PENETRATION TESTING REPORT · WEB & API SECURITY

crAPI

Completely Ridiculous API — API Security Assessment

Assessment date: 19 February 2026
Prepared by: Eric Muturi and David Munene

> **Lab assessment / demonstration.** This assessment was performed against **crAPI**, an intentionally vulnerable, open-source application published by the OWASP Foundation for security training. It is **not** a client engagement and contains no real customer data. Findings are shared to demonstrate methodology and hands-on API testing.

Executive Summary

This report presents the findings of a web and API security assessment conducted against crAPI (Completely Ridiculous API), an intentionally vulnerable, open-source application developed by the OWASP Foundation to teach and practise API security. The assessment was performed on 19 February 2026.

Overall Risk Level: CRITICAL

Total Vulnerabilities Identified	6
Critical Risk Items	4
High Risk Items	2
Medium / Low Risk Items	0
API Endpoints Tested	18

Scan Details

Target	crAPI Application (http://127.0.0.1:8888)
Application Type	Web Application
Technology Stack	Node.js, MongoDB, PostgreSQL
Scan Method	OWASP API Security Top 10, Burp Suite

Testing Methodology

This penetration test followed a structured methodology based on industry-standard frameworks including the OWASP Testing Guide, the Penetration Testing Execution Standard (PTES), and NIST SP 800-115. The assessment was conducted in the following phases:

- **Reconnaissance** — information gathering to understand the target architecture, technologies, and attack surface.
- **API Endpoint Enumeration** — scanning to identify open ports, services, API endpoints, and entry points.
- **Vulnerability Assessment** — identifying weaknesses using automated tools and manual techniques.

- **Exploitation** — controlled exploitation to demonstrate real-world impact and validate findings.
- **Post-Exploitation** — assessing data access, privilege escalation, and lateral movement.
- **Reporting & Remediation Guidance** — documenting findings with remediation and risk prioritisation.

Phase 1: Reconnaissance

1.1 Domain Information

- IP Address: 127.0.0.1
- Open Ports: 8443 and 8888

```
└─$ nmap -sV 127.0.0.1
Starting Nmap 7.98 ( https://nmap.org ) at 2026-02-17 04:35 +0300
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000050s latency).
Not shown: 998 closed tcp ports (reset)
PORT     STATE SERVICE  VERSION
8443/tcp open  ssl/http OpenResty web app server 1.27.1.2
8888/tcp open  http     OpenResty web app server 1.27.1.2

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.11 seconds
```

Fig 1.1 — Open ports (Nmap)

1.2 Technology Fingerprinting

- Web Server: openresty/1.27.1.2
- Frontend: React.js
- Database: PostgreSQL 14, MongoDB 4.4
- Authentication: JWT Token

```
└─$ curl -i http://127.0.0.1:8888
HTTP/1.1 200 OK
Server: openresty/1.27.1.2
Date: Tue, 17 Feb 2026 01:24:16 GMT
Content-Type: text/html
Content-Length: 1207
Last-Modified: Mon, 22 Sep 2025 18:05:09 GMT
Connection: keep-alive
ETag: "68d18fd5-4b7"
Accept-Ranges: bytes

<!doctype html><html lang="en"><head><meta charset="utf-8"/><link rel="icon" href="/images/favicon.ico"/><meta name="viewport" content=
"width=device-width,initial-scale=1"/><meta name="theme-color" content="#000000"/><meta http-equiv="cache-control" content="no-cache"/>
<meta http-equiv="Pragma" content="no-cache"/><meta http-equiv="Expires" content="0"/><meta name="keywords" content="OWASP, API, Top 10
, BOLA, IDOR, BFLA, Mass Assignment, Broken Object Level Authorization, Broken Function Level Authentication, Excessive Data Exposure,
SSRF, Lack of Resources & Rate Limiting"/><meta name="description" content="completely ridiculous API (crAPI) will help you to understa
nd the ten most critical API security risks. crAPI is intentionally vulnerable to the OWASP API Top 10, but you'll be able to safely ru
n it to educate/train yourself."/><link rel="apple-touch-icon" href="/images/logo192.png"/><link rel="manifest" href="/manifest.json"/>
<title>crAPI</title><script defer="defer" src="/static/js/main.8c78208c.js"></script><link href="/static/css/main.dd33429d.css" rel="st
ylesheet"></head><body><noscript>You need to enable JavaScript to run this app.</noscript><div id="root"></div></body></html>
```

