PREPARED FOR

Acme Inc.

Acme Platform · Application & Cloud Penetration Test

ENGAGEMENT	March 2026 Pentest
REPORT DATE	31 March 2026
TESTING WINDOW	03 — 28 March 2026
METHODOLOGY	White-Box · Trace exploit-gated · CVSS v3.1
TESTED BY	Brandon Helms, OSCP, CISSP · Trace Offensive Security
APPROVED BY	Tanvir Singh
REPORT VERSION	v1.0 — Final
REPORT ID	TR-PT-2026-03-ACME
CLASSIFICATION	CONFIDENTIAL

DOCUMENT CONFIDENTIALITY

The information in this document is confidential and privileged, intended solely for the recipient. It may not be used, published, or redistributed without the prior written consent of both Acme Inc. and Trace.

AUTHORIZATION

This assessment was conducted with the authorization of Acme Inc. Testing was performed between 03 March 2026 and 28 March 2026 against the systems enumerated in Appendix A (Assessment Scope) and within the agreed rules of engagement.

Contents

01 Executive Summary 3
Scope 3
Findings Overview 3
02 Key Themes & Recommendations 4
03 Threat Ranking Methodology 5
04 Findings Summary 6
A Appendix A — Assessment Scope 7
B Appendix B — Trace Methodology 9
C Appendix C — Findings Detail 10

SECTION 01

Executive Summary

Trace conducted an application and cloud penetration test of the Acme Platform for Acme, combining authenticated dynamic testing of app.acme.com with white-box source review of the backend-api, web-app, and infra-config repositories, plus a configuration review of the production AWS organization.

The highest-leverage issues clustered around trust extended to CI/CD identities and outbound network paths rather than the application's authentication core — most prominently an AWS IAM trust policy and a request-forgery in the webhook delivery service.

Scope

The engagement covered the multi-tenant developer-collaboration platform — dashboard, REST API, supporting Kubernetes workloads, and Terraform-managed AWS accounts. Detailed scope is listed in Appendix A.

Findings Overview

CRITICAL	0
HIGH	2
MEDIUM	2
LOW	3
INFORMATIONAL	3

TOTAL FINDINGS

10

- 8 remediated & verified
- 1 accepted with compensating controls
- 1 open

SECTION 02

Key Themes & Recommendations

CI/CD Trust Boundaries

Trust granted to CI/CD identities (GitHub OIDC, Terraform Cloud) was the highest-leverage class of issue observed. Roles trusting external OIDC providers should default to aud and sub claim restrictions; this is best codified as a Terraform module guardrail rather than reviewed per role.

→ RECOMMENDATION

Codify aud / sub trust restrictions as a reusable IAM Terraform module so every OIDC-trusting role inherits the claim conditions by default rather than relying on per-role review.

Server-Side Outbound HTTP

The outbound webhook delivery path in backend-api performed destination validation before DNS resolution, allowing TOCTOU and rebinding bypasses against internal metadata services. Centralizing all outbound HTTP behind an egress proxy that resolves and re-validates against an allow-list eliminates this class of bug.

→ RECOMMENDATION

Centralize outbound HTTP behind an egress proxy with allow-listed destinations that re-resolves and re-validates the target after DNS, closing TOCTOU and DNS-rebinding bypasses.

Tenant Isolation

A single endpoint missing the organizationId scope check exposed cross-tenant telemetry. A Fastify pre-handler that enforces tenancy on every authenticated route closes the class.

→ RECOMMENDATION

Enforce organizationId scoping via a global Fastify pre-handler so every authenticated route is tenant-checked by default, rather than relying on each handler to add the filter.

SECTION 03

Threat Ranking Methodology

Trace scores every confirmed finding using the Common Vulnerability Scoring System (CVSS v3.1). A finding's severity tier usually follows its score band in the table below, and each tier carries the remediation urgency described there. When Trace places a finding in a different tier than its score suggests, for example because a precondition makes it harder or easier to exploit in practice, the finding's Severity Considerations explain why.

