Got it 👍 let’s prepare **WSTG notes for “Test Upload of Unexpected File Types” (WSTG-BUSL-08)**.

**📝 Notes: Test Upload of Unexpected File Types**

**🎯 Purpose**

To verify whether the application enforces **strict file type restrictions** during upload.  
If checks are weak, attackers may upload **malicious files** (scripts, executables, or disguised payloads) leading to:

* Remote Code Execution (RCE)
* Malware distribution
* Information disclosure

**⚡ Common Attack Scenarios**

1. **Web Shell Upload**
   * Uploading .php, .jsp, .asp, .aspx instead of allowed files.
2. **MIME-Type Mismatch**
   * Sending Content-Type: image/png but uploading a PHP script.
3. **Double Extensions**
   * shell.php.jpg → server interprets as PHP.
4. **Case/Unicode Bypass**
   * file.PhP, file.asp;.jpg, or Unicode characters in extensions.
5. **Dangerous Document Macros**
   * Uploading .docm, .xlsm with malicious macros.
6. **Client-Side Only Validation**
   * Using Burp/ZAP to change blocked file type after client-side rejection.

**🔍 How to Test**

1. **Check Allowed File Types**
   * Upload legit formats first (e.g., .jpg, .png).
2. **Bypass Validation**
   * Upload forbidden types directly via **Burp/ZAP**.
   * Manipulate Content-Type headers.
3. **Test Extension Tricks**
   * .php.jpg, .asp;.png, .gif%00.php.
4. **Check File Execution**
   * Access uploaded file in browser → confirm if it executes.
5. **Check Storage Location**
   * Are files stored in /uploads/ accessible directly?
   * If yes → test for execution in web root.

**🛡️ Mitigation**

* **Whitelist extensions & MIME types** (server-side, not client-side).
* Rename uploaded files → generate **random UUIDs**.
* Store files **outside web root** (serve via download handler).
* Strip metadata (EXIF, scripts) from uploaded files.
* Scan files with **AV/antimalware** before processing.
* Forbid dangerous file types (.php, .exe, .sh, .js, .html).

✅ **Key Takeaway:**  
If file type validation is weak, attackers can upload **malicious files** → leading to **RCE, malware distribution, or stored XSS**.

👉 Do you want me to also make a **list of practical payload examples** (like double extensions, MIME spoofing tricks, polyglot files) that testers often use for this attack?