CLASSIFICATION : Confidential
REPORT ID : EXM-EXM-2026-001
VERSION : 1.0

PENETRATION TESTING REPORT

CLIENT : [redacted]
PROJECT : Web Penetration Testing
DATE : 20 Mei 2026

PREPARED BY : DPSK Security
Independent Penetration Testing
& Security Research

TESTER : Muhammad Khuluqil Karim

DOCUMENT CONTROL

DOCUMENT INFORMATION

Field	Value
Report Title	Web Application Penetration Testing Report
Client	[redacted]
Project	Web Penetration Testing
Report ID	EXM-EXM-2026-001
Version	1.0
Classification	Confidential
Date	20 Mei 2026

VERSION HISTORY

Version	Date	Author	Description
0.1	20 Mei 2026	DPSK Security	Initial Draft
1.0	20 Mei 2026	DPSK Security	Final Report

EXECUTIVE SUMMARY

Security testing was conducted on the [redacted] web application platform to identify potential security vulnerabilities that could be exploited by unauthorized parties.

During the assessment, the testing team uncovered critical configuration and authorization flaws. Most notably, a Critical-severity Vertical Privilege Escalation was successfully exploited via the public-facing API Gateway (PostgREST/Supabase), allowing a low-privileged account (clipper) to instantly elevate its privileges to an administrative role (admin) and bypass financial compliance controls (kyc_access). Additionally, a Low-severity Information Disclosure vulnerability was identified due to overly verbose database error messages that expose internal schema relationships.

Immediate remediation is required to secure the database access layers, restrict field-level permissions, and ensure the overall integrity of the platform's business logic.

ENGAGEMENT OVERVIEW

This penetration test evaluated the security posture of the web application, API gateways, database access layers, and server configurations.

The main objectives were to:

- Identify Vulnerabilities: Discover flaws in application logic, authentication, database security rules (RLS/CLS), and server setups.
- Evaluate Potential Impact: Analyze the real-world business and technical impact of potential exploitation (e.g., privilege escalation and data leaks).
- Provide Remediation Guidance: Deliver actionable technical recommendations to fix discovered flaws and harden the system.

SCOPE OF TESTING

Security testing was performed on the following systems:

Target	Type
[redacted]	Web Application
[redacted]	Web Application / API Gateway

OUT OF SCOPE

Target	Reason
Denial of Service (DoS/DDoS) Attacks	Out of scope to prevent operational downtime on production services.

METHODOLOGY

Security testing is conducted using an approach that adheres to industry standards, such as:

- OWASP Testing Guide
- OWASP Top 10
- NIST SP 800-115

Testing stages include:

- Reconnaissance
- Enumeration
- Vulnerability Identification
- Exploitation Validation
- Reporting

RISK RATING MODEL

Vulnerability risk assessment is done based on a combination of the probability of exploitation and the impact on the system.

Severity	Description
Critical	Vulnerability could lead to complete compromise
High	The vulnerability could lead to significant access
Medium	Limited impact
Low	Minimal risk

FINDINGS SUMMARY

VULNERABILITY FINDINGS SUMMARY

ID	Nama Kerentanan	Severity	Affected Asset	Description
EXM-SEC-001	Vertical Privilege Escalation via API Mass Assignment	Critical		The database layer (Supabase API) allows regular users to directly modify administrative fields (role, kyc_access) through HTTP PATCH requests.
EXM-SEC-002	Verbose Error Messages (Information Disclosure)	Low		The system returns overly detailed database error messages (error hints) that disclose the internal table schema and relationships when processing malformed requests.

EXPLANATION

Vertical Privilege Escalation via API Mass Assignment is a condition where the application's database layer (Supabase API) allows a regular user to manipulate profile control parameters and obtain administrative privileges. In a secure implementation, sensitive fields such as the account role (role: admin) or financial verification status (kyc_access) should be strictly protected and only modifiable by trusted internal system processes, rather than directly through the user's browser.

If exploited, an unauthorized user could elevate their account to Administrator within seconds. This creates a critical security risk, as the attacker could potentially gain full access to restricted application functionality, access internal company data, and even approve payment transactions without proper authorization. Therefore, the development team should immediately implement database-level protections, such as database triggers, to prevent unauthorized modification of these sensitive fields.

