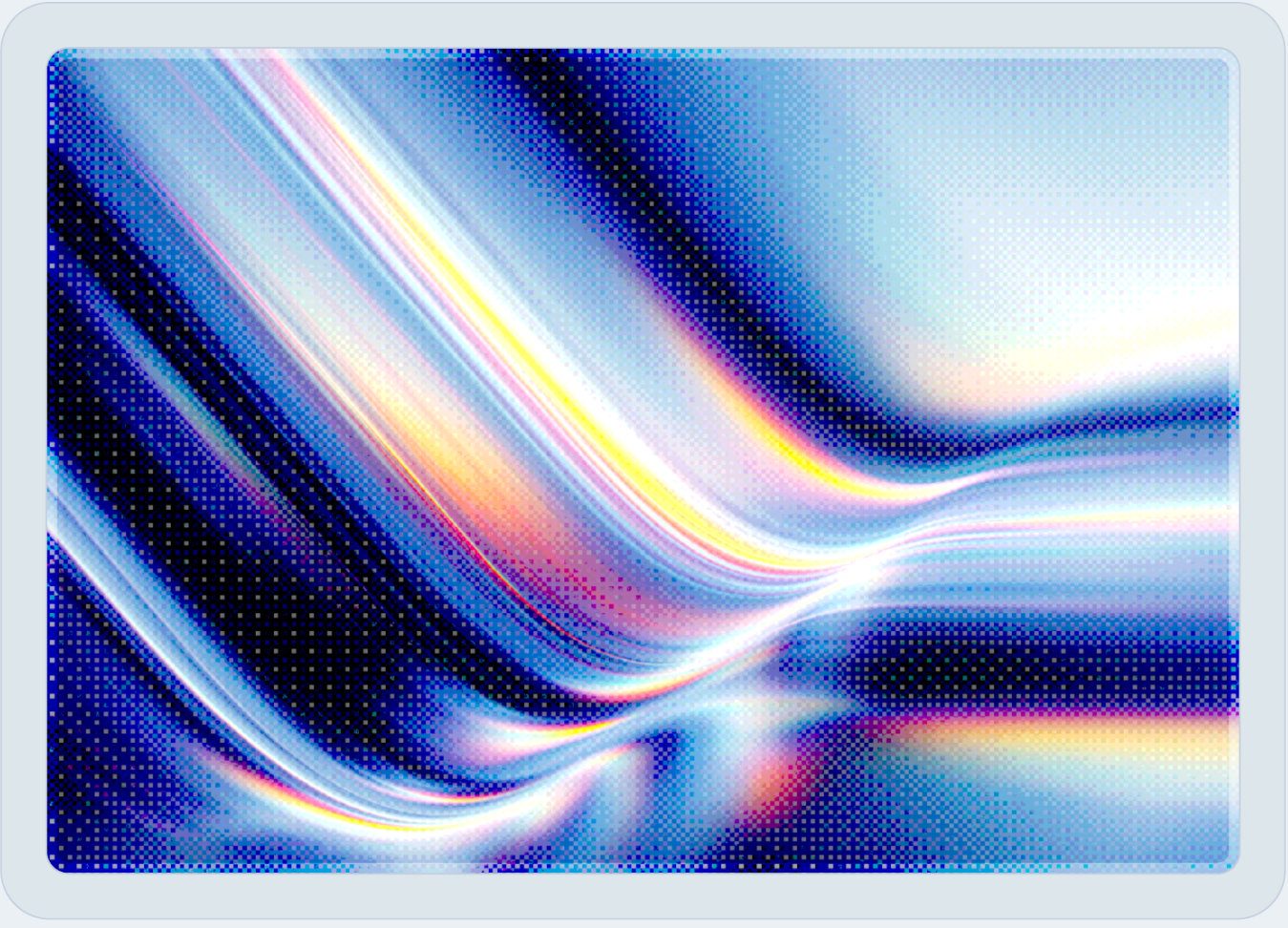
CSAI Foundation | Cloud Security Alliance

The wfx Campaign: Hijacked GitHub Accounts Harvest CI Secrets

How 4,837 compromised accounts turned GitHub Actions into AI-key and cloud-identity harvesters, and what defenders must do now

