An artifact without valid, verified attestations does not run, by default.** 1 Enforce signature and provenance verification in enforcing mode at cluster admission and at the deploy step, and at package install where the ecosystem supports it. 2 Treat a missing signature or attestation as a verification failure. 3 Attempt to deploy an unsigned image and one with a broken signature on a schedule, and assert that both are rejected.
PV·Q	**WRONG IDENTITY ACCEPTED** The verifier checks that an artifact is signed, not who signed it or which builder, repository and workflow produced it. STRIDE S (Clean) · CAPEC Partial: CAPEC-473 · CWE Direct: CWE-347, CWE-346	**Verification asserts who produced an artifact and from what, not merely that someone signed it.** 1 Pin the expected OIDC issuer and certificate identity in each policy, such as the workflow path, repository and ref, rather than accepting any signature from the issuer. 2 Check the provenance's builder ID, source repository, ref and workflow against expected values. 3 Define these expectations per artifact or artifact family in policy-as-code owned by the artifact's owners. 4 Test each policy with an artifact signed by another repository under the same issuer, and assert that it fails.
PV·J	**SIGNING KEY COMPROMISE** A long-lived signing key is stolen and used to sign malicious artifacts that verify as genuine. *Seen in: NVIDIA code-signing certificates (2022).* STRIDE S, I (Clean) · CAPEC Direct: CAPEC-474, CAPEC-206 · CWE Partial: CWE-654, CWE-522	**No single long-lived secret is enough to produce a signature that consumers trust.** 1 Replace long-lived keys with Sigstore keyless signing, using short-lived certificates bound to the workload identity. 2 Hold any remaining keys in a hardware-backed key service that logs every use, and alert on use outside the release pipeline. 3 Rehearse revocation, including rotating the key through TUF or the verifier's trust policy and rejecting signatures made after the compromise.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
PV·10	**BUILD LEVEL READ AS SOURCE TRUST** Provenance faithfully records a build of a commit that was never reviewed, and a high build level is read as a statement about the source. STRIDE T (Loose) · CAPEC Partial: CAPEC-673 · CWE Partial: CWE-654	**Policy evaluates source integrity and build integrity as separate claims and requires both.** 1 Require evidence that the commit was reviewed and merged to a protected branch, such as a SLSA Source track attestation, alongside build provenance. 2 Check at verification time that the provenance's source revision is reachable from a protected branch. 3 State in policy and consumer documentation what the build level claims and what it does not.
PV·9	**ATTESTATION SUBSTITUTION** A valid attestation for one artifact is accepted for another because the subject digest is not compared, or lookup is by tag. STRIDE T, S (Clean) · CAPEC Partial: CAPEC-477 · CWE Direct: CWE-354, CWE-347	**An attestation is accepted only for the exact artifact whose digest it names.** 1 Compare the attestation's subject digest with the digest of the artifact being verified, and fail on any mismatch. 2 Resolve tags to digests before verification, and look up attestations by digest only. 3 Restrict who can push attestations to a registry repository, and verify each attestation's signer as well as its subject.
PV·8	**UNWATCHED TRANSPARENCY LOG** Signatures go to a transparency log nobody monitors, so a signature made with your identity by someone else goes unnoticed. STRIDE S, R (Loose) · CAPEC None · CWE Partial: CWE-347	**Every signature made under the organisation's identities is seen by the organisation, and verification depends on the log.** 1 Monitor Rekor for entries made with the organisation's identities, and alert on any that does not match a known pipeline run. 2 Require proof of inclusion, such as a signed entry timestamp or inclusion proof, during verification. 3 Fail closed when the log is unavailable, relying on proofs bundled with the artifact rather than skipping the check.
PV·7	**INCOMPLETE PROVENANCE** Provenance omits inputs such as resolved dependencies, parameters or the build image, so it attests to less than consumers assume. STRIDE R (Clean) · CAPEC None · CWE Direct: CWE-223	**Provenance records every input that could change the output, and states plainly where it is incomplete.** 1 Include resolved dependencies with digests, external parameters and the build image digest in every provenance document. 2 Mark completeness explicitly in provenance, and fail verification for critical artifacts when required fields are missing. 3 Validate a sample of provenance documents by rebuilding from the recorded inputs and comparing digests.
PV·6	**DOWNGRADE TO UNSIGNED** Policy accepts older unsigned versions, or falls back to tag matching when attestations are missing. STRIDE T, S (Clean) · CAPEC Partial: CAPEC-620, CAPEC-554 · CWE Partial: CWE-636, CWE-347	**Once signing is required, policy never weakens because an attestation is absent.** 1 Set a cut-over date or version after which unsigned artifacts are rejected, and enforce it in admission policy. 2 Remove any fallback to tag matching, and fail verification when attestations are missing. 3 Give each exemption an exact digest, a named owner and an expiry date, and alert before it lapses. 4 Prevent rollback to older versions with a minimum-version policy or TUF metadata.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
PV·5	**TRUST ROOT DISTRIBUTION** Verifiers fetch their trust roots (keys, certificate bundles, TUF roots) insecurely, or never rotate them. STRIDE S, T (Clean) · CAPEC Partial: CAPEC-479, CAPEC-94 · CWE Partial: CWE-324, CWE-924	**Verifiers obtain and update their trust roots through an authenticated channel that survives key compromise and rotation.** 1 Distribute trust roots through TUF, as the Sigstore public instance does, or pin them in the verifier image. 2 Monitor trust root expiry, and rotate roots before it, testing each rotation on a non-production verifier first. 3 Fail closed and alert the owners when a root has expired or cannot be updated.
PV·4	**UNVERIFIED DELIVERY PATH** A manual deploy, break-glass procedure, side-loaded image or second cluster skips verification. STRIDE T, E (Clean) · CAPEC Partial: CAPEC-554 · CWE Direct: CWE-424, CWE-420	**Verification sits at a point every path to production must cross, so a new path cannot skip it.** 1 Enforce verification at cluster admission and in the runtime environment, not only in the CD pipeline. 2 Inventory production clusters and deploy paths from infrastructure configuration, and alert on any that lacks enforcing admission policy. 3 Route break-glass deploys through the same verification with a recorded, time-limited exception, and review every use.
PV·3	**ATTESTATION STORE TAMPERING** Attestations kept in a registry or database can be deleted or replaced. STRIDE T, R (Clean) · CAPEC Partial: CAPEC-690, CAPEC-268 · CWE Direct: CWE-732, CWE-471	**Attestations, once written, cannot be removed or altered without detection.** 1 Store attestations in a write-once store and record them in a transparency log. 2 Restrict delete and overwrite rights on registries that hold attestations to a break-glass role. 3 Alert on any deletion or replacement of an attestation.
PV·2	**CLAIM INFLATION** A supplier claims a SLSA level or "signed builds" it does not meet, and consumers rely on the label. STRIDE S (Loose) · CAPEC None · CWE Direct: CWE-807, CWE-1357	**A supplier's security claim is accepted only on evidence the consumer can verify.** 1 Require suppliers to deliver signed provenance with each artifact, and verify its build level against the claim. 2 Write procurement requirements as verifiable evidence, such as provenance, signatures and identity policy, rather than labels. 3 Assign an owner to check each new supplier release against those requirements, and record the result.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

STATION 10 · LINE 2 · ARTIFACT TO RUNTIME

REGISTRIES (RG)

A registry turns a name into bytes, and every consumer trusts that the name still means what it meant yesterday. These cards cover public package and image registries, internal artifact repositories, and the proxies and mirrors between them.

