INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-19 06:43 UTC

CISA Contractor Exposed AWS GovCloud Credentials and DevSecOps Pipeline Details on Public GitHub

DATA BREACH | CRITICAL | CVSS 9.5

SCC Item ID	SCC-DBR-2026-0131
Type	Data Breach
Severity	CRITICAL
CVSS Base Score	9.5
Affected Products	AWS GovCloud (CISA-managed accounts), CISA internal DevSecOps infrastructure, GitHub (public repository)
Published	2026-05-18T16:48:21
Discovery Source	Rss

Executive Summary

A contractor working for CISA committed AWS GovCloud credentials and internal DevSecOps pipeline documentation to a public GitHub repository, according to Krebs on Security reporting dated May 18, 2026. The exposure gives adversaries privileged cloud access and a detailed map of the agency's software build and deployment infrastructure. Because CISA is the federal authority responsible for protecting U.S. critical infrastructure, the breach carries outsized national security and reputational risk beyond a typical credential leak.

Technical Analysis

A public GitHub repository maintained by a CISA contractor contained hardcoded or plaintext AWS GovCloud account credentials alongside documentation describing CISA's internal software build, test, and deployment pipeline. Applicable weaknesses: CWE-798 (Use of Hard-coded Credentials), CWE-522 (Insufficiently Protected Credentials), CWE-312 (Cleartext Storage of Sensitive Information), CWE-200 (Exposure of Sensitive Information). No CVE is assigned; this is an operational security failure, not a software vulnerability. MITRE ATT&CK techniques relevant to adversarial exploitation of this exposure include T1552.001 (Credentials in Files), T1552.004 (Private Keys), T1078.004 (Valid Accounts: Cloud Accounts), T1530 (Data from Cloud Storage), T1213 (Data from Information Repositories), T1195 and T1195.002 (Supply Chain Compromise). Automated credential-harvesting bots, including those operated by criminal and nation-state actors, routinely scan public repositories; dwell time between commit and harvest is frequently measured in minutes (industry

observation; confirmed by SANS and academic research on GitHub secret exposure rates). No patch is applicable; remediation requires immediate credential rotation, repository access revocation, and secrets management process overhaul. Primary source: Krebs on Security, May 18, 2026.

Action Checklist

1. Immediately revoke and rotate all AWS GovCloud credentials exposed in the public repository.
2. Suspend the contractor account(s) with access to those credentials and to the GitHub repository.
3. Set the repository to private or delete it pending forensic preservation.
4. Contact AWS GovCloud support to flag the compromised key IDs for abuse monitoring.
5. Pull AWS CloudTrail logs for all API calls made using the exposed key IDs, covering the full window from the first commit date to credential revocation.
6. Search CloudTrail logs for calls originating from IPs outside known CISA/contractor infrastructure, unusual S3 ListBucket or GetObject activity (T1530), IAM role assumption chains, and any new IAM user or policy creation (T1078.004).
7. Query GitHub audit logs for repository clone, fork, and download events.
8. Run a GitHub secret scanning alert review across all contractor-affiliated repositories.
9. Rotate all affected AWS GovCloud IAM credentials, access keys, and any private keys or certificates documented in the repository.
10. Audit all IAM policies attached to exposed accounts and remove any permissions not required.
11. Scrub the repository of sensitive files and enforce branch protection rules.
12. Implement a pre-commit secrets scanning hook (e.g., git-secrets, truffleHog, or GitHub Advanced Security secret scanning) across all repositories touching CISA infrastructure.
13. Validate that rotated credentials are functioning and that no legacy keys remain active in any CI/CD pipeline, build script, or configuration file.
14. Monitor CloudTrail and AWS Config for anomalous activity for a minimum of 30 days post-rotation.
15. Confirm that the DevSecOps pipeline documentation referenced in the repository has been reviewed for additional sensitive disclosures and that affected pipeline components have been audited for unauthorized modification (T1195.002).
16. Conduct a full secrets inventory across all contractor and internal repositories.
17. Mandate use of a secrets management solution (e.g., AWS Secrets Manager, HashiCorp Vault) for all credential storage; prohibit plaintext credentials in version control by policy and technical control.
18. Review contractor onboarding and offboarding procedures to ensure repository access is scoped and audited.
19. Brief the DevSecOps team on the MITRE ATT&CK techniques this exposure enables, particularly supply chain compromise vectors (T1195, T1195.002).