2026-11-12

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

Beginning at approximately 12:45 UTC on November 10, 2026, a threat actor holding stolen GitHub credentials began pushing a credential-stealing shell script, which labels itself "wfx," into GitHub repositories at scale [1]. Aikido Security, which named the campaign "wfx-loot," counted 4,837 compromised accounts opening probe pull requests across 61,764 repositories by the afternoon of November 11, and found indications that the stealer actually executed in at least 1,806 repositories; its summary elsewhere cites 1,835 [1]. SafeDep found malicious commits in 8,093 repositories [3], and StepSecurity reported that GitHub search returned more than 63,000 probe pull requests by 07:40 UTC on November 12 [2]. Researchers measured at different times and by different methods, so the counts differ, but every source agrees that the campaign was still active when it was reported. StepSecurity is explicit that the attack exploits no vulnerability in GitHub Actions itself [2]. The attacker is writing code with legitimately issued but stolen tokens.

Five findings matter most for security leaders. First, one stolen token exposes every repository its owner can write to: Aikido counted 10,431 probe pull requests in repositories the compromised account did not own, reached through collaborator or organization access [1][5]. Second, AI provider keys are a primary target, with the payload explicitly hunting OpenAI, Anthropic, Gemini, Hugging Face, Cursor, Copilot, Codex, and other AI tool credentials alongside cloud and registry secrets [1]. Third, the stealer tries to escalate CI identity into cloud identity. It mints GitHub OIDC tokens and attempts to exchange them for AWS, Azure, and GCP sessions wherever the repository is trusted, uses harvested service-account keys for GCP impersonation, chains into other roles, and replays tokens against HashiCorp Vault and Kubernetes [1][2][3]. Fourth, its persistence survives secret rotation, because it rewrites AWS role trust policies to accept any AWS account and adds federated identity credentials named `wfx-ci-…` to Azure app registrations [1][3][5]. Finally, its anti-forensics defeat scanning of a repository's current state: malicious commits are reverted within about a minute and workflow runs and logs are deleted, so a gap in run numbers is often the only trace left [1][4].

Organizations that find evidence of the payload executing should treat every secret and cloud role reachable from the affected workflows as compromised. They must also hunt for and remove the cloud backdoors, because rotating keys alone does not remove them.

Background

The wfx campaign is the latest in a series of attacks that use hijacked developer accounts to inject secret-stealing workflows into GitHub repositories. Its most direct predecessor is GhostAction, a campaign first discovered in September 2025 that resurfaced in waves in August and October 2026 [6]. In October 2026, GitGuardian reported that GhostAction operators had used about 1,900 compromised accounts to target more than 30,000 repositories, with the largest burst on October 8 and 9 aimed at 3,174 secrets [6]. By account and repository counts (4,837 accounts and 61,764 repositories against roughly 1,900 and 30,000), wfx is substantially larger than GhostAction. GitHub began blocking the GhostAction workflow from running on October 9 [6]. GitGuardian also observed that some victims had been compromised in earlier waves and apparently never rotated their secrets, which let attackers come back with the same access [6].

Aikido found that accounts hit by GhostAction were significantly more likely to be hit by wfx, and that in more than 100 repositories the wfx payload was injected into GhostAction's own workflow files [1]. Aikido nevertheless assesses the two as distinct campaigns run by different actors. It points to separate infrastructure, different payload design, and Indonesian-language comments in some newer wfx variants that have not appeared in GhostAction activity [1]. StepSecurity similarly cautions that the overlap does not prove a single operator [2]. Both teams offer a plausible explanation: stolen GitHub credentials are often sold or shared and end up in the hands of several threat actors [1][2]. In that case, an unremediated compromise from one campaign becomes the entry point for the next.

How the credentials were originally obtained remains unconfirmed for the campaign as a whole. Aikido states only that the actor used stolen GitHub credentials and tokens [1]. Endor Labs relays one victim's report that a leaked GitHub CLI OAuth token was used to open the probe pull request [4]. StepSecurity observed 173 probe pull requests from 19 accounts that GitHub attributed to the Fly.io GitHub App. It found no evidence of a breach of Fly.io's infrastructure and notes that most activity used other credentials, such as personal access tokens [2]. Bot accounts were also abused. Commits from the WebdriverIO bot account bypassed branch protection and landed 56 commits across 28 repositories [2] [4].