ID	THREAT	MITIGATION
RG·A	**PUBLISHER ACCOUNT TAKEOVER** A maintainer's registry account is taken over through phishing, reused passwords or an expired email domain, and malicious versions are published as the project. *Seen in: ua-parser-js (2021); npm chalk and debug phishing (2025).* STRIDE S, E (Clean) · CAPEC Direct: CAPEC-538, CAPEC-560 · CWE Direct: CWE-308, CWE-640	**Publishing a release requires proof that it came from the project's own build, not only from a valid login to the maintainer's account.** 1 Require phishing-resistant MFA, such as FIDO2 security keys or passkeys, for every account with publish rights on packages above a set download threshold. 2 Use trusted publishing so releases come from an OIDC-authenticated CI workflow, and disable long-lived publish tokens for projects that adopt it. 3 Hold a release from a new device, location or token for a delay and notify every maintainer, so a second owner can cancel it. 4 Let a project mark provenance as mandatory, and have the registry reject any version without a provenance attestation that names the project's repository and workflow.
RG·K	**MUTABLE VERSION OR TAG** A published version or image tag can be overwritten, so one name returns different content over time. STRIDE T (Clean) · CAPEC Partial: CAPEC-148, CAPEC-29 · CWE Direct: CWE-471, CWE-494	**A published name and version resolve to the same bytes for as long as they exist.** 1 Make versions and tags immutable once published, so a fix always requires a new version number rather than a replacement. 2 Pull dependencies and images by content digest, and record the digest in the lockfile or manifest that the build reads. 3 Alert the owning team and every recorded consumer when the registry sees a delete and re-publish, or an attempted overwrite, of an existing version.
RG·Q	**INTERNAL RELEASE OVERWRITE** Pipelines or users can overwrite or re-upload artifacts in internal repositories, including releases already deployed. STRIDE T, E (Clean) · CAPEC Partial: CAPEC-669, CAPEC-558 · CWE Direct: CWE-732, CWE-471	**Release repositories are append-only, and promotion moves the artifact that was tested rather than producing a new one.** 1 Restrict write access on release repositories to one promotion identity, and give pipelines and people write access only to snapshot or staging repositories. 2 Enable immutability on release repositories so an existing version or digest cannot be overwritten, re-uploaded or deleted outside the retention process. 3 Promote by copying the same digest between repositories, and fail promotion when the digest in the target differs from the one that passed testing.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
RG·J	**VIRTUAL REPOSITORY SHADOWING** A virtual repository merges internal and public members, so a public package shadows an internal one. STRIDE S, T (Clean) · CAPEC Direct: CAPEC-159, CAPEC-616 · CWE Direct: CWE-427, CWE-829	**Each internal name resolves to exactly one source, and no public member can ever answer for it.** 1 Configure virtual repositories to search internal members first, and document and test the resolution order for every one of them. 2 Exclude internal namespaces and scopes from every remote member with explicit include and exclude patterns, so remote members never serve those names. 3 Reserve internal names and scopes on the public registry where the ecosystem allows it, and alert when a remote member is asked for a name that exists internally.
RG·10	**REGISTRY INFRASTRUCTURE COMPROMISE** The registry's own servers, storage, CDN or index are compromised, changing what every consumer receives. *Seen in: Docker Hub breach (2019); PEAR server (2019).* STRIDE T (Clean) · CAPEC Direct: CAPEC-669, CAPEC-439 · CWE Direct: CWE-494, CWE-347	**Clients verify that content is what its publisher released, using signed metadata that the registry's own infrastructure cannot forge.** 1 Verify packages against signed repository metadata, using a framework such as TUF, and reject content whose hash does not match a signed target. 2 Check signed metadata for expiry and version rollback, so a compromised index or CDN cannot serve stale or reordered entries without detection. 3 Compare hashes from the primary registry against an independent mirror or transparency log on a schedule, and alert on any mismatch.
RG·9	**STALE PROXY AND MIRROR** A proxy keeps serving a version upstream has removed as malicious, or a mirror serves old metadata to hide a fix. *Seen in: Go module proxy cached typosquat (Socket, 2025).* STRIDE T (Loose) · CAPEC Partial: CAPEC-141 · CWE Partial: CWE-1250, CWE-672	**A proxy or mirror reflects upstream removals and metadata changes within a bounded time, and a client can tell when it does not.** 1 Subscribe the internal proxy to upstream yank, removal and advisory events, and purge or block the affected versions on receipt. 2 Set a maximum metadata age on every proxy and mirror, and fail resolution rather than serve metadata older than that limit. 3 Use signed metadata with timestamps and version numbers, so a client rejects a mirror that presents an older snapshot than one it has already seen.
RG·8	**NAME AND NAMESPACE SQUATTING** Anyone can claim names close to popular ones, look-alike organisation scopes, or names freed by deletion. STRIDE S (Clean) · CAPEC Direct: CAPEC-630, CAPEC-616 · CWE Partial: CWE-1007	**Package names and scopes that consumers trust are bound to a verified owner, and cannot be claimed by someone else once freed.** 1 Reserve names freed by deletion or transfer for a cooling-off period, and never reassign a name that had significant downloads. 2 Block registration of names and scopes within a small edit distance of popular ones, and flag look-alike names for review. 3 Offer namespace verification through a DNS record or verified organisation, and show the verified status to consumers and in resolution output.
RG·7	**CROSS-ECOSYSTEM MIRRORING** Automated mirroring or repackaging carries a malicious package from one ecosystem into another. *Seen in: Shai-Hulud 2.0, npm to Maven Central (2025).* STRIDE T (Loose) · CAPEC Partial: CAPEC-538 · CWE Partial: CWE-669	**A package entering an ecosystem by automated mirroring or repackaging is judged in that ecosystem as if it were new.** 1 Keep an inventory of every automated mirror and repackaging job, with the source ecosystem and the target namespace each one writes to. 2 Run the target ecosystem's malware and policy checks on every mirrored or repackaged artifact before it becomes resolvable. 3 Propagate upstream takedowns to every mirrored copy by recording the source identifier and content hash with each repackaged artifact.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
RG·6	**TRUSTED DOMAIN ABUSE** Malware, miners and phishing payloads are hosted under the registry's trusted domain and pass network controls. STRIDE S, E (Loose) · CAPEC Partial: CAPEC-185 · CWE Direct: CWE-183, CWE-1220	**Network trust is granted to specific repositories and artifacts, not to every byte a trusted domain will serve.** 1 Route registry traffic through an internal proxy and deny direct egress to registry domains, so allowing a domain does not allow all of its content. 2 Scan downloaded content after fetch for malware and unexpected file types, regardless of the domain it came from. 3 Restrict pipelines to an allow list of repositories, organisations or image paths on each registry, rather than allowing the whole host.
RG·5	**POPULARITY SIGNAL MANIPULATION** Stars, download counts and linked source repositories are faked to make a package look established. *Seen in: StarJacking (Checkmarx, 2022).* STRIDE S (Clean) · CAPEC Direct: CAPEC-693, CAPEC-691 · CWE Direct: CWE-807	**Trust decisions about a package rest on verifiable facts about its origin, never on counts the publisher can inflate.** 1 Remove stars, download counts and similar popularity metrics from every automated admission and risk scoring rule. 2 Verify that a package's linked source repository is its real origin by checking build provenance (see PV) rather than the repository URL in its metadata. 3 Base trust on signals such as verified publisher, provenance, maintainer history and package age, and record which signal admitted each package.
RG·4	**PRIVATE ARTIFACT EXPOSURE** Private packages or images are published publicly by mistake, or repositories allow anonymous pull. STRIDE I (Clean) · CAPEC Partial: CAPEC-180, CAPEC-37 · CWE Direct: CWE-552, CWE-306	**Private artifacts can reach only private destinations, and every read of them requires an authenticated identity.** 1 Disable anonymous read on every internal repository by default, and report any repository where it is enabled with an owner and a reason. 2 Configure publish steps with an explicit internal registry URL and scope, and block CI egress to public publish endpoints except from approved release workflows. 3 Scan images and packages for secrets and internal hostnames before push, and fail the push when a finding is not marked as a false positive.
RG·3	**RETENTION GAP** Retention policies delete artifacts still deployed, or that an investigation needs. STRIDE R, D (Loose) · CAPEC None · CWE Partial: CWE-826	**An artifact that is running, or that an investigation depends on, cannot be removed by a retention rule.** 1 Exempt any digest currently deployed, as reported by the deployment system, from automated retention and cleanup. 2 Restrict deletion in release repositories to a separate role, log every deletion, and allow a legal or incident hold that blocks deletion entirely. 3 Retain the exact digest, SBOM and provenance of everything deployed for a defined period after it stops running, so investigators can retrieve what ran.
RG·2	**UPSTREAM AVAILABILITY** Public registry outages or rate limits stop builds, and the fallback widens trust. STRIDE D, T (Clean) · CAPEC Partial: CAPEC-603 · CWE Partial: CWE-636, CWE-348	**When a public registry is unavailable, builds slow down or stop rather than fetch from a source they would not otherwise trust.** 1 Serve all builds from an internal mirror or caching proxy that holds every locked dependency, so a public outage does not block them. 2 Remove secondary or fallback registry URLs from client configuration, so an outage cannot cause resolution from a less trusted source. 3 Authenticate to public registries with an organisation account to raise rate limits, and alert when throttling begins rather than switching source.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

FW

STATION 11 · LINE 2 · ARTIFACT TO RUNTIME

FIREWALL AND QUARANTINE (FW)

A dependency firewall works only if it is the single way in and judges what it serves, not what it once saw. These cards cover dependency firewalls, curated proxies, cooldowns, quarantine and the exceptions that keep builds moving.

ID	THREAT	MITIGATION
FW·A	**BYPASS ROUTE** Some clients reach public registries directly: hard-coded URLs, build plugins, CI images with default configuration, developer machines, install scripts. STRIDE E, T (Loose) · CAPEC Partial: CAPEC-554 · CWE Direct: CWE-424, CWE-420	**The firewall is the only network path from any factory client to a public registry, and that is enforced by the network rather than by configuration.** 1 Block direct egress to public registry hosts at the network layer for CI, build and developer networks, and allow only the firewall to reach them. 2 Inventory tools, build plugins and scripts that fetch outside the package manager, using egress logs, and route each one through the firewall or block it. 3 Apply the same proxy and egress rules to developer machines and remote workers through managed configuration and a device network policy.
FW·K	**PRE-POLICY CACHE** Packages cached before a policy existed, or before a version was flagged, keep being served unchecked. STRIDE T (Loose) · CAPEC Partial: CAPEC-141 · CWE Partial: CWE-672	**Every artifact is judged against current policy and intelligence each time it is served, however long it has been cached.** 1 Re-evaluate cached artifacts whenever a policy rule or intelligence feed changes, and block any that now fail before the next request. 2 Check pinned versions in lockfiles and CI caches against current intelligence on a schedule, and fail builds that still reference a flagged version. 3 Evaluate policy at serve time as well as at first fetch, so a flagged version cannot be returned from the firewall's cache.
FW·Q	**BROAD EXCEPTION** An exception for one package covers every version or skips every check, and nobody removes it. STRIDE E (Loose) · CAPEC Partial: CAPEC-554 · CWE Partial: CWE-183, CWE-1220	**An exception allows the smallest thing that unblocks the work, for a limited time, and is decided by someone other than the requester.** 1 Scope each exception to a package, version and content hash, and to the single check it waives rather than to all checks. 2 Give every exception an expiry date and an owner, and remove or renew it automatically when it lapses. 3 Require approval from a reviewer other than the requester, and record who granted each exception and why.
FW·J	**DETECTION WINDOW** The firewall blocks only packages already known to be malicious, so a new malicious release passes until intelligence catches up. *Seen in: Shai-Hulud (2025).* STRIDE T (Loose) · CAPEC Partial: CAPEC-538, CAPEC-186 · CWE Direct: CWE-184	**New releases are kept out of builds until there has been time for them to be analysed and reported, with a reviewed way through for urgent fixes.** 1 Enforce a minimum package age, or cooldown, of several days before a new version can be resolved, and apply it to transitive dependencies too. 2 Run behavioural and diff analysis on new releases during the cooldown, and quarantine any that show install-time execution or new network access. 3 Allow an urgent version to skip the cooldown only through a named, time-limited exception that pins its content hash and triggers a review.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
FW·10	**FAIL-OPEN FIREWALL** When the intelligence feed, policy service or proxy fails, requests pass through or clients fall back to the public registry. STRIDE D, T (Clean) · CAPEC Partial: CAPEC-607, CAPEC-554 · CWE Direct: CWE-636, CWE-348	**If the firewall cannot decide, it does not serve, and any choice to fail open is explicit, owned and logged.** 1 Fail closed by default when the intelligence feed, policy service or proxy is unavailable, and return a distinct error that names the cause. 2 Remove any client fallback to the public registry, and block public registry egress so a fallback cannot occur even if it is configured. 3 Make fail-open a recorded per-ecosystem setting with an owner, and log and alert every request served while it is in effect.
FW·9	**ECOSYSTEM COVERAGE GAP** The firewall covers some ecosystems but not others: Go modules, Cargo, container images, CI actions, IDE extensions, AI models. STRIDE T (Loose) · CAPEC Partial: CAPEC-554 · CWE Partial: CWE-424	**Every kind of artifact the factory consumes passes through the same admission policy, whatever ecosystem it belongs to.** 1 Inventory every artifact type the factory consumes, from egress logs and build configuration, and list which ones the firewall does not yet proxy. 2 Proxy CI actions, base images, Go modules and Cargo crates through governed repositories, and block their public sources at egress. 3 Bring IDE extensions and AI model downloads under the same policy through an internal marketplace or model registry with allow lists.
FW·8	**COORDINATES WITHOUT CONTENT** The firewall judges a name and version, not the content hash, so a different file with the same coordinates passes. STRIDE T (Clean) · CAPEC Partial: CAPEC-148, CAPEC-29 · CWE Direct: CWE-1023, CWE-494	**The firewall admits content, so a decision about a version holds only for the exact bytes it evaluated.** 1 Record the content hash of every artifact at first evaluation, and bind each policy decision to that hash rather than to name and version. 2 Block and alert when upstream serves different bytes for a version the firewall has already seen. 3 Verify signatures and provenance where the ecosystem publishes them, and reject an artifact whose attestation does not match its hash (see PV).
FW·7	**QUARANTINE RELEASE ABUSE** A quarantined artifact is released without re-evaluation, or by the person who requested it. STRIDE E, R (Clean) · CAPEC Partial: CAPEC-122 · CWE Partial: CWE-863, CWE-841	**Release from quarantine is a new decision taken by someone independent, using what is known at the time of release.** 1 Re-scan an artifact with current intelligence at the moment of release, and refuse release if the result is worse than when it was quarantined. 2 Prevent self-release by requiring that the approver is not the person who requested the artifact or the exception. 3 Limit release rights to a small security role, and log each release with the approver, the reason and the content hash.
FW·6	**FETCH VISIBILITY GAP** Fetches are not logged with requester and digest, so after an incident nobody can say which builds pulled the bad version. STRIDE R (Clean) · CAPEC None · CWE Direct: CWE-778	**Every fetch leaves a record that answers which builds and machines received which bytes, and when.** 1 Log every fetch with requester identity, build or job identifier, package coordinates, content hash and time. 2 Index fetch logs so that a query by version or hash returns every build and machine that pulled it within minutes. 3 Retain fetch logs for at least as long as the longest-lived deployed artifact, and protect them from deletion by firewall administrators.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
FW·5	**HEURISTIC EVASION** Malicious packages dodge firewall analysis: a clean first release, payloads fetched at run time, obfuscation, delayed activation. STRIDE T (Loose) · CAPEC Partial: CAPEC-655, CAPEC-636 · CWE Partial: CWE-506, CWE-511	**Analysis judges what a package does and how it changed, not only what its first release looked like.** 1 Run packages in an instrumented sandbox during install and import, and flag network fetches, process launches and file writes outside the package directory. 2 Analyse the diff between each new version and the previous one, and raise the risk score for new install scripts, binaries or obfuscated code. 3 Track publisher history and flag releases that follow a pattern of clean versions then sudden behaviour change, new maintainers or new dependencies.
FW·4	**FIREWALL AS TARGET** Whoever controls the firewall's console or policy store can allow anything into every build. STRIDE E, T (Clean) · CAPEC Partial: CAPEC-578, CAPEC-176 · CWE Partial: CWE-654, CWE-1390	**Control of the firewall is treated as control of every build, and protected to that level.** 1 Federate firewall administration through the organisation's identity provider with phishing-resistant MFA, and remove local admin accounts. 2 Manage firewall policy as code in a reviewed repository, so every rule change needs a second approver before it applies. 3 Send policy change and admin audit logs to a store the firewall administrators cannot modify, and alert on changes made outside the policy pipeline.
FW·3	**FRICTION BYPASS** Blocks with no fast exception path push teams to vendor code, use personal tokens or route around the proxy. STRIDE E (Loose) · CAPEC None · CWE Direct: CWE-655	**Following the policy is the fastest way to get an unblocked build.** 1 Return a block message that names the rule, the reason and a link to request an exception, rather than a bare error. 2 Set and measure a service level for exception decisions, and offer a self-service path for low-risk cases such as a newer version of an approved package. 3 Monitor for circumvention, such as vendored registry code, personal tokens on public registries and direct egress attempts, and follow up each finding with the team.
FW·2	**POLICY DRIFT** Rules set once are never reviewed, so deny lists go stale and disabled rules stay disabled. STRIDE T (Loose) · CAPEC None · CWE Partial: CWE-184	**Every rule has an owner and a review date, so policy reflects current threats rather than the day it was written.** 1 Assign a named owner and a review date to every rule, and report rules that are past their review date. 2 Require a reason and an expiry when a rule is disabled, and re-enable it or escalate to its owner when the expiry passes. 3 Test policy against a set of known malicious and known good packages on a schedule, and alert when a rule stops catching what it should.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

