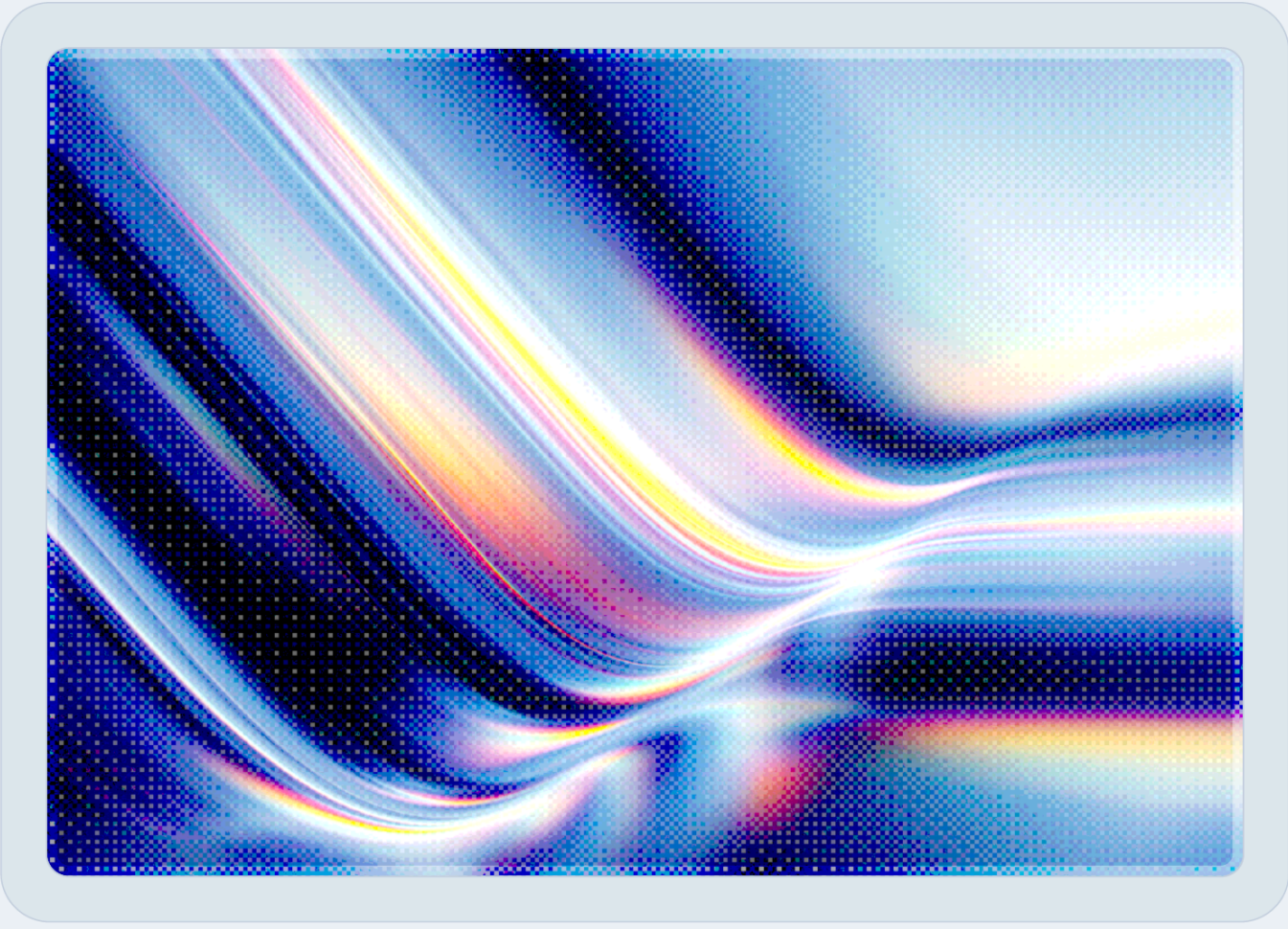
CSAI Foundation | Cloud Security Alliance

# Eight Seconds to First Use: Secrets in Public Container Images

## The Validate-Wait-Exploit Drain of AI API Keys

2026-11-27

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

**© 2026 Cloud Security Alliance. Some rights reserved.**

You may download, store, display, view, print, redistribute, and link to this document in its original, unmodified form, provided that attribution to the Cloud Security Alliance is maintained and all trademark and copyright notices remain intact.

