Sri Lanka Institute of Information Technology

IDOR (Insecure Direct Object Reference)

Vulnerability - Report 07

IT23187214

Web Security - IE2062

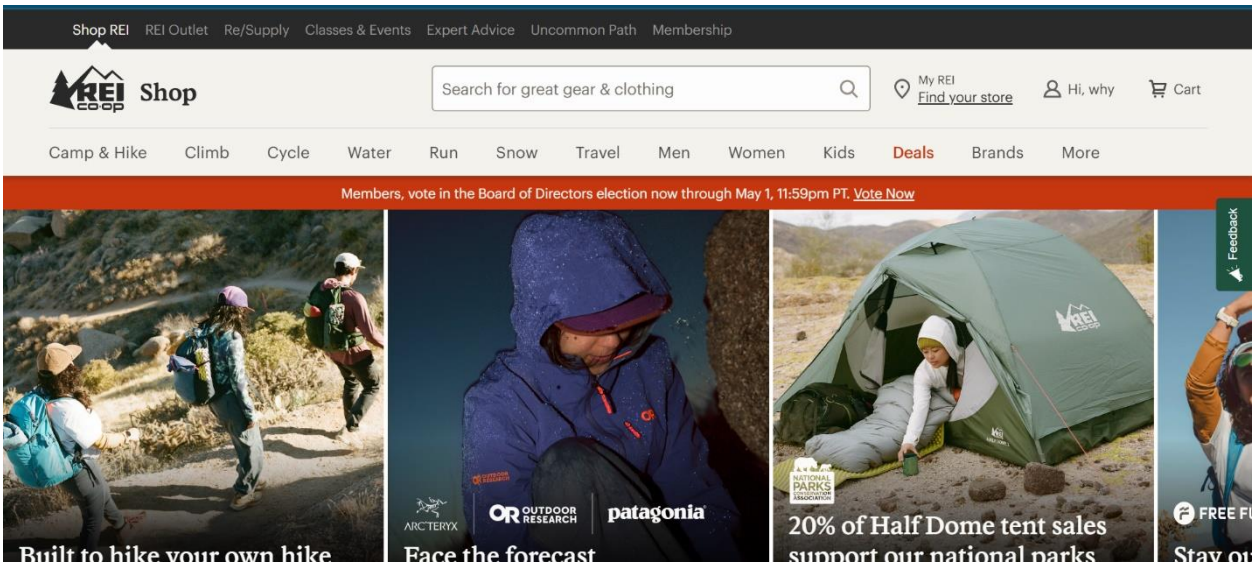
Vulnerability Title:

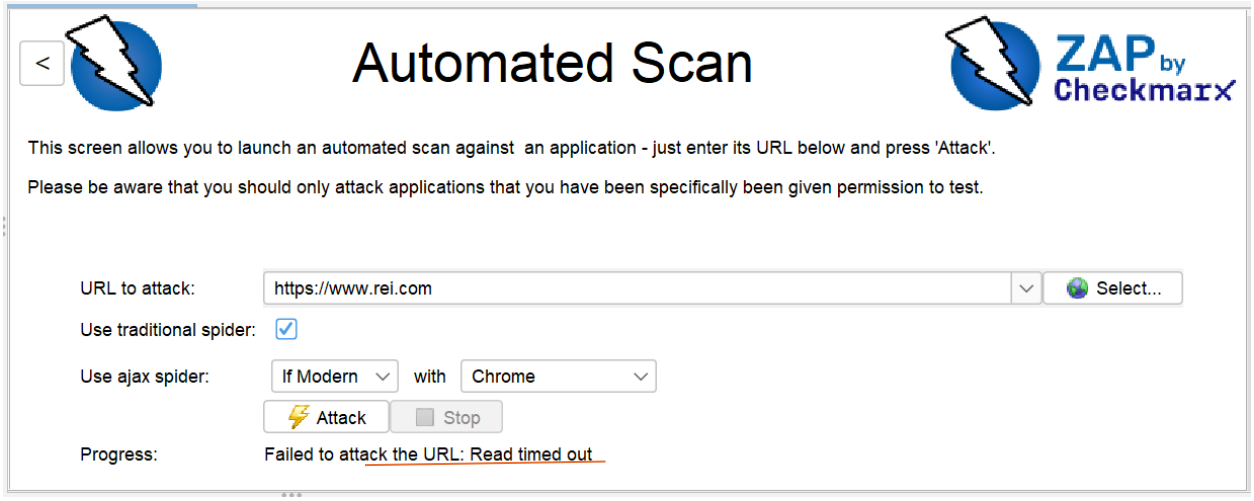
Insecure Direct Object Reference (IDOR) Vulnerability