IR / Forensic Enrichment

Triage Priority	IMMEDIATE

Escalation Criteria	Escalate to CISA leadership, US-CERT, and OMB immediately if CloudTrail analysis confirms API calls from non-CISA/contractor IPs during the exposure window, if any IAM privilege escalation or new resource creation is detected, if pipeline artifacts show signs of tampering (T1195.002), or if the exposed DevSecOps documentation contains details enabling attacks on CISA-monitored critical infrastructure systems — each of these conditions triggers federal incident reporting obligations under FISMA and CISA's own binding operational directives.
Recovery Notes	Validate all rotated AWS GovCloud credentials end-to-end through the CI/CD pipeline in a staging environment before re-enabling production deployments, and confirm that AWS Config's `access-keys-rotated` rule shows zero active keys predating the rotation event. Monitor CloudTrail for the exposed key IDs and for the contractor IAM user ARN for a minimum of 30 days post-rotation — any post-revocation API call attempt using the old key IDs would indicate an adversary testing cached credentials and must trigger immediate escalation. Treat all build artifacts (container images, deployment packages, Lambda ZIPs) produced during the exposure window as untrusted until each has been rebuilt from a verified clean pipeline state and its integrity hash re-established in AWS Config or a trusted artifact registry.
Forensic Artifacts	AWS CloudTrail event history for the exposed IAM access key IDs — scoped to the GovCloud region(s), covering the window from the first GitHub commit date to revocation — specifically filtering on sts:AssumeRole, iam:CreateUser, iam:AttachUserPolicy, s3:ListBucket, s3:GetObject, and ec2:DescribeInstances calls, with source IP and user-agent fields preserved to identify non-CISA actor activity | GitHub repository audit log and traffic API data — specifically clone events (GET /repos/{owner}/{repo}/traffic/clones), fork events, and raw download events timestamped during the public exposure window — to establish the full population of actors who accessed the repository before it was set to private | IAM credential report (aws iam get-credential-report) capturing the access key creation dates, last-used dates, last-used services, and last-used regions for all keys associated with the contractor IAM user — establishing the baseline of what was active and when | Git repository commit history forensic mirror — the full `git clone --mirror` output preserving all commit SHAs, author identities, timestamps, and file diffs for the commits that introduced AWS GovCloud credentials and DevSecOps pipeline documentation — including any subsequent commits that attempted to remove them (as removed secrets remain in git history and were already exposed) | AWS Config configuration timeline for IAM users, IAM policies, S3 bucket policies, and any EC2 or Lambda resources that the exposed keys had write access to — establishing whether any unauthorized resource creation or configuration change occurred during the exposure window consistent with T1078.004 (Valid Accounts: Cloud Accounts) or T1195.002 (Supply Chain Compromise: Compromise Software Supply Chain)

Per-Action IR Details

Step 1: Containment — Immediately revoke and rotate all AWS GovCloud credentials exposed in the public repository. Suspend the contractor account(s) with access to those credentials and to the GitHub repository. Set the repository to private or delete it pending forensic preservation. Contact AWS GovCloud support to flag the compromised key IDs for abuse monitoring.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy

Controls: NIST IR-4 (Incident Handling), NIST AC-2 (Account Management), NIST AC-6 (Least Privilege), CIS 5.3 (Disable Dormant Accounts), CIS 6.2 (Establish an Access Revoking Process)

Compensating: Use the AWS CLI to immediately deactivate the exposed IAM access keys: `aws iam update-access-key --access-key-id --status Inactive --profile govcloud-admin`. For the contractor IAM user, run `aws iam delete-login-profile --user-name ` and `aws iam detach-user-policy` for each attached policy. Archive the GitHub

repository contents locally using `git clone --mirror` before setting it to private, preserving all commit history, branch refs, and metadata for forensic review. Contact AWS GovCloud support via the GovCloud-specific support portal to place an abuse flag on the compromised key IDs.