Security Analysis

Attack Flow

The campaign follows a consistent four-stage pattern of probe, payload delivery, revert and cleanup, and exfiltration, which the major research teams documented in closely matching terms. It starts with a probe. A compromised account creates a `wfx-docs` branch and opens a pull request titled "docs: dependency audit notes" that adds a harmless file, `docs/wfx-audit.md` [2]. This confirms write access and can trigger pull-request workflows. A probe on its own means the account is compromised, even if no payload followed [1]. About 90 to 92 percent of these probe pull requests were still open at the time of reporting [1][2], which suggests that many repository owners have not yet noticed.

Where write access is confirmed, the attacker delivers the payload through one of several channels. The table below summarizes the variants and SafeDep's counts from its analysis of public default branches [3]. Some repositories received more than one variant, which is why the per-variant counts sum to slightly more than SafeDep's total of 8,093.

Delivery method	Commit message	Repositories (SafeDep)	Notes
Steps added to an existing workflow	`ci: refresh build cache`	7,309	Step named `Restore cache <4 chars>` or `Pass runtime` [1][2]
Hidden script plus npm lifecycle hook	`chore: add ci helper`	771	`.ci-xxxxxx.sh` run from `postinstall`; also executes on developer machines [1][3]
New workflow file (SafeDep: in an empty repository)	`chore: add dependency audit workflow`	16	`.github/workflows/build-audit-xxxxxx.yml` on `chore/xxxxxxxx` [1][3]
Appended to an existing CI shell script	Varies	Not counted	Reported by StepSecurity, e.g., `update.sh` [2]

Researchers also list branches matching `ci-check-[0-9]{4}` among the campaign's indicators [1][2]. StepSecurity counted more than 21,000 commits with the message "ci: refresh build cache" between November 10 and 12 [2]; that figure counts commits rather than repositories, and many repositories received multiple injection commits. After the payload runs, a revert commit ("ci: restore" or "ci: cleanup") removes it, usually within about a minute, and the workflow run and its logs are deleted [1][2]. Endor Labs calls these "sandwich commits." The payload lives on the default branch for only seconds between two ordinary-looking commits, so scanning the repository's current state finds nothing [4].

Payload Capabilities

The stealer is a roughly 1,500-line shell script. It is heavily commented and not obfuscated [1][4]. That suggests an operator who values speed and reliability more than stealth at the code level, and who relies on post-run cleanup for concealment instead. Its collection scope goes well beyond the secrets a workflow explicitly references. It serializes every repository secret and variable using `toJSON(secrets)` and `toJSON(vars)` [2][4], a pattern Endor Labs notes that legitimate workflows rarely use [4]. More significantly, it scans the memory of the `Runner.Worker` and `Runner.Listener` processes. StepSecurity explains that this exposes decrypted secrets the workflow never requested, which is how the script-based and npm-hook variants steal secrets without modifying any workflow file [1][2].

The payload's keyword filters and file targets show clear interest in AI credentials. Aikido documents targeting of OpenAI, Anthropic/Claude, Gemini, Hugging Face, Cursor, Copilot, Codex, Continue, Codeium, Aider, Weights & Biases, and Ollama credentials, alongside environment variables matching keywords such as `openai` and `anthropic` [1]. It also collects cloud CLI credentials for AWS, Azure, and GCP. Other targets include Docker, Kubernetes, npm, PyPI, SSH, GPG, Vault, and Terraform material, shell histories, `.env` files, keystores, and the last 100 commits [1]. SafeDep additionally notes targeting of `*.tfvars` and `*.tfstate` files and cryptocurrency wallets [3]. On self-hosted runners, the script also harvests cloud instance metadata credentials and the runner's own registration material [1].

From CI Token to Cloud Identity

Compared with GhostAction-style secret scraping, the distinguishing feature of wfx is that it actively converts stolen CI identity into cloud access and then plants new trust relationships to keep that access. The Miasma campaign (see CSA Resource Alignment) abused GitHub OIDC in a similar way, but wfx combines the technique with cloud-side persistence at a scale of thousands of repositories. The injected workflow requests `id-token: write` and `statuses: write` permissions, so it can mint

GitHub OIDC tokens and attempt to exchange them with every major cloud, even in repositories with no cloud configuration [3]. The research teams documented the following escalation and persistence behaviors.

