Great, let’s move into **WSTG-ATHZ-03: Testing for Privilege Escalation** 🚀

**🧾 Testing for Privilege Escalation**

**1. Why It Matters**

Privilege escalation occurs when a **low-privileged user** (e.g., normal account) gains **higher privileges** (e.g., admin, manager).

* **Vertical Escalation** → move from user → admin.
* **Horizontal Escalation** → access data/resources of another user with same privilege level.

**2. Common Scenarios**

**🔹 Parameter/Role Manipulation**

* Request body:
* POST /updateProfile
* Content-Type: application/json
* {
* "username": "user1",
* "role": "user"
* }

Modify → "role": "admin".

**🔹 Cookie Tampering**

* Cookie:
* role=user

Modify → role=admin.

**🔹 Insecure Direct Object References (IDOR)**

* Accessing another user’s data by changing ID:
* /account/view?id=2002

→ Should only see your own.

**🔹 Functionality Misuse**

* Low-privileged user can call **admin-only APIs** directly.  
  Example:
* POST /deleteUser?id=55

→ Works without being admin.

**🔹 Misconfigured Access Controls**

* Web server allows access to /admin/ directory because no proper server-side check is enforced.

**🔹 Business Logic Flaws**

* Example: During registration, change role field from user → admin.

**3. How to Test**

**🛠 Manual Testing (Burp Suite)**

* **Proxy & Repeater:** Capture requests, modify parameters like role, user\_id, etc.
* **Intruder:** Brute force role values, IDs.
* **Compare Responses:** 403 vs 200 (check for leakage).

**🛠 Automated Tools**

* **Nmap NSE (http-methods, http-auth-finder)** → detect weak auth.
* **wfuzz/ffuf** → brute force hidden admin endpoints.
* **Burp Suite Pro** → Authz Scanner can automate privilege escalation checks.

**4. Examples**

**Example 1: Cookie Role Tampering**

Before:

Cookie: session=abc123; role=user

After:

Cookie: session=abc123; role=admin

If server does not verify → escalated privileges.

**Example 2: Access Hidden Admin Function**

GET /admin/settings HTTP/1.1

Host: target.com

Cookie: session=xyz

If user with normal role can access → escalation vulnerability.

**Example 3: API Exploit**

POST /api/v1/updateUser

{

"id": 102,

"role": "admin"

}

If accepted, user becomes admin.

**5. Impact**

* Full **account takeover**.
* **Sensitive data exposure** (other users, financial info).
* **System compromise** if admin privileges are abused.

**6. Mitigation**

* Always enforce **role checks on server side** (not client).
* Use **RBAC (Role-Based Access Control)** properly.
* Apply **least privilege** principle.
* Protect cookies with **signatures/encryption** (JWT, HMAC).
* Ensure APIs validate **who is making the request** (user context).

✅ **Summary**:

Privilege Escalation happens when users can manipulate **roles, cookies, IDs, or API requests** to gain higher privileges. Testing involves tampering with **parameters, cookies, and hidden endpoints** using **Burp Suite, ffuf, and manual fuzzing**.

👉 Do you want me to prepare a **Burp Suite practical step-by-step flow** (intercept → modify → test access → confirm escalation) so you can apply it in your WSTG labs?