UNC6426: nx Supply Chain to AWS Admin via OIDC

How the s1ngularity Attack Enabled GitHub-to-AWS Trust Chain Exploitation in Under 72 Hours

Unofficial AI-assisted Research

Cloud Security Alliance AI Safety Initiative

2026-03-11

Key Takeaways

A threat actor cluster designated UNC6426 by Google/Mandiant exploited the downstream consequences of the August 2025 s1ngularity npm supply chain attack to achieve full AWS administrator access at a victim organization in under 72 hours. The attack chain required no zero-day exploits and no novel malware. Instead, UNC6426 combined three independently documented but rarely chained risks: a compromised open-source build tool package, an overly permissive GitHub-to-AWS OIDC trust policy, and a CloudFormation IAM capability that permitted a low-privilege role to create its own successor with administrator access. Each link in this chain represents a class of misconfiguration documented at scale — the OIDC trust policy flaw alone was confirmed across more than 275 AWS accounts [4] — suggesting these controls are frequently deprioritized in the initial deployment of developer infrastructure. Their combination produced a complete cloud takeover.

While the s1ngularity attack's technical mechanics have been extensively analyzed [1][2][6][8], the downstream exploitation of its stolen credential corpus has received less systematic treatment in public threat reporting. UNC6426's intrusion, documented in Google's Cloud Threat Horizons Report H1 2026, provides a rare end-to-end documented example of this second-order exploitation pattern, from initial credential acquisition through destructive post-compromise action [3][16].

Organizations that use GitHub Actions with AWS OIDC federation, deploy CloudFormation stacks from CI/CD pipelines, or distribute developer tooling through npm should treat this report as an active operational concern. The attack surface described here is not theoretical: AWS documented over 500 misconfigured IAM roles across more than 275 accounts exposed to exactly this class of OIDC trust bypass as recently as 2024, and the June 2025 AWS guardrail preventing new vulnerable role creation leaves every existing misconfigured role fully exploitable [4][10].

Background

The s1ngularity Supply Chain Attack (GHSA-cxm3-wv7p-598c)

The s1ngularity attack was disclosed on August 26, 2025, when security researchers identified that multiple packages in the Nx build system ecosystem—including the core `nx` package and several `@nx/` scoped packages—had been replaced with malicious versions on the npm registry [1][5]. The root cause was a workflow injection vulnerability in the Nx project's own GitHub Actions configuration, documented as GHSA-cxm3-wv7p-598c.

The attack exploited a `pull_request_target` workflow trigger that ran with write-level repository permissions and echoed unsanitized pull request titles as shell commands. An attacker submitted a pull request whose title contained a bash injection payload. Because `pull_request_target` executes in the context of the base repository—not the forking repository—the injected command ran with the Nx repository's elevated permissions, including access to the project's npm publishing token. The attacker extracted the token, published malicious versions of affected packages, and achieved supply chain distribution to every developer who updated their Nx installation during the exposure window [1][6].

The malicious packages executed a postinstall script loading a JavaScript credential stealer designated QUIETVAULT in subsequent threat analysis [3][7]. The malware collected environment variables, cloud provider credentials, GitHub personal access tokens (PATs), npm tokens, SSH private keys, and AI tool configuration files. QUIETVAULT exhibited a capability that warranted specific attention from researchers: it detected locally installed AI command-line tools—including Claude, Google Gemini CLI, and Amazon Q Developer—and invoked them with permission-bypass flags (`--dangerously-skip-permissions`, `--yolo`, `--trust-all-tools --no-interactive`) to recursively scan the filesystem for additional sensitive material beyond what environment variable enumeration would surface [7]. Exfiltrated data was encoded and written to attacker-controlled public GitHub repositories for subsequent retrieval. The attack resulted in the theft of at least 2,349 distinct secrets from 1,079 compromised repositories, along with GitHub personal access tokens and cloud credentials across affected organizations before remediation [2][8][17].

