INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-03-30 13:32 UTC

Secrets Sprawl at Scale: 29 Million New Hardcoded Credentials in 2025 Signal a Structural Breakdown in Non-Human Identity Control

SECURITY ANALYSIS | HIGH | CVSS 7.5

SCC Item ID	SCC-STY-2026-0035
Type	Security Analysis
Severity	HIGH
CVSS Base Score	7.5
Affected Products	GitHub (public and internal repositories), GitLab (self-hosted), Docker registries, Slack, Jira, Confluence, OpenAI, Anthropic, Brave Search, Firecrawl, Supabase, LiteLLM, MCP (Model Context Protocol) servers
Published	2026-03-30T07:30:00
Discovery Source	Rss

Executive Summary

GitGuardian's 2026 State of Secrets Sprawl report documents 29 million new hardcoded credentials exposed on public GitHub in 2025, a 34% year-over-year increase and the largest single-year volume on record. Organizations detect secrets but do not remediate them, leaving credentials exploitable for years. The 64% of secrets confirmed valid in 2022 that remain exploitable today demonstrates this gap transforms momentary developer errors into durable attack paths. The 81% surge in AI service credential leaks, driven by ungoverned adoption of API keys for platforms including OpenAI, Anthropic, and Supabase, signals that every new technology integration cycle compounds an already critical non-human identity control problem.

Technical Analysis

The GitGuardian 2026 State of Secrets Sprawl report establishes a pattern that security teams should treat as a systemic failure mode rather than an accumulation of individual developer mistakes. The 29 million newly exposed secrets on public GitHub in 2025 represent the visible surface; the report's finding that internal repositories expose secrets at six times the rate of public ones reframes the threat model entirely. Organizations that rely on repository visibility controls as a primary defense against credential leakage are operating on a false

assumption.

The AI integration vector is the sharpest emerging signal. An 81% year-over-year increase in AI service credential leaks tracks directly with the adoption curve of LLM APIs. Developers integrating OpenAI, Anthropic, LiteLLM, Supabase, and related services frequently treat API keys as low-risk configuration artifacts, embedding them in code, dotfiles, CI/CD pipeline definitions, and Slack or Jira attachments without applying the same discipline used for cloud provider credentials. The result is a new credential class, API tokens with broad programmatic access to AI inference, data pipelines, and vector stores, entering the same ungoverned sprawl pipeline that has accumulated cloud keys and database connection strings for years.

Model Context Protocol (MCP) servers represent the next exposure surface. Separately reported vulnerabilities in Anthropic's MCP Git Server (The Hacker News, January 2026) documented three flaws enabling unauthorized file access. MCP servers, by design, bridge AI agents to external tools, repositories, and data sources. A hardcoded credential in an MCP server configuration does not represent a single-service exposure; it can represent a pivot point into any system the server is authorized to reach. This architectural reality elevates the severity of MCP-related credential sprawl beyond what the credential type alone would suggest.

The MITRE ATT&CK mapping for this pattern is well-established: T1552.001 (credentials in files), T1552.004 (private keys), T1552.007 (container API keys), T1078.001 and T1078.004 (default and cloud account abuse), T1213 and T1213.003 (data from repositories and code), T1530 (data from cloud storage), and T1195.001 (supply chain compromise via CI/CD). The progression from initial discovery through exploitation typically moves from repository scanning, widely performed by both researchers and threat actors, through valid credential use that bypasses detection controls oriented toward anomalous authentication behavior. When a stolen API key authenticates successfully, most SIEM rules see a clean login.

The 64% long-term validity finding is the operationally critical number. Detection pipelines that generate alerts without enforced remediation workflows do not reduce risk; they document it. The gap between alert volume and remediation rate reflects a structural problem in how organizations assign ownership for non-human identity hygiene. Unlike compromised user credentials, which carry clear incident response procedures, a hardcoded API key in a legacy repository often has no clear owner, no rotation SLA, and no revocation workflow tied to detection.

CWE coverage for this story spans the core failure modes: CWE-798 (hardcoded credentials), CWE-321 (hardcoded cryptographic key), CWE-522 (insufficiently protected credentials), and CWE-312 (cleartext storage of sensitive information). These are not novel weaknesses; they have appeared in NIST guidance and OWASP documentation for years. Their persistence at this scale indicates that awareness-based approaches have reached their effectiveness ceiling. Control enforcement, pre-commit scanning, mandatory secrets management platform adoption, and automated rotation is the required response.