DETAILED FINDING

EXM-SEC-001 – VERTICAL PRIVILEGE ESCALATION VIA API MASS ASSIGNMENT

SEVERITY: Critical

AFFECTED ASSET:

Target	Endpoint
[redacted]	/rest/v1/users

EXAMPLE REQUEST:

PATCH https://[redacted]/rest/v1/users?id=eq.acb5ceaf-a8b1-4cf0-b535-1f6ce08fd41a

DESCRIPTION

Vertical Privilege Escalation via API Mass Assignment is a condition in which the application's database layer (Supabase API) allows a regular user to manipulate profile-related parameters and gain administrative privileges. In a secure implementation, sensitive fields such as the account role (role) or financial verification status (kyc_access) should be strictly protected and only modifiable internally by trusted system processes, rather than directly from the user's browser.

If exploited, an unauthorized user could elevate their account to Administrator within seconds. This creates a critical security risk, as an attacker could potentially gain full access to restricted application functionality, view internal company data, and even approve payment transactions without proper authorization. Therefore, the development team should immediately enforce server-side authorization controls and database-level protections, such as database triggers, to prevent unauthorized modification of these sensitive fields.

PROOF OF CONCEPT

The exploitation was performed by sending a data manipulation request (HTTP PATCH) directly to the database API server, bypassing the validation mechanisms implemented by the web application.

EXAMPLE REQUEST:

PATCH https://[redacted]/rest/v1/users?id=eq.acb5ceaf-a8b1-4cf0-b535-1f6ce08fd41a

DATA MODIFICATION PROCESS OBSERVED:

1. NITIAL USER DATA (BEFORE EXPLOITATION):

The user logged into the system as a regular user with a low-privileged role:


```
{
  "id": "acb5ceaf-a8b1-4cf0-b535-1f6ce08fd41a",
  "name": "[redacted]",
  "role": "clipper",
  "kyc_access": false
}
```


2. MANIPULATION REQUEST SENT:

The browser sent a crafted data modification request to force the database to change the administrative role field:


```
{
  "kyc_access": true,
  "role": "admin"
}
```


3. FINAL RESULT (AFTER EXPLOITATION):

The system accepted the unauthorized modification and successfully escalated the regular user's privileges to full Administrator access:


```
{
  "id": "acb5ceaf-a8b1-4cf0-b535-1f6ce08fd41a",
  "role": "admin",
  "kyc_access": true
}
```

EXPOSED INFORMATION INCLUDES:

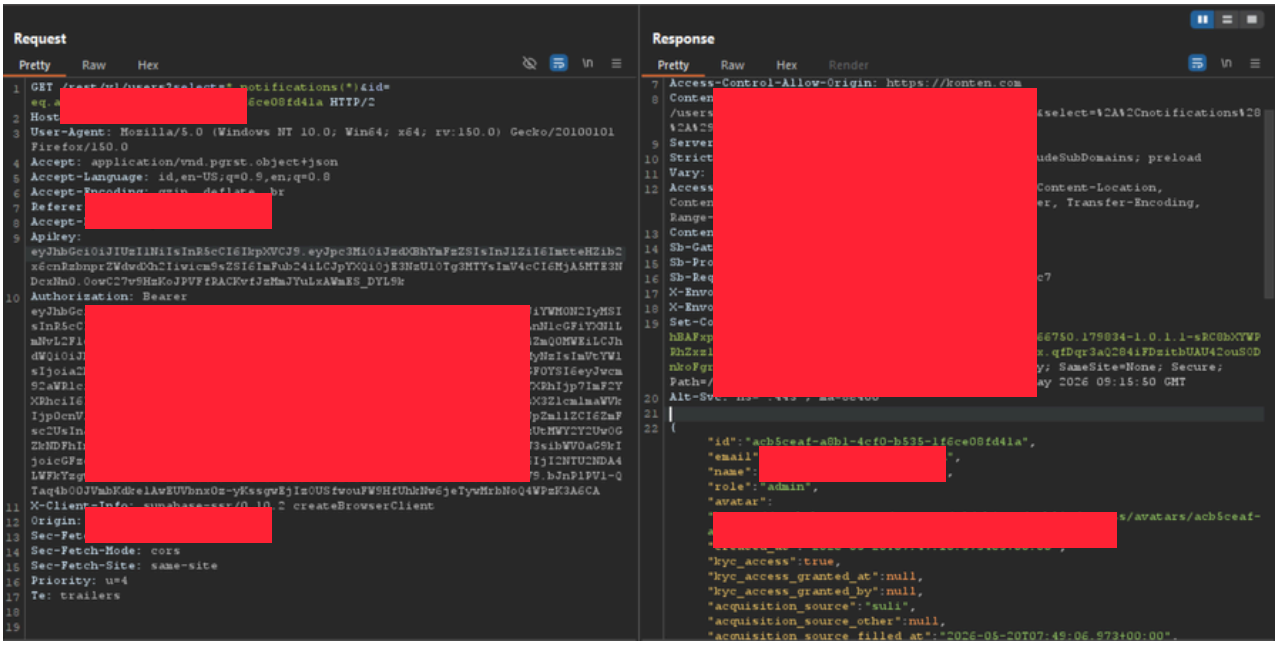
Full access to the main administrative control panel (Admin Dashboard).

