INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-20 03:55 UTC

CVE-2026-3605: An authenticated user with access to a kvv2 path through a policy containing a glob may be able to d...

CVE VULNERABILITY | HIGH | CVSS 8.1

SCC Item ID	SCC-CVE-2026-0053
Type	CVE Vulnerability
CVE ID	CVE-2026-3605
Severity	HIGH
CVSS Base Score	8.1
EPSS Score	0.0001 (2th percentile)
Affected Products	HashiCorp Vault Community Edition < 2.0.0; Vault Enterprise < 2.0.0, < 1.21.5, < 1.20.10, < 1.19.16
Published	2026-04-17T04:16:03.263
Discovery Source	Nvd

Executive Summary

A flaw in HashiCorp Vault's policy engine allows authenticated users to delete secrets outside their authorized scope when glob patterns are used in access policies, causing potential denial-of-service through data destruction. All Vault Community and Enterprise versions before the patched releases are affected. The business risk centers on availability and data integrity of secrets management infrastructure, which stores critical application credentials, encryption keys, and service authentication.

Technical Analysis

CVE-2026-3605 is an authorization bypass vulnerability (CWE-288, CVSS 8.1) in HashiCorp Vault's KVv2 secrets engine. When a policy grants access to a KVv2 path using a glob pattern, the authorization logic fails to restrict delete operations to only the explicitly permitted paths. An authenticated user can exploit this to destroy secrets they are not authorized to read or write. The vulnerability is constrained to the authenticated user's namespace and accessible paths; it does not permit cross-namespace deletion or secret data exposure. Attack chain maps to MITRE T1548 (Abuse Elevation Control Mechanism) for the policy bypass and T1485 (Data Destruction) for the resulting secret deletion. EPSS score is 0.012% (1.68th percentile), indicating low current exploitation probability. Not listed in CISA KEV. Affected: Vault Community Edition < 2.0.0; Vault Enterprise < 2.0.0, < 1.21.5, < 1.20.10, < 1.19.16. Fixed in Community Edition 2.0.0 and Enterprise 2.0.0, 1.21.5, 1.20.10,

1.19.16. Source: NVD (https://nvd.nist.gov/vuln/detail/CVE-2026-3605).

Action Checklist

1. Step 1: Immediate Containment (if deployment confirmed at risk), Audit all Vault policies using glob patterns on KVv2 paths. Temporarily restrict or revoke delete capabilities on glob-matched policies until patched. Identify which authenticated users hold policies with glob-pattern KVv2 access and assess their recent delete activity.
2. Step 2: Detection, Review Vault audit logs for delete operations on KVv2 paths by users whose policies rely on glob patterns. Look for delete events on paths not explicitly listed in the user's policy. Vault audit log fields to query: operation=delete, mount_type=kv, and correlate requesting_entity against policy definitions. Alert on deletions that do not match explicit (non-glob) policy paths.
3. Step 3: Eradication, Upgrade Vault Community Edition to 2.0.0 or Vault Enterprise to 2.0.0, 1.21.5, 1.20.10, or 1.19.16 per your current release branch. Follow HashiCorp's upgrade documentation for your deployment type (binary, Helm, HCP). After patching, audit and rewrite overly broad glob policies to use explicit path grants where feasible.
4. Step 4: Recovery, After upgrade, validate that delete operations on KVv2 paths are correctly restricted by running a controlled test with a low-privilege account holding a glob policy. Confirm audit logs reflect expected authorization decisions. Restore any deleted secrets from backup or regenerate affected credentials and rotate downstream services that relied on them.
5. Step 5: Post-Incident Review, Review your glob policy usage broadly across all Vault mounts, not only KVv2. Implement least-privilege policy standards that prefer explicit path grants over glob patterns. Add automated policy review to your Vault change management process. Document which secrets were at risk and verify all dependent services have valid credentials.