Vulnerability Description:

I found this program on the hackerone Bug hunting website. The website hosted at https://www.rei.com. **Insecure Direct Object Reference (IDOR)** occurs when an application exposes a reference to an internal object such as a user ID, file name, or record number, and fails to properly authorize the requesting user. An attacker can manipulate the value of such references to access data belonging to other users.

In this test, I attempted to identify IDOR vulnerabilities in the application by modifying object references (such as user IDs or order numbers) in various authenticated and unauthenticated requests. The goal was to access unauthorized resources or data by changing identifiers passed in URL parameters, form fields, or JSON bodies.





After attempting an automated vulnerability scan using OWASP ZAP, the scan failed due to a timeout while trying to reach the target URL (https://www.rei.com). This may have been caused by network restrictions, protective mechanisms like a WAF, or bot detection measures.

Due to the inability to complete the scan, I proceeded with manual testing to identify potential vulnerabilities. As part of this process, I focused on testing for Insecure Direct Object Reference (IDOR) issues, which do not require automated scanning to identify and can often be discovered through manual inspection of URL parameters and access control behavior.

Affected Components:

- User/account-specific endpoints
- Request parameters and paths involving identifiers (e.g., user_id, profile_id, account_id)

Impact Assessment:

- **Risk Level:** High (Based on website reaction)

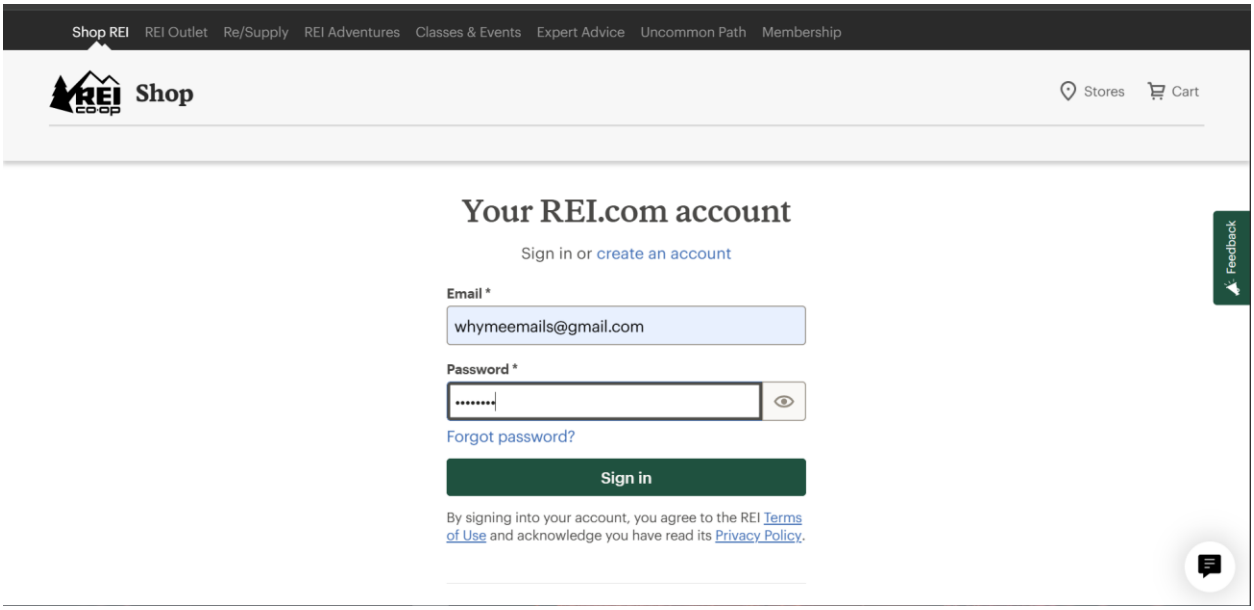
If vulnerable, IDOR could allow attackers to:

- Access to other users' personal or sensitive information
- Edit or delete resources belonging to other users
- Perform unauthorized actions using predictable object references

However, in this case, all attempts to manipulate object references were either rejected by the server or redirected to error pages. No unauthorized access or information disclosure occurred. This indicates proper authorization checks are in place for object-level access.

Steps to Reproduce:

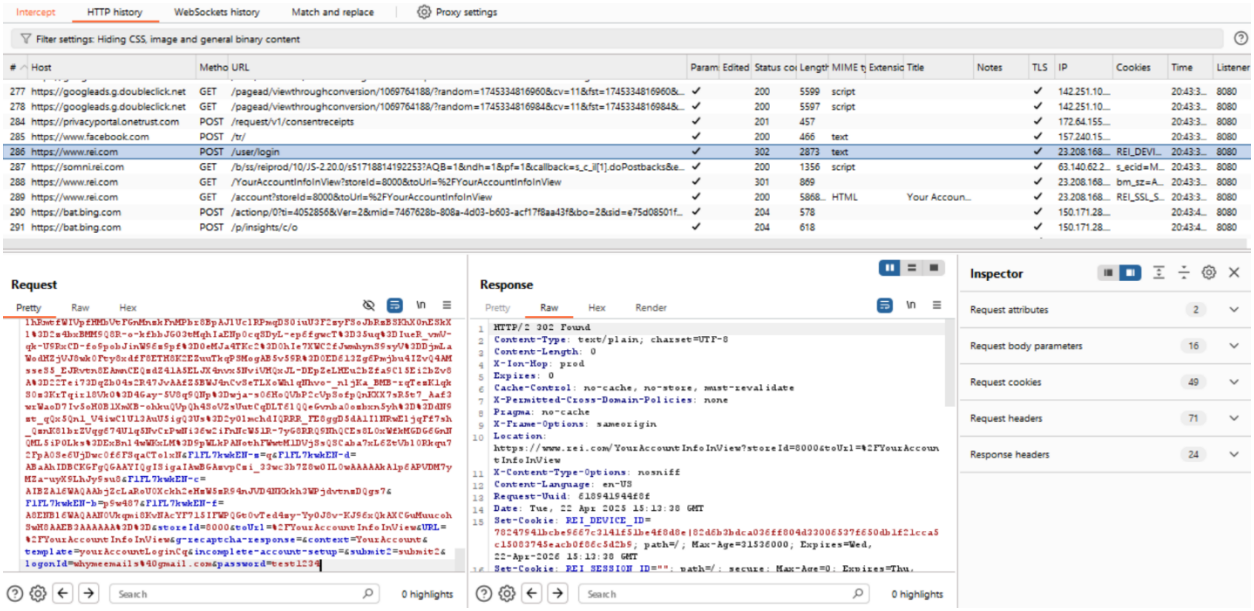
1. Logged into the application with a test user account and captured requests using **Burp Suite**.