This document may not be modified or altered. You may quote portions of the document as permitted by the Fair Use provisions of the United States Copyright Act, provided that attribution is given to the Cloud Security Alliance.

This document may be shared on professional and social media platforms in its original form with attribution.

*This document was generated with AI assistance and has not undergone official CSA review and approval processes.*

## Key Takeaways

Secrets baked into public container images are harvested at machine speed. Stony Brook University researchers published 300 honeypot images across eight public registries over four months, and the fastest observed use of a planted AWS key came 7.86 seconds after publication, with a median time to first API call of 51.19 seconds [1]. All 100 planted AWS keys were used within 15 minutes. Attackers also used unrestricted OpenAI and Anthropic keys to exhaust the roughly $60 in test-account credit, although no restricted key was used that way [1][2]. The scanning bots overwhelmingly extracted secrets through static analysis of image layers and registry APIs rather than by running the containers [1], which means runtime controls on a published image offer little protection against discovery.

The study also points to a multi-stage abuse pattern that this note calls "validate-wait-exploit." Leaked AWS keys were checked almost immediately, and about half of the scanning IPs stopped at that check [2]. The AI keys showed a different timeline: OpenAI keys sat quiet for weeks before one was used to burn through the remaining balance in under 24 hours. The researchers speculate that the wait reflects validated keys being resold and held until a buyer uses them [2]. That interpretation is the researchers' hypothesis, but it is consistent with independently documented key-checker tooling and storefronts that resell LLM access [8][9].

The exposure base is large, and AI tooling has opened a new leak path. Flare identified 10,456 Docker Hub images uploaded in a single month that contained exposed keys, with AI model keys the most frequently leaked category at almost 4,000 [4]. GitGuardian separately found 613 distinct secrets, 99 of them valid, in Claude Code settings files shipped inside public Docker Hub images, although it also observed that monthly leaks of that file have fallen sharply since an April 2026 peak [6]. Deleting a secret from an image does not end the incident: Flare reported that 75% of developers who removed leaked secrets did not revoke or rotate the underlying keys [4].

Organizations should therefore treat any secret that has appeared in a pushed image as compromised from the moment of publication. The practical response combines pre-push scanning, build-time secret mounts, scoped and short-lived credentials, provider-side spend caps, and rotation playbooks that run in minutes rather than days.

# Background

Container images are layered archives. Each instruction in a Dockerfile can add a filesystem layer, and anything written into a layer — an `.env` file swept up by `COPY . .`, a token passed as a build argument, a configuration file left in a working directory — remains recoverable from that layer even if a later instruction deletes it. When the image is pushed to a public registry, every layer becomes downloadable by anyone. Docker's own documentation directs builders to use secret mounts (`RUN --mount=type=secret`) precisely because those values are exposed to a single build step and are not persisted into the final image [10].

Flare's December 2025 study quantified how often this goes wrong. Scanning Docker Hub images uploaded between November 1 and November 30, 2025, Flare identified 10,456 images containing one or more exposed keys, and approximately 42% of affected images contained five or more sensitive values [4]. After filtering to high and critical findings, Flare associated 101 of 205 namespaces with identifiable companies, including a Fortune 500 company and a major national bank, with many exposures originating in personal or contractor-owned accounts that sit outside corporate visibility [4][5]. Flare identified `.env` files copied into the build context as the most frequent cause, followed by hard-coded API tokens in source files and secrets written directly into Dockerfiles, where they remain visible in the image's configuration and build history (Flare describes this as the image manifest) [4].