IMPACT:

The successful exploitation of this unauthorized privilege escalation introduces several critical security risks, including:

- Full Backend Functionality Takeover: A regular user can act as an Administrator and gain control over the application's entire backend ecosystem.
- Mass Data Exposure: An attacker can access sensitive information belonging to other users across the platform.

EVIDENCE OF FINDING



DETAILED FINDING

EXM-SEC-002 – VERBOSE ERROR MESSAGES (INFORMATION DISCLOSURE)

SEVERITY: Low

AFFECTED ASSET:

Target	Endpoint
https://[redacted]	Global PostgREST API Endpoints

CONTOH AKSES:

GET https://[redacted].com/rest/v1/users?select=database

DESKRIPSI

Verbose Error Messages / Information Disclosure is a condition where an application's API Gateway is configured to be overly permissive when handling invalid or malformed user requests. In a secure production environment, when a user submits an invalid command or parameter, the system should return a generic error message (e.g., "An error occurred") without exposing internal database details or structure to the public.

If left unaddressed, the system may automatically disclose internal table names and column relationships whenever it receives an invalid request. Although this vulnerability does not directly modify or compromise data, the exposed information effectively provides attackers with a "free map" of the application's internal data structure. This can assist attackers in identifying valuable data sources, such as financial transaction records and user notifications, and facilitate more targeted attacks.

PROOF OF CONCEPT

The issue can be passively triggered by sending an intentionally invalid search parameter to cause the database to return detailed error hints.

ERROR RESPONSE OBSERVED:

When random keywords are provided in the parameter, the system rejects the request but discloses the existence of the payment transaction table (payout_batches) along with its approval-related column structure (approved_by and created_by):


```
{
  "code": "PGRST201",
  "hint": "Try changing 'payout_batches' to 'payout_batches!payout_batches_approved_by_fkey' or 'payout_batches!payout_batches_created_by_fkey'.",
  "message": "Could not embed because more than one relationship was found for 'users' and 'payout_batches'"
}
```


EXPOSED INFORMATION INCLUDES:

- Confirmation of the existence of a sensitive table named payout_batches containing financial transaction data.
- Confirmation of the existence of a sensitive table named notifications containing user activity history.
- Disclosure of the application's internal database relationship structure, including Foreign Key relationships.

IMPACT:

Although classified as a low-risk issue, overly detailed error messages provide attackers with several tactical advantages, including:

- Instant Database Mapping: Attackers do not need to perform suspicious mass guessing or brute-force attempts, as the system voluntarily discloses valid table names.
- Business Logic Reverse Engineering: Attackers can gain insight into the platform's internal logic, such as the existence of transaction submission processes (created_by) and dedicated approval mechanisms (approved_by).
- Force Multiplier for Further Attacks: Disclosed internal table names can serve as valuable reconnaissance for identifying data exposure vulnerabilities, such as IDOR/BOLA, in secondary application tables.

REMEDIATION SUMMARY

SUMMARY OF RECOMMENDED REMEDIATION ACTIONS

ID	Nama Kerentanan	Rekomendasi
EXM-SEC-001	Vertical Privilege Escalation via API Mass Assignment	Implement database triggers to lock down the role and kyc_access columns. Ensure that these sensitive columns can only be modified by the internal system (service_role) and automatically reject any direct modification attempts from the user's browser.
EXM-SEC-002	Verbose Error Messages (Information Disclosure)	Switch the API configuration to Production Mode. Disable error hints and database relationship descriptions to prevent the system from exposing internal table structures when request errors occur.