SEVERITY	CVSS V3.1 SCORE RANGE	DESCRIPTION
CRITICAL	9.0 – 10.0	Easily exploited; severe impact on confidentiality, integrity, or availability with little or no precondition. Requires immediate remediation.
HIGH	7.0 – 8.9	Significant impact; exploitable by an authenticated or remote actor with realistic effort. Should be addressed promptly.
MEDIUM	4.0 – 6.9	Moderate impact or non-trivial preconditions. Typically resolved on the next planned engineering cycle.
LOW	0.1 – 3.9	Limited impact or significant preconditions. Often resolved alongside related work; risk acceptance reasonable with documented compensating controls.
INFO	0.0	Observation, hardening recommendation, or defense-in-depth opportunity. No immediate exploitability.

CVSS vectors are reproduced verbatim in Appendix C.

SECTION 04

Findings Summary

Every confirmed finding from this engagement, ordered by severity then finding ID. Detailed reproduction steps, affected assets, and remediation guidance for each finding are documented in Appendix C.

ID	SEVERITY	FINDING	CWE	CVSS	STATUS
PTF-1	HIGH	Privileged AWS IAM role trusted by GitHub OIDC without aud / sub restriction	CWE-284	8.2	Remediated & Verified
PTF-2	HIGH	SSRF via webhook URL validation bypass in outbound webhook service	CWE-918	7.6	Remediated & Verified
PTF-3	MEDIUM	JWT refresh tokens issued without rotation allow replay after logout	CWE-613	5.8	Remediated & Verified
PTF-4	MEDIUM	Tenant isolation gap: project activity endpoint missing organizationId scoping	CWE-639	5.4	Remediated & Verified
PTF-5	LOW	Container backend-api runs as root with allowPrivilegeEscalation: true	CWE-250	3.7	Remediated & Verified
PTF-6	LOW	Long-lived API keys without default expiration or last-used tracking	CWE-732	3.5	Remediated & Verified
PTF-7	LOW	Stored XSS via unsanitized project name in Slack webhook payload	CWE-79	3.4	Remediated & Verified
PTF-8	INFO	RDS instance acme-staging-rds deployed in public subnet	CWE-668	0.0	Accepted with Controls
PTF-9	INFO	Missing pre-commit hooks for secrets and IaC scanning	CWE-1059	0.0	Remediated & Verified
PTF-10	INFO	Weak Content Security Policy on app.acme.com permits unsafe-inline styles	CWE-693	0.0	In Progress

APPENDIX A

Assessment Scope

TYPE	ASSET	DESCRIPTION	STATUS
Hostname	app.acme.com	Production dashboard (multi-tenant)	**In Scope**
Hostname	api.acme.com	Public REST API	**In Scope**
Pattern	*.collab.acme.com	Tenant-scoped collaboration and webhook endpoints	**In Scope**
Repository	acme/backend-api	Whitebox source review	**In Scope**
Repository	acme/web-app	Whitebox source review	**In Scope**
Repository	acme/infra-config	IaC and Kubernetes manifests	**In Scope**
AWS Account	481234-prod	Production account configuration review	**In Scope**
AWS Account	481234-staging	Staging account configuration review	**In Scope**
Pattern	*.dev.acme.com	Pre-production sandboxes — high-level configuration review only	**Out of Scope**
Other	Self-hosted runners & on-prem agents	Customer-deployed integration agents	**Out of Scope**
Other	Third-party providers (Datadog, Segment, Stripe)	Relied upon as trusted dependencies; not assessed	**Out of Scope**

Engagement Approach

Trace performed a hybrid engagement combining white-box source review with authenticated and unauthenticated dynamic testing. Trace paired manual analysis with its exploit-gated tooling, so every finding in this report rests on evidence gathered from the running system in scope. Where a finding's claim depends on reading the source code rather than on an exploit observed during testing, the finding says so. Validated findings are scored using the Common Vulnerability Scoring System (CVSS v3.1) and tagged against OWASP Top 10 and CWE for downstream ticketing and vulnerability-management workflows.