Fig 1.2.1 — Web server: openresty/1.27.1.2

```
└─$ sudo docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED       STATUS                    PORTS
                    NAMES
e3f6b9086758   postgres:14                    "docker-entrypoint.s…"   4 days ago    Up 53 minutes (healthy)   5432/tcp
                    postgresdb
478f7b94f9b4   chromadb/chroma:latest         "dumb-init -- chroma…"   4 days ago    Up 53 minutes (healthy)   8000/tcp
                    chromadb
c1b4716dc704   crapi/crapi-web:latest         "/etc/nginx/nginx-wr…"   4 weeks ago   Up 49 minutes (healthy)   127.0.0.1:8888→80/tcp, 127.0.0.1:30080→80/tcp, 127.0.0.1:8443→443/tcp, 127.
0.0.1:30443→443/tcp   crapi-web
c5fef3d75f7a   crapi/crapi-workshop:latest    "/app/runner.sh"         4 weeks ago   Up 50 minutes (healthy)
                    crapi-workshop
31a5ee0a9417   crapi/crapi-chatbot:latest     "/bin/sh -c /app/ent…"   4 weeks ago   Up 51 minutes             5002/tcp, 5500/tcp
                    crapi-chatbot
6cc75b788c99   crapi/crapi-community:latest   "/bin/sh -c /app/main"   4 weeks ago   Up 51 minutes (healthy)   6060/tcp
                    crapi-community
a03894f9cdd7   crapi/crapi-identity:latest    "/__cacert_entrypoin…"   4 weeks ago   Up 52 minutes (healthy)   8080/tcp, 8989/tcp, 10001/tcp
                    crapi-identity
90f4b999fa3c   mongo:4.4                      "docker-entrypoint.s…"   4 weeks ago   Up 53 minutes (healthy)   27017/tcp
                    mongodb
2c02e517a821   crapi/mailhog:latest           "MailHog"                4 weeks ago   Up 53 minutes (healthy)   1025/tcp, 127.0.0.1:8025→8025/tcp
                    mailhog
559e7245d2b1   crapi/gateway-service:latest   "/app/server"            4 weeks ago   Up 53 minutes (healthy)   443/tcp
                    api.mypremiumdealership.com
```

Fig 1.2.2 — Database: PostgreSQL 14, MongoDB 4.4

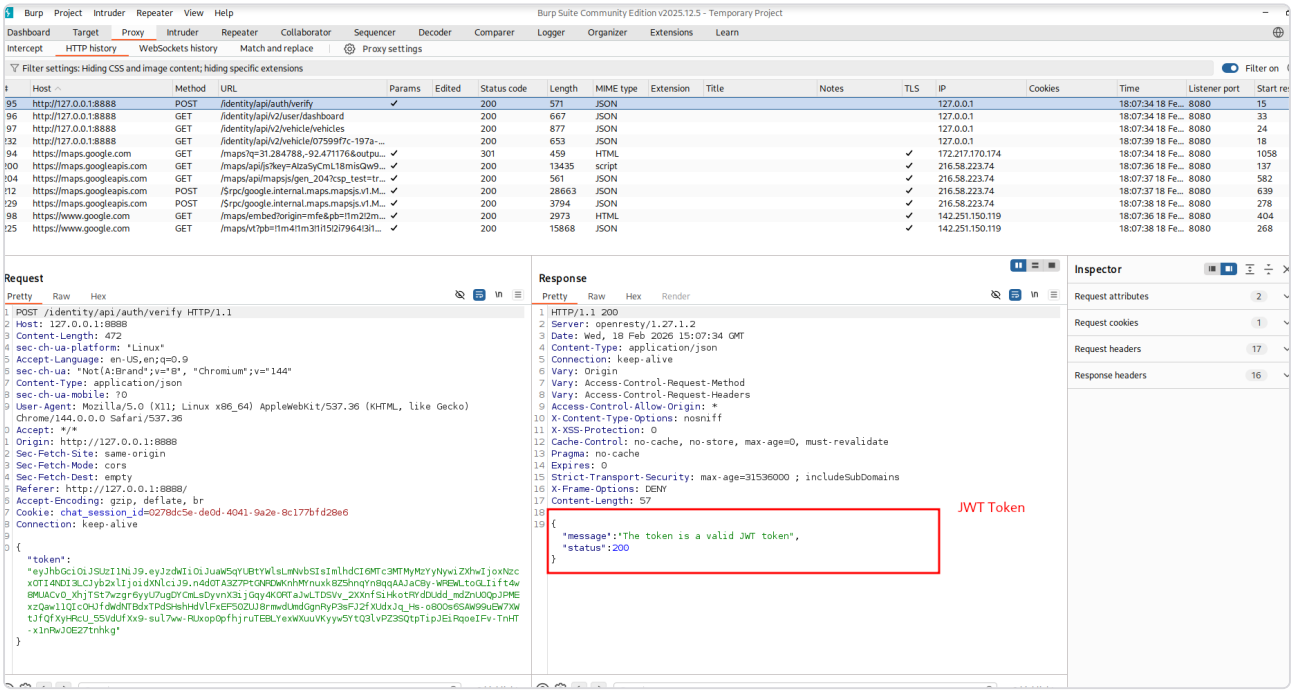

Fig 1.2.3 — Authentication: JWT token

Phase 2: API Endpoint Enumeration

Endpoints enumerated (selected):

- `POST /identity/api/auth/login`
- `GET /identity/api/v2/user/dashboard`
- `GET /workshop/api/shop/products`
- `POST /community/api/v2/coupon/validate-coupon`
- `GET /community/api/v2/community/posts/recent`
- `POST /identity/api/v2/user/reset-password`
- `GET /workshop/api/shop/orders/all`
- `GET /workshop/api/shop/orders/{id}`
- `GET /identity/api/v2/vehicle/vehicles`
- `GET /workshop/api/merchant/service_requests/{id}`

Phase 3: Vulnerability Assessment

Both automated scanners and manual techniques were used, tested against the OWASP API Security Top 10 and general web application vulnerabilities.