Action Checklist

1. Step 1: Assess exposure, audit all repositories (public and internal) for hardcoded credentials using a secrets scanning tool; prioritize internal repositories given the 6x higher exposure rate documented in the report; include AI service API keys (OpenAI, Anthropic, Supabase, LiteLLM) as explicit scan targets
2. Step 2: Review controls, verify pre-commit hooks and CI/CD pipeline scanning are enforced, not advisory; confirm secrets management platforms (HashiCorp Vault, AWS Secrets Manager, Azure Key Vault, or equivalent) are in use for all service-to-service credentials; validate that detection alerts have documented, assigned remediation workflows with SLAs

3. Step 3: Update threat model, add hardcoded AI service API keys and MCP server configurations to your secrets sprawl threat register; incorporate T1552.001, T1552.007, T1213.003, and T1195.001 into detection coverage reviews; treat internal repository exposure as equivalent in risk priority to public exposure
4. Step 4: Communicate findings, brief leadership on the 64% long-term validity finding as a business risk: credentials discovered years ago and never rotated remain live attack paths; quantify the organization's own backlog of unresolved secrets alerts to establish a concrete remediation gap
5. Step 5: Monitor threats, track Anthropic MCP server vulnerability disclosures via CISA advisories and security news outlets; monitor emerging regulatory guidance on non-human identity controls as AI integration governance matures; establish baseline metrics for secrets detection and remediation time-to-action

IR / Forensic Enrichment

Triage Priority	URGENT
Escalation Criteria	Escalate immediately to CISO and legal counsel if scanning confirms any OpenAI, Anthropic, Supabase, or cloud provider (AWS, Azure, GCP) credentials with confirmed-valid status in a public repository, or if internal repository exposure includes credentials with access to systems storing PII, PHI, or PCI-scoped data, as this may trigger breach notification obligations under GDPR Article 33, HIPAA §164.412, or applicable state privacy law.
Recovery Notes	After rotating all identified credentials, verify revocation is complete by re-running Trufflehog with `--only-verified` against the same repository set — any credential that still resolves as valid indicates the rotation was incomplete or a second copy exists in a branch or fork not included in the initial scan. Monitor AI service provider usage dashboards (OpenAI usage API, Anthropic Console) for anomalous API call volume in the 30 days following rotation, as threat actors who harvested credentials before revocation may have already established downstream automation that will surface as usage spikes from unrecognized IP ranges immediately post-rotation. Maintain enhanced logging of all secrets management platform (Vault, AWS Secrets Manager) access events for a minimum of 90 days post-remediation to detect deferred exploitation attempts using credentials that were captured prior to rotation.

Forensic Artifacts	GitHub Secret Scanning alert export (JSON) via API `GET /orgs/{org}/secret-scanning/alerts` — captures all detected credential types, affected repositories, introduction commit SHAs, and alert age; directly maps the 64% long-term validity finding to the organization's specific backlog | Full git patch history (`git log --all -p`) for each affected repository — establishes the exact commit, committer identity, timestamp, and branch context in which each hardcoded AI service credential (OpenAI `sk-`, Anthropic `sk-ant-`, Supabase `eyJ` JWT) was introduced, preserving chain of custody before rotation invalidates verification | AI service provider audit logs — OpenAI Organization Usage logs (platform.openai.com/usage), Anthropic Console API usage logs, and Supabase project logs — filtered for API calls originating from IP ranges outside the organization's known egress; unauthorized usage prior to revocation constitutes evidence of active exploitation | CI/CD pipeline execution logs (GitHub Actions workflow run logs, GitLab CI job logs, Jenkins build logs) for the 90 days preceding discovery — searched for environment variable injection patterns and any step that echoes, prints, or exports values matching AI service credential prefixes, which indicates secrets were passed through the pipeline in plaintext | MCP server configuration files (`mcp.json`, `~/.config/mcp/`, project-level `.mcp/` directories) and their git history — MCP server configs frequently embed API keys for Anthropic, OpenAI, Firecrawl, and Brave Search as inline values; git history of these files reveals credential introduction events and any prior rotation attempts that were applied to config but not to the underlying secret store

Per-Action IR Details

Step 1: Assess exposure — audit all repositories (public and internal) for hardcoded credentials using a secrets scanning tool; prioritize internal repositories given the 6x higher exposure rate documented in the report; include AI service API keys (OpenAI, Anthropic, Supabase, LiteLLM) as explicit scan targets