Platform	Escalation behavior	Persistence behavior
AWS	Exchanges OIDC tokens for role sessions (named `wfx-*`), chains into other IAM roles [1][3]	Calls `iam:CreateAccessKey` for long-lived keys; rewrites role trust to `"Principal": {"AWS": "*"}` [1][2]
Azure / Entra ID	Exchanges OIDC tokens or client secrets for management tokens; replays FOCI refresh tokens from cached Azure CLI sessions for Graph, ARM, and Key Vault [1]	Adds federated identity credentials named `wfx-ci-<n>` trusting the compromised repository [1][3]
GCP	Signs JWTs with harvested service-account keys; impersonates accounts via `generateAccessToken` and `signJwt` [1]	No dedicated persistence mechanism reported
Vault / Kubernetes	Replays OIDC tokens against Vault and against API servers found in kubeconfig files [1][3]	No dedicated persistence mechanism reported
GitHub	Uses `GITHUB_TOKEN` and runtime tokens [1]	Code to forge passing required status checks; not enabled in any of 101 samples StepSecurity examined [2]

The persistence mechanisms are the most consequential part of the campaign for incident responders, although the two cloud backdoors carry different levels of exposure. An AWS role whose trust policy names `"AWS": "*"` can be assumed from any AWS account in the world (absent restrictive trust-policy conditions), which gives the attacker immediate access that is independent of GitHub. The Azure federated credential, by contrast, lets any workflow on the compromised repository's main branch authenticate as the application with no stored secret [1][3]. It therefore grants renewed access to anyone who later regains write access to that repository, whether the original attacker or someone holding

another compromised account. Both backdoors remain in place after every secret is rotated [3][5]. An organization that rotates its keys and declares the incident closed could therefore leave the attacker with continuing or recoverable access.

Exfiltration and Evolving Infrastructure

Stolen data is packed into a single gzipped archive. The code supports several covert channels, including Actions cache entries, step summaries, artifacts named `wfx-loot-<run id>`, encrypted repository commits, and base32-encoded log output. However, all of these were disabled in the samples the researchers analyzed [1][2][3]. Early samples sent data by HTTP POST to `194.150.166[.]244` and `18.143.168[.]96` on port 8080 [1]. StepSecurity then observed the operator adapt. A third collection server, `153.75.82[.]159`, appeared together with a Telegram backup at 01:10 UTC on November 12, and around 05:30 UTC new copies switched to Telegram-only exfiltration through `api.telegram.org` [2]. Blocking the original IP addresses therefore does not stop the newest variants [2]. Because these channels send data directly off the runner, missing cache entries or artifacts do not prove that no data was taken [1].

Downstream Package Exposure

Most of the impact falls on CI secrets, but at least 50 compromised accounts publish to npm, Go, Packagist, PyPI, or crates [1]. Confirmed malicious versions include Packagist `digitalelvis/neuronai-studio` 3.14.3, whose payload runs when `npm install` executes inside `vendor/`. Several Go module versions are also affected (`github.com/onuragtas/docker-env` v1.0.278, v1.0.280, and per SafeDep v1.0.282, and a `github.com/vkuznet/gocurl` pseudo-version), but in those cases the payload sits only in workflow files that Go builds do not execute [1][2][3]. SafeDep scanned 218 npm, Go, and Packagist packages from affected publishers and found no other malicious published versions, although PyPI and some other registries were not checked [3]. Any package built and published between a malicious commit and its revert should be considered suspect.

Recommendations

Immediate Actions

Organizations using GitHub should first determine whether they were touched. That means searching all repositories, including private ones, for the probe pull request title "docs: dependency audit notes," the `wfx-docs`, `chore/[a-z0-9]{8}`, and `ci-check-[0-9]{4}` branches, and the commit messages listed above, from November 10 onward [1][2]. Because logs are deleted, teams should look for gaps in Actions run numbers following suspicious commits; Aikido and Endor Labs both describe this as the most reliable signal of execution [1][4]. A probe without a payload still means the authoring account is compromised. That account's personal access tokens, OAuth grants, SSH keys, and sessions should be revoked, and a password reset with phishing-resistant MFA should be required [2][4].