Tools: Burp Suite, Postman, jwt.io, CyberChef.

3.1 Vulnerability Summary

Severity	Count	CVSS Range
CRITICAL	4	9.1
HIGH	2	7.5 to 8.8
MEDIUM	0	–
LOW	0	–

3.2 Testing Coverage (OWASP API Security Top 10)

- API1:2023 Broken Object Level Authorization (BOLA) — **Vulnerabilities found**
- API2:2023 Broken Authentication — **Vulnerabilities found**
- API5:2023 Broken Function Level Authorization — **Vulnerabilities found**
- Unrestricted File Upload — **Vulnerabilities found**
- Excessive Data Exposure — **Vulnerabilities found**

Detailed Findings

Broken Object Level Authorization (BOLA) — Service Reports & Orders

1) Service Reports

Severity: CRITICAL | CVSS 9.1 | Endpoint: `/service-report`

Description: The `/service-report` endpoint is vulnerable to BOLA. The application does not verify whether the requested report ID belongs to the authenticated user. By modifying the `id` parameter, an attacker can access other users' reports.

Proof of concept: Authenticated as the AH Ninja user, requesting `/service-report?id=6` returns AH Ninja's report. Changing the parameter to `?id=5` returns another user's report with no authorization check, exposing owner information and vehicle number.