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis: establishing scope and identifying affected assets before classification

Controls: NIST SI-7 (Software, Firmware, and Information Integrity) — integrity scanning of repository contents for unauthorized credential exposure, NIST AU-6 (Audit Record Review, Analysis, and Reporting) — systematic review of repository commit history and access logs for secret introduction events, CIS 2.1 (Establish and Maintain a Software Inventory) — extend inventory scope to include credential artifacts embedded in code artifacts, CIS 7.1 (Establish and Maintain a Vulnerability Management Process) — treat each identified hardcoded secret as a vulnerability entry requiring tracked remediation

Compensating: Run Trufflehog v3 (open source) against all repos: `trufflehog git file://./repo --only-verified` to surface only confirmed-live credentials. For GitHub public exposure, use `trufflehog github --org= --only-verified`. Pipe output to a CSV for triage: `trufflehog git file://./repo --json | jq -r '[.SourceMetadata.Data.Git.commit, .DetectorName, .Raw] | @csv'`. For AI service keys specifically, use Gitleaks with a custom rule targeting OpenAI `sk-` prefixes, Anthropic `sk-ant-` prefixes, and Supabase `eyJ` JWT patterns. A 2-person team can complete an org-wide public repo scan in a single shift using Trufflehog's GitHub org-level sweep.

Evidence: Before scanning, preserve point-in-time evidence: capture full git commit history via `git log --all --oneline > commit_history.txt` and `git log --all -p > full_patch_history.txt` for each repository to establish when each credential was first introduced. Export GitHub audit log events for the org (Admin > Audit Log > Export) covering the past 12 months, filtering for `repo.create`, `repo.access`, and `secret_scanning_alert` event types. For internal GitLab instances, export the Rails audit log at `/var/log/gitlab/gitlab-rails/audit_json.log` and filter for `push` events on affected repositories. This preserves chain of custody before any remediation invalidates the commit record.

Step 2: Review controls — verify pre-commit hooks and CI/CD pipeline scanning are enforced, not advisory; confirm secrets management platforms (HashiCorp Vault, AWS Secrets Manager, Azure Key Vault, or equivalent) are in use for all service-to-service credentials; validate that detection alerts have documented, assigned remediation workflows with SLAs

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: evaluating the adequacy of existing controls and detection infrastructure prior to or during an active secrets sprawl condition

Controls: NIST IR-4 (Incident Handling) — verify the incident handling capability includes a documented workflow for secrets remediation alerts with assigned ownership and SLAs, NIST IR-8 (Incident Response Plan) — confirm the IR plan addresses non-human identity compromise as a named incident category, not subsumed under generic credential abuse, NIST SI-2 (Flaw Remediation) — pipeline-enforced scanning is the enforcement mechanism for flaw remediation at the code artifact level; advisory-only hooks represent a gap, NIST CM-1 (Policy and Procedures for Configuration Management) — policy must explicitly prohibit hardcoded credentials in CI/CD pipeline definitions and IaC templates, CIS 4.6 (Securely Manage Enterprise Assets and Software) — CI/CD pipeline configuration must be version-controlled and audited for secrets injection points, CIS 6.1 (Establish an Access Granting Process) — service-to-service credentials provisioned through Vault/Secrets Manager must be tied to the access granting process, not ad hoc developer provisioning

Compensating: Enforce pre-commit scanning without enterprise tooling by installing the `pre-commit` framework with the `detect-secrets` hook: add `.pre-commit-config.yaml` with `repo: https://github.com/Yelp/detect-secrets` and set `fail_on_unaudited: true` so commits are blocked, not warned. For CI/CD enforcement in GitHub Actions without a paid tier, add a Trufflehog action step with `fail: true` to the pipeline YAML — this blocks merges if secrets are detected. Validate HashiCorp Vault adoption by running `vault audit list` and confirming at least one audit device (file or syslog) is enabled; absence of an audit device means Vault access is unlogged, which is a critical gap for this threat. Document remediation SLAs in a shared Markdown runbook committed to a private ops repo, with GitHub Issues used as the ticket system for tracking open secrets alerts.