Limitations of Testing

Penetration testing is a point-in-time exercise. The findings in this report reflect the state of the in-scope systems during the testing window above. Code or infrastructure changes after that window may alter the risk surface. Trace did not perform load testing, physical security assessment, or social-engineering exercises unless explicitly noted in the scope above.

APPENDIX B

Trace Methodology

Every Trace engagement follows the same five-phase workflow, from reconnaissance through retest verification.

Trace's engagement methodology proceeds through five phases:

PHASE	ACTIVITIES
1. Reconnaissance	Asset enumeration, attack-surface mapping, authentication flow inventory.
2. Threat Modeling	Building a threat model of the target from the customer's intake, the product's documentation and, where supplied, the source code, to map the assets worth protecting, the trust boundaries between users and organizations, and the abuse cases to test first.
3. Exploit-Gated Discovery	Hypothesis-driven candidate generation; exploitation validated against the running system; reproducible evidence capture.
4. Triage & Scoring	CVSS v3.1 scoring, OWASP / CWE tagging, customer-impact narrative drafting.
5. Reporting & Retest	Findings delivery via the Trace dashboard and as a written report; retest verification of remediations.

Findings in this report are ordered by severity. Finding identifiers (PTF-N) are stable across the engagement; once assigned, an identifier is never reused. Where an identifier does not appear in this report, the corresponding finding was investigated and later dismissed, for example as not exploitable, out of scope, or withdrawn after review with the customer, and is therefore not among the published findings.

APPENDIX C

Findings Detail

HIGH PTF-1 Remediated & Verified

Privileged AWS IAM role trusted by GitHub OIDC without aud / sub restriction

CVSS	8.2
VECTOR	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:H/A:N
CATEGORY	Cloud IAM / Trust Policy
CWE	CWE-284
OWASP	A02:2025 Security Misconfiguration
REPORTED	31 March 2026

Summary

Acme's production cloud account had a door that opened for anyone who could run a build on GitHub, not just Acme's own builds. The rule that decides who gets in checked the wrong detail, so a stranger's workflow could walk in with full administrator access to everything in production. This has since been fixed and re-tested.

Description

The deploy role `arn:aws:iam::481234:role/acme-ci-deploy` trusted the GitHub Actions OIDC provider (`token.actions.githubusercontent.com`) but its trust policy constrained only the `aud` claim to `sts.amazonaws.com` — the default audience every GitHub workflow presents. The `sub` claim, which encodes the calling repository, ref, and environment, was left unconstrained.

Any GitHub Actions workflow in any repository — including a public fork running an attacker-authored workflow — could therefore request an OIDC token with the default audience and assume the role. The role held `AdministratorAccess`-equivalent permissions across the production account.

↑ PTF-1 CONTINUED FROM PREVIOUS PAGE

Root Cause Analysis

The trust policy was scaffolded from the AWS console's OIDC wizard, which writes the `aud` condition but leaves `sub` as a commented placeholder. The role was created when a single repository deployed everything, and the `sub` restriction was never added as the org grew to dozens of repositories. No Terraform guardrail required a `sub` condition on roles trusting the GitHub provider, so the gap was invisible to review.

Impact

An attacker able to run a GitHub Actions workflow under any repository could mint a token accepted by the trust policy and assume an administrator-equivalent role in the production account — reading every S3 bucket, RDS snapshot, and Secrets Manager value, and modifying infrastructure at will. Exploitation requires no Acme credentials.

Attack Scenario

1. Attacker opens a pull request that adds a workflow with `permissions: id-token: write`.
2. The workflow requests an OIDC token with the default audience `sts.amazonaws.com`.
3. The workflow calls `sts:AssumeRoleWithWebIdentity` against the Acme deploy-role ARN.
4. The trust policy checks only `aud` — which matches — and issues credentials.
5. The attacker now holds administrator-equivalent credentials in the production account.

Severity Considerations

Scored High rather than Critical: exploitation requires the attacker to know the role ARN (recoverable from public Terraform or a leaked workflow log) and to land a workflow run, and GitHub's `pull_request` token is read-only for `id-token` unless the workflow opts in. The blast radius — the full production account — argues for the upper end of High.