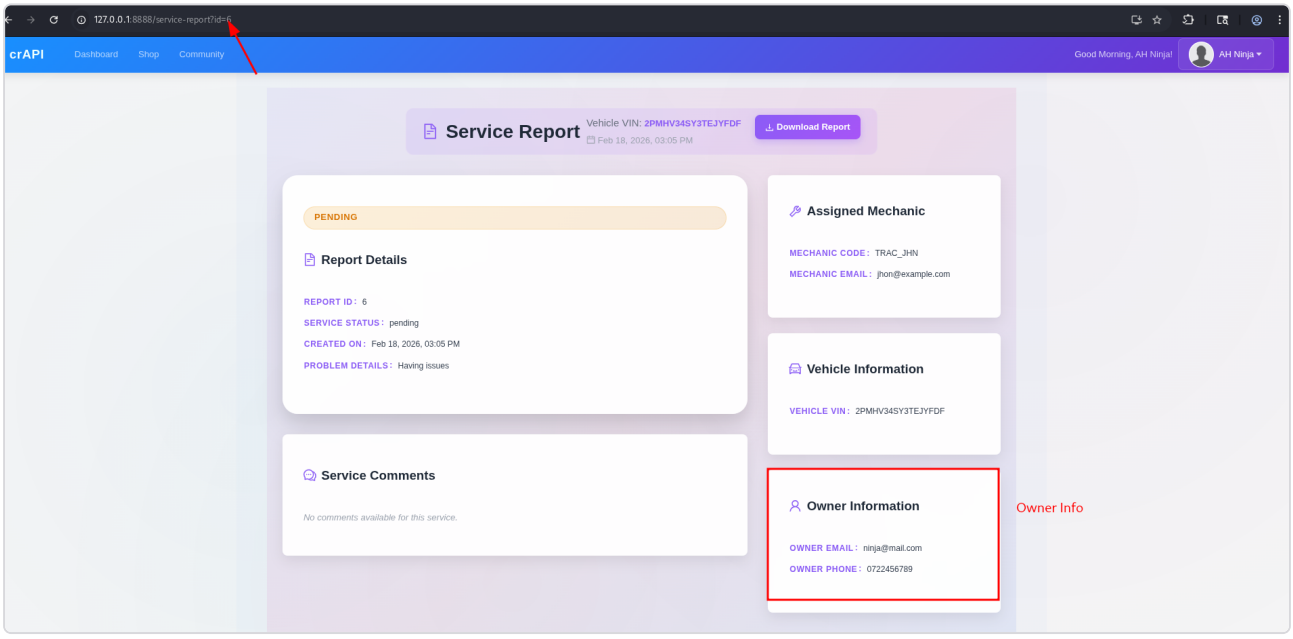

Fig 2.1.1 — User 1 service report

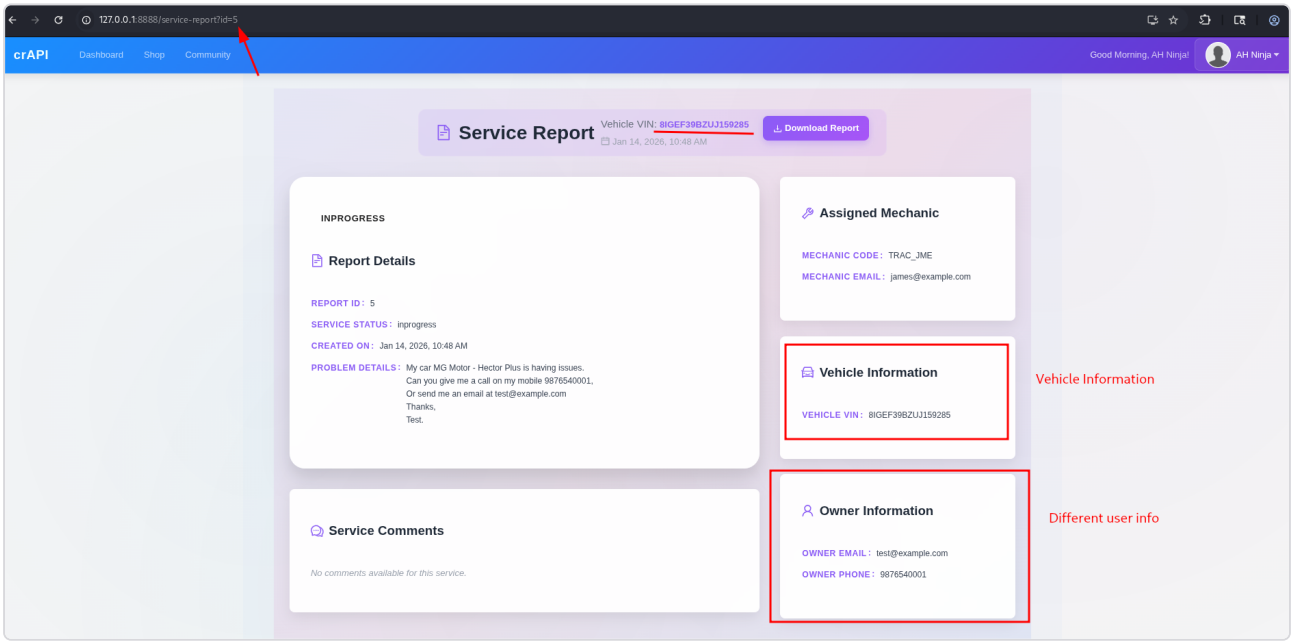

Fig 2.1.2 — A different user's service report accessed via ID tampering

Impact:

- Unauthorized access to sensitive service reports
- Exposure of personal / confidential data
- Enables enumeration across sequential report IDs

Recommendation: Implement server-side authorization to validate report ownership. Return `403 Forbidden` for unauthorized access, and use non-sequential / UUID-based IDs to reduce enumeration risk.

2) Orders

Severity: CRITICAL | CVSS 9.1 | Endpoint: `/workshop/api/shop/orders/`

Description: The orders endpoint fails to enforce object-level authorization. By modifying the `orderId`, an authenticated user can access other users' orders; sequential IDs can be enumerated without ownership verification, exposing email addresses and phone numbers.

Proof of concept: Logged in as AH Ninja, `/workshop/api/shop/orders/6` returns AH Ninja's order. Intercepting in Burp Suite and changing the ID to `/workshop/api/shop/orders/1` returns another user's order without validation, confirming BOLA.