AI coding assistants have introduced a newer leak path. GitGuardian's November 2026 analysis found that when a developer approves a Claude Code shell command with "Yes, and don't ask again," the literal command string — including any credential typed inline — is saved as an allow rule in `.claude/settings.local.json` [6][7]. Although Claude Code ignores that file in the local Git configuration, Docker does not honor `.gitignore`, so a broad `COPY` instruction carries the file into the image unless `.dockerignore` excludes it [6][7]. GitGuardian counted 109,284 secret occurrences in `.claude/settings.local.json` and 29,613 in `.claude/settings.json` across 543 Docker Hub image repositories and 7,625 tags, collapsing to 613 distinct secrets because a single committed file is rebuilt into every tag [6]. One valid Cloudflare API token appeared in 559 tags of a single image [6]. In practice, agent configuration files can function as an unintended secrets store.

The trend for this particular leak path is improving. GitGuardian observed that monthly leaks of `settings.local.json` fell 92%, from a peak of 12,535 occurrences in April 2026 to 959 by October, and Anthropic stated that it is investigating and prototyping built-in secret scanning [6]. Images that have already been pushed, however, retain the exposed values regardless of later improvements.

The Stony Brook work, by Billy Tsouvalas and Nick Nikiforakis and presented at the ACM Conference on Computer and Communications Security (CCS) in November 2026 [1][3], moves the question from how often secrets leak to how quickly leaked secrets are used. Its honeypot images carried AWS, OpenAI, and Anthropic API keys alongside decoy SSH and database credentials, with each key tied to monitoring so that any use would appear in provider logs [1][2]. The study figures cited in this note are drawn from published coverage of the paper [1][2].

# Security Analysis

## Discovery Is Continuous, Automated, and Static

The researchers use the umbrella term "ContainerBots" for the scanners they observed, while noting that the activity did not appear to come from a single operator [1]. Their images were never advertised, yet they were pulled more than 75,000 times [1]. Monitoring rarely detected anyone actually running the images. Instead, as reported by BleepingComputer, the bots worked "through static analysis, e.g., unpacking image layers or querying the registry API to extract filesystem contents, without ever instantiating the container" [1]. Docker Hub accounted for nearly all AWS key activity, with Harbor also seeing some abuse; public coverage does not name the other six registries [1].

The defensive implication is that the window between `docker push` and compromise is effectively zero for planning purposes. A median of roughly 51 seconds to first use leaves no practical room for human review or for remediation triggered by a post-publication finding. The researchers' recommendations reflect this: they argue that Docker Hub should scan images for secrets before they go live and that access to the public feed of recently published or updated images should be restricted or rate-limited to slow mass scanners [1]. They reported their findings to Docker in May 2026. According to the paper, Docker acknowledged the disclosure and its security team was reviewing the findings, but a small follow-up test in August found no platform changes [1][2]. Until registries act, scanning remains the publisher's responsibility, and it must happen before the push.

## The Validate-Wait-Exploit Pattern

Synthesizing the Stony Brook honeypot results with Sysdig's earlier LLMjacking investigations, this note describes the abuse of leaked credentials in four phases, each with different timing and different defensive opportunities. Not every key passes through each phase, and no single study observed the full

sequence end to end. "Validate-wait-exploit" is a descriptive label used in this note, not terminology from the underlying studies; in that label, probing is treated as part of validation. The table below summarizes what is reported for each phase.