↑ PTF-1 CONTINUED FROM PREVIOUS PAGE

Proof of Concept

```
# .github/workflows/poc.yml in any repository
jobs:
  assume:
    runs-on: ubuntu-latest
    permissions:
      id-token: write
    steps:
      - uses: aws-actions/configure-aws-credentials@v4
        with:
          role-to-assume: arn:aws:iam::481234:role/acme-ci-deploy
          aws-region: us-east-1
      - run: aws sts get-caller-identity && aws s3 ls
```

The `configure-aws-credentials` step succeeds and `aws s3 ls` enumerates every production bucket.

Remediation

The trust policy now pins `token.actions.githubusercontent.com:sub` to `repo:acme/infra-config:ref:refs/heads/main` alongside the existing `aud` condition. A reusable Terraform module (`modules/github-oidc-role`) was introduced so every OIDC-trusting role inherits both conditions, and a `tflint` rule fails CI if a role trusting the GitHub provider omits a `sub` condition.

Verification

Trace re-ran the proof-of-concept workflow from a repository other than `acme/infra-config`; `sts:AssumeRoleWithWebIdentity` was rejected with `AccessDenied` (sub-claim mismatch). A workflow on `acme/infra-config@main` continued to assume the role as expected.

HIGH PTF-2 Remediated & Verified

SSRF via webhook URL validation bypass in outbound webhook service

CVSS	7.6
VECTOR	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:L/A:N
CATEGORY	Server-Side Request Forgery
CWE	CWE-918
OWASP	A01:2025 Broken Access Control
REPORTED	31 March 2026

Summary

When a customer set up a webhook, the platform would fetch whatever address they gave it, including addresses that only exist inside Acme's own network. That let a logged-in user reach internal systems the public should never see, and on a few older servers pick up the credentials those servers use. Fixed and re-tested.

Description

The outbound webhook service (`backend-api/src/webhooks/deliver.ts`) validated user-supplied destination URLs against a deny-list of private CIDR ranges, then performed the HTTP request in a separate step. Validation resolved the hostname to an IP and checked it; the subsequent delivery re-resolved the hostname independently. A DNS name that resolved to a public IP during validation and a private IP during delivery (DNS rebinding), or a `302` redirect to an internal address, bypassed the check entirely.

Root Cause Analysis

Destination validation and the outbound request performed two independent DNS resolutions — a time-of-check/time-of-use gap — and redirects were followed without re-validating the redirect target. The deny-list approach also omitted the link-local metadata range, which mattered on the subset of older hosts that still answered IMDSv1.

↑ PTF-2 CONTINUED FROM PREVIOUS PAGE

Impact

An authenticated user configuring a webhook could coerce the platform into issuing requests to internal services. On the subset of older hosts that still answered IMDSv1 this reached the instance metadata endpoint, exposing the task role's temporary credentials; internal-only admin endpoints and the Kubernetes API server were reachable from the delivery path across the fleet.

Attack Scenario

1. Attacker registers a webhook pointing at a hostname they control.
2. The hostname's DNS record alternates between a public IP (passes validation) and `169.254.169.254` (used at delivery), or returns a `302` to the metadata endpoint.
3. The platform delivers the webhook to the metadata service and echoes the response, leaking internal data.

Severity Considerations

High rather than Critical because most of the production fleet already enforced IMDSv2 (hop limit 1): the metadata-credential path was viable only on the residual IMDSv1 hosts, while the rebinding-to-internal-services impact applied fleet-wide. Had IMDSv1 been reachable across the whole fleet the rating would be Critical.