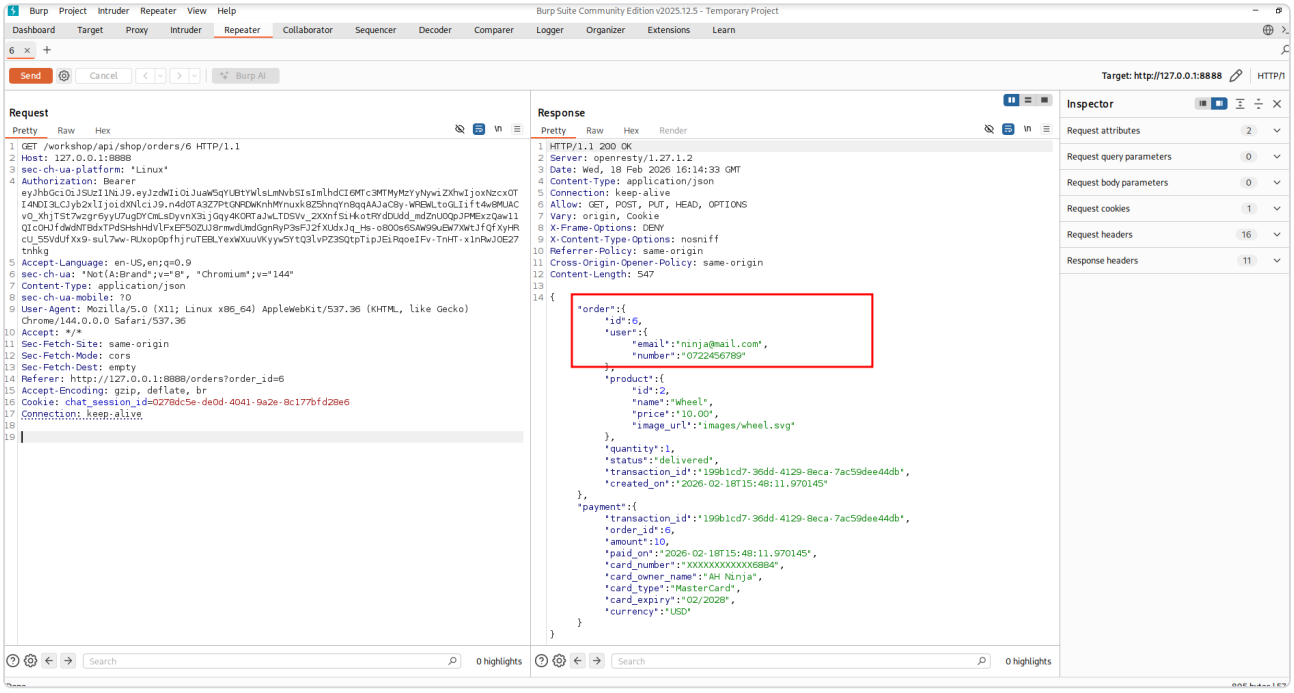

Fig 2.2.1 — User 1 orders

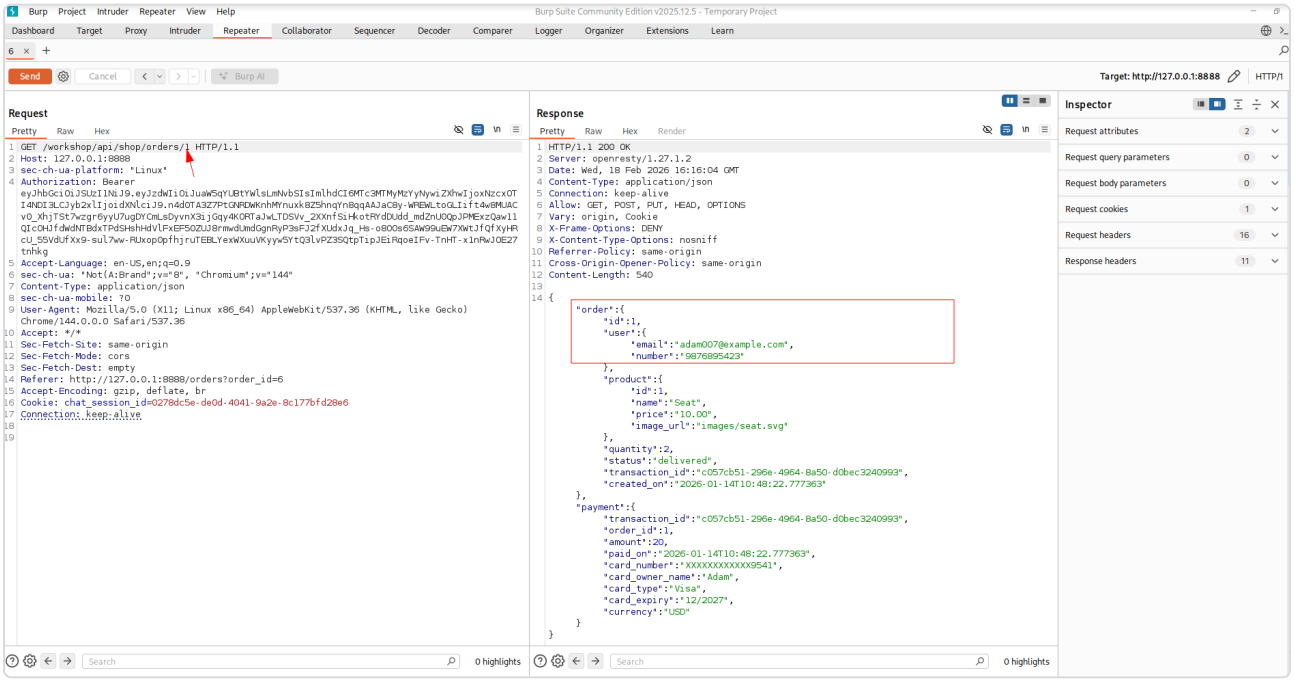

Fig 2.2.2 — A different user's orders accessed via ID tampering

Impact:

- Unauthorized access to other users' orders
- Exposure of PII (email address, phone number)
- Enables sequential ID enumeration attacks

Recommendation: Enforce server-side ownership checks, return `403 Forbidden` for unauthorized requests, and adopt non-sequential order IDs.

Algorithm Confusion (alg:none Signature Bypass)

Severity: CRITICAL | CVSS 9.1 | Endpoint: `/identity/api/auth/verify`

Description: The `/identity/api/auth/verify` endpoint improperly validates JSON Web Tokens, accepting tokens with the algorithm set to `"none"` and no valid signature. During testing the `alg` header was changed from RS256 to `"none"` and the signature removed; the server still responded `HTTP 200 OK` and reported the token as valid. This allows attackers to forge arbitrary JWTs, potentially escalating privileges and impersonating other users.