Evidence: Before modifying any CI/CD configuration, export current pipeline definitions (`.github/workflows/*.yml`, `.gitlab-ci.yml`, `Jenkinsfile`) and store them as forensic baselines — these establish whether scanning was ever configured and whether it was set to advisory or blocking. Pull the GitHub Secret Scanning alert history via API: `GET /repos/{owner}/{repo}/secret-scanning/alerts?state=open` and export all open alerts with their `created_at` timestamps to measure alert age and identify secrets matching the 64% long-term validity pattern (alerts older than 90 days with no resolution). For HashiCorp Vault, export the audit log from the configured audit device (default path `/var/log/vault/audit.log`) and search for `operation=read` events on paths corresponding to AI service credential mounts to detect whether legitimate or unauthorized access occurred.

Step 3: Update threat model — add hardcoded AI service API keys and MCP server configurations to your secrets sprawl threat register; incorporate T1552.001, T1552.007, T1213.003, and T1195.001 into detection coverage reviews; treat internal repository exposure as equivalent in risk priority to public exposure

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: updating threat models and detection coverage based on documented incident patterns constitutes lessons-learned integration

Controls: NIST RA-3 (Risk Assessment) — the threat register update is a formal risk assessment activity; hardcoded AI service API keys represent a newly quantified risk category per the 81% YoY surge in AI credential leaks, NIST SI-4 (System Monitoring) — detection coverage review must explicitly map monitoring rules to T1552.001 (Credentials in Files), T1552.007 (Container API), T1213.003 (Code Repositories), and T1195.001 (Compromise Software Dependencies and Development Tools), NIST IR-8 (Incident Response Plan) — the IR plan must be updated to reflect MCP server configurations as a new credential exposure surface, with specific playbook entries for AI gateway credential revocation, CIS 7.1 (Establish and Maintain a Vulnerability Management Process) — the vulnerability management process must be extended to include non-human identity exposure as a tracked vulnerability class alongside traditional CVEs

Compensating: Map ATT&CK coverage gaps using the free ATT&CK Navigator (https://mitre-attack.github.io/attack-navigator/): import the four technique IDs (T1552.001, T1552.007, T1213.003, T1195.001) and score current detection coverage as 0 (none), 1 (partial), or 2 (full). For T1552.001 detection without a SIEM, deploy a Sigma rule converted to grep: search application and build logs for patterns matching `sk-`, `sk-ant-`, `eyJ` (Supabase JWT), and `AKIA` (AWS key prefix) using `grep -rE '(sk-[a-zA-Z0-9]{20,}|sk-ant-[a-zA-Z0-9]{20,}|AKIA[0-9A-Z]{16}|eyJhbGciOi)' /path/to/build/logs`. For MCP server configs, add a cron job scanning MCP configuration files (typically `~/.config/mcp/` or project-level `mcp.json`) for inline

credential patterns on a daily basis.

Evidence: Before updating the threat model, collect threat intelligence artifacts that document actual exploitation of these techniques against AI service credentials: export any existing SIEM or log alerts that fired on T1552.001 or T1213.003 patterns in the past 90 days to establish whether this threat class was already active in the environment. Pull GitHub Advanced Security code scanning results filtered to the `secret` category via API: `GET /repos/{owner}/{repo}/code-scanning/alerts?tool_name=secret-scanning`. For MCP server environments, enumerate running MCP server processes and their configuration file paths: `ps aux | grep mcp` and inspect each config file for inline API keys for OpenAI, Anthropic, Firecrawl, and Brave Search endpoints. Document all findings before threat model updates to establish a pre-update baseline.

Step 4: Communicate findings — brief leadership on the 64% long-term validity finding as a business risk: credentials discovered years ago and never rotated remain live attack paths; quantify the organization's own backlog of unresolved secrets alerts to establish a concrete remediation gap

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: communicating lessons learned and quantified risk to leadership is an explicit post-incident output that drives resource allocation and policy change

Controls: NIST IR-6 (Incident Reporting) — leadership briefing on a systemic secrets sprawl condition constitutes an incident report to organizational management, not merely a technical finding, NIST IR-5 (Incident Monitoring) — the backlog of unresolved secrets alerts is a monitoring artifact; presenting its age distribution (alerts open >30, >90, >365 days) directly maps to the 64% long-term validity finding, NIST RA-3 (Risk Assessment) — quantifying the remediation gap (number of open alerts × estimated blast radius per credential type) constitutes a risk assessment output suitable for executive briefing, CIS 7.2 (Establish and Maintain a Remediation Process) — the briefing must include a proposed SLA-based remediation process; presenting the gap without a proposed close plan does not satisfy this control

