Perfect, now let’s tackle **Testing for Weaker Authentication in an Alternative Channel** — another **WSTG** test case.

**🧾 Testing for Weaker Authentication in an Alternative Channel**

**1. Why It Matters**

Web applications sometimes provide **multiple channels** for user access (e.g., web portal, mobile app, API, legacy interface).

* If **one channel enforces weaker authentication**, an attacker can bypass stronger protections by abusing the weaker entry point.

👉 Example:

* Web login requires **MFA + strong password**.
* Mobile API only requires **username & password** without MFA.
* Attacker targets the mobile API → bypass MFA.

**2. What to Test**

**🔹 Identify All Channels**

* Web login page.
* Mobile applications (analyze APK/IPA).
* APIs (REST, SOAP, GraphQL).
* Legacy portals (admin interfaces, FTP, SSH).

**🔹 Compare Authentication Requirements**

* Is MFA enforced everywhere?
* Is password complexity enforced equally?
* Are lockout/rate-limiting protections consistent?

**🔹 Session Handling**

* Are tokens/cookies shared across channels?
* Can a session started in a weaker channel access a stronger one?

**3. How to Test (Methodology)**

**🛠 Step 1: Enumerate Channels**

* Use **Burp Suite** proxy with mobile apps.
* Use **API discovery tools** like postman, mitmproxy, OWASP ZAP, or nmap for API endpoints.

**🛠 Step 2: Replay Authentication Requests**

* Intercept login request from each channel.
* Compare headers, cookies, MFA requirements.

**🛠 Step 3: Try Downgrade**

* If the web app enforces MFA, attempt login via API/mobile endpoint → does it skip MFA?
* If strong password policy on web, test a weak password via API/mobile.

**🛠 Step 4: Token Reuse**

* Login via weak channel → capture session token.
* Try using that token in the stronger web interface.

**4. Example Attack Scenarios**

* **Mobile App Bypass**:  
  Web requires CAPTCHA + MFA. Mobile login only asks for username/password → attacker brute-forces via mobile API.
* **Legacy Endpoint**:  
  Web login enforces strong password policy. Legacy SOAP endpoint still accepts admin:admin.
* **Session Hijack**:  
  Mobile app generates a session cookie without Secure/HttpOnly flags. Attacker steals it and reuses on the main site.

**5. Tools**

* **Burp Suite** – intercept & replay mobile/API traffic.
* **Mitmproxy / Fiddler** – sniff traffic from mobile apps.
* **Nmap** – detect open legacy services (FTP, Telnet).
* **Postman / curl** – replay API authentication requests.

**6. Mitigations**

* Apply **uniform authentication controls across all channels**.
* Enforce **MFA** consistently.
* Apply the **same password complexity policy** everywhere.
* Ensure **session tokens** are equally protected.
* Monitor for **unusual login patterns** (API brute force, mobile bypass).

✅ **Summary**:  
This test ensures that **alternative channels (API, mobile, legacy)** don’t weaken security compared to the main web app. An attacker will always go for the **weakest door**, so all authentication channels must align.

👉 Do you want me to also build a **Burp Suite mobile API lab exercise** showing how to intercept and test for weaker authentication in practice?