Proof of concept:

```
Reconstructed token (alg set to none, signature stripped):
eyJhbGciOiJub25lIn0.eyJzdWIiOiJqb2huZG9lQGdtYWlsLmNvbSIsImlhdCI6...LCJyb2xlIjoidXNlciJ9.

Response:
{ "message": "The token is a valid JWT token", "status": 200 }
```

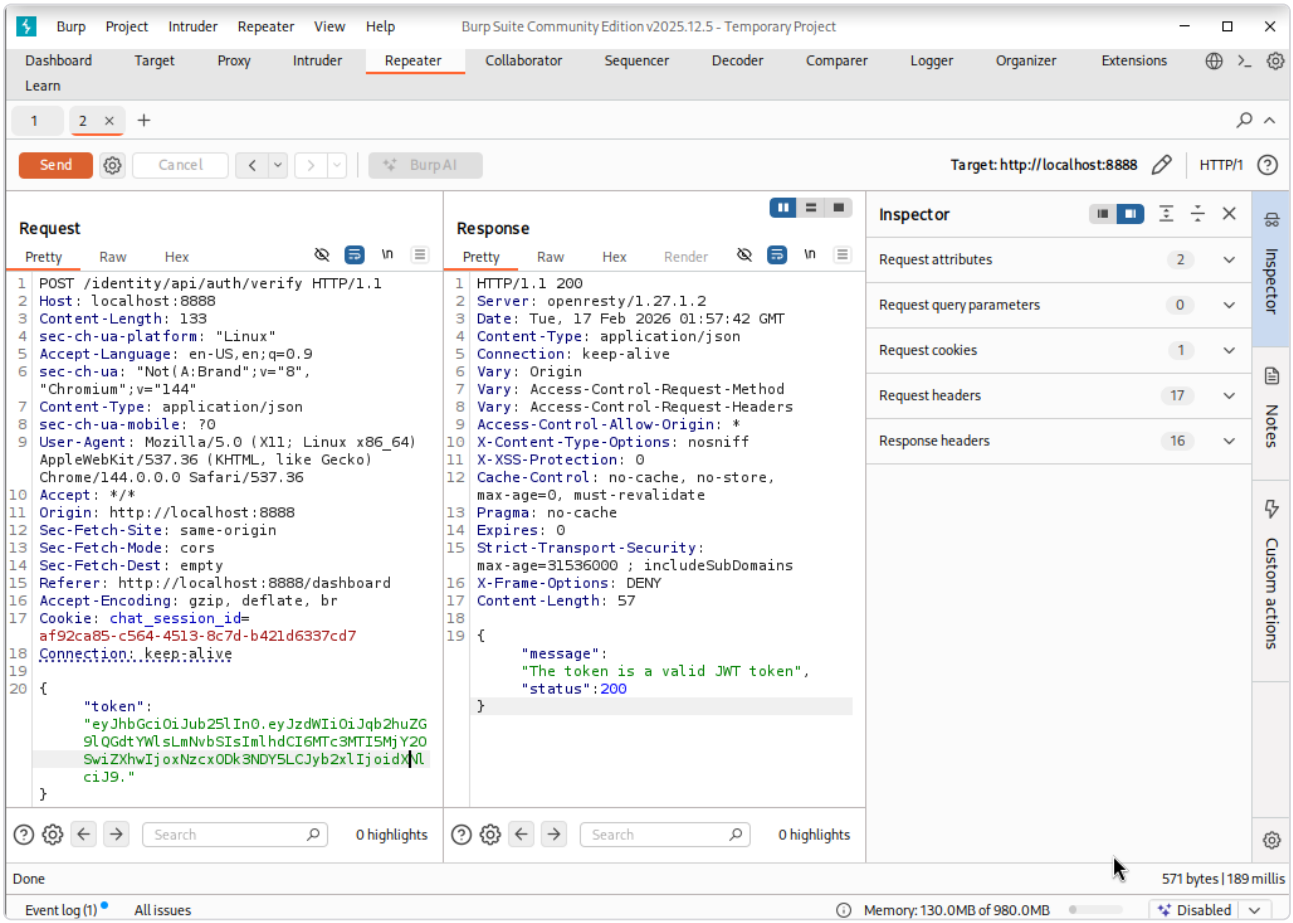

Fig 3.1 — Server accepts an unsigned (alg:none) token

Impact: authentication bypass, privilege escalation, unauthorized access to protected resources, account impersonation.

Recommendation: enforce strong algorithm validation, always verify signatures, reject `alg:none`, whitelist algorithms (e.g. RS256 only), and do not trust user-controlled claims.

Privilege Escalation

Severity: CRITICAL | CVSS 9.1 | Endpoint: `/identity/api/auth/verify`

Description: Because the application accepts `alg:none` tokens and does not verify signatures, an attacker can modify the `role` claim and escalate from a regular user to administrator. During testing, changing `"role": "user"` to `"role": "admin"` and removing the signature produced a token the server accepted with `HTTP 200 OK`.

