Perfect — this one is **WSTG-ATHZ-04: Testing for Insecure Direct Object References (IDOR)** 🔎

**🧾 Testing for Insecure Direct Object References (IDOR)**

**1. What is IDOR?**

* An **authorization flaw** where an application exposes internal objects (IDs, filenames, keys) and does not properly verify **ownership**.
* Example: A user changes a parameter (like id=1001 → id=1002) to access another user’s data.

**2. Common Places Where IDOR Appears**

* **URL Parameters**
* /account/details?id=1001
* **Form Inputs**
* POST /transfer
* amount=500&toAccount=2002
* **Cookies**
* Cookie: user\_id=1001
* **Headers**
* X-User-ID: 1001
* **APIs**
* GET /api/user/1001/profile

**3. Testing Approach**

**🔹 Step 1: Identify Direct Object References**

* Look for **numeric IDs**, **UUIDs**, **file paths**, or **keys** in requests.

**🔹 Step 2: Tamper with Values**

* Change id=1001 → id=1002.
* Modify file paths: /files/report1.pdf → /files/report2.pdf.

**🔹 Step 3: Observe Server Response**

* ✅ If application gives **someone else’s data** → IDOR exists.
* ❌ If it returns **403 Forbidden** or error → access control in place.

**🔹 Step 4: Use Burp Suite Tools**

* **Proxy + Repeater** → intercept & modify ID values.
* **Intruder** → brute force ID ranges.
* **Comparer** → analyze differences in responses.

**4. Examples**

**Example 1: User Account**

GET /user/profile?id=101

Change → /user/profile?id=102  
👉 If you see another user’s profile → IDOR.

**Example 2: File Download**

GET /download?file=invoice101.pdf

Change → invoice102.pdf  
👉 If accessible without restriction → IDOR.

**Example 3: API Request**

GET /api/v1/orders/2001

Authorization: Bearer <user1\_token>

Change order ID → /api/v1/orders/2002  
👉 If User1 can see User2’s order → IDOR.

**5. Tools for Automation**

* **Burp Suite Intruder** → fuzz IDs (1000-2000).
* **ffuf / wfuzz** → brute force parameters in API endpoints.
* **Autorize (Burp Extension)** → check access control across roles.

**6. Impact of IDOR**

* Data leakage (PII, financials, health records).
* Unauthorized actions (fund transfer, delete files).
* Account takeover in worst cases.

**7. Mitigation**

* Enforce **server-side authorization checks**.
* Avoid predictable IDs (use random/UUIDs).
* Implement **object-level ACLs** (access control lists).
* Apply **principle of least privilege**.

✅ **Summary**:  
IDOR occurs when developers rely on **client-supplied identifiers** without validating ownership.  
Testing involves **modifying parameters, fuzzing IDs, and comparing responses**. Burp Suite is the main tool here, alongside automation like ffuf and extensions such as Autorize.

👉 Do you want me to also give you a **Burp Suite lab-style hands-on workflow** (step-by-step intercept → modify → confirm IDOR) so you can practice this in WSTG labs?