Great one — this is **WSTG-SESS-10: Testing for Session Puzzling**.  
It’s less commonly understood, but very dangerous if implemented badly. Let’s break it down clearly 👇

**🧪 Testing for Session Puzzling**

**🔎 What is Session Puzzling?**

* Occurs when an application **reuses or merges multiple session tokens** incorrectly.
* Attacker exploits confusion in session management (e.g., mixing **unauthenticated** and **authenticated** sessions).
* Leads to **privilege escalation**, **bypassing authentication**, or **unauthorized access**.

Think of it like the app saying:  
👉 *“This user is partially logged in with one session and fully logged in with another, so let’s just combine them.”*

**⚡ Example Scenarios**

1. **Anonymous → Authenticated Merge**
   * User browses anonymously (session A).
   * Then logs in (session B).
   * App **doesn’t destroy session A** → attacker can hijack A to inherit B’s privileges.
2. **Multiple Session Cookies Accepted**
   * App accepts more than one Set-Cookie.
   * Browser might send both → app gets confused and picks wrong one.
3. **Weak Session Renewal**
   * After login, session cookie is not regenerated.
   * Attacker who knew the old session ID can still use it.
4. **Privilege Escalation by Session Role Mixing**
   * Same session token used across multiple privilege levels.
   * Example: a “guest” session suddenly becomes “admin” after puzzle merge.

**🔧 How to Test (Practical Steps)**

**With Burp Suite:**

1. **Start Anonymous Session**
   * Browse app without logging in. Capture session cookie (e.g., PHPSESSID=xyz).
2. **Login Normally**
   * Log in → capture the new session cookie (if issued).
3. **Try Mixing**
   * Replay requests with **old session cookie**.
   * Try sending both cookies in the request.
   * Observe if old session inherits authenticated privileges.
4. **Check Session Regeneration**
   * If the session ID **remains the same** before and after login → ❌ vulnerable.
5. **Check Multiple Set-Cookie Headers**
   * If app sets 2+ cookies with different roles → attempt sending combinations.

**✅ Mitigations**

* **Regenerate session IDs** after login or privilege escalation.
* **Invalidate old sessions** on login/logout.
* Ensure app uses **only one active session cookie** at a time.
* Enforce **strict separation** between anonymous & authenticated sessions.
* Set **HttpOnly, Secure, SameSite** cookie flags.

⚡ **Summary:**  
Session puzzling = when session state gets **confused or merged** → attacker escalates access.  
Testing = check if session IDs change properly, if multiple cookies are allowed, or if old session IDs remain valid.  
Secure design = regenerate, invalidate, and strictly separate sessions.

👉 Do you want me to also prepare a **realistic Burp Suite lab workflow** (step-by-step with repeater, proxy, intruder) to simulate session puzzling attacks, just like I did for CSRF & logout testing?