STATION 12 · LINE 2 · ARTIFACT TO RUNTIME

BM SBOM AND VEX (BM)

An SBOM is a claim about an artifact, and every decision built on it inherits its gaps and its author's honesty. These cards cover SBOM generation, storage and exchange, VEX statements, and the vulnerability and licence decisions driven by inventory.

ID	THREAT	MITIGATION
BM·A	**UNVERIFIABLE SBOM** An SBOM is accepted without a signature or a binding to the artifact's digest, so anyone can supply one claiming any contents. STRIDE S, T (Clean) · CAPEC Partial: CAPEC-690 · CWE Direct: CWE-347, CWE-354	**An SBOM is trusted only when its signer is known and it is bound to the exact artifact it describes.** 1 Publish each SBOM as a signed in-toto attestation whose subject is the artifact's digest, and store it alongside the artifact. 2 Verify the signer's identity against an expected issuer and build identity before ingesting an SBOM, and reject unsigned ones. 3 Reject a supplier SBOM whose subject digest does not match the artifact actually received.
BM·K	**INCOMPLETE INVENTORY** The SBOM omits components: static libraries, vendored code, OS packages, copied-in binaries, build tools, code downloaded at run time. STRIDE R (Loose) · CAPEC None · CWE Direct: CWE-223	**The inventory is derived from what actually ships, and states which kinds of component it could not see.** 1 Generate the SBOM from the final artifact or image, not only from source manifests, and merge in build-time data from the build system. 2 Detect copied-in and statically linked binaries by file hash and signature matching, and list unidentified binaries as unknown components rather than omitting them. 3 Record in the SBOM which component types the generator covers, and mark the SBOM as incomplete when known gaps such as run-time downloads apply.
BM·Q	**MALICIOUS VEX** A VEX statement marks an exploitable vulnerability "not affected", suppressing it for every consumer, and its author is not verified. STRIDE S, T (Clean) · CAPEC Partial: CAPEC-194, CAPEC-690 · CWE Direct: CWE-807, CWE-347	**A VEX statement suppresses a finding only when it comes from an authorised, verified author with a justification someone has checked.** 1 Accept only signed VEX documents from an allow list of authors, and reject unsigned or unknown-author statements. 2 Require a justification code and supporting evidence for every not_affected statement, and route third-party statements to human review before they take effect. 3 Keep your own analysis authoritative, so an external VEX can propose a status but cannot override an internal affected decision automatically.
BM·J	**SBOM DRIFT** The SBOM describes a different build from the one deployed: generated at the wrong stage, not refreshed after a patch, served from cache. STRIDE T, R (Loose) · CAPEC None · CWE Partial: CWE-1250	**The SBOM and the artifact are produced together and bound by digest, so one cannot change without the other.** 1 Generate the SBOM in the same build step that produces the artifact, and attach it as an attestation to that artifact's digest. 2 Regenerate and re-attach the SBOM on every rebuild or patch, and never copy or cache an SBOM from a previous build. 3 Check at deployment and in inventory queries that the deployed digest matches the SBOM's subject, and flag any deployment without a match.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
BM·10	**IDENTIFIER AMBIGUITY** Components are named inconsistently (PURL and CPE, forks, renamed or repackaged packages), so vulnerability matching misses or misattributes. STRIDE S (Loose) · CAPEC None · CWE Partial: CWE-694	**Each component carries an identifier precise enough to match vulnerabilities to the code actually shipped.** 1 Use package URLs (PURL) as the primary identifier, with CPE only as a secondary mapping, and record the upstream source for forks and repackaged components. 2 Include file hashes for components, so a fork or rebuilt package can be told apart from the original it was derived from. 3 Test vulnerability matching against a fixed set of known renamed, forked and repackaged cases, and fail the pipeline when a known match is missed.
BM·9	**HOSTILE SBOM DOCUMENT** SBOMs from suppliers are untrusted input to parsers, databases and dashboards. STRIDE D, E (Clean) · CAPEC Direct: CAPEC-230, CAPEC-201 · CWE Partial: CWE-611, CWE-79, CWE-770	**Supplier SBOMs are handled as untrusted input by every system that reads them.** 1 Parse supplier SBOMs in an isolated worker with no network access or database credentials, and validate them against the SPDX or CycloneDX schema first. 2 Enforce limits on document size, nesting depth and component count, and reject documents that exceed them. 3 Disable external entities and DTD processing in XML parsers, and encode SBOM fields before they are rendered in dashboards.
BM·8	**INVENTORY AS RECONNAISSANCE** A published or shared SBOM tells an attacker which vulnerable components run where. STRIDE I (Clean) · CAPEC Partial: CAPEC-310, CAPEC-169 · CWE Direct: CWE-497, CWE-213	**SBOMs reach only the parties who need them, under terms that limit further sharing, and every access is recorded.** 1 Share SBOMs through an authenticated portal under agreed terms rather than publishing them openly, especially for internet-facing systems. 2 Log every read of the SBOM store with requester identity, and review access by external parties periodically. 3 Pair each shared SBOM with its VEX, so recipients see which listed vulnerabilities are not exploitable rather than treating each one as an open target.
BM·7	**SBOM STORE TAMPERING** The central SBOM repository can be edited, so inventory queries during an incident return false answers. STRIDE T, R (Clean) · CAPEC Partial: CAPEC-268 · CWE Direct: CWE-732, CWE-471	**Inventory records cannot be changed after they are written, and any change would be visible.** 1 Store SBOMs as signed attestations in append-only storage, so a correction is a new record rather than an edit. 2 Restrict write access to the SBOM store to the build pipeline's identity, and deny update and delete to people and other services. 3 Verify signatures on retrieval and record SBOM hashes in a transparency log, so a changed or replaced record is detected.
BM·6	**STALE VEX** A "not affected" statement stays in force after a code change makes the component reachable. STRIDE T (Loose) · CAPEC None · CWE Partial: CWE-672	**A not_affected status holds only for the code it was assessed against, and lapses when that code or time moves on.** 1 Bind each VEX statement to specific product versions or digests, and give every statement an expiry date. 2 Re-evaluate VEX statements automatically when the affected component, its callers or the product version change, and revert the status to under_investigation until reviewed. 3 Assign each VEX statement a named owner who is notified on expiry or re-evaluation.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
BM·5	**CONVERSION LOSS** Converting between SPDX and CycloneDX, or between versions, drops hashes, licences or relationships. STRIDE T (Loose) · CAPEC None · CWE Direct: CWE-223	**Converting an SBOM preserves every field that later decisions depend on, or says what was lost.** 1 Document which fields each conversion tool drops, and choose tools that preserve hashes, licences and relationships. 2 Compare component counts, hashes and relationships before and after conversion, and fail the conversion when any are missing. 3 Re-validate converted SBOMs against the target schema and keep the original document as the authoritative copy.
BM·4	**FLAT INVENTORY** The SBOM lists components without their relationships, so reachability and blast radius cannot be worked out. STRIDE R (Loose) · CAPEC None · CWE Direct: CWE-223	**The inventory records how components relate, so blast radius and reachability can be worked out from it.** 1 Record dependency relationships in the SBOM using CycloneDX dependencies or SPDX relationships, marking direct and transitive dependencies. 2 Reject SBOMs with no dependency graph at ingestion, or flag them as flat so decisions built on them are marked lower confidence. 3 Load SBOMs into a queryable store that can answer which artifacts contain a component and which direct dependency brought it in.
BM·3	**COMPLIANCE THEATRE** SBOMs are produced to meet a requirement and consumed by nothing, so a vulnerable component in one blocks nothing. STRIDE T (Loose) · CAPEC None · CWE None	**Every SBOM feeds at least one decision that can block or change what ships.** 1 Feed every SBOM into a vulnerability and licence policy check that can fail the build or block deployment. 2 Re-check stored SBOMs against new advisories continuously, and raise findings for deployed artifacts without waiting for a rebuild. 3 Name the system and owner that consume each SBOM, and report SBOMs that no query or policy has read in a set period.
BM·2	**MISSING SUPPLIER SBOM** Third-party software arrives with no SBOM, or one made by a tool that sees different things from yours. STRIDE R (Loose) · CAPEC None · CWE Partial: CWE-1357	**Third-party software has an inventory before it is deployed, either from its supplier or produced locally.** 1 Make a signed, digest-bound SBOM a procurement and contract requirement for third-party software. 2 Generate your own SBOM for every third-party artifact on receipt, and compare it with the supplier's to find components either one misses. 3 Track third-party artifacts with missing or partial SBOMs in a register with an owner and a date for resolution.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

STATION 13 · LINE 2 · ARTIFACT TO RUNTIME

INFRASTRUCTURE AS CODE (IC)

Infrastructure as code makes the estate reviewable, and makes whoever runs the code the owner of the estate. These cards cover Terraform and OpenTofu, Pulumi, CloudFormation and configuration management, their state, modules and providers, and the plan and apply pipeline.