IR / Forensic Enrichment

Triage Priority	URGENT
Escalation Criteria	Escalate immediately to CISO and initiate formal incident if Vault audit logs confirm delete operations on KVv2 paths outside a user's explicit policy scope (indicating active exploitation rather than misconfiguration risk), if any deleted secrets supported production authentication for regulated systems (PCI-DSS cardholder data environment, HIPAA-covered services), or if the Vault deployment lacks audit logging enabled — rendering exploitation scope unverifiable.
Recovery Notes	After patching and credential rotation, monitor Vault audit logs daily for 30 days for any recurrence of cross-scope delete operations, particularly from the same entity identifiers (`auth.display_name`, `auth.accessor`) observed in the pre-patch exploitation window. Validate that all downstream services that consumed secrets from affected KVv2 paths have successfully authenticated with rotated credentials — failed authentication events in application logs or service health checks are the primary indicator of incomplete recovery. Retain the pre-patch Raft snapshot and the full audit log from the incident window for a minimum of 90 days to support any deferred regulatory or legal review.

Forensic Artifacts	Vault JSONL audit log (configured `file` audit device, default `/var/log/vault/vault_audit.log`): contains `request.operation=delete`, `request.mount_type=kv`, `request.path`, `auth.display_name`, `auth.policies`, and `auth.accessor` fields that directly map delete actions to glob-policy holders and the specific KVv2 paths they deleted outside explicit policy scope. | Vault policy HCL files (captured via `vault policy read ` for each policy): forensic record of which glob patterns were in place, on which mount paths, with which capabilities — establishes the authorization misconfiguration scope and the set of users who could have exploited CVE-2026-3605. | Vault Raft storage snapshot (via `vault operator raft snapshot save`): taken pre-patch, this provides a recoverable state of the KVv2 secrets store and allows enumeration of which secret paths existed before any exploitation-window deletes — the authoritative source for determining data loss scope. | KVv2 metadata endpoint responses (via `vault kv metadata get ` for each known path): the `deletion_time` and `destroyed` fields in KVv2 metadata will show whether secrets were soft-deleted (recoverable) or permanently destroyed, and the timestamp of the delete operation, corroborating audit log entries. | Identity provider authentication logs (LDAP, OIDC, or AWS IAM auth method logs): for Vault deployments using external auth backends, these logs provide session context (source IP, authentication time, MFA status) for the entity identifiers found in Vault audit logs, establishing whether the deletes were performed by legitimate authenticated users exploiting the flaw versus a compromised credential.

Per-Action IR Details

Step 1: Containment — Audit all Vault policies using glob patterns on KVv2 paths. Temporarily restrict or revoke delete capabilities on glob-matched policies until patched. Identify which authenticated users hold policies with glob-pattern KVv2 access and assess their recent delete activity.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST AC-6 (Least Privilege), CIS 5.4 (Restrict Administrator Privileges to Dedicated Administrator Accounts), CIS 6.2 (Establish an Access Revoking Process)

Compensating: Export all Vault policies using the Vault CLI: `vault policy list` followed by `vault policy read ` for each result, then grep for glob patterns: `vault policy list | xargs -I{} vault policy read {} | grep -E '\*|\+'`. Pipe output to a file for review. For policy revocation without enterprise tooling, use `vault token revoke -accessor ` for each token associated with glob-policy holders, sourced from `vault token lookup` and audit log `auth.accessor` fields.

Evidence: Before modifying any policy, capture a full policy snapshot: `vault policy list > policy_inventory_$(date +%F).txt` and `vault policy read >> policy_snapshot_$(date +%F).txt`. Capture current token accessors and their associated policies via `vault auth list` and `vault token lookup -accessor `. Preserve Vault audit log files in their current state (default path: `/var/log/vault/vault_audit.log` or configured `file` audit device path) — these are your pre-containment baseline for delete activity attribution.

