Perfect 👌 — let’s cover **Testing for Browser Cache Weaknesses** as per **WSTG (Web Security Testing Guide)**.

**🧾 Testing for Browser Cache Weaknesses**

**1. Why It Matters**

When web applications don’t properly configure **caching headers**, sensitive information may be stored in the **browser’s cache**.  
Attackers with **local access** (shared computers, stolen devices) or through **cache poisoning attacks** can retrieve:

* Account details
* Session tokens
* Credit card info
* Private documents

**2. What to Test**

**🔹 Check HTTP Response Headers**

Using **Burp Suite**, **curl**, or browser DevTools, inspect sensitive pages (e.g., account, checkout, dashboard).

Headers to look for:

* **No security controls** → risky
* Cache-Control: public
* **Best practice** → secure
* Cache-Control: no-store, no-cache, must-revalidate
* Pragma: no-cache
* Expires: 0

**🔹 Browser Back/Forward Navigation**

* Login, view sensitive info (profile, payment).
* Logout.
* Hit **Back button** → does private data still appear?
* If yes → cache not cleared properly.

**🔹 Private vs Shared Cache**

* Test from **incognito mode** and **normal mode**.
* Sensitive data must **never** be cached in shared/public caches (e.g., proxy servers).

**🔹 File Downloads**

* Download bank statements, invoices, or reports.
* Are they cached in /tmp or browser cache directories?
* Can another user on the same system read them?

**3. Tools & Commands**

**🛠 Burp Suite**

* Intercept HTTPS responses → check for **Cache-Control** headers.
* Look for public or missing directives.

**🛠 Curl Example**

curl -I https://target.com/account

Sample insecure response:

HTTP/1.1 200 OK

Cache-Control: public, max-age=3600

**🛠 Manual Browser Testing**

* Login → open DevTools → **Network tab**.
* Refresh → check headers for caching misconfigurations.

**4. Example Vulnerable Cases**

1. **Cached Bank Statement**
2. Cache-Control: public, max-age=3600

→ attacker using shared kiosk can retrieve previous user’s statement.

1. **Logout Bypass via Back Button**
   * User logs out, but pressing **Back** reloads cached page with private info.

**5. Mitigations**

✅ Set correct headers for sensitive responses:

Cache-Control: no-store, no-cache, must-revalidate

Pragma: no-cache

Expires: 0

✅ Use Cache-Control: private for personal data if caching is necessary.  
✅ Always **invalidate session tokens** on logout.  
✅ Avoid storing sensitive documents in predictable cache locations.

✅ **Summary**:  
Testing for browser cache weaknesses means checking whether **sensitive data is cached** in browsers or shared proxies. Use **Burp Suite / curl** to inspect headers, and **manual testing** with logout + back button. If caching headers are missing or misconfigured → attacker could retrieve sensitive info.

👉 Do you want me to also design a **mini lab exercise** for you, where you’ll test a vulnerable app with Burp and see private data still available after logout?