GitHub-to-AWS OIDC Federation and Its Trust Model

GitHub Actions OIDC federation allows CI/CD workflows to authenticate to AWS without storing static long-lived credentials. When a workflow runs, GitHub's OIDC provider at `token.actions.githubusercontent.com` issues a signed JSON Web Token containing claims that identify the specific workflow's execution context: the repository path, branch reference,

deployment environment, triggering actor, and a subject (`sub`) claim that encodes this identity in a canonical form [9][18]. The workflow exchanges this short-lived token (valid for five minutes) for temporary AWS STS credentials by calling `sts:AssumeRoleWithWebIdentity`.

The security model depends critically on the IAM role's trust policy validating not only the token's issuer and audience but also its subject claim. A trust policy that validates only `aud: sts.amazonaws.com` without a `sub` condition will accept OIDC tokens from any GitHub Actions workflow on the entire GitHub platform, not merely workflows belonging to the intended organization or repository. This misconfiguration—scoping trust to the identity provider rather than to the specific tenant within it—is a confused deputy vulnerability: AWS becomes a relying party that cannot distinguish legitimate workflows from attacker-controlled ones because it has not been told to ask. Datadog Security Labs and Rezonate independently confirmed that over 500 such vulnerable role ARNs were publicly discoverable across more than 275 AWS accounts via Sourcegraph OSINT on public workflow files [4]. AWS implemented a backend guardrail in June 2025 blocking creation of new roles with this pattern, but the control is not retroactive; every previously created misconfigured role remains exploitable [10].

Security Analysis

The UNC6426 Attack Chain

The UNC6426 intrusion documented in Google's Cloud Threat Horizons Report H1 2026 unfolded across approximately 72 hours beginning around August 26, 2025, coincident with the s1ngularity attack window [3][16]. The victim organization was a software development firm whose engineers routinely used the Nx monorepo build system.

Initial Credential Acquisition. A developer at the victim organization used an IDE with the Nx Console extension installed. Updating the extension triggered installation of a malicious nx package version, executing the QUIETVAULT postinstall script. Among the credentials harvested was the developer's GitHub PAT—a token with access to the victim organization's private repositories and, crucially, the ability to trigger GitHub Actions workflow runs via the GitHub API. QUIETVAULT exfiltrated this token to one of its attacker-controlled public staging repositories, making it available to any downstream actor monitoring those collection points [3][7].

UNC6426 should not be understood as the same actor who executed the s1ngularity attack. The original supply chain compromise is attributed to a separate, as-yet-unnamed threat cluster. UNC6426 appears to have operated as a downstream consumer of the stolen credential corpus, either through direct monitoring of the exfiltration repositories or through a secondary market for stolen credentials. This distinction matters operationally: it means the original attacker's objectives and UNC6426's objectives were independent, and organizations may face multiple distinct adversaries pursuing different goals from the same initial access event.

Reconnaissance via NORDSTREAM. With the stolen PAT in hand, UNC6426 conducted reconnaissance of the victim's GitHub organization using the Nord Stream open-source CI/CD extraction tool (designated NORDSTREAM in the Mandiant report) [3][11]. Nord Stream, maintained by Synacktiv, enumerates repository workflow files, identifies cloud credential configuration patterns, and—given a PAT and a target IAM role ARN—triggers workflow executions that request GitHub OIDC tokens and call `sts:AssumeRoleWithWebIdentity` to obtain temporary AWS credentials. The tool automates cleanup of triggered workflow runs by default, limiting forensic visibility into its use.

UNC6426 identified a GitHub Actions deployment workflow in the victim's environment that used the `aws-actions/configure-aws-credentials` action configured for OIDC authentication. The associated IAM role, named `Github-Actions-CloudFormation`, had a trust policy that included the GitHub OIDC provider as a federated principal but lacked an adequately scoped `sub` condition. The attacker-controlled PAT was sufficient to trigger a workflow run that obtained a valid GitHub OIDC JWT. That JWT exchanged for temporary STS credentials for the `Github-Actions-CloudFormation` role within minutes.