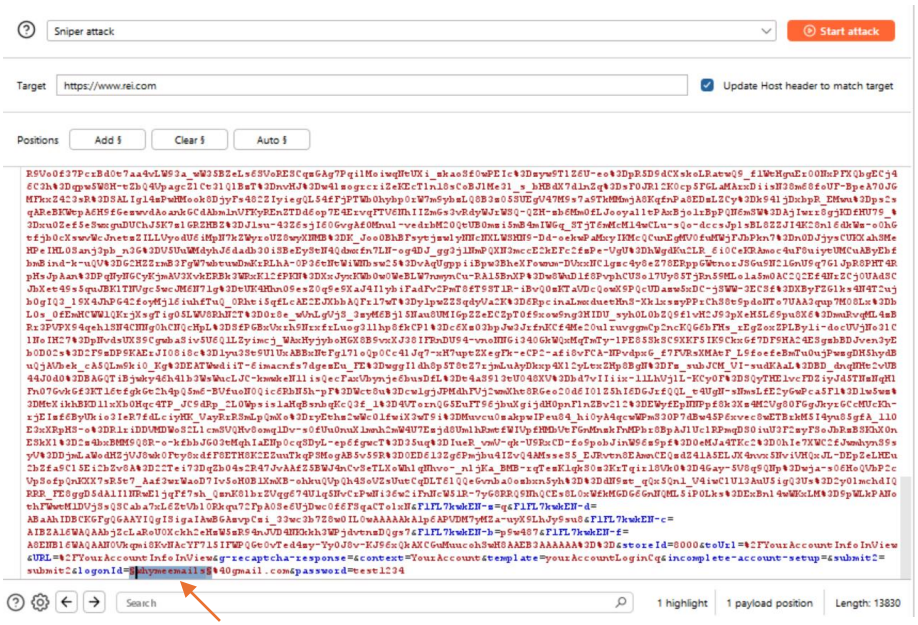
2. Identified and targeted endpoints containing identifiers such as user_id, profile_id, order_id, etc.
3. Sent requests to **Burp Repeater** and manually modified the ID values to those of other users or sequential IDs (e.g., 123 → 124, 9999, etc.).
4. Observed the response to determine whether unauthorized data was returned.
5. Repeated the process for both GET and POST requests across multiple parts of the application.
6. All attempts returned responses such as:
    - 403 Forbidden
    - 401 Unauthorized
    - Redirection to login or generic error pages
    - No exposure of other users' data

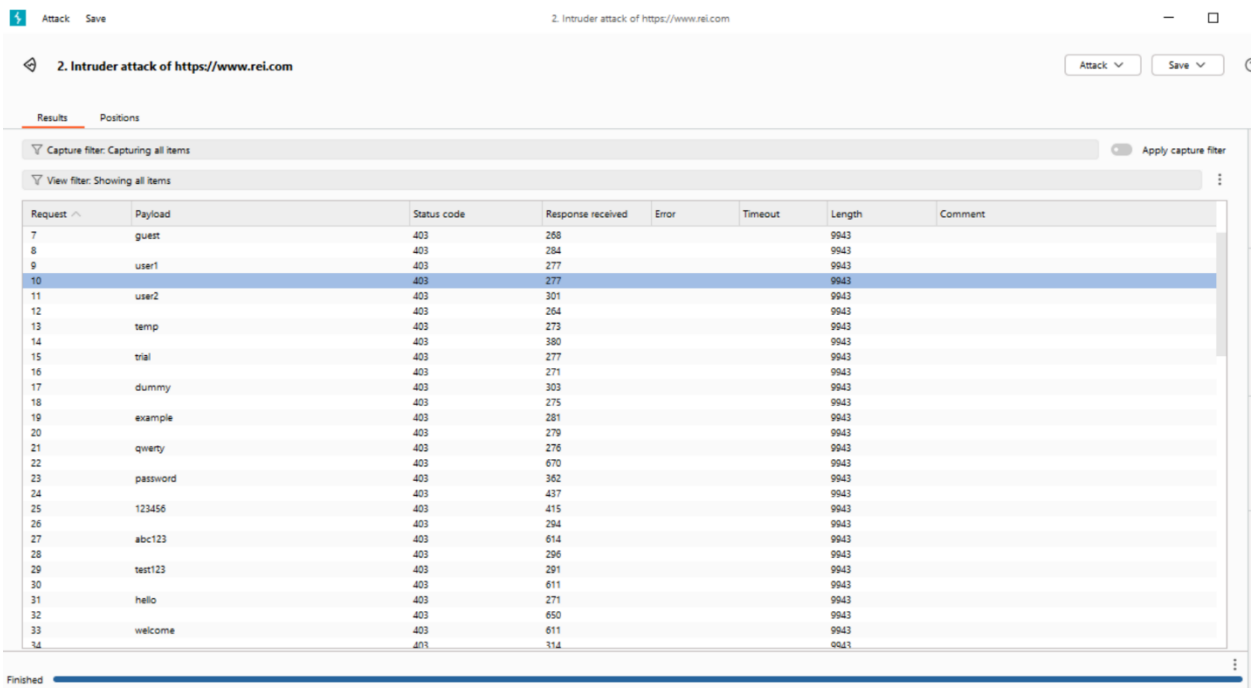
Proof of Concept (PoC):