Where the payload ran, responders should rotate every secret the job could reach, not just those it referenced. That includes repository, environment, and organization secrets, package-registry publish tokens, cloud keys, Vault and Kubernetes credentials, self-hosted runner host credentials, and all AI provider API keys [1][2]. They must then remove cloud persistence explicitly. In AWS, that means auditing CloudTrail for `AssumeRoleWithWebIdentity` or `AssumeRole` sessions named `wfx-*` followed by `CreateAccessKey` or `UpdateAssumeRolePolicy`, and searching for trust policies containing `"AWS": "*"` [2][3]. In Entra ID, responders should delete any federated credential named `wfx-ci-*`. In GCP, they should review `GenerateAccessToken` and `SignJwt` calls from GitHub workload identity pools, and in Vault, JWT logins from the GitHub OIDC issuer [2][3]. Developers who ran `npm install` on an affected commit should treat their machines as compromised [2][3]. Runner egress to the three collection IPs and to `api.telegram.org` should be blocked unless business-justified [2].

Short-Term Mitigations

Once credentials were stolen, the campaign worked because write access to a repository amounted to the ability to run arbitrary code with access to all of its secrets. Organizations should break that equivalence. All pushes to default and release branches, not only changes to workflow files, should require pull requests with review, and CODEOWNERS protection for workflow files should extend to the scripts and package manifests that CI executes, such as `package.json` lifecycle hooks and shell scripts invoked by workflows. This matters because two of the wfx variants never touch a workflow file. Administrators should be included in branch protection, and bypass lists should be audited; bot accounts

with protection bypass, as in the WebdriverIO case, deserve particular scrutiny [2][4]. The default `GITHUB_TOKEN` should be set to read-only. Deploy-capable secrets should move into protected environments that require approval, so they are not available to every workflow [4][7].

Cloud OIDC trust should be scoped as tightly as GitHub's subject claims allow, binding roles to specific repositories, branches, and environments rather than to whole organizations [8]. Scoping alone is not sufficient, however, because wfx's Azure backdoor works by adding a trust relationship that points at the compromised repository itself. Alerting on any creation or modification of federated identity credentials and IAM trust policies is therefore as important as scoping them correctly. Teams should also use `npm ci --ignore-scripts` where feasible, alert on newly introduced lifecycle hooks, and monitor runner egress for connections to raw IP addresses on unusual ports [4].

Strategic Considerations

The wfx campaign shows that a developer's GitHub token is a high-value non-human credential in its own right. Depending on its scope, it can reach every cloud account and AI service that the developer's repositories are trusted to touch. GhostAction victims who never rotated their secrets were re-compromised [6], and Aikido found GhostAction-hit accounts disproportionately represented among wfx victims [1]. One plausible reading is that the GitHub account credentials themselves, and not only the repository secrets, were never revoked after the first incident, or that both campaigns draw on the same stolen-credential supply. Either way, prior compromise appears to be a strong risk signal, and remediation that rotates repository secrets without revoking the developer's tokens and sessions is incomplete. Organizations should treat developer tokens with the lifecycle discipline they apply to production service credentials: short lifetimes, fine-grained scopes, inventory, and verified revocation after any incident.

AI provider keys deserve specific attention. These keys are increasingly stored as CI secrets for evaluation pipelines, coding agents, and AI-enabled tests. When stolen, they convert directly into compute spend and, for providers that retain files, fine-tunes, or conversation state, potential access to that data. Organizations should place AI keys behind per-project spending caps and usage anomaly alerting, prefer short-lived or proxy-brokered credentials over static keys in CI, and include AI providers in incident rotation runbooks alongside cloud and registry credentials. Finally, the anti-forensic design of this campaign argues for streaming GitHub audit logs and workflow-run metadata to an external, tamper-resistant store. When attackers can delete run logs, evidence that never left GitHub cannot be relied on.