Proof of concept:

```
Original payload:
{ "sub": "johndoe@gmail.com", "iat": 1771292669, "exp": 1771897469, "role": "user" }

Modified in CyberChef: "role": "user"  ->  "role": "admin", header set to {"alg":"none"},
signature removed.

Response:
{ "message": "The token is a valid JWT token", "status": 200 }
```

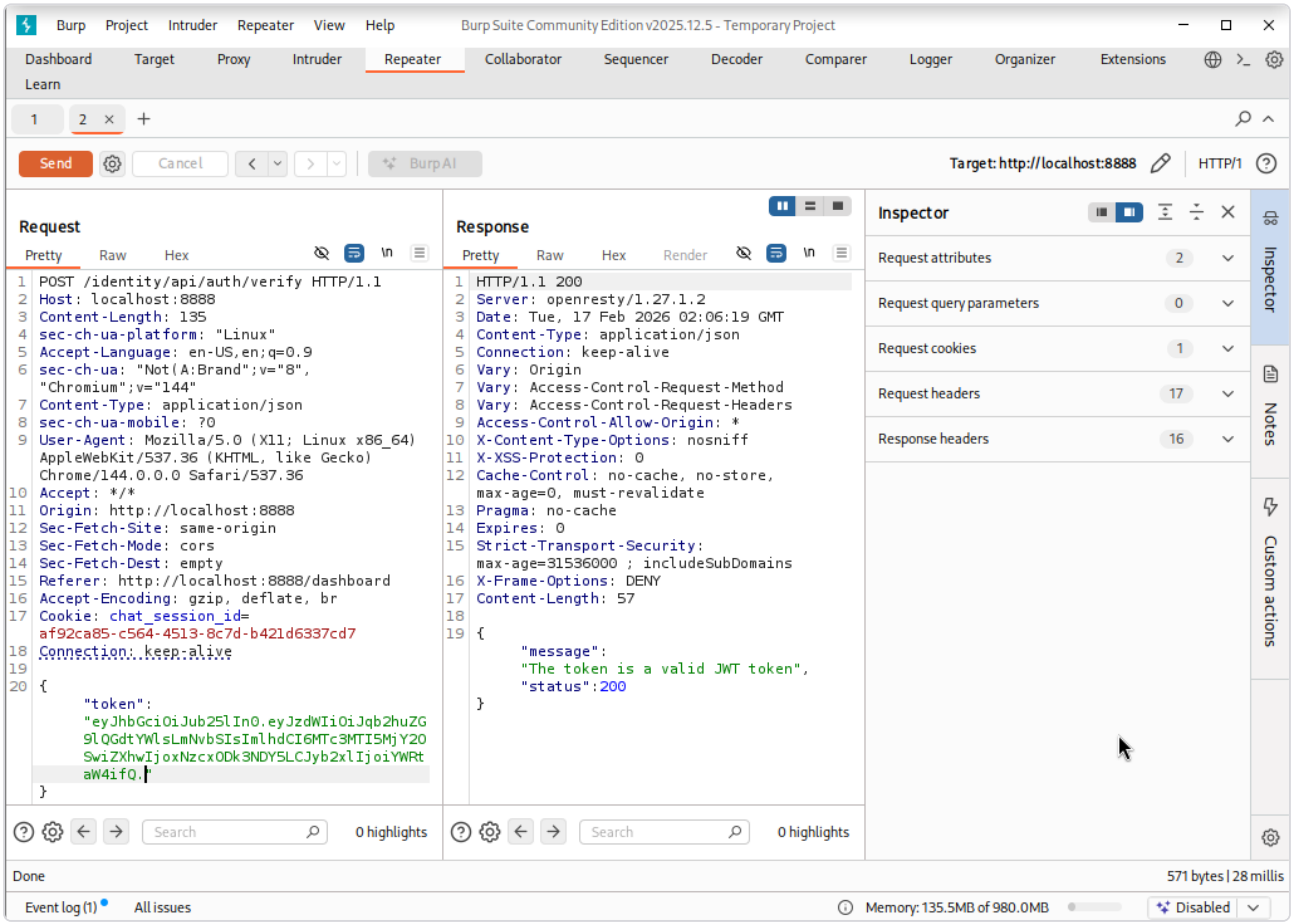

Fig 4.1 — Forged admin token accepted (privilege escalation)

Impact: full administrative account compromise, unauthorized data access, modification or deletion of resources, creation of malicious admin accounts, complete RBAC bypass, potential full system compromise.

Recommendation: enforce strict signature verification, reject `alg:none`, whitelist algorithms, perform server-side authorization against trusted data sources, and return `HTTP 401` for any invalid or unsigned token.

Excessive Data Exposure

Severity: HIGH | CVSS 7.5 | Endpoint: `/community/api/v2/community/posts/`

Description: The community posts endpoint exposes sensitive information in API responses without proper access controls. Data from other users — email addresses and vehicle IDs — is included even when the authenticated user should only see their own data, leading to privacy violations and information leakage.