ID	THREAT	MITIGATION
IC·A	**STATE FILE EXPOSURE** State files hold secrets and full resource detail in plain text, in storage readable by too many identities. STRIDE I (Clean) · CAPEC Direct: CAPEC-639 · CWE Direct: CWE-312, CWE-732, CWE-922	**State is treated as a secret store, encrypted, readable only by the identities that apply it, and audited.** 1 Use an encrypted state backend with a key that only the apply role can use, and block public access to the backend. 2 Restrict state read access to the apply and plan roles for each workspace, and remove read access from developers and broad admin groups. 3 Keep secrets out of state by reading them at run time from a secret store or using ephemeral values, and mark unavoidable outputs as sensitive. 4 Log every read and write of state objects, and alert on access from an identity that is not a pipeline role.
IC·K	**PLAN-TIME CODE EXECUTION** Running plan on an untrusted change executes code (external data sources, provider and module downloads), so a pull request borrows the pipeline's cloud credentials. STRIDE E, I (Clean) · CAPEC Partial: CAPEC-175, CAPEC-233 · CWE Direct: CWE-829, CWE-250	**Planning an untrusted change cannot reach credentials or execute code with more privilege than its author already has.** 1 Run plan with a separate, read-only OIDC role that cannot write resources or read secrets, distinct from the apply role. 2 Require a maintainer to approve plan runs on changes from forks or first-time contributors before the job receives any cloud credentials. 3 Block external data sources, local-exec provisioners and unapproved providers during plan with policy-as-code on the configuration. 4 Install providers and modules only from an internal mirror during plan, so plan cannot download and run arbitrary code.
IC·Q	**MALICIOUS MODULE OR PROVIDER** Modules and providers are pulled by name and floating version, then run with the cloud credentials. STRIDE T, E (Clean) · CAPEC Direct: CAPEC-538, CAPEC-185 · CWE Direct: CWE-494, CWE-829, CWE-1357	**Modules and providers that run with cloud credentials are fixed to reviewed content and fetched from a controlled source.** 1 Pin modules and providers to exact versions, and commit the dependency lock file with checksums so a changed package fails installation. 2 Mirror approved modules and providers internally, and configure the tool to install only from that mirror. 3 Require security review and an allow-list entry before a new module source or provider namespace can be used.
IC·J	**OVER-PRIVILEGED APPLY IDENTITY** The identity that applies the code is admin across accounts, so compromising the IaC pipeline means owning the estate. STRIDE E (Clean) · CAPEC Partial: CAPEC-122 · CWE Direct: CWE-250, CWE-266	**Each apply can change only the resources its workspace owns, and never the permissions that bound it.** 1 Issue a separate OIDC-scoped apply role per workspace or account, trusted only for that repository, branch and environment. 2 Gate apply on approval of the reviewed plan by someone other than the author, through a protected environment. 3 Deny apply roles the ability to create identities or modify their own policies, using permission boundaries or organisation-level service control policies.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
IC·10	**PLAN AND APPLY DIVERGENCE** What is applied differs from what was reviewed: the plan is regenerated at apply time, or state changed in between. STRIDE T (Clean) · CAPEC Partial: CAPEC-29 · CWE Partial: CWE-367	**The changes applied are exactly the changes that were reviewed, and nothing else.** 1 Save the plan to a file in the plan job, and apply that saved plan file rather than generating a new plan at apply time. 2 Fail apply when state has changed since the plan was made, which saved plan files do by default, and require a fresh plan and review. 3 Store the plan file as an artifact bound to its commit hash and a content hash, and verify both before apply.
IC·9	**CONFIGURATION CONTROLLER** A configuration management controller pushes root-level commands to every host, so its playbooks, inventory and credentials are a single point of compromise. STRIDE E, T (Clean) · CAPEC Partial: CAPEC-445, CAPEC-234 · CWE Partial: CWE-250, CWE-653	**The configuration controller's reach is limited to what each run needs, and every change to what it runs is reviewed.** 1 Keep playbooks and inventory in a protected repository with required review, and run only from reviewed commits. 2 Replace standing root credentials with short-lived, per-run credentials or signed SSH certificates scoped to the target hosts. 3 Split the controller by environment or host group, so one compromised controller cannot reach every host. 4 Log every run with its commit, operator, target hosts and outcome to a store the controller cannot modify.
IC·8	**DRIFT AND OUT-OF-BAND CHANGE** Console and CLI changes create infrastructure the code does not describe, so review no longer reflects reality and backdoors survive. STRIDE T, R (Clean) · CAPEC Partial: CAPEC-176, CAPEC-554 · CWE None	**The code is the only routine way to change infrastructure, and any difference between code and reality is found and resolved.** 1 Run drift detection on a schedule for every workspace, and alert the owner on any difference from the last applied state. 2 Limit console and CLI write access to read-only for most people, with write access only through a logged break-glass role. 3 Open a reconciliation ticket automatically for every break-glass session, and close it only when the change is imported into code or reverted.
IC·7	**SECRETS IN VARIABLES AND OUTPUT** Secrets are hard-coded or passed as plain variables that appear in logs, plan output and state. STRIDE I (Clean) · CAPEC Direct: CAPEC-37 · CWE Direct: CWE-798, CWE-532	**Secrets reach infrastructure tools at run time from a secret store, and never appear in code, logs or review output.** 1 Read secrets from a secret store at apply time using the pipeline's workload identity, and block hard-coded secrets with a pre-merge secret scan. 2 Mark secret variables and outputs as sensitive so plan output and logs mask them, and fail review when a known secret pattern appears unmasked. 3 Post only a summary of plan output to pull request comments, and keep the full plan in access-controlled pipeline storage.
IC·6	**INSECURE DEFAULTS PROPAGATED** Templates and shared modules encode public storage, open network rules or disabled encryption, and copying spreads them. *Seen in: IaC security smells (Rahman et al., 2019).* STRIDE I, T (Loose) · CAPEC Partial: CAPEC-180 · CWE Direct: CWE-1188	**Shared templates are secure by default, and every plan is checked against policy before it can be applied.** 1 Evaluate policy-as-code on every plan, checking for public storage, open network rules and disabled encryption, and fail the pipeline on a violation. 2 Build shared modules with encryption, private access and logging on by default, and require an explicit, named input to turn any of them off. 3 Allow a policy failure to be overridden only through a recorded exception approved by the policy owner, not by the change author.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
IC·5	**DESTRUCTIVE CHANGE** A change, typo or rename destroys and recreates databases, keys or DNS records. STRIDE D, T (Clean) · CAPEC None · CWE None	**Destroying a critical resource requires a deliberate, separately approved action, never a side effect of an ordinary change.** 1 Show destroy and replace actions prominently in the plan summary, and require additional approval for any plan that contains them. 2 Enable deletion protection on databases, keys and DNS zones in both the code and the cloud provider, such as prevent_destroy lifecycle rules. 3 Block plans that destroy resources tagged as protected using policy-as-code, and use moved blocks so renames do not cause replacement.
IC·4	**ORPHANED RESOURCES** Resources removed from code are not destroyed, and run on unmanaged: forgotten endpoints, dangling DNS records others can claim. STRIDE S, I (Loose) · CAPEC Partial: CAPEC-154 · CWE Partial: CWE-672	**Every resource in the estate is either managed by code or known and owned, and nothing is left running unmanaged.** 1 Require resource removal to go through a destroy in the plan, and flag state rm or removed blocks without destroy for explicit review. 2 Compare cloud inventory with all state files on a schedule, and report resources that no workspace manages to an owner. 3 Detect DNS records that point at released IP addresses, buckets or hostnames, and alert before someone else can claim the target.
IC·3	**CROSS-WORKSPACE REACH** Workspaces read each other's state or outputs, so a low-trust workspace learns secrets from a high-trust one. STRIDE I, E (Clean) · CAPEC Partial: CAPEC-180, CAPEC-639 · CWE Direct: CWE-653, CWE-732	**A workspace can read only the specific outputs it has been granted, and never another workspace's full state.** 1 Share values between workspaces through explicit outputs or a parameter store, rather than granting read access to another workspace's state. 2 Expose only non-secret outputs that consumers need, and keep secrets in a secret store with its own access control. 3 Grant remote state access per workspace with least privilege, and deny low-trust workspaces read access to high-trust ones.
IC·2	**STATE LOCKING AND CORRUPTION** Concurrent applies, force-unlocks or edited state make the code adopt or orphan resources. STRIDE T, D (Clean) · CAPEC Partial: CAPEC-26 · CWE Partial: CWE-667, CWE-362	**State changes happen one at a time, every version can be recovered, and manual edits are rare, controlled and recorded.** 1 Enable state locking in the backend, and fail any apply that cannot acquire the lock. 2 Turn on versioning for the state backend so any previous state can be restored, and test restoring it. 3 Restrict force-unlock and state editing commands to a break-glass role, and log each use with the operator and reason.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

IM

STATION 14 · LINE 2 · ARTIFACT TO RUNTIME

CONTAINER IMAGES (IM)

An image freezes a whole operating system and its contents into one artifact, including whatever the build let in. These cards cover base images, Dockerfiles, layers, image builders, tags and digests.