Step 2: Detection — Review Vault audit logs for delete operations on KVv2 paths by users whose policies rely on glob patterns. Look for delete events on paths not explicitly listed in the user's policy. Vault audit log fields to query: operation=delete, mount_type=kv, and correlate requesting_entity against policy definitions. Alert on deletions that do not match explicit (non-glob) policy paths.

NIST Phase: Detection Analysis

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

Controls: 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: Vault audit logs are JSONL format. Use `jq` to isolate relevant delete events: `cat vault_audit.log | jq 'select(.type=="request" and .request.operation=="delete" and .request.mount_type=="kv")' | jq '{time: .time, entity: .auth.display_name, accessor: .auth.accessor, path: .request.path, policy: .auth.policies}'`. Cross-reference the

`.request.path` values against each user's policy definition to identify paths resolved via glob that would not have been reachable under an explicit-path policy. For ongoing alerting without SIEM, set up a `cron` job running this `jq` filter every 15 minutes and diff against a known-good baseline of expected delete paths.

Evidence: Vault audit log entries for `operation=delete` and `mount_type=kv` are the primary artifact — specifically the fields `request.path`, `auth.display_name`, `auth.policies`, `auth.accessor`, and `time`. Capture the full audit log segment covering the window between when the vulnerable Vault version was deployed and when containment was applied. Also collect `response.auth.token_policies` from the same log entries to verify which glob policy was active at the time of each delete. If Vault is integrated with an identity provider (LDAP, OIDC), pull authentication events from that provider for the same entities and timeframe to establish session context.

Step 3: Eradication — Upgrade Vault Community Edition to 2.0.0 or Vault Enterprise to 2.0.0, 1.21.5, 1.20.10, or 1.19.16 per your current release branch. Follow HashiCorp's upgrade documentation for your deployment type (binary, Helm, HCP). After patching, audit and rewrite overly broad glob policies to use explicit path grants where feasible.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST CM-3 (Configuration Change Control), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: Before upgrading, verify the current Vault binary version with `vault version` and confirm your release branch to select the correct target: CE to 2.0.0, or Enterprise to 2.0.0 / 1.21.5 / 1.20.10 / 1.19.16. For binary deployments, download the signed HashiCorp release from releases.hashicorp.com, verify the SHA256 checksum against HashiCorp's published checksums file, and replace the binary during a maintenance window with a Vault seal/unseal cycle. For Helm deployments, update the `vault` chart image tag to the patched version in your `values.yaml` and run `helm upgrade`. After patching, rewrite glob policies using the principle of explicit enumeration: replace `secret/data/app/*` with `secret/data/app/credential1`, `secret/data/app/credential2`, etc., using `vault policy write `.

Evidence: Before initiating the upgrade, capture a snapshot of the Vault storage backend if self-hosted (Raft snapshot via `vault operator raft snapshot save pre_patch_snapshot_$(date +%F).snap`). Document the pre-patch binary hash (`sha256sum $(which vault)`) and the output of `vault status` to confirm HA mode and active node. Preserve all existing policy HCL files before rewriting them — these are forensic evidence of the misconfiguration scope and will be needed for post-incident review and potential regulatory documentation of the vulnerability window.

Step 4: Recovery — After upgrade, validate that delete operations on KVv2 paths are correctly restricted by running a controlled test with a low-privilege account holding a glob policy. Confirm audit logs reflect expected authorization decisions. Restore any deleted secrets from backup or regenerate affected credentials and rotate downstream services that relied on them.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST CP-10 (System Recovery and Reconstitution), NIST SI-7 (Software, Firmware, and Information Integrity), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Create a dedicated test token with a glob-pattern KVv2 policy scoped to a non-production mount: `vault token create -policy=test-glob-policy -ttl=1h`. Attempt a delete on a path that the glob matches but is outside the explicitly intended scope; the patched version should return a 403. Confirm the Vault audit log entry shows `response.auth.token_policies` and a denied `operation=delete`. For secrets restoration, if a Raft snapshot was taken pre-incident, restore to a test cluster first to enumerate deleted paths: compare `vault kv list` output against the snapshot inventory. For credentials that cannot be restored (e.g., API keys, database passwords), trigger rotation using each service's native rotation mechanism and update Vault's KVv2 store with new values via `vault kv put`.