Proof of Concept

```
# Attacker-controlled DNS: rebind.evil.example alternates A records.
# First resolution → 203.0.113.10 (public, passes validation)
# Second resolution → 169.254.169.254 (metadata, used at delivery)
curl -X POST https://api.acme.com/v1/webhooks \
  -H 'authorization: Bearer <user-jwt>' \
  -d '{"url":"http://rebind.evil.example/latest/meta-data/"}'
# Trigger delivery; the stored response contains internal metadata.
```

Remediation

All outbound webhook delivery now flows through a dedicated egress proxy that resolves the destination once, pins the connection to the validated IP, rejects RFC-1918 / link-local / ULA ranges, and refuses cross-host redirects. IMDSv2 is now enforced fleet-wide — IMDSv1 disabled, hop limit 1 — via the launch template, closing the residual metadata path on the older hosts.

↑ PTF-2 CONTINUED FROM PREVIOUS PAGE

Verification

Trace re-ran the rebinding and redirect payloads; both were rejected at the proxy with `403 destination_not_allowed`, and the metadata endpoint was unreachable from the delivery path.

MEDIUM PTF-3 Remediated & Verified

JWT refresh tokens issued without rotation allow replay after logout

CVSS	5.8
VECTOR	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:H/I:N/A:N
CATEGORY	Session Management
CWE	CWE-613
OWASP	A07:2025 Authentication Failures
REPORTED	31 March 2026

Summary

Logging out didn't really end a session. A token that keeps you signed in kept working for up to 30 days even after you logged out, so anyone who had copied it could keep using the account quietly. The account's own access only, but a long, silent window. Fixed and re-tested.

Description

Refresh tokens issued by `backend-api` were long-lived (30-day) opaque tokens stored server-side, but logout deleted only the access-token session and did not revoke the associated refresh token. A refresh token captured before logout — from a shared machine's local storage or a proxy log — could be exchanged for a fresh access token after the user believed they had signed out.

Root Cause Analysis

Logout revoked the access-session row, but the refresh-token table had no `revokedAt` linkage to the session, and refresh tokens were not rotated on use. A single captured token therefore remained valid for its full 30-day lifetime.

Impact

An attacker who captured a refresh token retained authenticated access to the victim's account for up to 30 days, surviving the victim's logout. Account scope only — no privilege escalation — but a meaningful, silent replay window.

↑ PTF-3 CONTINUED FROM PREVIOUS PAGE

Attack Scenario

1. Victim authenticates on a shared or compromised browser; the refresh token persists in local storage.
2. Victim logs out, believing the session ended.
3. Attacker reads the stored refresh token and exchanges it at `POST /v1/auth/refresh`, receiving a valid access token.

Severity Considerations

Medium: requires prior capture of the refresh token, but the impact — post-logout account access for up to 30 days, invisible to the victim — is non-trivial.

Proof of Concept

```
curl -X POST https://api.acme.com/v1/auth/refresh \
  -H 'content-type: application/json' \
  -d '{"refreshToken":"<captured-token>"}'
# 200 OK with a fresh access token, issued after the victim's logout.
```

Remediation

Logout now revokes the refresh token bound to the session, refresh tokens rotate on every use (single-use with reuse detection that revokes the entire token family on replay), and the lifetime was reduced to 7 days.

Verification

A refresh token captured pre-logout was rejected with `401 token_revoked` after logout, and replaying a rotated token revoked the family as designed.

MEDIUM PTF-4 Remediated & Verified

Tenant isolation gap: project activity endpoint missing organizationId scoping

CVSS	5.4
VECTOR	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N
CATEGORY	Broken Object-Level Authorization
CWE	CWE-639
OWASP	A01:2025 Broken Access Control
REPORTED	31 March 2026

Summary

One page of project activity forgot to check which company the viewer belonged to. Anyone with an account who knew a project's identifier could see another company's project name, member names, and when they were last active. No files or credentials were exposed. Fixed and re-tested.

Description

`GET /v1/projects/:projectId/activity` loaded the activity feed by `projectId` alone, without scoping the query by the caller's `organizationId`. Any authenticated user who knew or guessed a project ID belonging to another tenant could read that project's activity metadata.

Root Cause Analysis

The handler issued `prisma.activity.findMany({ where: { projectId } })` without the composite `(projectId, organizationId)` filter the rest of the API applies through a shared scoping helper. This route predated the helper and was never migrated.