ID	THREAT	MITIGATION
IM·A	**UNTRUSTED BASE IMAGE** Images build on public base images pulled by tag from unverified publishers, so a compromised base enters every image built on it. STRIDE T, S (Clean) · CAPEC Direct: CAPEC-446 · CWE Direct: CWE-1357, CWE-829, CWE-494	**Every image starts from a base the organisation has chosen, can identify by content, and will rebuild from when that base changes.** 1 Maintain a short list of approved base images mirrored into an internal registry, and reject builds whose FROM line points anywhere else. 2 Pin each FROM line by digest and verify the publisher's signature on that digest before the build starts. 3 Assign every approved base an owning team that tracks upstream releases and triggers rebuilds of all dependent images when the base digest changes.
IM·K	**TAG INSTEAD OF DIGEST** Dockerfiles and deployments reference tags, so what runs changes whenever the tag moves. STRIDE T (Clean) · CAPEC Partial: CAPEC-690, CAPEC-186 · CWE Direct: CWE-494	**What is built and what is deployed are named by content, so a reference resolves to the same bytes every time it is used.** 1 Resolve tags to digests at build time and write the digest, not the tag, into Dockerfiles, manifests and Helm values. 2 Run an update bot that proposes digest bumps as reviewed changes, so a moved tag becomes a visible diff rather than a silent change. 3 Reject at admission any pod spec whose image reference lacks a digest (see OR·J).
IM·Q	**SECRETS IN LAYERS** Secrets copied or set during the build remain in a layer or in build history, even if a later step deletes them. *Seen in: Codecov Docker image credential (2021).* STRIDE I (Clean) · CAPEC Direct: CAPEC-37 · CWE Direct: CWE-212, CWE-538	**Build-time secrets are available to the step that needs them and never become part of any layer, cache entry or history record.** 1 Provide credentials to build steps through BuildKit secret mounts or SSH forwarding, and forbid ARG, ENV and COPY for secret material. 2 Use multi-stage builds that copy only named artifacts into the final stage, so build tooling and intermediate files stay behind. 3 Scan every layer and the image history for secrets before push, and fail the build on any finding. 4 Rotate any credential found in a published image, because deleting the tag does not recall copies already pulled.
IM·J	**FETCH DURING IMAGE BUILD** Build steps download scripts and install unpinned packages, outside the controls on application dependencies. STRIDE T (Clean) · CAPEC Direct: CAPEC-538, CAPEC-185 · CWE Direct: CWE-494, CWE-829	**An image build draws its inputs through the same controlled, pinned path as the application's own dependencies.** 1 Route all network access from image builds through the organisation's dependency proxy, and deny direct egress from builders. 2 Pin operating system and language packages installed in Dockerfiles to exact versions, and verify a checksum for any file fetched by URL. 3 Build a sample of images twice from the same inputs and compare the results, treating any difference as an unpinned input to track down.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
IM·10	**BUILDER ESCAPE** The image builder runs privileged, and a hostile Dockerfile or build context breaks out onto the build host. *Seen in: Leaky Vessels BuildKit flaws (2024).* STRIDE E (Clean) · CAPEC Direct: CAPEC-480 · CWE Direct: CWE-250, CWE-653	**Building an image grants no more authority over the build host than running an untrusted program would.** 1 Run image builds with a rootless builder in a user namespace, with no access to the host's container daemon socket. 2 Build Dockerfiles from forks, external contributors or tenants on ephemeral builders, ideally microVMs, that hold no secrets and are destroyed after each job. 3 Patch builders and their runtimes on a defined clock, and track builder versions in the same inventory as production components.
IM·9	**FETCH ON START** The container downloads code or configuration when it starts, so what runs is not what was scanned or signed. STRIDE T (Clean) · CAPEC Partial: CAPEC-185 · CWE Direct: CWE-494, CWE-829	**Everything a container executes is either in the image that was scanned and signed, or verified at load against an identity the build recorded.** 1 Move plugin, script and model downloads from the entrypoint into the build, so they are scanned and covered by the image signature. 2 Pin each object that must still be loaded at run time by digest, and verify its signature before executing it. 3 Deny egress from workloads by default with NetworkPolicy, so an undeclared startup fetch fails visibly in testing.
IM·8	**MALICIOUS PUBLIC IMAGE** Public hubs host images with miners or backdoors, some named to pass for official images. *Seen in: docker123321 images on Docker Hub (2018).* STRIDE S, T (Clean) · CAPEC Direct: CAPEC-616, CAPEC-185 · CWE Direct: CWE-1357, CWE-829	**Workloads run only images from sources the organisation has reviewed, and adding a source is a decision rather than a pull.** 1 Pull public images only through an internal registry proxy that allows listed namespaces and verified publishers. 2 Require a recorded review, with an owner and a justification, before a new upstream namespace joins that allow list. 3 Enforce allowed registries at admission, so an image referenced directly from a public hub is rejected.
IM·7	**ROOT AND TOOLS BY DEFAULT** Images run as root and ship shells, package managers and setuid binaries that help an attacker after entry. STRIDE E (Clean) · CAPEC Partial: CAPEC-69 · CWE Direct: CWE-250	**A runtime image contains only what the application needs and starts with no privilege it does not use.** 1 Build runtime images on distroless or scratch bases in a final stage that contains no shell, package manager or compiler. 2 Set a numeric non-root USER in the image, and enforce runAsNonRoot and allowPrivilegeEscalation false at admission. 3 Strip setuid and setgid bits from the final filesystem during the build, and fail the build if any remain.
IM·6	**STALE IMAGE** Images are built once and run for months, collecting unpatched components. STRIDE E (Loose) · CAPEC Partial: CAPEC-310 · CWE Direct: CWE-1395, CWE-1104	**Images have a maximum age, and a patched base reaches running workloads by automatic rebuild rather than by someone remembering.** 1 Trigger a rebuild and redeploy of every dependent image when an approved base image publishes a new digest. 2 Rebuild all images on a fixed schedule, for example weekly, even when no source has changed. 3 Record build time in signed provenance, and block admission of images older than the age limit unless an owned exception exists.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
IM·5	**BUILD CONTEXT LEAKAGE** The build context sends the whole repository, including VCS metadata and environment files, into the image. STRIDE I (Clean) · CAPEC Direct: CAPEC-37, CAPEC-497 · CWE Direct: CWE-527, CWE-538	**The build sees only the files it needs, and the final image contains only files someone meant to ship.** 1 Commit a deny-by-default .dockerignore that admits only listed paths, so .git, .env and credential files never reach the builder. 2 Copy named paths into the final stage rather than whole directories, so new files in the repository do not ship by default. 3 Compare each image's file list with the previous release, and fail the build when VCS metadata, key files or environment files appear.
IM·4	**INVISIBLE LAYER CONTENTS** Image scanners read package databases, so binaries copied in without a package manager escape the inventory. STRIDE T (Loose) · CAPEC None · CWE Direct: CWE-223	**Image inventory is built from the files the image contains, not only from what package managers recorded.** 1 Use scanners that hash and identify binaries, archives and other files on disk as well as reading package databases (see BM·K). 2 Record any software copied in without a package manager in the image's SBOM, with its source and checksum. 3 Scan the flattened filesystem the container will see, with whiteouts and overwrites applied, so files removed or replaced by later layers are reported correctly.
IM·3	**MULTI-ARCHITECTURE INCONSISTENCY** Variants of one image for different architectures differ in content, and only one is scanned or signed. STRIDE T (Loose) · CAPEC Partial: CAPEC-477 · CWE Partial: CWE-347	**Every platform variant of an image receives the same checks as the one people test, and the index binds them together.** 1 Build all variants from the same source commit in one pipeline run, and emit provenance for each platform manifest. 2 Scan and sign each platform manifest as well as the image index, and make verification fail if any variant lacks either. 3 Verify at admission the digest of the platform manifest the node will actually pull, not only the index.
IM·2	**SELF-DECLARED METADATA** Labels for source, version and maintainer are written by the builder and trusted by downstream tools. STRIDE S, T (Clean) · CAPEC Direct: CAPEC-691, CAPEC-690 · CWE Direct: CWE-348, CWE-807	**Downstream decisions rely on facts attested by the build platform, and labels written by the builder are treated as hints.** 1 Take source repository, commit and builder identity from signed provenance (see PV), and ignore labels for any policy or ownership decision. 2 Fail verification when an image's source or version label disagrees with its provenance. 3 Inventory the tools that read labels, such as scanners, dashboards and finding routers, and move each one to provenance fields.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

STATION 15 · LINE 2 · ARTIFACT TO RUNTIME

CONTAINER RUNTIME (RT)

A container is a set of kernel features, not a security boundary unless it is configured as one. These cards cover container runtimes, isolation, the host kernel and sandboxes.

ID	THREAT	MITIGATION
RT·A	**CONTAINER ESCAPE** A runtime or kernel flaw lets a process break out of its container onto the host. *Seen in: runc CVE-2019-5736 (2019); runc CVE-2024-21626 (2024).* STRIDE E (Clean) · CAPEC Direct: CAPEC-480 · CWE Partial: CWE-653, CWE-1395	**Hosts are patched on a measured clock and treated as expendable, so an escape reaches little beyond the node itself.** 1 Patch container runtimes and node kernels within a defined window by replacing nodes from current images rather than patching in place. 2 Run untrusted or multi-tenant code in a microVM or user-space kernel sandbox selected through a RuntimeClass. 3 Give nodes no credentials beyond their own, restrict kubelets with Node authorisation, and block pod access to the metadata service.
RT·K	**PRIVILEGED CONTAINER** Workloads run privileged or with powerful capabilities, which amounts to root on the host. STRIDE E (Clean) · CAPEC Partial: CAPEC-480, CAPEC-233 · CWE Direct: CWE-250	**Workloads receive only the kernel privileges they need, and any exception is visible, owned and time-limited.** 1 Enforce Pod Security Admission at the restricted level, which forbids privileged mode and requires all capabilities to be dropped except NET_BIND_SERVICE. 2 Grant exemptions per namespace through a reviewed policy change that records an owner and an expiry date (see OR·J). 3 Report every running privileged pod or added capability from the cluster's own state each day, and alert on any without a current exemption.
RT·Q	**HOST MOUNTS AND NAMESPACES** Host paths, the runtime socket, or the host's process and network namespaces are shared into containers. STRIDE E, I (Clean) · CAPEC Partial: CAPEC-480 · CWE Direct: CWE-653, CWE-250	**A container sees only its own filesystem, processes and network unless a reviewed exception says otherwise.** 1 Deny hostPath volumes, hostPID, hostIPC and hostNetwork at admission by default. 2 Never mount the container runtime socket into workloads, and run agents that need host access on dedicated nodes. 3 Allow any remaining hostPath mount only read-only, by exact path, in namespaces named in the exception policy.
RT·J	**SHARED KERNEL ACROSS TRUST LEVELS** Workloads of different trust share one kernel, so one kernel exploit crosses them all. STRIDE E (Clean) · CAPEC Partial: CAPEC-480 · CWE Direct: CWE-653	**Workloads share a kernel only with workloads of the same trust level.** 1 Label nodes by trust level and use taints, tolerations and node affinity so untrusted and sensitive workloads never schedule together. 2 Run untrusted tenants and customer code in microVM sandboxes, giving each its own guest kernel. 3 Place the most sensitive workloads, such as signing services and secret stores, on dedicated node pools or clusters.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
RT·10	**UNCONFINED SYSCALLS** No seccomp, AppArmor or SELinux profile applies, so the whole kernel attack surface is open to the container. STRIDE E (Clean) · CAPEC Partial: CAPEC-480 · CWE Partial: CWE-250, CWE-653	**Every container runs under a kernel confinement profile, and running unconfined needs an explicit, reviewed exception.** 1 Set the RuntimeDefault seccomp profile for all pods, through the kubelet default or the pod spec, and enforce it at admission. 2 Apply AppArmor or SELinux profiles to every workload, with custom profiles for services that warrant tighter confinement. 3 Alert on any running container that the runtime reports as unconfined for seccomp or its LSM.
RT·9	**CREDENTIAL REACH FROM CONTAINERS** Containers can reach the cloud metadata service, node credentials or mounted service-account tokens they do not need. STRIDE I, E (Clean) · CAPEC Partial: CAPEC-150, CAPEC-560 · CWE Partial: CWE-1220, CWE-668	**A container can reach only the credentials its own workload needs, and node and cloud credentials stay out of its reach.** 1 Block pod access to the cloud metadata endpoint with NetworkPolicy, and require IMDSv2 with a hop limit of one where the provider offers it. 2 Disable service-account token automount by default, and mount projected, audience-bound tokens only into pods that call the API. 3 Give each workload its own cloud identity through workload identity federation instead of relying on the node's role.
RT·8	**EXPOSED RUNTIME API** The container daemon's API listens on the network without authentication, so anyone who reaches it can start containers. *Seen in: Graboid worm (Unit 42, 2019).* STRIDE S, E (Clean) · CAPEC Direct: CAPEC-36 · CWE Direct: CWE-306	**The container runtime's API is reachable only locally, and any remote management is authenticated and authorised.** 1 Bind the runtime API to a local Unix socket only, and disable TCP listeners in the daemon configuration. 2 Require mutual TLS with client certificates for any remote management that remains, and firewall it to named management hosts. 3 Scan node and network ranges for runtime API ports such as 2375 and 2376, and alert on any listener that appears.
RT·7	**UNDETECTED RUNTIME DRIFT** Processes, binaries and connections absent from the image appear at run time and nothing notices. STRIDE R, T (Loose) · CAPEC Partial: CAPEC-549 · CWE Partial: CWE-778	**Expected behaviour is derived from the image and the workload, and any departure from it raises an alert that someone owns.** 1 Deploy eBPF or audit-based runtime monitoring that alerts on processes, binaries and outbound connections absent from the image's baseline. 2 Write detection rules for high-value signals such as shells in production containers, connections to mining pools and new executables in writable paths. 3 Route runtime alerts to a named on-call rota with a playbook for isolating the pod and preserving evidence (see RT·2).
RT·6	**WRITABLE ROOT FILESYSTEM** The container's filesystem is writable, so an attacker can drop tools and persist across process restarts. STRIDE T (Clean) · CAPEC Partial: CAPEC-558 · CWE Partial: CWE-732	**A container's filesystem matches its image for its whole life, and writable space is small, named and temporary.** 1 Set readOnlyRootFilesystem to true for every container and enforce it at admission. 2 Provide writable paths as named emptyDir volumes with size limits, backed by memory where possible. 3 Alert when an executable is written to or run from a writable volume.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
RT·5	**NO USER NAMESPACE** Root inside the container maps to root on the host. STRIDE E (Clean) · CAPEC Partial: CAPEC-480 · CWE Direct: CWE-250, CWE-653	**Privilege inside a container does not carry over to the host, whatever user the process runs as.** 1 Enable user namespaces for pods with hostUsers set to false, so container root maps to an unprivileged host UID range. 2 Run workloads as a non-root UID set in the image and enforced at admission with runAsNonRoot. 3 Check on each node pool that a container's root process appears on the host with an unprivileged UID.
RT·4	**RESOURCE EXHAUSTION** Without limits on CPU, memory, processes and disk, one container starves its neighbours or the node. STRIDE D (Clean) · CAPEC Partial: CAPEC-130 · CWE Direct: CWE-770	**Every workload runs inside limits it cannot raise, so one container cannot exhaust the node it shares.** 1 Require CPU, memory and ephemeral storage requests and limits on every container, with namespace defaults set by LimitRange. 2 Set a per-pod PID limit in the kubelet configuration, and cap each namespace's totals with ResourceQuota. 3 Reserve node resources for the kubelet and system daemons, and place tenants with unpredictable load on separate node pools.
RT·3	**UNPACKING FLAWS** The runtime unpacks image layers with path or symlink bugs that write outside the container's filesystem. STRIDE T, E (Clean) · CAPEC Direct: CAPEC-132, CAPEC-126 · CWE Direct: CWE-22, CWE-59	**Image layers from outside the organisation are unpacked by a current runtime in a place where a path escape reaches nothing of value.** 1 Patch container runtimes and image unpacking libraries on a defined clock, tracking advisories for path traversal and symlink flaws. 2 Pull and unpack images from untrusted sources on isolated, ephemeral nodes that hold no secrets. 3 Verify layer digests against the signed manifest before unpacking, and reject layers containing absolute paths, parent-directory entries or device files.
RT·2	**EPHEMERAL EVIDENCE** Container logs and filesystems disappear when the container is deleted, taking the evidence with them. STRIDE R (Clean) · CAPEC Partial: CAPEC-571, CAPEC-268 · CWE Partial: CWE-778	**Evidence of what a container did outlives the container, in a store the workload cannot alter.** 1 Ship container logs and runtime audit events off the node as they are written, to a store with separate administrators. 2 Follow a response procedure that checkpoints or snapshots a suspect container's filesystem before the container is deleted. 3 Retain runtime and API audit records for longer than typical detection delays, and test that a terminated pod's history can be retrieved.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ORCHESTRATION (OR)