CSA Resource Alignment

CSA's rapid-research analysis **"Miasma: Red Hat npm Supply Chain Worm"** (June 2026) is the closest precedent to wfx and should be read alongside this note [9]. Miasma also started from a stolen GitHub credential, abused GitHub Actions OIDC, and harvested credentials across GitHub, AWS, Azure, GCP, Vault, and Kubernetes. The Miasma note identified the structural tension that wfx now exploits at far larger scale: OIDC federation and trusted publishing remove long-lived secrets but become trust surfaces in their own right once an attacker controls a workflow. The companion note **"Miasma: Cross-Registry Supply Chain Credential Harvesting"** (June 2026) follows that campaign's extension into Go and GitHub Actions credential harvesting, the same attack surface wfx targets [10]. wfx extends the pattern by planting new federated trust relationships for persistence, which makes the Miasma recommendations on OIDC scoping and infostealer credential hygiene directly applicable.

"AI Toolchain Hijacked: IDE Plugin API Key Theft" (June 2026) documented coordinated campaigns that stole AI API keys and developer credentials through IDE plugin ecosystems [11]. wfx's explicit targeting of OpenAI, Anthropic, Copilot, Cursor, and other AI tool credentials adds to growing evidence that AI keys are now a routine collection target in developer supply-chain attacks, whether the entry point is an editor extension or a CI runner. The financial consequences of such theft are analyzed in CSA's **"Who Pays for Stolen Inference? Loss Allocation After Black October"** (November 2026), which examines the October 2026 incidents, named "Black October" by Wiz Research, in which attackers stole AI provider API keys stored as environment variables on at least nine AI developer and hosting platforms. That note finds that standard provider terms generally leave the account holder liable for unauthorized usage [12], which strengthens the case for the spending caps and rotation discipline recommended above.

At the control-framework level, the **AI Controls Matrix (AICM) v1.1** provides the governance mapping for these recommendations [13]. Its Identity and Access Management domain covers lifecycle management and least privilege for developer tokens, bot accounts, and federated workload identities. The Cryptography, Encryption and Key Management domain addresses AI provider key rotation. The Supply Chain Management, Transparency, and Accountability domain and the Application and Interface Security domain address CI/CD pipeline integrity and the review of workflow changes. The Logging and Monitoring domain supports the off-platform audit log retention that wfx's log deletion makes necessary.

References

[1] Smith, Oliver / Aikido Security. "wfx-loot campaign pushes CI secret stealer to tens of thousands of repositories." Aikido Security, November 12, 2026.

[2] Shaikh, Owais / StepSecurity. "WFX Phantom: The GitHub Actions Secret Stealer That Deletes Its Tracks." StepSecurity, November 12, 2026.

[3] Singh, Kunal / SafeDep. "wfx: GitHub Actions Credential Stealer Hits 8,000 Repos." SafeDep, November 12, 2026.

[4] Subramanian, Sriram and Henrik Plate / Endor Labs. "Hijacked GitHub Accounts Push 'wfx' CI Secret Stealer to Thousands of Repositories." Endor Labs, November 12, 2026.

[5] Zorz, Mirko. "Check your GitHub repos: A secret stealer hit tens of thousands of them." Help Net Security, November 12, 2026.

[6] Lakshmanan, Ravie. "GhostAction Compromises About 1,900 GitHub Accounts, Targets Over 30,000 Repositories." The Hacker News, October 15, 2026.

[7] GitHub. "Secure use reference." GitHub Docs, accessed November 12, 2026.

[8] GitHub. "OpenID Connect." GitHub Docs, accessed November 12, 2026.

[9] Cloud Security Alliance. "Miasma: Red Hat npm Supply Chain Worm." CSA Lab Space, June 3, 2026.

[10] Cloud Security Alliance. "Miasma: Cross-Registry Supply Chain Credential Harvesting." CSA Lab Space, June 27, 2026.

[11] Cloud Security Alliance. "AI Toolchain Hijacked: IDE Plugin API Key Theft." CSA Lab Space, June 21, 2026.

[12] Cloud Security Alliance. "Who Pays for Stolen Inference? Loss Allocation After Black October." CSA Lab Space, November 11, 2026.

[13] Cloud Security Alliance. "AI Controls Matrix (AICM) v1.1." Cloud Security Alliance, 2025.