| Phase | Observed behavior | Evidence | Defensive opportunity |
|---|---|---|---|
| Validate | Near-immediate liveness check. For AWS keys, nearly every bot began with `GetCallerIdentity`; about half of scanning IPs stopped after that check. | Stony Brook study [1][2] | Alert on identity and model-list calls from unfamiliar sources; revoke on first touch |
| Probe | Capability and quota discovery without paying for inference, such as invoking a model with invalid parameters or querying logging configuration | Sysdig LLMjacking analysis [8] | Monitor validation errors and logging-configuration queries; enable invocation logging |
| Wait | Validated OpenAI keys sat idle for weeks; researchers speculate resale | Stony Brook study [2]; resale markets [9] | The longest window for rotation, available only if the leak is detected |
| Exploit | Balance exhausted rapidly; one OpenAI key burned its remaining balance in under 24 hours. For AWS keys, only around 6% of scanning actors moved to sustained resource use. | Stony Brook study [2] | Hard spend caps, per-key budgets, anomaly detection on token volume |

The validation phase is also cheap for attackers. Sysdig's 2024 investigation of LLMjacking documented attackers checking stolen cloud credentials against ten AI services and deliberately invoking Amazon Bedrock with an invalid `max_tokens_to_sample` value: a validation error confirmed model access without generating billable output [8]. The same attackers queried model-invocation logging settings, which Sysdig assessed as an effort to avoid detection, and noted that the OAI Reverse Proxy tool states it will not use AWS keys with logging enabled [8]. In 2025, Sysdig described concurrent checker scripts that sort stolen keys by model access, subscription tier, and remaining quota, and documented a storefront selling 30-day proxy access built on stolen credentials for $30 [9].

The wait phase is the least understood. The researchers' own explanation — "We speculate that the wait phase reflects validated keys being resold and held idle until a buyer eventually uses them" — is plausible given the resale economy described above, but it has not been independently confirmed [2]. Other explanations, such as operators batching keys for later campaigns, cannot be ruled out from public data, and the public coverage does not say whether the Anthropic keys showed the same delay as the OpenAI keys. Whatever the cause, the wait is the defender's best opportunity. An organization that detects the leak during that period can revoke the key before the spend arrives, which is why fast detection still matters even though validation itself cannot be prevented once a key is public.

The exploit phase is where cost becomes material. In the Stony Brook experiments, attackers consumed the full balance of roughly $60 in prepaid OpenAI and Anthropic credit loaded by the researchers [2]. That small figure reflects the experimental budget, not the ceiling. Sysdig calculated a worst-case cost exceeding $46,000 per day for a single compromised cloud account abusing Claude 2.x at quota limits across four regions; this is a theoretical maximum based on 2024 pricing rather than an observed loss, and current model pricing differs [8]. In one observed reverse proxy, Sysdig estimated close to $50,000 in model usage accumulated in about 4.5 days [9].

## Key Scope Shapes the Outcome

One finding from the study deserves particular attention from AI platform teams. In the second round of tests, every OpenAI and Anthropic key that attackers used to exhaust credits had unrestricted permissions, and no restricted key was abused that way [2]. The sample is small and the result should not be over-generalized. It does suggest that scanners check what a key can do before investing in it, and that narrowly scoped keys are less attractive to resell. Combined with the AWS observation that only around 6% of scanning actors moved to sustained use [2], the data indicates that, under the study's conditions, exposure is near-certain to be detected by attackers, while exploitation is selective and depends partly on what the key permits.

## Deletion Is Not Remediation

Because image layers and tags persist and may already have been pulled by scanners, removing a secret from the latest tag does not withdraw it. Flare found that roughly 25% of developers removed exposed secrets from their containers or manifests within one to two days, but that in most cases the credential itself was not revoked; Flare puts the share of removals without revocation or rotation at 75% [4][5]. GitGuardian likewise warns that deleting a settings file from GitHub does nothing for images that have already been published [7]. The same logic applies to CSA's earlier analysis of Gitea CVE-2026-27771, where images thought to be private were in fact anonymously pullable for years and had to be treated as a credential-disclosure event rather than a patching exercise [13].