The orchestrator's API is a remote shell on every node for whoever holds the right role. These cards cover the Kubernetes API, RBAC, admission, operators and Helm, nodes and cluster networking.

ID	THREAT	MITIGATION
OR·A	**EXPOSED CONTROL PLANE** The API server, kubelet, etcd or a dashboard is reachable from the internet or accepts anonymous requests. *Seen in: Tesla Kubernetes console (2018).* STRIDE S, E (Clean) · CAPEC Direct: CAPEC-36, CAPEC-180 · CWE Direct: CWE-306	**Control plane components accept only authenticated requests over private paths, and etcd answers only the API server.** 1 Expose the API server on a private endpoint or to named source ranges, and reach it from outside through an authenticated access proxy. 2 Disable anonymous authentication on the API server and kubelets, and set kubelet authorisation to Webhook mode. 3 Require client certificates for etcd, and firewall it so that only API server hosts can connect. 4 Scan public address ranges for API server, kubelet, etcd and dashboard ports, and alert on any response.
OR·K	**OVER-BROAD RBAC** Roles grant wildcards, secret reads, pod create and exec, or bind and impersonate, so many identities can become cluster-admin. STRIDE E (Clean) · CAPEC Partial: CAPEC-180, CAPEC-233 · CWE Direct: CWE-266, CWE-1220	**Each identity holds the narrowest roles its job needs, and the paths by which a role can grow into cluster-admin are known and few.** 1 Replace wildcard verbs and resources with explicit lists, and reject new wildcard roles in policy checks on RBAC manifests. 2 Treat pod create, exec, secret read, bind, escalate and impersonate as privileged rights granted only to named groups. 3 Run an RBAC escalation analysis in CI and on a schedule, and fail when any identity gains a path to cluster-admin.
OR·Q	**ADMISSION WEBHOOK COMPROMISE** Admission webhooks see every object with high privilege, so one flaw gives cluster-wide control, and a failing webhook may fail open. *Seen in: IngressNightmare, CVE-2025-1974 (2025).* STRIDE E, T (Clean) · CAPEC Partial: CAPEC-234, CAPEC-554 · CWE Partial: CWE-636, CWE-250	**Admission webhooks are held to the standard of the cluster they guard, and their failure behaviour is a deliberate choice.** 1 Run webhooks in a dedicated namespace whose NetworkPolicy admits only traffic from the API server. 2 Set failurePolicy to Fail for security webhooks, scope them with namespace and object selectors, and alert when a webhook times out. 3 Pin webhook images by digest, grant them only the permissions they use, and patch them on the same clock as the control plane. 4 Prefer in-process ValidatingAdmissionPolicy where a rule can be written in CEL, removing both a network hop and a privileged service.
OR·J	**ADMISSION POLICY ABSENT** Nothing at admission blocks privileged pods, host mounts, untrusted registries or unsigned images. STRIDE E, T (Clean) · CAPEC Partial: CAPEC-233 · CWE Partial: CWE-693	**The cluster refuses workloads that break its security baseline, whoever or whatever submits them.** 1 Enforce Pod Security Admission at the restricted level on every namespace, with a lower level only under a reviewed exemption. 2 Admit images only from allowed registries, by digest, carrying a signature and provenance from the expected build identity (see PV). 3 Grant exemptions per namespace in version-controlled policy with an owner and an expiry, and report them on a schedule.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
OR·10	**CLUSTER SECRETS EXPOSURE** Secrets sit unencrypted in etcd, readable by broad roles and copied into many namespaces. STRIDE I (Clean) · CAPEC Partial: CAPEC-639, CAPEC-180 · CWE Direct: CWE-312, CWE-732	**Secrets are encrypted at rest under a key held outside the cluster, and readable only by the workloads that use them.** 1 Enable etcd encryption at rest with a KMS provider, so the key encryption key never sits on control plane disks. 2 Grant get and list on secrets only to named service accounts in their own namespace, and remove secret access from broad roles. 3 Scope secret operators and external secret controllers to the namespaces they serve, instead of copying secrets across the cluster.
OR·9	**OPERATOR PRIVILEGE** Operators and controllers run with cluster-admin to manage resources, so one compromised controller owns the cluster. STRIDE E (Clean) · CAPEC Partial: CAPEC-234, CAPEC-122 · CWE Direct: CWE-250, CWE-266	**A controller holds only the permissions for the resources and namespaces it manages.** 1 Install operators in namespaced mode where supported, with Roles limited to the resource types they reconcile. 2 Review the RBAC each third-party operator requests before install, and pin its image and manifests by digest. 3 Run operators on system node pools away from tenant workloads, and audit their API calls for actions outside their declared purpose.
OR·8	**HOSTILE CHART OR MANIFEST** Charts from public repositories carry unexpected RBAC, hooks or images. STRIDE T, E (Clean) · CAPEC Direct: CAPEC-538, CAPEC-446 · CWE Direct: CWE-829, CWE-1357	**What is installed in the cluster is rendered output someone reviewed, from a chart whose origin is verified.** 1 Mirror third-party charts into an internal repository, pin each by version and digest, and verify its signature or provenance file. 2 Render charts in CI and review the full manifest diff, including RBAC, CRDs and images, before every install or upgrade. 3 Disable chart hooks unless allow-listed, and apply the same admission policy to hook jobs as to any other workload.
OR·7	**FLAT CLUSTER NETWORK** With no network policies, any pod reaches any other pod, the control plane and the metadata service. STRIDE E, I (Clean) · CAPEC None · CWE Partial: CWE-653	**Pods reach only the services they are meant to call, and every other path is closed by default.** 1 Apply a default-deny ingress and egress NetworkPolicy to every namespace, then allow named flows. 2 Block pod egress to the cloud metadata service and the API server, except for workloads that need them. 3 Encrypt traffic between workloads with mutual TLS from a service mesh or the network plugin.
OR·6	**NAMESPACE AS TENANT BOUNDARY** Teams or customers are separated only by namespace, while sharing nodes, CRDs and cluster-scoped resources. STRIDE E, I (Clean) · CAPEC Partial: CAPEC-124 · CWE Direct: CWE-653	**Tenants who might be hostile to each other share nothing that one can read or change on the other's behalf.** 1 Deny tenants the right to create cluster-scoped resources such as CRDs, ClusterRoles and webhook configurations. 2 Place mutually untrusted tenants on separate clusters, or at the least on dedicated node pools with sandboxed runtimes. 3 List what tenants share in each cluster, including nodes, CRDs, ingress controllers and DNS, and review the list whenever a tenant is added.
OR·5	**NODE TO CLUSTER ESCALATION** A compromised node's credentials read secrets for pods on other nodes or change objects beyond the node. STRIDE E (Clean) · CAPEC Partial: CAPEC-560, CAPEC-233 · CWE Direct: CWE-1220, CWE-266	**A node's credentials allow only the actions needed for the pods scheduled on that node.** 1 Enable the Node authoriser and the NodeRestriction admission plugin, so a kubelet can read only secrets for pods bound to it. 2 Keep privileged workloads such as controllers and CI runners off nodes that run untrusted pods. 3 Replace nodes after a short maximum lifetime from immutable images, and rotate kubelet certificates automatically.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
OR·4	**SERVICE ACCOUNT TOKEN ABUSE** Default service accounts with automounted tokens are usable from any compromised pod. STRIDE E, S (Clean) · CAPEC Partial: CAPEC-560, CAPEC-150 · CWE Partial: CWE-1188, CWE-250	**Pods carry an API credential only when they need one, and that credential is short-lived and narrowly scoped.** 1 Set automountServiceAccountToken to false on every default service account, and enable it per workload only where needed. 2 Use projected service-account tokens with a short expiry and a bound audience, and remove legacy long-lived token secrets. 3 Bind no roles to default service accounts, and check that in policy for every namespace.
OR·3	**AUDIT LOGGING GAP** API audit logging is off, too coarse or kept on the cluster, so actions cannot be reconstructed. STRIDE R (Clean) · CAPEC Partial: CAPEC-268, CAPEC-571 · CWE Direct: CWE-778	**Every change to the cluster can be reconstructed from records that the cluster's own administrators cannot rewrite.** 1 Enable API server audit logging with a policy that records metadata for all requests and full request bodies for RBAC, workload and exec changes. 2 Ship audit events as they are written to an external store under separate administration. 3 Alert when audit events stop arriving, when the audit policy changes, or when event objects are deleted in bulk.
OR·2	**VERSION LAG** Clusters and add-ons run unsupported versions with known flaws. STRIDE E (Loose) · CAPEC Partial: CAPEC-310 · CWE Direct: CWE-1104, CWE-1395	**Clusters and their add-ons stay within supported versions on a planned schedule rather than one forced by an incident.** 1 Upgrade clusters on a fixed cadence that keeps each within the upstream support window, replacing nodes as part of each upgrade. 2 Inventory add-ons such as ingress controllers, CNI plugins, CSI drivers and operators with their versions, and track their advisories. 3 Check manifests for deprecated APIs in CI before each upgrade, and fail on any API removed in the next version.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

STATION 17 · LINE 2 · ARTIFACT TO RUNTIME

DELIVERY AND DEPLOYMENT (DL)

The last hop decides what actually runs, and every earlier control is worth only what this hop enforces. These cards cover CD tools, GitOps controllers, promotion, environments, approvals, rollback and release channels.

