Perfect 👍 you’ve now moved to **Testing File Extensions Handling for Sensitive Information**. This is part of WSTG (Web Security Testing Guide) under *information leakage / arbitrary file handling*. Let’s structure it for your notes:

**🗂️ Test File Extensions Handling for Sensitive Information**

**Goal** → Check if the application mishandles file extensions or exposes sensitive files (source code, backups, configs).

**1. Common Sensitive File Types**

Attackers look for files accidentally left on the server:

* **Backup files**: .bak, .old, .zip, .tar, .gz
* **Temp files**: .swp, ~, .tmp
* **Config / Env files**: .env, config.php.old, web.config.bak
* **Source files**: .php.swp, .aspx.old, .java~

👉 Example: index.php~ or config.php.bak may leak DB credentials.

**2. Wordlists**

* Use **SecLists** → Discovery/Web-Content/ has wordlists for file extensions & backups.
  + Example wordlist: raft-large-files.txt, web-extensions.txt, wordpress.fuzz.txt

**3. Fuzzing Approaches**

**a) Burp Suite Intruder**

* Add payload like:
* index.php
* index.php~
* index.php.bak
* index.php.old
* index.php.zip
* Look at response **status codes** (200 OK, 403 Forbidden, 500 Error).

**b) ffuf (Fuzz Faster U Fool)**

* Command example:
* ffuf -u http://target.com/indexFUZZ -w /path/to/wordlist/extensions.txt
* If a backup exists → you might see 200 OK with sensitive content.

**c) Dirsearch / Gobuster**

* Can also be used with extension lists.

**4. Practical Example**

* Target file: login.php
* Try:
* login.php.bak
* login.php~
* login.php.old
* login.php.save
* If 200 OK → attacker downloads it → extracts DB credentials or logic.

✅ **Summary**:

* Use **SecLists** for extension wordlists.
* Fuzz with **BurpSuite Intruder** or **ffuf**.
* Watch for responses that indicate real files (200, 403, sometimes 500).
* Focus on backups, configs, and source code.

Do you want me to also make a **ready-to-use ffuf + seclists command set** for you (like a small cheat sheet) so you can run tests directly?