Evidence: Post-patch, capture `vault status` output and the patched binary hash to confirm the correct version is running. Run `vault kv list -recursive /` and diff against your pre-incident inventory to enumerate any KVv2 paths that were deleted during the exploitation window. For each deleted path, identify all services that referenced that secret by querying Vault audit logs for `operation=read` on those paths in the 30 days preceding deletion — these are the

downstream services requiring credential rotation. Retain the post-patch audit log entries from the validation test as evidence of correct authorization enforcement.

Step 5: Post-Incident — Review your glob policy usage broadly across all Vault mounts, not only KVv2. Implement least-privilege policy standards that prefer explicit path grants over glob patterns. Add automated policy review to your Vault change management process. Document which secrets were at risk and verify all dependent services have valid credentials.

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 AU-6 (Audit Record Review, Analysis, and Reporting), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 5.1 (Establish and Maintain an Inventory of Accounts)

Compensating: Script a full policy audit across all mounts using: `vault policy list | xargs -I{} sh -c 'echo "=== {} ==="; vault policy read {}'` and pipe through `grep -n '\*'` to flag any remaining glob usage across all mount types (PKI, transit, database, AWS, etc.), not only KVv2. Implement a pre-commit hook or CI check in your Vault policy repository that rejects HCL files containing unqualified glob patterns on delete-capable paths. Use osquery with Vault's audit log as a file target to create a scheduled query that flags new `delete` operations on KVv2 mounts for continuous monitoring post-incident.

Evidence: Compile the complete post-incident artifact package: (1) the pre-patch policy snapshot showing all glob-pattern policies in scope, (2) the Vault audit log segment covering the full exploitation window with all delete operations highlighted, (3) the diff of KVv2 path inventory pre- and post-incident showing confirmed deleted secrets, (4) the list of downstream services mapped to deleted secrets via read-access audit log correlation, and (5) the post-patch validation test log entries confirming correct authorization enforcement. This package supports lessons-learned review, potential regulatory notification assessment, and serves as the baseline for the updated Vault policy governance standard.

Detection Guidance

Enable and review Vault's audit device logs (file or syslog). Filter for events where: (1) operation=delete, (2) mount_type=kv (KVv2), and (3) the requesting principal holds a policy containing a glob ('*' or '+') on the affected path prefix. A deletion event on a path that does not appear as an explicit grant in the user's policy is the primary behavioral indicator. There are no public IOCs (IPs, hashes, domains) associated with this vulnerability; exploitation is authenticated and internal. Monitor for unexpected secret absence or application authentication failures that may indicate secrets were destroyed.

Framework Mappings

MITRE-ATTACK

- **T1548** — Abuse Elevation Control Mechanism
- **T1485** — Data Destruction

NIST-800-53R5

- **AC-6** — Least Privilege
- **CM-6** — Configuration Settings

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1548	Abuse Elevation Control Mechanism	Privilege-Escalation
T1485	Data Destruction	Impact

Sources

Source	URL	Tier
nvd	https://nvd.nist.gov/vuln/detail/CVE-2026-3605	T1
CVE-2026-3605 - Red Hat Customer Portal	https://access.redhat.com/security/cve/cve-2026-3605	T3
CVE-2026-3605 Security Vulnerability Analysis & Exploit Details	https://cve.akaoma.com/cve-2026-3605	T3
CVE-2026-3605: denial-of-service vulnerability in vault (Go)	https://www.resolvedsecurity.com/vulnerability-catalog/CVE-2026-3605	T3
CVE-2026-3605 - High Vulnerability - TheHackerWire	https://www.thehackerwire.com/vulnerability/CVE-2026-3605/	T3

Generated 2026-04-20 03:55 UTC by TJS Security Command Center