ID	THREAT	MITIGATION
DL·A	**GITOPS REPOSITORY AS PRODUCTION** Whoever can write to the repository a GitOps controller syncs from controls production, and it is often protected less than application code. STRIDE T, E (Clean) · CAPEC Direct: CAPEC-445 · CWE Partial: CWE-732, CWE-494	**The repository a controller syncs from is protected as production access, and the controller applies only changes it can attribute.** 1 Protect the synced branch with required review by production owners and no bypass for administrators. 2 Require signed commits, and configure the GitOps controller to verify signatures against an allowed key set before syncing. 3 Keep configuration repositories separate from application repositories, with membership no wider than the group that holds production access.
DL·K	**CD CONTROLLER COMPROMISE** The CD controller holds credentials for every cluster and environment, so a flaw or an exposed console hands over all of them. *Seen in: Argo CD CVE-2022-24348 (2022).* STRIDE E, I (Clean) · CAPEC Partial: CAPEC-180, CAPEC-122 · CWE Direct: CWE-653, CWE-250	**No single delivery component holds every environment's credentials, and its interfaces are guarded as closely as the environments it reaches.** 1 Run a CD controller per environment or cluster that pulls from inside its target, so no central instance holds credentials for all of them. 2 Keep the controller's console and API off the internet, and federate administrator access through single sign-on with MFA. 3 Patch the controller on a defined clock, and give its service account rights only in the namespaces it deploys to.
DL·Q	**PROMOTION WITHOUT RE-VERIFICATION** An artifact reaches production by rebuild, retag or a different digest, so what was tested is not what ships. STRIDE T (Clean) · CAPEC Partial: CAPEC-29 · CWE Partial: CWE-367, CWE-494	**Promotion moves one immutable artifact between environments, and each environment verifies it again before it runs.** 1 Promote by digest, referencing the exact image that passed earlier stages, and forbid rebuilds in promotion pipelines. 2 Record test results as attestations bound to the digest, so a different digest carries no evidence to pass on. 3 Verify signature, provenance and test attestations for the digest at each environment's admission (see PV).
DL·J	**APPROVAL BYPASS** Deployment approvals can be skipped by direct API calls, re-runs, admin override or self-approval. STRIDE E (Clean) · CAPEC Partial: CAPEC-207, CAPEC-36 · CWE Partial: CWE-841, CWE-424	**Approval is enforced by the deploy target, which accepts only changes carrying a valid approval from someone other than the requester.** 1 Issue approvals as signed attestations that the deploy target or its admission controller verifies, not as a step in the pipeline's interface. 2 Reject approvals from the person who authored or triggered the change, and require fresh approval for any re-run or changed input. 3 Remove administrator override from deployment environments, or treat each use as a break-glass event that alerts the owning team.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
DL·10	**DIRECT-TO-PRODUCTION PATH** Engineers and tools hold standing access to deploy outside the pipeline. STRIDE T, E (Clean) · CAPEC Partial: CAPEC-554 · CWE Direct: CWE-424, CWE-266	**The pipeline is the only standing route to production, and any other access is temporary, recorded and reconciled.** 1 Remove standing write access to production from people and tools, and grant it just in time with approval and an expiry. 2 Log every break-glass session, alert the owning team, and review each one afterwards. 3 Configure the GitOps controller to detect and revert out-of-band changes, and alert on every revert.
DL·9	**RELEASE CHANNEL HIJACK** The update server, download page, CDN or update channel that delivers software to customers is compromised. *Seen in: 3CX (2023); Polyfill.io (2024).* STRIDE S, T (Clean) · CAPEC Direct: CAPEC-669, CAPEC-186 · CWE Direct: CWE-494, CWE-347	**Clients trust an update because they have verified its signature and freshness, not because of where it came from.** 1 Sign every release with a key held in an HSM, and have clients verify the signature before installing. 2 Protect update metadata with TUF or an equivalent scheme, using separate roles, threshold signing and expiry to resist freeze, rollback and mix-and-match attacks. 3 Hold the download domain, DNS and CDN accounts under dedicated MFA-protected identities, and monitor for changes to records and published files.
DL·8	**ENVIRONMENT BOUNDARY LEAK** Staging and production share clusters, credentials, networks or data, so compromising staging reaches production. STRIDE E, I (Clean) · CAPEC Partial: CAPEC-124, CAPEC-560 · CWE Direct: CWE-653	**Lower environments share nothing with production that would let a compromise of one reach the other.** 1 Give each environment its own cloud accounts, clusters, credentials and network, with no routes from staging into production. 2 Issue pipeline identities per environment, so production refuses the credentials of a staging job. 3 Use synthetic or masked data in lower environments, and block automated copying of production data into them.
DL·7	**FLAG AND REMOTE CONFIG TAMPERING** Feature flags and remote configuration change production behaviour without a code change or review. STRIDE T, R (Clean) · CAPEC Partial: CAPEC-176, CAPEC-75 · CWE Partial: CWE-15, CWE-912	**Changes to production behaviour through flags or remote configuration are reviewed, attributable and reversible, like code changes.** 1 Restrict production flag and configuration changes to named roles, and require a second approver for flags that enable features or relax controls. 2 Write every flag change to an immutable audit log with actor, time, old value and new value. 3 Remove code behind retired flags, and treat activation of any flag that gates dormant code as a release.
DL·6	**VALUES AND OVERLAY INJECTION** Values files and overlays from contributors or tenants inject arbitrary manifests at render time. STRIDE T, E (Clean) · CAPEC Partial: CAPEC-77, CAPEC-153 · CWE Direct: CWE-1336, CWE-915	**Contributors and tenants set only declared values, and what renders is checked as the manifest it really is.** 1 Define a schema for each chart's values, and reject undeclared keys or types, including raw YAML blocks. 2 Never pass tenant-supplied strings through template functions such as tpl, so a value cannot become markup. 3 Validate the fully rendered manifests against admission policy in CI before they reach the controller.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
DL·5	**ROLLBACK TO VULNERABLE VERSION** Rollback restores a version with a known vulnerability, or an attacker forces a downgrade. STRIDE T (Clean) · CAPEC Partial: CAPEC-166 · CWE Partial: CWE-1328	**A rollback is a deployment like any other, subject to the same vulnerability policy and authorisation.** 1 Maintain a deny list of digests with known critical vulnerabilities, and reject them at admission whether deployed forward or back. 2 Limit rollback to the roles that can approve deployments, and log each one with its target version. 3 Keep a patched recent version available as the rollback target, so restoring service does not mean restoring a known flaw. 4 Enforce version monotonicity in update clients, so an older signed release cannot be served as an update (see DL·9).
DL·4	**CANARY BLIND SPOT** Progressive rollout watches availability, not security signals, so a malicious build passes the canary. STRIDE T (Loose) · CAPEC None · CWE None	**Progressive rollout judges a build on its security behaviour as well as its health, and its stages cannot be skipped quietly.** 1 Include security signals in canary analysis, such as new outbound destinations, unexpected processes and shifts in authentication errors. 2 Compare a build's runtime behaviour with a baseline from previous releases, because a malicious build can keep its error rate normal. 3 Require approval to skip or shorten rollout stages, and record each skip with the actor and the reason.
DL·3	**DEPLOYMENT RECORD GAP** There is no reliable record of what was deployed where, when and by whom. STRIDE R (Clean) · CAPEC Partial: CAPEC-268 · CWE Direct: CWE-778	**What runs where is answerable from a record that the deploying systems write but cannot change.** 1 Emit a deployment record with digest, environment, actor, approval and time from the deploy target rather than from the pipeline. 2 Write deployment records to an append-only store outside the delivery platform. 3 Join deployment records with live cluster inventory, so a query for a digest or version returns every location within minutes.
DL·2	**ORPHANED PREVIEW ENVIRONMENTS** Preview and ephemeral environments outlive their purpose, running old code with real credentials. STRIDE I, E (Loose) · CAPEC None · CWE Partial: CWE-772, CWE-459	**Preview environments live only as long as the change that created them and hold nothing a stranger could use.** 1 Tear down preview environments automatically when their pull request closes, and after a maximum age regardless. 2 Give previews their own short-lived, low-privilege credentials and synthetic data, never production secrets. 3 Put previews behind authentication by default, and report any that are publicly reachable each day.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

STRUCTURAL HAZARDS (T)

These ten hazards are properties of the software factory itself, so no single station can fix them. Apply them to the whole design after the stations.

