

EXECUTIVE SUMMARY

Target: <http://testphp.vulnweb.com>

Scan ID: AG-0DAFB3ED

Scan Date: 2026-02-08 00:39:37

Findings: 1 vulnerabilities detected

- Detected 1 security issue(s) requiring attention.
- Immediate remediation recommended for critical findings.
- Review each finding below for detailed impact analysis.
- Prioritize fixes based on severity and exploitability.

DETAILED FINDINGS

Finding #1: Insecure Direct Object Reference (IDOR)

HIGH

CWE: CWE-639

CVSS Score: 8.6 (High)

CVSS Score: 8.6 (High)

THREAT SCORE: **75/100**

A horizontal bar consisting of a yellow segment on the left and a grey segment on the right, representing a threat score of 75 out of 100.

Description:

- Application exposes internal object references without authorization checks.
- Attackers can access resources belonging to other users.
- Object IDs are predictable and not properly validated.

Impact:

- Unauthorized access to other users' data.
- Privacy breach affecting multiple users.
- Potential for mass data harvesting.
- Regulatory compliance violations (GDPR, etc.).

Forensic Analysis

Method: GET | Param: user_id

URL: /api/v1/user/1005

Analysis: The 'user_id' parameter is sequential and lacks authorization checks.

Table 1: Payload Decomposition

Component	Value	Technical Function
Target ID	1005	Direct reference to a specific database record ID.
Access Check	Missing	The application fails to verify if the requester owns ID 1005.
Result	200 OK	Server returns data for the unauthorized object ID.

Payload Specifications:

Vector Category: Insecure Direct Object Reference

Raw Payload: 1005

```
Encoded: 1005
Encoding Type: None
```

Reproduction Command:

```
curl -X GET 'http://target/api/v1/user/1005' -H 'Authorization: Bearer <attacker_token>
```

HTTP Traffic Snapshot:

Request:

```
GET /api/v1/user/1005 HTTP/1.1
Host: target
Authorization: Bearer <attacker_token>
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{"id": 1005, "username": "victim_user", "email": "victim@corp.com", "role": "admin"}
```

Remediation:

- Implement proper authorization checks on all resource access.
- Use indirect references or UUIDs instead of sequential IDs.
- Validate user permissions before returning data.
- Log and monitor access patterns for anomalies.

Recommended Code Fix:

```
# VULNERABLE CODE:
@app.get("/user/{user_id}")
def get_user(user_id: int):
    return db.get_user(user_id)

# SECURE CODE:
@app.get("/user/{user_id}")
def get_user(user_id: int, current_user: User):
    if user_id != current_user.id and not current_user.is_admin:
        raise HTTPException(403, "Access denied")
    return db.get_user(user_id)
```

SCAN TIMELINE

- [Orchestrator] TARGET_ACQUIRED - 2026-02-08 00:36:10
- [Sigma] JOB_ASSIGNED - 2026-02-08 00:36:10
- [Beta] LOG - 2026-02-08 00:36:10
- [Gamma] VULN_CONFIRMED - 2026-02-08 00:36:10

- [Kappa] GI5_LOG - 2026-02-08 00:36:10