Proof of concept: Logged in as a regular user, created and viewed a community post, and intercepted the API response in Burp Suite. The response included other users' email addresses and vehicle IDs, confirming excessive data exposure.

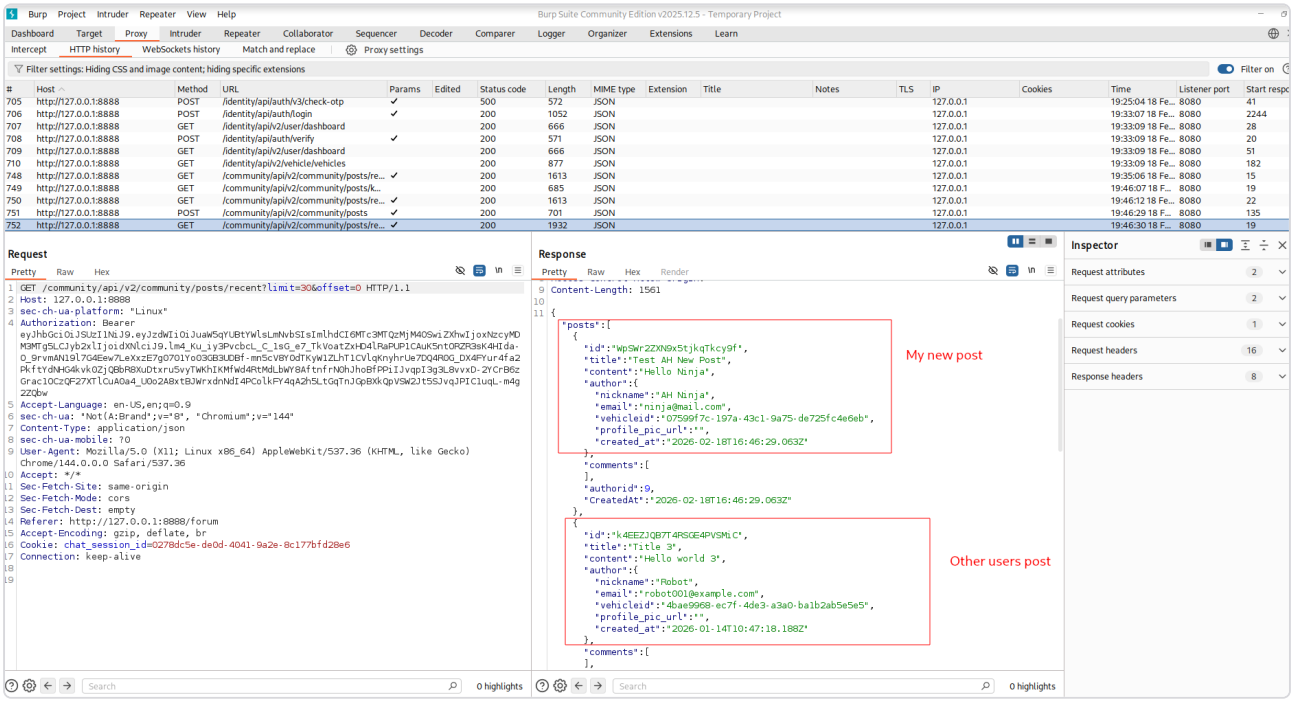

Fig 5.1 — API response leaking other users' email addresses and vehicle IDs

Impact: exposure of PII, disclosure of sensitive user data (emails, vehicle IDs), potential privacy and compliance issues.

Recommendation: restrict API responses to only the data relevant to the authenticated user; implement backend access control and data filtering; review endpoints for unnecessary data exposure.

Unrestricted File Upload — User Pictures

Severity: HIGH | CVSS 8.8 | Endpoint: `/identity/api/v2/user/pictures`

Description: The `/identity/api/v2/user/pictures` endpoint does not properly validate uploaded file types. A `.txt` and an `.html` file were accepted where only images are intended. This lack of restrictions indicates insufficient server-side controls and could allow dangerous files such as `.php` to be uploaded.

Impact: upload of unintended file types; potential remote code execution or XSS if executable or HTML files are served; malware hosting; application compromise where file handling is insecure.

Recommendation: restrict allowed types to safe image formats (.png, .jpg, .jpeg); validate media type and file signatures (magic bytes); rename uploaded files server-side to prevent execution; return `400 Bad Request` for unsupported types.

Conclusion

The assessment identified six vulnerabilities — four CRITICAL and two HIGH — indicating a fundamentally insecure application. The issues map to common OWASP API Security Top 10 patterns, including broken authentication, broken object-level authorization, broken function-level authorization, and insufficient security controls.

Immediate actions:

- Address all CRITICAL findings within 24 to 48 hours.
- Implement HIGH severity fixes within 7 days.
- Conduct a security code review focused on JWT validation and object-level authorization.
- Integrate security testing into the SDLC.
- Retest after remediation to confirm effectiveness.

Assessment conducted with manual testing supported by Burp Suite, Postman, jwt.io, and CyberChef. crAPI is a public OWASP practice target; this document is a demonstration of methodology, not a confidential client engagement.