Finding sensitive information leaked through metadata files

What this test is about

This test focuses on **finding sensitive information leaked through metadata files** on a website. These files are often not meant for users but can expose:

- Hidden directories or paths
- Web application functionality
- Technology used
- Contact info, social media, or team members
- Security policies

Why it's important: Attackers can use this info to plan attacks, perform social engineering, or find weak points in the system.

Tools Required

- Browser (View Source / DevTools)
- curl
- wget
- Burp Suite
- OWASP ZAP
- GPG (for OpenPGP keys, optional)

Step-by-Step Testing

Step 1: Check robots.txt

- **Purpose:** Find hidden paths that are disallowed for crawlers.
- **How to Test:**

```
curl -O -Ss https://target.com/robots.txt && cat robots.txt

or

wget https://target.com/robots.txt
```

- **What to Look For:** Lines starting with Disallow:. These are hidden or restricted paths, e.g., /admin or /private.
- **Optional:** Use Google Webmaster Tools → Analyze robots.txt to see how Google interprets it.
- **Outcome:** Discover directories or pages that may not be linked publicly.

Step 2: Analyze META Tags

- **Purpose:** Check for crawler instructions and hidden information in HTML.

- **How to Test:**

1. Open the webpage in a browser.
2. Right-click → **View Page Source** (or press CTRL+U).
3. Search for <meta> tags such as:

```
<meta name="robots" content="noindex, nofollow">
<meta property="og:title" content="Site Name">
<meta property="og:image" content="https://target.com/image.jpg">
```

- **What to Look For:**
  - robots tag → tells search engines whether to index or follow links.
  - Open Graph / Twitter tags → may contain hidden URLs or images.
  - Technology or framework hints.
- **Outcome:** Instructions for crawlers, technology info, and URLs that may be unlinked elsewhere.

Step 3: Check sitemap.xml

- **Purpose:** Map all URLs of the website, including hidden pages.
- **How to Test:**

```
wget https://target.com/sitemap.xml
or open in browser:
https://target.com/sitemap.xml
```

- **What to Look For:** All <loc> entries listing URLs.
- **Optional:** Explore hidden pages in browser or Burp Suite/ZAP.
- **Outcome:** Full URL mapping, discover unlinked pages or endpoints.

Step 4: Check security.txt

- **Purpose:** Find contact info, bug bounty programs, or security-related info.
- **How to Test:**

```
wget https://target.com/.well-known/security.txt
or open in browser:
https://target.com/.well-known/security.txt
```

- **What to Look For:**
  - Contact emails (Contact:)

- Bug bounty links (Policy:)
- Encryption info (Encryption:)

- **Outcome:** Information useful for responsible disclosure, social engineering, or bug bounty.

Step 5: Check humans.txt

- **Purpose:** Learn about developers or contributors behind the site.
- **How to Test:**

```
wget https://target.com/humans.txt
or open in browser:
https://target.com/humans.txt
```

- **What to Look For:** Names, roles, emails, or contributions.
- **Outcome:** Identify people behind the site; useful for social engineering or OSINT.

Step 6 (Optional): Check OpenPGP Public Keys

- **Purpose:** Learn about cryptography or key ownership.
- **How to Test:**

```
Download public key from the website.
Use GPG to view metadata: gpg --list-packets keyfile.asc
```

- **What to Look For:** Algorithm, key size, creation date, user IDs.
- **Outcome:** Understand the cryptography in use and owner info.

Step 7 :Similarly using burp and using ZAP also u can try to testing