Impact

Cross-tenant disclosure limited to project activity metadata — project name, member display names, and last-activity timestamps. No file contents, credentials, or PII beyond display names were exposed. Required a valid authenticated session and a target project UUID.

↑ PTF-4 CONTINUED FROM PREVIOUS PAGE

Attack Scenario

1. Attacker authenticates as an ordinary user with no special role.
2. Attacker obtains a foreign `projectId` (e.g. from a shared link).
3. Attacker calls the activity endpoint and receives the other tenant's activity feed.

Severity Considerations

Medium — held down by the limited data class (metadata only) and the need for a valid project UUID; held up by the clean cross-tenant boundary break with no additional precondition.

Proof of Concept

```
curl https://api.acme.com/v1/projects/<foreign-project-uuid>/activity \
  -H 'authorization: Bearer <attacker-jwt>'
# 200 OK — returns activity for a project in another organization.
```

Remediation

The handler now resolves the project through the shared `(projectId, organizationId)` scoping helper and returns `404` when the project does not belong to the caller's organization. A Fastify pre-handler was added to assert org context on every authenticated route.

Verification

Requesting a foreign project ID now returns `404 not_found`; the caller's own projects resolve normally.

LOW PTF-5 Remediated & Verified

Container backend-api runs as root with allowPrivilegeEscalation: true

CVSS	3.7
VECTOR	CVSS:3.1/AV:L/AC:H/PR:H/UI:N/S:C/C:L/I:L/A:N
CATEGORY	Container Hardening
CWE	CWE-250
OWASP	A02:2025 Security Misconfiguration
REPORTED	31 March 2026

Summary

The main backend service ran with the highest level of permission inside its container, more than it needs. That doesn't let anyone in by itself, but if a future bug ever did, it would remove a safety layer that limits how far an attacker could get. Fixed and re-tested.

Description

The `backend-api` Kubernetes Deployment ran without a `securityContext`: the container executed as UID 0 (root) and `allowPrivilegeEscalation` defaulted to `true`. A code-execution bug in the application would run with unnecessary in-container privilege and a larger path toward host or cluster escalation.

Root Cause Analysis

The Deployment manifest omitted pod- and container-level `securityContext`, inheriting the base image's default `USER root`. No admission policy enforced non-root execution.

Impact

Defense-in-depth weakness, not directly exploitable on its own. It widens the blast radius of any future remote-code-execution bug in `backend-api` and removes a containment layer the cluster otherwise relied on.

↑ PTF-5 CONTINUED FROM PREVIOUS PAGE

Severity Considerations

Low: no standalone exploit path. The rating reflects the increased severity it would lend to a chained code-execution finding rather than independent impact.

Remediation

The Deployment now sets `runAsNonRoot: true`, `runAsUser: 10001`, `allowPrivilegeEscalation: false`, drops all Linux capabilities, and mounts a read-only root filesystem. A Kyverno `require-non-root` policy enforces this cluster-wide.

Verification

The running pod reports UID 10001, and a test Deployment attempting `runAsUser: 0` was rejected by the admission policy.

LOW PTF-6 Remediated & Verified

Long-lived API keys without default expiration or last-used tracking

CVSS	3.5
VECTOR	CVSS:3.1/AV:N/AC:H/PR:H/UI:N/S:U/C:L/I:L/A:N
CATEGORY	Credential Management
CWE	CWE-732
OWASP	A07:2025 Authentication Failures
REPORTED	31 March 2026

Summary

API keys never expired and nobody could tell whether a key was still in use. A key that leaked into a repository, a ticket, or a log stayed valid until someone noticed and revoked it by hand, with no signal that anything was wrong. Fixed and re-tested.

Description

Personal and service API keys (`acme_sk_*`) were created without an expiration, and the platform did not record a `lastUsedAt` timestamp. Leaked or forgotten keys remained valid indefinitely and could not be identified as stale for rotation.

Root Cause Analysis

The API-key model had no `expiresAt` or `lastUsedAt` column; key creation defaulted to no expiry and there was no usage-tracking middleware.

