Perfect — this is **WSTG-SESS-06: Testing for Cross-Site Request Forgery (CSRF)**.  
Let’s break it down clearly 👇

**🧪 Testing for Cross-Site Request Forgery (CSRF)**

**🔎 What is CSRF?**

* **CSRF** is when an attacker tricks a victim into making an **unintended, authenticated request** to a web application.
* The app **trusts the victim’s browser** (because cookies/session tokens are sent automatically) → action executes **without victim’s intent**.

Example:  
If a bank site doesn’t verify requests properly, attacker can embed a hidden form that silently sends:

<form action="https://bank.com/transfer" method="POST">

<input type="hidden" name="to" value="attacker\_account">

<input type="hidden" name="amount" value="1000">

<input type="submit">

</form>

<script>document.forms[0].submit();</script>

If the victim is logged into bank.com, their browser **auto-sends cookies** → money transferred. 🚨

**🔧 How to Test for CSRF**

**1. Identify State-Changing Requests**

* Use **Burp Suite Proxy** to monitor requests.
* Look for actions like:
  + Change password
  + Transfer funds
  + Update profile
  + Delete records

⚠️ GET requests that change state = suspicious.

**2. Check for Anti-CSRF Protections**

* Look in requests for:
  + **CSRF token** (random unique string per request/session).
  + Double-submit cookies.
  + SameSite cookies.
* If **no protection** → likely vulnerable.

**3. Replay the Request Without Token**

* Copy the request in Burp → **send via Repeater**.
* Remove/modify CSRF token field.
* If request still succeeds → 🔴 CSRF possible.

**4. Craft a Proof-of-Concept (PoC) CSRF Attack (Lab Only)**

Example HTML PoC:

<html>

<body>

<form action="https://victim.com/change-email" method="POST">

<input type="hidden" name="email" value="attacker@evil.com">

</form>

<script>document.forms[0].submit();</script>

</body>

</html>

Open this file in a browser while logged into victim.com.

* If victim’s email changes → site vulnerable.

**5. Burp Suite CSRF PoC Generator**

* Burp → **Engagement tools → Generate CSRF PoC**.
* Creates a ready-to-use HTML snippet for testing.

**✅ Mitigations**

* Use **unique CSRF tokens** tied to session & request.
* Reject **state-changing GET requests**.
* Use **SameSite=Lax or Strict** cookie flag.
* Require **re-authentication** for critical actions (e.g., money transfer).
* Add **CAPTCHA** or user interaction confirmation.

⚡ **Summary:**

* CSRF = attacker abuses victim’s authenticated session.
* Test by removing CSRF tokens & replaying request.
* Burp has built-in CSRF PoC generator.
* Mitigate with CSRF tokens, SameSite cookies, re-authentication.

👉 Do you want me to also give you a **Burp Suite lab workflow** (step-by-step, like you did for session fixation) so you can test CSRF in practice?