Privilege Escalation via CloudFormation IAM Capabilities. The `Github-Actions-CloudFormation` role's permissions were scoped to CloudFormation stack deployment operations—not direct IAM management. However, the role held `cloudformation:CreateStack` and `cloudformation:DescribeStacks` permissions, along with the `CAPABILITY_NAMED_IAM` and `CAPABILITY_IAM` stack deployment capabilities. These capabilities authorize a CloudFormation stack to create and modify IAM resources as part of its deployment, and they are commonly granted to deployment roles that need to provision application infrastructure including service accounts and execution roles [12].

UNC6426 deployed a CloudFormation stack whose template contained a single IAM resource: a new IAM role with the `arn:aws:iam::aws:policy/AdministratorAccess` managed policy attached. Because CloudFormation acts as an intermediary with its own service role that holds the necessary IAM permissions, the constraint that `Github-Actions-CloudFormation` itself lacked

direct `iam:CreateRole` permissions did not prevent this operation. The stack deployed successfully, and UNC6426 assumed the newly created role to obtain full administrative access to the victim AWS account [3].

Post-Compromise Actions. With administrator access, UNC6426 enumerated and exfiltrated objects from S3 buckets, terminated production EC2 instances and RDS database instances, and decrypted application secrets. The group then used the stolen GitHub PAT to rename internal repositories to match the `s1ngularity-repository-[random]` naming pattern used by the original s1ngularity attacker and made them public—an action that both exfiltrated source code and created reputational damage. The mimicry of the original attacker's signature behavior likely served dual purposes: it seeded forensic confusion about attribution and ensured that secondary credential harvesters monitoring s1ngularity staging repositories would receive any additional tokens or secrets embedded in the exposed source [3].

Compounding Misconfigurations: Why Each Control Failure Mattered

The attack's success required the simultaneous presence of three distinct misconfigurations, none of which is unusual individually. Understanding each failure independently is essential for defenders prioritizing remediation across their own environments.

The OIDC trust policy failure was the pivot point from GitHub to AWS. Had the `Github-Actions-CloudFormation` role's trust policy included a `sub` condition scoped to the victim organization's specific repository and deployment environment—a targeted IAM policy change—the stolen PAT could not have produced AWS credentials regardless of how the attacker-controlled workflow was constructed. The PAT gave UNC6426 the ability to trigger workflows; the misconfigured trust policy gave those workflows the ability to assume an AWS identity they should not have been able to reach.

The CloudFormation IAM capability failure was the pivot from limited cloud access to full administrative control. The deployment role's CloudFormation permissions were, by themselves, not obviously dangerous—CloudFormation is a standard infrastructure-as-code tool, and `CAPABILITY_NAMED_IAM` is commonly required for application stacks that deploy Lambda execution roles or EC2 instance profiles. The failure was the absence of a service control policy (SCP) or permission boundary that restricted what IAM resources CloudFormation could create on behalf of that role. A permission boundary preventing the `Github-Actions-CloudFormation` role from creating any IAM principal with more permissions than itself would have contained the escalation [13].

The credential exfiltration mechanism—QUIETVAULT writing stolen tokens to public GitHub repositories—represents an exfiltration channel that many organizations' data loss prevention controls are not configured to detect. GitHub API traffic to external repositories is structurally similar to routine developer activity, which means signature-based data loss prevention controls are unlikely to distinguish malicious exfiltration from legitimate open-source contribution; behavioral analytics are required to detect this pattern reliably.

The AI Tool Weaponization Angle

QUIETVAULT's use of locally installed AI CLI tools merits specific treatment because it represents, to the authors' knowledge, the first publicly documented instance of AI CLI tool weaponization within a supply chain postinstall script. By invoking Claude Code, Gemini CLI, and Amazon Q Developer with permission-bypass flags from within the developer's own environment, the malware leveraged AI assistants as filesystem reconnaissance agents operating with the full filesystem permissions of the developer's user account [7]. This technique exploits the trust that AI-assisted development workflows inherently require: these tools are deliberately granted broad local access to repositories, configuration files, and credential stores because that access is necessary for their intended function.