Impact

A leaked key — committed to a repo, pasted in a ticket, or captured in a log — stayed valid until manually revoked, with no signal to operators that it was unused or compromised. Scope is the key's own permissions.

↑ PTF-6 CONTINUED FROM PREVIOUS PAGE

Severity Considerations

Low: a credential-lifecycle/hygiene weakness rather than a direct vulnerability; impact depends on a separate key-leak event.

Remediation

New keys default to a 90-day expiry (configurable, capped at one year), `lastUsedAt` is recorded on every authenticated request, and the dashboard surfaces keys unused for 30+ days for rotation.

Verification

Newly minted keys carry an `expiresAt`; the key list shows `lastUsedAt`, and an expired key was rejected with `401 key_expired`.

LOW PTF-7 Remediated & Verified

Stored XSS via unsanitized project name in Slack webhook payload

CVSS	3.4
VECTOR	CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:N/A:N
CATEGORY	Cross-Site Scripting
CWE	CWE-79
OWASP	A05:2025 Injection
REPORTED	31 March 2026

Summary

A project name could carry hidden code that ran in the browser of teammates who previewed the Slack notification. Someone able to create projects could use that to act as another member of the same company inside the dashboard. It stays within one company. Fixed and re-tested.

Description

Project names were rendered into the Slack notification payload without escaping Slack `mrkdwn` control characters, and the same unescaped name was reflected into the dashboard's notification-preview pane via `dangerouslySetInnerHTML`. A crafted project name executed as stored XSS in the context of other org members who opened the preview.

Root Cause Analysis

The notification builder interpolated `project.name` directly into both the Slack `mrkdwn` block and the dashboard preview HTML, and the preview component rendered the template with `dangerouslySetInnerHTML`.

Impact

An attacker with project-create permission in an org could store markup that executed in the browser of other org members viewing the notification preview, enabling session-scoped actions in the dashboard. Cross-user within a single tenant; not cross-tenant.

↑ PTF-7 CONTINUED FROM PREVIOUS PAGE

Attack Scenario

1. Attacker (an org member) creates a project whose name carries an event-handler payload; the preview sink applied no sanitization, so the markup was rendered as-is.
2. A teammate opens the notification preview for that project.
3. The payload executes in the teammate's authenticated session.

Severity Considerations

Low: requires authenticated project-create access within the same org and a victim viewing the preview, and the CSP (see PTF-10) partially mitigated inline execution. Stored persistence and cross-user reach kept it above Informational.

Proof of Concept

```
Project name: "><img src=x onerror=alert(document.domain)>
Opening the notification preview for this project triggers the handler.
```

Remediation

Project names are now HTML-escaped before insertion into the dashboard preview (the `dangerouslySetInnerHTML` path was removed) and escaped for Slack `mrkdwn` before delivery. A length and character allow-list was added at the project-name input boundary.

Verification

The payload above now renders as inert text in both the dashboard preview and the delivered Slack message.

INFO PTF-8 Accepted with Controls

RDS instance acme-staging-rds deployed in public subnet

CVSS	0.0
CATEGORY	Network Exposure
CWE	CWE-668
OWASP	A02:2025 Security Misconfiguration
REPORTED	31 March 2026

Summary

The staging database sat on a network segment that's reachable from the internet, where a database normally wouldn't be. The firewall rules still blocked every outside connection and the database holds only test data, so nothing was exposed; Acme accepted the setup with those controls in place.

Description

The RDS instance `acme-staging-rds` was provisioned in a public subnet with an auto-assigned route to the internet gateway. `PubliclyAccessible` was correctly `false`, so no public endpoint was assigned, and inbound reachability was restricted by the `sg-acme-staging-rds` security group to the staging application subnets only.

Root Cause Analysis

A staging Terraform module reused the public-subnet group from an earlier single-tier layout. The placement deviated from the private-subnet standard used in production, though `PubliclyAccessible` remained false.

Impact

No exploitable exposure observed: the security group denied all inbound except the staging application subnets, and no public endpoint was assigned. This is a configuration-drift / defense-in-depth observation, and the instance holds only synthetic test data.