Evidence: Before revoking keys, capture the full IAM access key metadata via `aws iam list-access-keys --user-name ` to record key IDs, creation dates, and last-used timestamps. Preserve the GitHub repository's full commit history — specifically the commit SHAs, timestamps, author identity, and diff of the commit(s) that introduced credentials. Run `git log --all --full-history --stat` and `git show ` on the offending commits. Document the exact AWS GovCloud account IDs, IAM user ARNs, and attached policy ARNs before any modification.

Step 2: Detection — Pull AWS CloudTrail logs for all API calls made using the exposed key IDs, covering the full window from the first commit date to credential revocation. Look for calls originating from IPs outside known CISA/contractor infrastructure, unusual S3 ListBucket or GetObject activity (T1530), IAM role assumption chains, and any new IAM user or policy creation (T1078.004). Query GitHub audit logs for repository clone, fork, and download events. Run a GitHub secret scanning alert review across all contractor-affiliated repositories.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis

Controls: NIST IR-4 (Incident Handling), NIST IR-5 (Incident Monitoring), NIST AU-2 (Event Logging), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST SI-4 (System Monitoring), CIS 8.2 (Collect Audit Logs)

Compensating: Use AWS CLI to extract CloudTrail events scoped to the exposed key IDs without a SIEM: `aws cloudtrail lookup-events --lookup-attributes AttributeKey=AccessKeyId,AttributeValue= --start-time --end-time --region us-gov-west-1 --output json > cloudtrail_exposed_key.json`. Pipe through `jq` to filter for high-risk API calls: `jq '.Events[] | select(.EventName | test("CreateUser|AttachUserPolicy|AssumeRole|ListBuckets|GetObject|PutBucketPolicy"))' cloudtrail_exposed_key.json`. For GitHub audit logs, use the GitHub REST API: `curl -H "Authorization: token " "https://api.github.com/orgs//audit-log?phrase=action:repo.access&per_page=100"` filtered to the repository name and the exposure window.

Evidence: Capture the full CloudTrail event history for the exposed key IDs from the first GitHub commit timestamp to revocation — specifically recording source IP addresses, user-agent strings, AWS service endpoints called, and any `sts:AssumeRole` chains that would indicate the keys were used to pivot to other roles or accounts. Pull GitHub repository traffic data via `GET /repos/{owner}/{repo}/traffic/clones` and `GET /repos/{owner}/{repo}/traffic/views` APIs to establish a download/clone baseline and identify spikes correlating with the public exposure window. Capture any GitHub secret scanning alerts generated for the repository under Settings > Security > Secret scanning alerts before the repository is set to private.

Step 3: Eradication — Rotate all affected AWS GovCloud IAM credentials, access keys, and any private keys or certificates documented in the repository. Audit all IAM policies attached to exposed accounts and remove any permissions not required. Scrub the repository of sensitive files and enforce branch protection rules. Implement a pre-commit secrets scanning hook (e.g., git-secrets, truffleHog, or GitHub Advanced Security secret scanning) across all repositories touching CISA infrastructure.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST IR-4 (Incident Handling), NIST SI-2 (Flaw Remediation), NIST SI-7 (Software, Firmware, and Information Integrity), NIST AC-6 (Least Privilege), CIS 5.4 (Restrict Administrator Privileges to Dedicated Administrator Accounts), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: Install `truffleHog3` (free, open source) and run a full historical scan of all contractor-affiliated repositories: `trufflehog git https://github.com// --since-commit --json > secrets_scan_results.json`. For IAM policy audit without commercial tooling, run `aws iam list-attached-user-policies --user-name ` and `aws iam simulate-principal-policy` to enumerate effective permissions and identify over-privileged access. Use `git filter-repo` (not the deprecated `filter-branch`) to permanently rewrite repository history and remove committed credential files: `git filter-repo --path --invert-paths`. Install `git-secrets` hooks across all repos: `git secrets --install && git secrets

--register-aws`.