The security implication is that any developer tool capable of invoking AI CLI tools with elevated or bypass flags becomes a force-multiplier for credential theft rather than a component to be evaluated in isolation. AI tool vendors may begin implementing input validation to prevent invocations originating from automated contexts; however, organizations should not assume such controls are currently in place. The technique demonstrates that the attack surface of an AI-augmented development environment extends across every tool in the workflow, not just the AI assistant itself.

Recommendations

Immediate Actions

Organizations using GitHub Actions with AWS OIDC federation should audit all IAM roles whose trust policies reference `token.actions.githubusercontent.com` within the next 48 hours. The Rezonate open-source `github-oidc-checker` tool (archived as of April 2025; available for reference use) and Datadog Cloud Security Management both provide automated detection of roles with absent or insufficiently scoped `sub` conditions [4][14]. Every role discovered without explicit subject scoping should be treated as actively exploitable. The remediation is a targeted IAM trust policy

change: add a `StringEquals` condition on `token.actions.githubusercontent.com:sub` scoped to the specific repository and deployment environment. For production deployment roles, the subject should be scoped to a named environment (`repo:org/repo:environment:production`) and GitHub environment protection rules should be configured to permit only protected branches to target that environment.

Any organization that used nx packages published to npm between August 26 and August 29, 2025—specifically versions 20.9.0, 20.10.0, 20.11.0, 20.12.0, 21.5.0, 21.6.0, 21.7.0, or 21.8.0 of the `nx` package, or any of the `@nx/devkit`, `@nx/js`, `@nx/workspace`, `@nx/node`, `@nx/eslint`, `@nx/key` (version 3.2.0), or `@nx/enterprise-cloud` (version 3.2.0) packages [1][5]—should rotate all credentials that may have been present in the development environment at time of installation. This includes GitHub PATs, npm tokens, SSH private keys, AWS access keys stored in environment variables or `~/.aws/credentials`, and any API keys or service account tokens in the developer's home directory. The QUIETVAULT postinstall script executes once on package installation, not on build or runtime, meaning rotation is required only if the malicious package was installed, not merely listed as a dependency.

CloudFormation deployment roles with `CAPABILITY_NAMED_IAM` or `CAPABILITY_IAM` should be evaluated for privilege escalation exposure. Any role that holds these capabilities and can deploy stacks without IAM permission boundaries is a potential escalation path of the type UNC6426 exploited. A permission boundary policy preventing the creation of IAM roles with permissions exceeding those of the deployment role itself is the minimal control required [13].

Short-Term Mitigations

GitHub Actions workflows that use the `pull_request_target` trigger should be reviewed and, where possible, migrated to `pull_request` or restructured to avoid executing any code from the forking repository in the elevated-permission context. Where `pull_request_target` is required for legitimate functionality—for example, workflows that need to post review comments or update statuses on PRs from forks—the workflow must not check out or execute the forking branch's code. This is the root cause class of the original s1ngularity pwn-request vulnerability [1].

AWS CloudTrail should be configured to alert on `AssumeRoleWithWebIdentity` calls where `userIdentity.identityProvider` matches `token.actions.githubusercontent.com` and the `userIdentity.userName` field—which contains the OIDC `sub` claim—does not match expected repository patterns for the organization. Unexpected repository paths in this field are a direct indicator of OIDC trust chain abuse. Similar

monitoring should be applied to any `cloudformation:CreateStack` call that includes `CAPABILITY_NAMED_IAM` or `CAPABILITY_IAM` capabilities originating from a service principal associated with a CI/CD role outside of normal deployment windows.

Developer workstations that install AI CLI tools (Claude Code, Gemini CLI, Amazon Q Developer, GitHub Copilot CLI) should have endpoint controls that monitor or restrict invocation of those tools with permission-bypass arguments. While these flags serve legitimate use cases in controlled environments, their presence in a postinstall script or other automated context outside an interactive developer session is anomalous and warrants investigation.