Compensating: Generate the backlog quantification without a SIEM using the GitHub Secret Scanning API: `gh api /orgs/{org}/secret-scanning/alerts --paginate --jq '[.[] | {number: .number, created_at: .created_at, secret_type: .secret_type, state: .state}]' > secrets_backlog.json`. Calculate age buckets with: `jq '[.[] | select(.state=="open") | {secret_type, age_days: (now - (.created_at | fromdateiso8601) | /86400 | floor)}] | group_by(.age_days > 365) | map({bucket: (.[0].age_days > 365 | if . then "over_365_days" else "under_365_days" end), count: length})' secrets_backlog.json`. Present AI credential types (OpenAI, Anthropic, Supabase) separately from generic secrets to highlight the 81% surge category. A 2-slide executive summary with the age distribution chart and a table of credential types by count is sufficient for a leadership briefing.

Evidence: Before the briefing, preserve a point-in-time snapshot of the unresolved alert backlog with full metadata (creation date, secret type, repository name, committer identity) as a dated export — this establishes the baseline against which future remediation progress is measured and satisfies NIST IR-5 (Incident Monitoring) documentation requirements. Cross-reference the backlog against the organization's OAuth app authorization list (GitHub: Settings > Developer Settings > OAuth Apps) to identify whether any compromised service accounts have active OAuth tokens that extend the blast radius beyond API keys. Retain this snapshot with a hash (SHA-256 of the JSON export) to ensure the evidentiary record is tamper-evident for potential regulatory or legal review.

Step 5: Monitor developments — track GitGuardian's 2026 State of Secrets Sprawl report for follow-up releases; monitor Anthropic MCP server vulnerability disclosures via The Hacker News and CISA advisories; watch for regulatory guidance on non-human identity controls as AI integration governance matures

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: maintaining situational awareness of emerging threats in the AI credential and MCP server space is a preparation-phase activity that feeds detection and response readiness

Controls: NIST SI-5 (Security Alerts, Advisories, and Directives) — formal process for receiving and acting on CISA advisories covering MCP server vulnerabilities and AI service credential disclosures, NIST IR-2 (Incident Response Training) — monitoring regulatory guidance on non-human identity controls informs training updates; as AI governance matures, IR teams must be trained on the new playbook entries, NIST RA-3 (Risk Assessment) — continuous monitoring of the secrets sprawl landscape (including GitGuardian annual reports and CISA NHI guidance) is a risk assessment input that triggers reassessment when the threat landscape materially changes, CIS 7.1 (Establish and

Maintain a Vulnerability Management Process) — the vulnerability management process must include a defined intake mechanism for threat intelligence from sources like CISA advisories and vendor security bulletins covering AI service providers

Compensating: Operationalize monitoring without a commercial threat intel platform: create a free CISA Known Exploited Vulnerabilities (KEV) RSS feed subscription filtered to relevant vendors (Anthropic, OpenAI ecosystem tools) using a free RSS reader or a cron job fetching `https://www.cisa.gov/sites/default/files/feeds/known_exploited_vulnerabilities.json` daily and diffing the output. Subscribe to GitGuardian's public blog RSS and the CISA Cybersecurity Advisories feed. For MCP-specific monitoring, set a GitHub repository watch on `modelcontextprotocol/servers` (the official MCP reference implementation) to receive email notifications on new security issues and releases. Route all alerts to a shared Slack or Teams channel designated for threat intel triage, reviewed by the team on a weekly cadence.

Evidence: Establish a monitoring baseline before beginning ongoing surveillance: export the current CISA KEV catalog (`curl https://www.cisa.gov/sites/default/files/feeds/known_exploited_vulnerabilities.json > kev_baseline_$(date +%Y%m%d).json`) and store it as the reference point for detecting new entries related to AI service providers or MCP server components. Document the current version of any MCP server software in use (check `package.json` or equivalent manifest in MCP server deployments) so that future vulnerability disclosures can be immediately assessed for applicability without requiring a new asset discovery sweep. Retain records of all CISA advisories reviewed and the disposition decision (applicable/not applicable with rationale) to satisfy NIST SI-5 (Security Alerts, Advisories, and Directives) documentation requirements.

Detection Guidance

Detection for secrets sprawl requires coverage across three distinct phases: exposure, discovery by adversaries, and exploitation.