Evidence: Before scrubbing the repository, preserve a forensic clone (`git clone --mirror`) and generate a complete list of all sensitive files ever committed using `git log --all --full-history -- "*.pem" "*.key" "*.env" "*credentials*" "*config*"`. Document the full IAM policy JSON for each policy attached to the exposed accounts via `aws iam get-policy-version` before any policy modification — this establishes the blast radius of what an adversary with those keys could have done. Record all IAM roles that trust the exposed IAM user or access key via `aws iam list-roles` filtered on the trust policy principal.

Step 4: Recovery — Validate that rotated credentials are functioning and that no legacy keys remain active in any CI/CD pipeline, build script, or configuration file. Monitor CloudTrail and AWS Config for anomalous activity for a minimum of 30 days post-rotation. Confirm that the DevSecOps pipeline documentation referenced in the repository has been reviewed for additional sensitive disclosures and that affected pipeline components have been audited for unauthorized modification (T1195.002).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST SI-7 (Software, Firmware, and Information Integrity), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST CM-2 (Baseline Configuration), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: Enumerate all remaining active access keys across the GovCloud account to confirm no legacy keys survive: `aws iam generate-credential-report && aws iam get-credential-report --query Content --output text | base64 -d | grep -v ',false,,' | awk -F',' '{print $1,$9,$10,$14,$15}'` — this surfaces any key created before the rotation date that is still active. For CI/CD pipeline integrity verification without enterprise tooling, grep all pipeline definition files (`.github/workflows/*.yml`, `buildspec.yml`, `Jenkinsfile`, `.gitlab-ci.yml`) for hardcoded key patterns: `grep -rE "(AKIA|AGPA|AIPA|ANPA|ANVA|ASIA)[A-Z0-9]{16}" ./ --include="*.yml" --include="*.json" --include="*.sh"`. Set an AWS Config rule `access-keys-rotated` to alert on any key older than 1 day during the 30-day monitoring window.

Evidence: Capture the AWS Config configuration timeline for all IAM resources modified during the incident window via `aws configservice get-resource-config-history --resource-type AWS::IAM::User --resource-id ` to establish a verified pre- and post-incident configuration baseline. Pull the complete list of pipeline artifact hashes (checksums of build outputs, container images, deployment packages) produced during the exposure window from the CI/CD system — any artifact built while the exposed credentials were active must be treated as potentially tampered per T1195.002 supply chain compromise risk. Document the S3 bucket versioning history for any buckets the exposed keys had write access to, checking for unauthorized object uploads or deletions.

Step 5: Post-Incident — Conduct a full secrets inventory across all contractor and internal repositories. Mandate use of a secrets management solution (e.g., AWS Secrets Manager, HashiCorp Vault) for all credential storage; prohibit plaintext credentials in version control by policy and technical control. Review contractor onboarding and offboarding procedures to ensure repository access is scoped and audited. Brief the DevSecOps team on the MITRE ATT&CK techniques this exposure enables, particularly supply chain compromise vectors (T1195, T1195.002).

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity

Controls: NIST IR-4 (Incident Handling), NIST IR-8 (Incident Response Plan), NIST SI-2 (Flaw Remediation), NIST SI-5 (Security Alerts, Advisories, and Directives), NIST AU-11 (Audit Record Retention), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 6.1 (Establish an Access Granting Process), CIS 6.2 (Establish an Access Revoking Process)

Compensating: Run `truffleHog3` or `gitleaks` (free, open source) as a scheduled cron job across all contractor and internal GitHub organizations: `gitleaks detect --source . --report-format json --report-path gitleaks-report.json` — schedule weekly via cron and route output to a shared security inbox. For contractors lacking AWS Secrets Manager budget, implement `git-secrets` pre-commit hooks organization-wide via a one-time setup script pushed to all repositories. Enforce a GitHub organization-level policy (free tier) requiring branch protection and mandatory status checks that include secrets scanning before merge. Document contractor repository access grants and revocations in

a simple audit log (spreadsheet or ticketing system) with mandatory manager sign-off.