Strategic Considerations

The UNC6426 incident illustrates the convergence of three trends that individually receive significant attention in security discourse—open-source supply chain risk, CI/CD pipeline credential exposure, and cloud IAM misconfiguration—but are rarely analyzed as an integrated attack surface. Organizations should conduct threat modeling exercises that explicitly trace the path from developer workstation to cloud control plane, asking at each step what credentials are present, what trust relationships those credentials can activate, and what the most permissive action reachable from each trust relationship is. This exercise frequently reveals escalation paths like the CloudFormation IAM capability chain that are invisible when IAM policies are reviewed in isolation.

Adopting npm Trusted Publishers for any internal or external package publishing eliminates the class of static npm token theft that enabled the s1ngularity distribution phase. The npm Trusted Publishers feature—which the Nx project adopted as part of its post-incident remediation—replaces long-lived publishing tokens with short-lived OIDC-based credentials scoped to specific repository workflows, substantially raising the cost of publishing unauthorized package versions [1]. Organizations that maintain private npm registries or publish internal packages should evaluate this migration as a supply chain integrity control.

The downstream credential market dynamic demonstrated here—where a supply chain attack generates a credential corpus that multiple unaffiliated threat actors subsequently exploit—argues for treating mass supply chain compromise events as an immediate credential rotation trigger across every affected development environment, not merely as a vendor incident requiring a patch. The five-day window between the s1ngularity attack and UNC6426's destructive actions suggests that credential rotation within 24–48 hours of confirmed exposure would have been sufficient to prevent this specific intrusion.

CSA Resource Alignment

This incident maps directly to multiple Cloud Security Alliance frameworks and guidance publications. The MAESTRO threat modeling framework for agentic and AI-augmented systems addresses the manipulation of AI tool trust assumptions that QUIETVAULT exploited—specifically the risk that AI agents operating with broad local permissions can be directed to conduct reconnaissance beyond their intended scope [15]. The technique of weaponizing AI CLI tools with permission-bypass flags represents an AI agentic threat that MAESTRO Layer 4 (execution environment integrity) addresses through recommendations for sandboxing and input validation of AI tool invocations.

The Cloud Controls Matrix (CCM) v4.0 provides directly applicable controls across multiple domains. CCM IAM-01 through IAM-09 address identity lifecycle, privilege management, and the federated identity controls that the OIDC trust policy misconfiguration violated. CCM SEF-01 addresses supply chain security and the provenance verification requirements that npm package integrity monitoring would satisfy. CCM DSP-07 addresses data classification and protection in ways that would constrain the S3 exfiltration phase of the attack.

The CSA STAR program's continuous monitoring criteria provide an audit framework for the specific controls this incident exposed as absent: periodic review of IAM role trust policies (particularly those involving federated identity providers), validation of CloudFormation deployment role permission boundaries, and monitoring of CI/CD pipeline credential usage against behavioral baselines. Organizations seeking to demonstrate assurance over their cloud-native CI/CD pipeline security posture should incorporate GitHub Actions OIDC trust policy review as a standing STAR control.

CSA's Zero Trust guidance applies to the lateral movement phase: a Zero Trust architecture in which the `Github-Actions-CloudFormation` role's effective permissions were continuously validated against a policy prohibiting IAM resource creation above a defined privilege ceiling would have limited the CloudFormation escalation even after OIDC trust was abused, provided no other escalation paths were reachable from the deployment role. The incident reinforces the Zero Trust principle that no single authenticated principal—regardless of how that authentication was obtained—should hold unconstrained escalation paths to administrative access.

References

[1] Nx Project, "S1ngularity - What Happened, How We Responded, What We Learned," nx.dev, September 5, 2025. https://nx.dev/blog/s1ngularity-postmortem

[2] Wiz Research, "s1ngularity: supply chain attack leaks secrets on GitHub: everything you need to know," Wiz Blog, August 2025. https://www.wiz.io/blog/s1ngularity-supply-chain-attack

[3] Google Cloud / Mandiant, "Cloud Threat Horizons Report H1 2026," Google Cloud, March 2026. https://services.google.com/fh/files/misc/cloud_threat_horizons_report_h12026.pdf (Note: URL requires verification; article identity per report title and H1 2026 publication.)