↑ PTF-8 CONTINUED FROM PREVIOUS PAGE

Severity Considerations

Informational (CVSS 0.0): no exploitable path under the current security-group posture. Raised as configuration drift from the private-subnet standard, not as a live exposure.

Compensating Controls

Accepted as risk with documented compensating controls. Acme elected not to migrate the staging instance this cycle. Compensating controls: (1) `sg-acme-staging-rds` restricts inbound 5432 to the staging application subnets only; (2) the instance contains synthetic test data exclusively, with no production or customer data; (3) `PubliclyAccessible` remains `false`, so no public endpoint is assigned. Migration to a private subnet is tracked for the next infrastructure cycle.

INFO PTF-9 Remediated & Verified

Missing pre-commit hooks for secrets and IaC scanning

CVSS	0.0
CATEGORY	Secure SDLC
CWE	CWE-1059
OWASP	A03:2025 Software Supply Chain Failures
REPORTED	31 March 2026

Summary

Nothing stopped a developer from accidentally committing a password or an insecure infrastructure setting before it reached the main branch. That isn't a hole on its own, but it makes the kind of leaks found elsewhere in this report more likely to happen again. Fixed and re-tested.

Description

Repositories in the Acme org had no pre-commit or CI gate for secret detection or infrastructure-as-code scanning. Secrets accidentally committed would only be caught after push, and IaC misconfigurations such as the trust-policy gap in PTF-1 had no automated guardrail.

Root Cause Analysis

No org-level template enforced pre-commit hooks or a CI scanning stage; adoption was left to individual repositories, most of which had none.

Impact

A secure-SDLC / defense-in-depth observation. It raises the likelihood that secret leaks and IaC misconfigurations reach the default branch undetected. No direct exploitability.

Severity Considerations

Informational: a process gap rather than a vulnerability; its value is in preventing recurrence of higher-severity findings such as PTF-1.

↑ PTF-9 CONTINUED FROM PREVIOUS PAGE

Remediation

A shared `pre-commit` config (gitleaks + `tflint` + `checkov`) was added to the org template repository and adopted across the in-scope repositories, and a required CI stage runs the same scanners on every pull request, blocking merge on findings.

Verification

Trace confirmed the CI scanning stage is required on `acme/backend-api`, `acme/web-app`, and `acme/infra-config`, and that a test commit containing a dummy secret was blocked pre-merge.

INFO PTF-10 In Progress

Weak Content Security Policy on app.acme.com permits unsafe-inline styles

CVSS	0.0
CATEGORY	Security Headers
CWE	CWE-693
OWASP	A02:2025 Security Misconfiguration
REPORTED	31 March 2026

Summary

The web app's rules about which styles a page may load are looser than they could be. On their own they don't let anyone run code, but they weaken a safety net that would otherwise limit the damage from an injection bug. Acme is working on the fix.

Description

The dashboard at `app.acme.com` served a Content-Security-Policy that included `style-src 'self' 'unsafe-inline'`. While `script-src` was correctly nonce-based, the `'unsafe-inline'` allowance for styles weakens defense against CSS-based data exfiltration and reduces the mitigating value of CSP against injection findings such as PTF-7.

Root Cause Analysis

The CSP was authored when the frontend relied on inline styles from a third-party component library. The inline-style dependency has since been largely removed, but the directive was not tightened.

Impact

A defense-in-depth weakness. `'unsafe-inline'` on `style-src` does not by itself grant code execution, but it lowers the bar for CSS-injection exfiltration and weakens CSP as a mitigation layer for stored-injection findings.

Severity Considerations

Informational: no direct exploit. The rating reflects reduced defense-in-depth rather than independent impact.

↑ PTF-10 CONTINUED FROM PREVIOUS PAGE

Recommended Remediation

Migrate the remaining inline styles to nonced or hashed `<style>` blocks and remove `'unsafe-inline'` from `style-src`, matching the nonce-based `script-src` already in place. This change is in progress; Trace will re-test once it ships.