Evidence: Retain all CloudTrail logs, GitHub audit logs, IAM credential reports, and repository forensic clones for a minimum period consistent with NIST AU-11 (Audit Record Retention) and applicable federal records retention requirements — at minimum 1 year for GovCloud environments. Preserve the original public repository mirror, the diff of all commits containing sensitive material, and the GitHub traffic clone/view reports as evidentiary artifacts. Document the full timeline from first commit date to detection, notification, and containment for the lessons-learned report, including the dwell time window during which the credentials were publicly accessible and actionable by adversaries.

Detection Guidance

If your organization operates or monitors AWS GovCloud systems: Primary log source is AWS CloudTrail. Query for API activity on the exposed access key IDs across all GovCloud regions, filtering for: calls from unexpected source IPs or ASNs, AssumeRole events, CreateUser or AttachUserPolicy actions, S3 GetObject or ListBucket calls on sensitive buckets, and any activity outside business hours. Secondary: GitHub audit log for repository clone, fork, and download events tied to the exposed repository, including anonymous access if the repo was public. Behavioral indicator: any AWS API call originating from cloud provider IP ranges (e.g., AWS, GCP, Azure compute) against CISA GovCloud accounts that does not correspond to known automation may indicate automated bot harvesting (T1552.001). If a SIEM is ingesting CloudTrail, search for the specific exposed key ID strings as a literal match across all indexed log data to identify any prior undetected use.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`https://github.com/[contractor-repository — not publicly named in available reporting]`	Public GitHub repository containing exposed AWS GovCloud credentials and CISA DevSecOps pipeline documentation. Specific repository URL not confirmed in available sources.	LOW

Framework Mappings

MITRE-ATTACK

- **T1552.001** — Credentials In Files
- **T1195.002** — Compromise Software Supply Chain
- **T1552.004** — Private Keys
- **T1078.004** — Cloud Accounts
- **T1213** — Data from Information Repositories
- **T1530** — Data from Cloud Storage
- **T1195** — Supply Chain Compromise

NIST-800-53R5

- **CM-7** — Least Functionality
- **SA-9** — External System Services

- **SR-3** — Supply Chain Controls and Processes
- **SI-7** — Software, Firmware, and Information Integrity
- **SR-2** — Supply Chain Risk Management Plan
- **IA-5** — Authenticator Management
- **AC-3** — Access Enforcement
- **SC-28** — Protection of Information at Rest

OWASP-TOP10-2021

- **A07:2021** — Identification and Authentication Failures
- **A01:2021** — Broken Access Control
- **A04:2021** — Insecure Design

CIS-V8

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

ISO-27001-2022

- **A.8.28** — Secure coding
- **A.8.8** — Management of technical vulnerabilities
- **A.5.23** — Information security for use of cloud services

HIPAA-SECURITY

- **164.312(a)(1)** — Access Control
- **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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1552.001	Credentials In Files	Credential-Access
T1195.002	Compromise Software Supply Chain	Initial-Access
T1552.004	Private Keys	Credential-Access
T1078.004	Cloud Accounts	Defense-Evasion
T1213	Data from Information Repositories	Collection
T1530	Data from Cloud Storage	Collection

Technique ID	Technique Name	Tactic
T1195	Supply Chain Compromise	Initial-Access

Sources

Source	URL	Tier
Security News	https://krebsonsecurity.com/wp-content/uploads/2026/05/privatecisa.png	T3
CISA Admin Leaked AWS GovCloud Keys on Github	https://krebsonsecurity.com/2026/05/cisa-admin-leaked-aws-govcloud-...	T3
CISA Admin Leaked AWS GovCloud Keys on Github - Reddit	https://www.reddit.com/r/technology/comments/1th9qu6/cisa_admin_lea...	T3
CISA Admin Leaked AWS GovCloud Keys on Github	https://www.cryptika.com/cisa-admin-leaked-aws-govcloud-keys-on-git...	T3
CISA Admin Leaked AWS GovCloud Keys on Github - Facebook	https://www.facebook.com/groups/2600net/posts/4562192467337176/	T3

Generated 2026-05-19 06:43 UTC by TJS Security Command Center