### Upstream Detection Is Uneven

Source-code platforms have established detection mechanisms that registries currently lack in public practice. Under GitHub's secret scanning partner program, tokens found in public repositories are reported to the issuing provider, which may revoke them [12]. In October 2026 GitHub added new Anthropic service and user API key types and made push protection block them by default [11]. These mechanisms do not apply to secrets inside container image layers pushed to a registry, and the Stony Brook researchers' call for registry-side pre-publication scanning addresses exactly that gap [1]. GitGuardian also observed that inline credentials in command strings and environment assignments often do not match provider-specific detection patterns, so even good scanners will miss some of what agent configuration files capture [7].

## Recommendations

### Immediate Actions

Organizations should inventory every public image published under corporate, team, personal, and contractor namespaces, scan all historical tags and layers rather than only the latest tag, and rotate any credential found. Rotation must be performed at the provider; deleting the image or file is not sufficient [4][7]. Security teams should add `.env`, `.claude/`, and other AI-assistant configuration directories to `.dockerignore` across repositories and reject broad `COPY . .` patterns in review [6][7]. For AI provider accounts, teams should enable hard spending limits where the provider supports them — checking whether a budget setting only alerts or actually caps usage — and confirm that billing alerts reach an on-call channel [2]. Any key that has appeared in a public image should be treated as compromised, even if it was public for only a minute [2].

### Short-Term Mitigations

Over the next quarter, secret scanning should become a blocking gate before `docker push` in CI, covering image layers and manifests in addition to source code, because a check after publication is too late [2]. Builds that need credentials should use BuildKit secret mounts rather than `ARG`, `ENV`, or copied files [10]. AI API keys should be issued per service and per environment with the narrowest permissions and project scoping the provider allows, given the study's small-sample finding that only unrestricted keys were drained [2]. Where possible, applications should route model calls through an internal gateway that holds provider keys server-side, so that no long-lived provider credential ever

needs to exist in an image. That design moves the secret rather than eliminating it, and the gateway's own credentials then become a high-value target that requires hardening, as CSA's analysis of the LiteLLM vulnerability chain illustrates [18]. Detection engineering should cover the validation phase: alerts on `GetCallerIdentity` and Bedrock validation errors from unfamiliar IP ranges, queries against model-invocation logging configuration, and first use of a key from a new network [8]. Planting canary tokens in images and build contexts can provide early warning when a pipeline begins leaking [4].

### Strategic Considerations

Longer term, organizations should move AI workloads away from static API keys toward workload identity and short-lived credentials, treating AI keys as non-human identities subject to the same lifecycle governance as cloud service accounts. Developer use of AI coding assistants should be paired with policy on where those tools persist approvals and command history, and on avoiding inline credentials in commands that an agent may record [7]. Referencing secrets by environment-variable name in commands (for example, `$N8N_API_KEY`) ensures that saved allow rules record the name rather than the value [6]. Organizations should also bring personal and contractor registry namespaces into scope through contract terms and monitoring, since Flare found that many corporate exposures, including one tied to a Fortune 500 company, originated there [4][19]. Finally, enterprises and industry groups should press registry operators to adopt pre-publication secret scanning and provider-notification programs comparable to those that already exist for source-code hosting [1][12], and AI providers should extend automatic revocation to secrets found outside Git.

## CSA Resource Alignment

CSA's analysis of **Gitea CVE-2026-27771: Private Container Registry Exposure** [13] is the most directly related prior work. That note argued that container images are serialized snapshots of production configuration and that any image exposure must be handled as a credential-disclosure event requiring systematic rotation, rebuilding, and access-log review, with BuildKit secret mounts recommended to reduce secret density. The Stony Brook findings extend that argument from an access-control flaw to deliberately public images and add a timing constraint: where the Gitea case involved a multi-year lookback, public registries compress the time from exposure to first use to seconds.