[4] Datadog Security Labs (Christophe Tafani-Dereeper), "No keys attached: Exploring GitHub-to-AWS keyless authentication flaws," Datadog Security Labs, 2024. https://securitylabs.datadoghq.com/articles/exploring-github-to-aws-keyless-authentication-flaws/

[5] GHSA-cxm3-wv7p-598c, "nx pull_request_target workflow injection enabling malicious package publication," GitHub Security Advisory, August 2025. https://github.com/nrwl/nx/security/advisories/GHSA-cxm3-wv7p-598c

[6] StepSecurity, "s1ngularity: Popular Nx Build System Package Compromised with Data-Stealing Malware," StepSecurity Blog, August 2025. https://www.stepsecurity.io/blog/supply-chain-security-alert-popular-nx-build-system-package-compromised-with-data-stealing-malware

[7] Endor Labs, "Nx build platform compromised by supply chain attack – How attackers collude with AI code assistants," Endor Labs Research, August 2025. https://www.endorlabs.com/learn/nx-build-platform-compromised-by-supply-chain-attack---how-attackers-collude-with-ai-code-assistants

[8] GitGuardian (Guillaume Valadon and Anna Nabiullina), "The Nx 's1ngularity' Attack: Inside the Credential Leak," GitGuardian Blog, August 2025. https://blog.gitguardian.com/the-nx-s1ngularity-attack-inside-the-credential-leak/

[9] GitHub, "About security hardening with OpenID Connect," GitHub Docs, 2025. https://docs.github.com/en/actions/security-for-github-actions/security-hardening-your-deployments/about-security-hardening-with-openid-connect

[10] Amazon Web Services, "Require a GitHub Actions `sub` claim condition when using OIDC," AWS Security Blog, June 2025. https://aws.amazon.com/blogs/security/ (Note: specific article URL returns HTTP 404 at time of citation validation; cited as AWS Security Blog index pending URL verification.)

[11] Synacktiv, "Nord Stream: Extracting Secrets from CI/CD Environments," GitHub, 2024. https://github.com/synacktiv/nord-stream

[12] Amazon Web Services, "AWS CloudFormation: Acknowledging IAM Resources in AWS CloudFormation Templates," AWS Documentation. https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-iam-template.html

[13] Amazon Web Services, "IAM Permission Boundaries," AWS Identity and Access Management User Guide. https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_boundaries.html

[14] Rezonate, "github-oidc-checker: Scan Your AWS Account for Vulnerable GitHub OIDC Trust Policies," GitHub, 2024. (Archived April 14, 2025; read-only, no longer maintained.) https://github.com/Rezonate-io/github-oidc-checker/

[15] Cloud Security Alliance, "MAESTRO: A Multi-Agent Environment Threat and Risk Operational Framework," CSA AI Safety Initiative, 2025. https://cloudsecurityalliance.org/research/working-groups/ai-technology-and-risk

[16] The Hacker News, "UNC6426 Exploits nx npm Supply Chain Attack to Gain AWS Admin Access in 72 Hours," The Hacker News, March 11, 2026. https://thehackernews.com/2026/03/unc6426-exploits-nx-npm-supply-chain.html

[17] Wiz Research, "s1ngularity's Aftermath: AI, TTPs, and Impact in the Nx Supply Chain Attack," Wiz Blog, September 3, 2025. https://www.wiz.io/blog/s1ngularitys-aftermath

[18] National Institute of Standards and Technology, "Digital Identity Guidelines: Federation and Assertions (SP 800-63-4)," NIST, August 2025. https://pages.nist.gov/800-63-4/sp800-63c.html (Note: Supersedes SP 800-63C rev. 2022, effective August 1, 2025.)