For exposure detection: Deploy pre-commit scanning with enforced blocking (not advisory warnings) using tools such as Gitleaks, TruffleHog, or GitGuardian's native tooling. Instrument CI/CD pipelines to fail builds on detected secrets. Monitor GitHub Advanced Security or equivalent for new secret scanning alerts and measure time-to-remediation; an alert with no remediation within SLA is an open exposure.

For adversary discovery signals: Review GitHub audit logs for unusual repository enumeration or cloning activity from unfamiliar IPs or service accounts. Monitor for access to repository contents by accounts that have not recently committed. Watch for bulk API calls to secrets scanning endpoints, which may indicate competitor or adversary reconnaissance.

For exploitation detection: Prioritize authentication anomaly detection for non-human identities. Alert on API key usage from new geographic locations, new ASNs, or at unusual hours when the key has a consistent prior usage pattern. For AI service API keys (OpenAI, Anthropic, Supabase), monitor usage volume spikes and unexpected model or endpoint calls, which may indicate a stolen key being used for unauthorized inference or data extraction. For MCP server credentials specifically, audit authorization logs for file access patterns outside normal agent workflows.

For backlog remediation auditing: Query your secrets management platform and SIEM for credentials that were flagged but not rotated. Any secret with an open detection age exceeding your defined SLA should be treated as a confirmed exposure and escalated for immediate rotation, not remediation-queue processing.

MITRE ATT&CK techniques to hunt against: T1552.001 (search for credentials in source code, config files, environment variable logs), T1552.007 (API key abuse in container and cloud workload logs), T1213.003 (unusual data access from repository or code management systems), T1530 (unexpected cloud storage access using service account credentials).

Framework Mappings

MITRE-ATTACK

- **T1552.007** — Container API
- **T1078.001** — Default Accounts
- **T1078.004** — Cloud Accounts
- **T1552.004** — Private Keys
- **T1213** — Data from Information Repositories
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1530** — Data from Cloud Storage
- **T1213.003** — Code Repositories
- **T1552.001** — Credentials In Files

OWASP-TOP10-2021

- **A07:2021** — Identification and Authentication Failures
- **A04:2021** — Insecure Design

NIST-800-53R5

- **IA-5** — Authenticator Management
- **SC-13** — Cryptographic Protection
- **SI-4** — System Monitoring

CIS-V8

- **16.10** — Apply Secure Design Principles in Application Architectures
- **5.2** — Use Unique Passwords
- **6.3** — Require MFA for Externally-Exposed Applications
- **8.2** — Collect Audit Logs

ISO-27001-2022

- **A.8.28** — Secure coding
- **A.5.34** — Privacy and protection of personal information
- **A.8.24** — Use of cryptography
- **A.5.23** — Information security for use of cloud services

HIPAA-SECURITY

- **164.308(a)(5)(ii)(D)** — Password Management
- **164.312(d)** — Person or Entity Authentication

SOC2-TSC

- **CC6.1** — Logical access security software, infrastructure, and architectures

NIST-CSF-2

- **DE.CM-01** — Networks and network services are monitored

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1552.007	Container API	Credential-Access
T1078.001	Default Accounts	Defense-Evasion
T1078.004	Cloud Accounts	Defense-Evasion
T1552.004	Private Keys	Credential-Access
T1213	Data from Information Repositories	Collection
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1530	Data from Cloud Storage	Collection
T1213.003	Code Repositories	Collection
T1552.001	Credentials In Files	Credential-Access

Sources

Source	URL	Tier
Security News	https://thehackernews.com/2026/03/the-state-of-secrets-sprawl-2026-...	T3
modelcontextprotocol/servers: Model Context Protocol ... - GitHub	https://github.com/modelcontextprotocol/servers	T3
On Stream: Prevent GitHub Actions Attacks - YouTube	https://www.youtube.com/watch?v=zkT-Bg8rcJM	T3
Security overview · openanalytics/rtq-docker · GitHub	https://github.com/openanalytics/rtq-docker/security	T3
Three Flaws in Anthropic MCP Git Server Enable File Access and ...	https://thehackernews.com/2026/01/three-flaws-in-anthropic-mcp-git-...	T3

Organizations should conduct their own validation and consult qualified security professionals before taking action based on this report. Tech Jacks Solutions is not liable for any damages resulting from the use of this information.

Generated 2026-03-30 13:32 UTC by TJS Security Command Center