Got it 👍 Here are **precise WSTG-style notes for “Testing WebSockets” (WSTG-CLNT-11)**.

**📝 Testing WebSockets**

**🎯 Purpose**

To test the **security of WebSocket communications**, which provide **persistent, bidirectional connections** between client and server (ws:// or wss://).

* If insecure, attackers can intercept, inject, or manipulate data.

**⚡ Common Risks**

1. **Unencrypted Transport** (ws://) → MITM attacks.
2. **Missing Authentication** → anyone can connect.
3. **Weak Origin Checks** → attacker site hijacks session.
4. **Input Manipulation** → messages lack server-side validation.
5. **Session Hijacking** → tokens stored in client-side storage stolen via XSS.
6. **DoS Risk** → unlimited messages, oversized payloads.

**🔍 How to Test**

**1. Identify WebSocket Endpoints**

* Look for ws:// or wss:// in:
  + Browser DevTools → Network → WS.
  + JavaScript files.

**2. Transport Security**

* Ensure **wss:// (TLS)** is used.
* Check for **valid certificates**.

**3. Authentication & Authorization**

* Verify WebSocket connection requires **authenticated session**.
* Test with invalid/expired tokens.
* Attempt privilege escalation by replaying tokens.

**4. Origin Validation**

* Check if server restricts Origin header.
* Try connecting from attacker-controlled domain.

**5. Input Validation**

* Intercept traffic with Burp/ZAP.
* Send malformed JSON, oversized payloads, or injections (SQLi, XSS).
* Look for error leakage or bypass of server checks.

**6. Message Reuse / Replay**

* Capture valid WebSocket message → resend → check if actions repeat (e.g., duplicate order, transfer).

**7. Session Handling**

* Test if WebSocket session is terminated after **logout**.
* If still active → session fixation risk.

**🛡️ Mitigation**

* Use **wss:// only** (never plaintext ws://).
* Authenticate each connection with **secure tokens (short-lived JWT, cookies with SameSite/HttpOnly)**.
* Validate **Origin and Sec-WebSocket-Protocol** headers.
* Apply **input validation & rate limiting** on messages.
* Close sockets on **logout / session expiry**.
* Monitor abnormal traffic (DoS protection).

✅ **Key Takeaway:**  
WebSockets = always-on communication channel.  
Treat every message as **untrusted input**, enforce **TLS, authentication, and server-side validation**.

👉 Do you want me to also prepare a **step-by-step Burp Suite workflow** for testing WebSocket security (interception, fuzzing, replay)?