Peak Security

Northstar Labs Security Report

by **Peak Security**

Table of Contents

1. Executive Summary

Peak Security assessed Northstar Labs' customer portal and supporting API to identify exploitable weaknesses that could affect customer data, account integrity, or service availability. The assessment produced two reportable findings: one High cross-tenant authorization issue and one Medium password-reset token issue. The authorization issue gives an authenticated user access to another tenant's invoice metadata and should be fixed first. The password-reset issue allows one reset token to be used twice before expiry, which leaves an account-takeover path after a legitimate reset completes.

Findings

Vulnerability Distribution

The following chart shows validated findings by severity.

0	1	1	0	0
CRITICAL	HIGH	MEDIUM	LOW	INFO

Vulnerabilities Listing

#	Severity	Finding
PS-2026-001	High	Broken object-level authorization exposes cross-tenant invoice metadata
PS-2026-002	Medium	Password reset tokens remain valid after a successful password change

Critical & High Severity Findings

PS-2026-001

Broken object-level authorization exposes cross-tenant invoice metadata

Severity	High	Affected Asset	api.northstar.example

Summary

The invoice endpoint accepts a caller-controlled invoice identifier but does not verify that the invoice belongs to the authenticated caller's organization. A low-privileged user can therefore request invoice metadata belonging to another tenant.

Evidence

HTTP REQUEST AND RESPONSE

```
GET /v1/invoices/inv_demo_tenant_b HTTP/1.1
Host: api.northstar.example
Authorization: Bearer <tenant-a-user-token>

HTTP/1.1 200 OK
Content-Type: application/json

{
  "id": "inv_demo_tenant_b",
  "organization_id": "org_demo_tenant_b",
  "amount_due": 18400,
  "billing_email": "billing@tenant-b.example"
}
```

Root Cause

The route queries invoices by globally unique invoice_id and returns the first match. The data-access layer does not include the authenticated organization_id in the query, and the route performs no ownership check before serializing the response.

Impact

An authenticated attacker can read another customer's billing contact, invoice state, plan information, and amount due. The same authorization pattern is also used by update and export handlers, creating a credible path to unauthorized modification and bulk disclosure if identifiers are discovered at scale.

- Cross-tenant exposure of billing metadata and customer contact information
- Potential unauthorized invoice-state changes through adjacent handlers
- Increased compliance and customer-trust impact because tenant boundaries are bypassed

Proof of Concept

The issue was reproduced with two test organizations supplied for the engagement. No production customer data was accessed.

1. Sign in as a member of test organization A and capture a valid access token.
2. Create an invoice in test organization B and record its synthetic invoice ID.
3. Request /v1/invoices/<tenant-b-invoice-id> with organization A's token.
4. Observe a 200 OK response containing organization B's invoice metadata.

REPRODUCTION COMMAND

```
curl -sS \
-H 'Authorization: Bearer <tenant-a-user-token>' \
'https://api.northstar.example/v1/invoices/inv_demo_tenant_b' | jq
```

Remediation

1. Scope every invoice lookup to the authenticated organization, for example WHERE id = ? AND organization_id = ?.
2. Centralize resource-ownership checks in the authorization layer and require them for read, update, export, and delete handlers.
3. Return a consistent 404 Not Found response for nonexistent and unauthorized resources to reduce identifier probing.
4. Add negative integration tests that attempt cross-tenant access for every invoice operation.
5. Review other routes that accept object identifiers for the same missing tenant constraint.

Preferred control: Enforce tenant ownership in the data-access query itself. A route-level check alone is easier to omit and more likely to regress.

Medium Severity Findings

PS-2026-002

Password reset tokens remain valid after a successful password change

Severity	Medium	Affected Asset	app.northstar.example

Summary

A password reset link can be used more than once until its 30-minute expiry. Completing the reset does not revoke the underlying token or invalidate other outstanding reset tokens for the account.

Observed Behavior

The same reset URL successfully changed the test account password twice, including after the account owner had already signed in with the first new password.

Impact

Anyone who obtains a reset link from browser history, logs, an email compromise, or accidental forwarding retains an account-takeover path even after the legitimate user completes the reset.

Evidence

TWO SUCCESSFUL USES OF ONE TOKEN

```
POST /auth/password/reset -> 204 No Content
Token: rst_demo_7f4...
New password: <first-test-password>

POST /auth/password/reset -> 204 No Content
Token: rst_demo_7f4...
New password: <second-test-password>
```

Remediation

1. Store reset-token state server-side and atomically mark a token consumed on first successful use.
2. Invalidate every outstanding reset token for the account after a password change.
3. Revoke active sessions or provide a clear option to do so during reset.
4. Avoid writing raw reset tokens to application, proxy, analytics, or error logs.

Peak Security

peaksecurity.us

contact@peaksecurity.us