To test for **Insecure Direct Object Reference (IDOR)**, I used **Burp Suite** to intercept and modify parameters during login and post-login account access requests. The goal was to check if changing predictable identifiers like logonId, user_id, or account_id could allow unauthorized access to another user's data.



1. I captured the login request using Burp Proxy and sent it to Burp Repeater for manual testing.
2. The original request included:
   logonId=whymeemails@gmail.com and password=test1234
3. I modified the logonId parameter to different values, such as:
   - logonId=admin@rei.com
   - logonId=user1@rei.com
   - logonId=samplecustomer@rei.com
4. The rest of the request remained unchanged to check if the application would accept the alternate identity without proper verification.
5. Each request was submitted, and I monitored the response code, message, and content.
6. All responses returned standard login error messages, and none resulted in successful login or unauthorized session tokens.
7. There was no change in response content length or structure that would indicate user enumeration or access to other accounts.


Sniper attack
Start attack
Target
https://www.rei.com
Update Host header to match target
Positions
Add §
Clear §
Auto §
1 highlight
1 payload position
Length: 13830


Attack
Save
2. Intruder attack of https://www.rei.com
Results
Positions
Capture filter: Capturing all items
Apply capture filter
View filter: Showing all items
Request
Payload
Status code
Response received
Error
Timeout
Length
Comment
7 guest 403 268 9943
8 403 284 9943
9 user1 403 277 9943
10 403 277 9943
11 user2 403 301 9943
12 403 264 9943
13 temp 403 273 9943
14 403 380 9943
15 trial 403 277 9943
16 403 271 9943
17 dummy 403 303 9943
18 403 275 9943
19 example 403 281 9943
20 403 279 9943
21 qwerty 403 276 9943
22 403 670 9943
23 password 403 362 9943
24 403 437 9943
25 123456 403 415 9943
26 403 294 9943
27 abc123 403 614 9943
28 403 296 9943
29 test123 403 291 9943
30 403 611 9943
31 hello 403 271 9943
32 403 650 9943
33 welcome 403 611 9943
34 403 314 9943
Finished

Proposed Mitigation or Fix:

Although no vulnerability was found, the following security practices should be maintained:

1. Implement object-level authorization checks for every sensitive action or resource.
2. Never rely solely on client-side controls (e.g., hiding IDs or buttons).
3. Use unpredictable identifiers such as UUIDs instead of incremental integers.
4. Validate access on the server side before processing any object-related request.
5. Log and monitor access to sensitive resources for anomalies.
6. Conduct regular access control reviews and penetration tests to confirm coverage.

Conclusion:

An IDOR vulnerability assessment was performed using Burp Suite by manipulating object identifiers across multiple endpoints. All unauthorized attempts were blocked by the application. This indicates that proper access control and object-level authorization checks are in place, preventing insecure access to resources.