The AI-key abuse described here connects to CSA's **JetBrains Marketplace: AI Plugin Campaign Steals LLM API Keys** [14], which documented trojanized AI coding plugins exfiltrating developer LLM keys; to **LLMjacking Evolves: Stolen AI Compute as Attack Infrastructure** [15], which examined how stolen AI credentials are validated, resold, and used; and to **LiteLLM AI Gateway: Critical**

**Vulnerability Chain Exposes API Keys** [18], which examined key exposure through a compromised AI gateway. Taken together, they suggest that LLM API keys are being collected through several channels — malicious plugins, compromised gateways, and now passive registry scanning — and the resale storefronts documented by Sysdig [9] offer one plausible destination. If the Stony Brook researchers' resale hypothesis is correct, the validate-wait-exploit pattern would be the registry-scanning branch of that economy.

At the governance level, CSA's **Defining Non-Human Identity (NHI)** [16] provides the framing for treating AI API keys as managed machine identities with ownership, scoping, rotation, and decommissioning requirements, rather than as configuration strings. CSA's work on **CISA GovCloud Credential Exposure: Capacity and Contractor Governance** [19] addresses the related problem of static credentials and contractor identities that sit outside corporate visibility. The **AI Controls Matrix (AICM)** [17] supplies the control vocabulary for implementing these recommendations, particularly its Cryptography, Encryption and Key Management domain for secret lifecycle and rotation, its Identity and Access Management domain for least-privilege scoping of AI credentials, its Application and Interface Security domain for secure build pipelines, and its Logging and Monitoring domain for detecting validation-phase activity.

# References


[1] Toulas, Bill. "Cloud keys leaked in Docker Hub images are stolen within seconds." BleepingComputer, November 17, 2026.

[2] Arntz, Pieter. "A bot tested an AWS key within 8 seconds of exposure." Malwarebytes, November 24, 2026.

[3] Tsouvalas, Billy and Nikiforakis, Nick. "Gone in 8 Seconds: Uncovering Malicious Bots Operating on Public Container Registries." ACM Digital Library (ACM CCS), November 2026.

[4] Morag, Assaf. "Thousands of Exposed Secrets Found on Docker Hub, Putting Organizations at Risk." Flare, December 10, 2025.

[5] Page, Carly. "10K Docker images spray live cloud creds across the internet." The Register, December 11, 2025.

[6] Valadon, Guillaume. "Claude Code settings.local.json Secrets: 17 months of exposure on GitHub, & Docker Hub." GitGuardian, November 9, 2026.

[7] Sharma, Shweta. "Claude Code's 'Don't Ask Again' button has a secrets-spilling problem." CSO Online, November 11, 2026.

[8] Brucato, Alessandro. "LLMjacking: Stolen Cloud Credentials Used in New AI Attack." Sysdig, May 6, 2024.

[9] Sysdig Threat Research Team. "LLMjacking targets DeepSeek." Sysdig, February 7, 2025.

[10] Docker. "Build secrets." Docker Documentation, accessed November 27, 2026.

[11] GitHub. "Secret scanning blocks Anthropic keys by default, adds Salesforce." GitHub Changelog, October 13, 2026.

[12] GitHub. "Secret scanning for partners." GitHub Docs, accessed November 27, 2026.

[13] Cloud Security Alliance. "Gitea CVE-2026-27771: Private Container Registry Exposure." CSA Labs, May 28, 2026.

[14] Cloud Security Alliance. "JetBrains Marketplace: AI Plugin Campaign Steals LLM API Keys." CSA Labs, June 19, 2026.

[15] Cloud Security Alliance. "LLMjacking Evolves: Stolen AI Compute as Attack Infrastructure." CSA Labs, June 18, 2026.

[16] Cloud Security Alliance. "Defining Non-Human Identity (NHI)." Cloud Security Alliance, 2026.

[17] Cloud Security Alliance. "AI Controls Matrix." Cloud Security Alliance, 2025.

[18] Cloud Security Alliance. "LiteLLM AI Gateway: Critical Vulnerability Chain Exposes API Keys." Cloud Security Alliance.

[19] Cloud Security Alliance. "CISA GovCloud Credential Exposure: Capacity and Contractor Governance." Cloud Security Alliance.