ID	THREAT	MITIGATION
T10	**TRANSITIVE TRUST** Every component trusts its upstream, recursively, so the factory is as strong as the weakest link in a tree nobody has fully drawn. *Seen in: SolarWinds (2020).* STRIDE T, S (Loose) · CAPEC Partial: CAPEC-446 · CWE Partial: CWE-1357, CWE-829	**Trust in an artifact rests on verification at each link the organisation can see, and the links it cannot verify are named and owned.** 1 Draw the trust chain for each critical artifact, from source, tools, actions, base images and dependencies through to runtime, and keep it as a maintained architecture record. 2 Mark on that chain where verification is enforced and where it is assumed, and give each assumption an owner. 3 Identify the upstreams whose fall would reach every artifact, and give each a compensating control such as mirroring, pinning or independent verification.
T9	**NAMES THAT MOVE** Tags, branches, versions and "latest" are pointers, not contracts, and most of the factory addresses things by name. *Seen in: trivy-action (2026).* STRIDE T, S (Clean) · CAPEC Partial: CAPEC-690 · CWE Direct: CWE-471, CWE-494	**Every hand-off between stations refers to content by an immutable identifier, and mutable names are resolved once, visibly, and on record.** 1 Resolve tags, branches and version ranges to digests or commit hashes at one recorded point, and carry the resolved identifier through every later stage. 2 Treat a moved name as an event, with update bots proposing the new digest as a reviewed change. 3 Test the design by assuming every mutable name moves overnight, and list the stations that would silently run different content.
T8	**THE PIPELINE OUTRANKS EVERYONE** Automation concentrates write access to source, registries, clouds and production in identities nobody logs in as. *Seen in: Codecov (2021).* STRIDE E (Clean) · CAPEC Partial: CAPEC-122, CAPEC-560 · CWE Partial: CWE-250, CWE-653	**Automation holds no more authority than the people who can direct it, and that authority is split so no one pipeline holds it all.** 1 Split pipeline identities by stage and environment, so building, signing, publishing and deploying use separate credentials with separate trust policies. 2 Map who can make each pipeline act, through commits, triggers, workflow edits and runner access, and keep that group no larger than the group approved for its targets. 3 Require human approval or a second independent identity for actions no single person may take alone, such as publishing a release or deploying to production.
T7	**RUNNING STRANGERS' CODE IS THE JOB** Install scripts, build plugins, actions and test suites execute third-party code by design at every stage. *Seen in: Shai-Hulud (2025).* STRIDE E, T (Loose) · CAPEC Partial: CAPEC-538, CAPEC-446 · CWE Direct: CWE-829, CWE-653	**Each stage decides what third-party code can reach before that code runs, because running it is unavoidable.** 1 Run third-party code such as install scripts, plugins, actions and test suites in ephemeral sandboxes with no secrets and egress limited to the proxy. 2 Separate stages that execute third-party code from stages that hold signing keys or publishing credentials, passing only digest-addressed artifacts between them. 3 Inventory where third-party code executes and with which privileges, and review the inventory whenever a stage gains a secret.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
T6	**THE GUARD IS INSIDE THE WALL** Scanners, firewalls, signers and SBOM generators run with the widest reach, so compromising a control compromises everything it protects. *Seen in: Trivy and KICS (2026).* STRIDE E, T (Loose) · CAPEC Partial: CAPEC-670, CAPEC-578 · CWE Partial: CWE-250, CWE-654, CWE-1357	**Security controls meet a stricter standard than the code they check, and no single control's verdict is final.** 1 Pin, verify and update scanners, firewalls, signers and SBOM generators through a supply chain held to the highest internal tier. 2 Give each control read access to what it checks and write access only to its own results, with signing and publishing keys held elsewhere. 3 Cross-check critical verdicts with an independent second control, so a control that lies is caught by disagreement.
T5	**CHECKED IS NOT RUN** Each station checks an artifact, and the next receives something that may differ. STRIDE T (Clean) · CAPEC Partial: CAPEC-29 · CWE Partial: CWE-367, CWE-494	**One digest is carried end to end, and every check is bound to the digest it examined.** 1 Fix the artifact digest at build and carry it unchanged through scanning, signing, promotion and admission, with no rebuild, retag or re-fetch. 2 Record each check's result as an attestation over that digest, so its evidence cannot be reused for different bytes. 3 Verify at runtime admission that the digest about to run holds every attestation the policy requires.
T4	**CONFIGURATION IS CODE** YAML, HCL, Dockerfiles and policy files are executable instructions that are reviewed and owned as settings. STRIDE T (Loose) · CAPEC Partial: CAPEC-75, CAPEC-35 · CWE None	**Files that change what executes are treated as code, with owners, review and tests to match.** 1 Classify workflow files, Dockerfiles, IaC, Helm values and policy files as executable, and assign each a code owner. 2 Enforce review of those files through code-owner rules by people who understand what the files execute. 3 Lint, policy-test and diff the rendered output of configuration changes in CI, as for application code.
T3	**SELF-PROPAGATION** Compromised output becomes someone else's input: stolen publish tokens republish, pipelines infect pipelines, and a factory becomes a worm's carrier. *Seen in: Shai-Hulud (2025).* STRIDE T, E (Loose) · CAPEC Partial: CAPEC-538, CAPEC-442 · CWE Partial: CWE-494, CWE-829	**Nothing is published or consumed automatically without a step that a compromised peer cannot satisfy.** 1 Publish only through short-lived trusted-publishing identities scoped to one package, with human approval of each release, so stolen tokens cannot republish. 2 Apply a cooldown before new upstream versions enter builds, giving compromised releases time to be found and withdrawn. 3 Require provenance from the expected build identity on consumed packages, so output from a compromised factory without that identity is refused.
T2	**SHARED FATE** Most factories depend on the same few platforms, registries, actions and base images, so their outages and compromises are everyone's. *Seen in: tj-actions (2025).* STRIDE D, T (Clean) · CAPEC Partial: CAPEC-538 · CWE Partial: CWE-1357	**Dependence on shared providers is known, verified at each use and survivable when a provider fails.** 1 List the external platforms, registries, actions and base images without which the factory cannot build or deploy. 2 Mirror critical dependencies, images and actions internally, so builds and rollbacks continue while a provider is down. 3 Verify signatures and digests on everything taken from a shared provider, so its compromise yields refused content rather than accepted content.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

ID	THREAT	MITIGATION
T1	**EVIDENCE LIVES WHERE THE ATTACKER WORKS** Logs, provenance, audit trails and SBOMs are stored in the systems an attacker compromises, and are rewritten or deleted with the breach. STRIDE R, T (Clean) · CAPEC Direct: CAPEC-268, CAPEC-571 · CWE Partial: CWE-653, CWE-732	**Evidence is written to an independent store as it is produced, under administrators who cannot change the systems it describes.** 1 Stream logs, audit trails, provenance and SBOMs to an append-only store in a separate account with separate administrators. 2 Record signatures and attestations in a transparency log, so later deletion or rewriting is detectable. 3 Retain the inputs needed to reconstruct a past build, such as source commits, lockfiles and base digests, and test a reconstruction periodically.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

SOURCES

Every known case in the tables was checked against primary or near-primary sources in October 2026, and eight citations were corrected or replaced where the incident did not show the card's mechanism. The frameworks and taxonomies come first, then a source for each known case, then the facts that are still contested or rest on a single analysis.

- Taxonomy of Attacks on Open-Source Software Supply Chains, Ladisa, Plate, Martinez, Barais
- Risk Explorer for Software Supply Chains, SAP
- SLSA v1.0 threats and mitigations and supply chain threats overview
- OSC&R launch announcement, OX Security
- OWASP Top 10 CI/CD Security Risks
- Threat Matrix for Kubernetes, Microsoft
- Catalog of Supply Chain Compromises, CNCF TAG Security
- NIST SP 800-204D
- Enhancing Software Supply Chain Security through STRIDE-Based Threat Modelling of CI/CD Pipelines (2025)
- Systematic Review of Identity-Centric Security in Cloud-Native CI/CD Pipelines
- Guidance for the Trivy supply chain compromise, Microsoft
- Trivy compromised by TeamPCP, Wiz
- Trivy supply chain compromise timeline, SafeDep
- Shai-Hulud research note, Cloud Security Alliance
- Dependency Firewall product initiative, GitLab (source for the pre-policy cache gap)
- Package Firewall policy, Endor Labs (source for exceptions that skip every check)

Known case sources

- Workstation: CircleCI incident report (2023); s1ngularity post-mortem, Nx, and s1ngularity's aftermath, Wiz (2025); hackerbot-claw GitHub Actions exploitation, StepSecurity (2026); GlassWorm, Koi Security (2025); Contagious Interview VS Code tasks, OpenSourceMalware, and Void Dokkaebi, Trend Micro; package hallucination study, Spracklen et al. (2024)
- Version control: Bypassing required reviews using GitHub Actions, Cider Security (2021); tj-actions/changed-files alert, CISA (2025); Trivy incident discussion, Aqua Security (2026); Gentoo GitHub incident report (2018); Stolen OAuth user tokens, GitHub (2022); xz Utils disclosure, oss-security (2024); Trojan Source (2021); CAPEC-695 repo jacking; Cross Fork Object Reference, Truffle Security (2024); CVE-2022-36804

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

- Pipeline definition and runners: Ultralytics attack analysis, PyPI, and zizmor and the Ultralytics injection, William Woodruff (2024); reviewdog/action-setup, Wiz (2025); Codecov security update (2021); AWS-2025-015, AWS (2025); Playing with fire: PyTorch, John Stawinski (2024); CVE-2024-27198; Jenkins advisory 2024-01-24; Artifact poisoning in Rust, Legit Security (2022); ArtiPACKED, Unit 42 (2024)
- Identity and build: GitHub to AWS keyless authentication flaws, Datadog Security Labs (2023); Leaked MSI source code with Intel OEM keys, Binarly (2023); SUNSPOT technical analysis, CrowdStrike (2021); Webmin exploit page and CVE-2019-15107; XcodeGhost, Unit 42 (2015)
- Dependency resolution: GHSA-pjwm-rvh2-c87w, ua-parser-js (2021); Dependency confusion, Alex Birsan (2021); Compromised nightly dependency, PyTorch (2022); colors and faker, Snyk (2022); crossenv malware, npm (2017); event-stream issue 116 (2018); Revival Hijack, JFrog (2024); npm lockfiles as a blind spot, Liran Tal (2019); CVE-2022-23812, node-ipc; The massive hole in the npm ecosystem, Darcy Clarke (2023); kik, left-pad and npm, npm (2016)
- Scanners and provenance: Trivy supply chain attack, Aqua Security, and KICS GitHub Action, Wiz (2026); Malware using NVIDIA's stolen certificates, BleepingComputer (2022)
- Registries and firewalls: npm debug and chalk compromised, Aikido (2025); Docker Hub database breach, BleepingComputer (2019); PEAR compromise, Rapid7 (2019); Go module proxy caching, Socket (2025); Shai-Hulud v2 spreads from npm to Maven, The Hacker News (2025); StarJacking, Checkmarx (2022)
- Infrastructure, images and runtime: Security smells in IaC scripts, Rahman et al., ICSE (2019); Leaky Vessels, Snyk (2024); Backdoored images removed from Docker Hub, Ars Technica (2018); CVE-2019-5736; CVE-2024-21626; Graboid, Unit 42 (2019)
- Orchestration and delivery: Tesla cloud cryptojacking, Wired (2018); IngressNightmare, Wiz (2025); CVE-2022-24348, Argo CD; 3CX software supply chain compromise, Mandiant (2023); Polyfill supply chain attack, Sansec (2024)

Contested or single-source facts

- The link from the reviewdog compromise to the tj-actions token theft is described by Wiz as likely, not proven.
- Travis CI disputes that its key stolen in the 2022 Heroku attack gave access to customer repositories.
- The motive of the hackerbot-claw account, and any link between it and TeamPCP, are not established.
- The identity of "Jia Tan" in the xz Utils backdoor is unknown, and claims of state sponsorship are speculation.
- GlassWorm's self-propagation rests on Koi Security's analysis.
- The Ultralytics cache poisoning is confirmed by PyPI, but the exact poisoning steps are inferred.
- Package hallucination is shown by research; no attack in the wild exploiting a hallucinated name was found.
- Webmin's own page cites CVE-2019-15231, while NVD records the backdoor as CVE-2019-15107.
- The docker123321 case rests on Kromtech's analysis as reported by Ars Technica: 17 images pulled more than 5 million times, most carrying cryptominers, and one reported by a GitHub user as backdoored. The roughly $90,000 mined is Kromtech's estimate from a single wallet address. The report does not say the images were named to pass for official ones; that part of IM·8 is the general pattern, not this case.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.

- In the Tesla case, Wired reports RedLock's finding: an unprotected Kubernetes console with a pod holding AWS credentials, used to run a hidden miner. What was exposed is unresolved. Tesla says only internal engineering test cars were affected, while RedLock did not check what the attackers could reach in the S3 bucket. How much was mined is unknown.

© 2026 Jonathan Brett Crawley. CC BY-NC-ND 4.0, with additional permissions. Share it unchanged, with credit, for non-commercial use; ask before shipping it in a product.