Good one 👍 Let’s prepare **WSTG notes for “Test Defenses Against Application Misuse” (WSTG-BUSL-10)**.

**📝 Notes: Test Defenses Against Application Misuse**

**🎯 Purpose**

To check whether the application has **controls that prevent abuse of its features**.  
Even if a feature is working as designed, attackers may **misuse it at scale** (automation, scraping, spam, DoS, fraud).

**⚡ Common Attack Scenarios**

1. **Credential Stuffing / Brute Force**
   * Automated login attempts without lockouts → account takeover.
2. **Mass Account Creation**
   * Attackers register thousands of fake accounts → spam or resource abuse.
3. **Web Scraping**
   * Automated bots harvest pricing, product catalogs, or personal data.
4. **Denial of Service (DoS)**
   * Automated requests to resource-heavy endpoints → degrade service.
5. **Feature Abuse**
   * Using messaging features to spam/phish users.
   * Using password reset repeatedly to harass victims.

**🔍 How to Test**

1. **Check for Automation Resistance**
   * Try scripting requests with curl, Burp Intruder, OWASP ZAP.
   * Is CAPTCHA, rate-limiting, or IP throttling in place?
2. **Mass Request Simulation**
   * Attempt 50+ account registrations or password resets.
3. **Check for Abuse-Prone Features**
   * Messaging, file uploads, search, exports.
   * Can they be misused for spam/data exfiltration?
4. **Examine API Endpoints**
   * Are APIs accessible without proper throttling or authentication?
5. **Look for Logging & Alerting**
   * Does the app detect and respond to abuse attempts?

**🛡️ Mitigation**

* Implement **rate limiting** & **throttling** (per user, per IP, per device).
* Enforce **CAPTCHA** for high-risk functions (login, registration, password reset).
* Deploy **bot detection** (behavioral analysis, device fingerprinting).
* Apply **monitoring & anomaly detection** for abnormal usage patterns.
* Limit exposure of **resource-heavy endpoints**.
* Enforce **account verification** (email/SMS/2FA) to prevent fake accounts.

✅ **Key Takeaway:**  
Even legitimate features can be **weaponized by attackers**. Applications need strong **defenses against misuse** such as rate-limiting, CAPTCHA, monitoring, and anomaly detection.

👉 Do you want me to also build a **practical tester’s misuse checklist** (e.g., try 100 logins, 50 password resets, scraping product pages, etc.) so you can directly apply it in assessments?