Great, let’s dive into **Testing for Vulnerable “Remember Password” Functionality** 🛑 — another **WSTG authentication-related** test.

**🧾 Testing for Vulnerable “Remember Password”**

**1. Why It Matters**

The “Remember Password” / “Keep me logged in” feature is convenient but often insecure.  
Poor implementations can:

* Store **plaintext passwords** in cookies or local storage.
* Use **predictable tokens** that attackers can steal and reuse.
* Keep sessions alive too long, exposing accounts if a machine is shared or stolen.

**2. What to Test**

**🔸 Cookie Storage**

* Check cookies via browser DevTools or **Burp Suite → Proxy → HTTP history**.
* Look for:
  + Password stored in cleartext (password=123456).
  + Base64-encoded credentials (dXNlcjpwYXNz = user:pass).
  + Long-lived session cookies without **expiry flags**.

**🔸 Local Storage / Session Storage**

* Inspect localStorage or sessionStorage.
* If credentials or tokens are stored insecurely, an **XSS attack** could steal them.

**🔸 Token Predictability**

* Some apps store a static token like rememberme=abcd1234.
* Use **Burp Intruder** to fuzz token values and check if you can log in as another user.

**🔸 Session Reuse**

* After logout, check if the remember-me cookie still works.
* If yes → attacker with stolen cookie can bypass logout.

**🔸 Transmission Security**

* Confirm the cookie/token is only transmitted over **HTTPS**.
* Missing Secure flag → exposed on HTTP.
* Missing HttpOnly flag → exposed to JavaScript.

**3. Burp Suite Testing Approach**

1. **Login with "Remember me" checked.**
   * Intercept with Burp → look at new cookies or headers.
2. **Analyze Cookie / Storage**
   * Check if it contains **plaintext or encoded passwords**.
   * Decode any base64 strings in Burp Decoder.
3. **Test Session Persistence**
   * Close browser → reopen → visit app.
   * Does the cookie still authenticate you? For how long?
4. **Replay with Another User**
   * Try stealing the cookie (remember\_token) and replay in Burp → see if access is granted.

**4. Example Vulnerable Cases**

* **Bad Cookie Storage**
* Set-Cookie: rememberme=admin:password123

→ attacker with cookie = full account access.

* **Predictable Tokens**
* remember\_token=1001
* remember\_token=1002

→ sequential tokens allow user enumeration.

* **No Expiration**
* Set-Cookie: remember\_token=xyz; Expires=2099-12-31

→ token stays valid indefinitely.

1. **Mitigations**
2. Never store passwords in cookies or local storage.
3. Use **strong, random tokens** bound to a device and user.
4. Enforce **short expiration times** (e.g., 2 weeks).
5. Always use **Secure** + **HttpOnly** flags.
6. Revoke tokens on logout, password change, or suspicious activity.

✅ **Summary**:

When testing "Remember Password," focus on **cookie inspection, storage methods, token predictability, session persistence, and security flags**. Burp Suite helps by capturing cookies, replaying them, and checking how resilient they are to tampering or theft.

👉 Do you want me to create a **challenge lab** for this, where you intercept a vulnerable “remember me” cookie in Burp, decode it, and use it to log in without a password?