**Search Engine Discovery and Reconnaissance for Information Leakage**

Search engines (Google, Bing, DuckDuckGo, etc.) **crawl websites** and index pages.

Sometimes, **sensitive information** (that was never meant to be public) gets indexed such as

* Internal documents (PDFs, Word files)
* Config files (config.php, .env, .bak)
* Test or staging websites (e.g., dev.example.com)
* Error messages, debug logs, or code snippets
* Usernames, passwords, API keys

Attackers (and testers) can use search engines to **gather intelligence (recon)** before trying to hack.

This is called **Search Engine Reconnaissance**.

**1) Google Hacking / Google Dorking**

[https://www.google.com](https://www.google.com/)

**What it is**: Using Google’s advanced operators (site:, filetype:, intitle:, etc.) to find sensitive/exposed data. Check if sensitive data, backups, or configs are publicly indexed.

**Type**: Web-based

**Step 1: Basic Domain Search**

Use the site: operator

**site: example.com (target url)**

**Step 2: Look for Exposed Files**

site: example.com filetype:pdf

site:example.com filetype:xls

site:example.com filetype:doc

**🡪May reveal internal docs or spreadsheets.**

**Step 3: Search for Sensitive Keywords**

site: example.com password

site:example.com confidential

site:example.com "internal use only"

**🡪Could reveal hardcoded secrets.**

**Step 4: Find open directories**

site:example.com intitle:"index of"

**→ find open directories**

**2) Shodan-** [**https://www.shodan.io**](https://www.shodan.io)

**What it is: A search engine for internet-connected devices, servers, IoT, APIs.**

**How to use:**

**On web: hostname:example.com → list servers/services.**

**Use in WSTG: Check if the organization’s servers/devices are exposed or misconfigured.**

**What is Shodan?**

* Shodan is like Google, but instead of searching websites, **it searches for devices and servers on the internet.**
* When you connect to a service (like a web server, FTP, or SSH), that service tells some information about itself (called a banner).

**Why Shodan in Recon/VAPT?**

In Reconnaissance phase → helps you map attack surface of target infrastructure.

* In Infra VAPT → helps find exposed services (databases, webcams, routers, industrial systems, misconfigured cloud storage).
* Used to detect misconfiguration, outdated services, weak security controls.
* **You don’t scan the target** → Shodan already did.
* This makes it **passive reconnaissance** (safe, no alerts on client’s IDS/IPS).
* Shodan **scans the entire internet**, saves banners + metadata in its database, indexes it like a search engine, and lets you query it later.
* Google shows webpages. While Shodan shows servers, routers, IoT, APIs, webcams, SCADA systems — basically anything online with an IP and open port.

Every time a device “talks” (HTTP, FTP, SSH), it leaks a banner → Shodan stores this.

Example:

Apache/2.4.49 (Debian) OpenSSL/1.1.1

**From this, you know:**

* **What software** is running
* **What version**
* **Which port**
* **Where it’s located**

**How you (a pentester) use it**

**Step 1 — Reconnaissance**

Before scanning, check if the target already leaks info on Shodan.

Example query:

hostname: demo.testfire.net

**You may get:**

1. IP address
2. Open ports (80, 443, 22)
3. Banner (Apache/2.4.49, OpenSSH 7.6)

Now you already know the target runs **Apache + SSH** without even scanning.

**Step 2: Technology Identification during penetration testing.**

These are **Shodan search filters** used in

**ssl:"demo.testfire.net"**  
Searches SSL certificates across the internet for demo.testfire.net.  
**Purpose:** find hidden subdomains (like staging.demo.testfire.net) that are also covered by the SSL cert.

**Run SSL Query**

Type:

ssl:"demo.testfire.net"

What happens:

* Shodan looks at all SSL/TLS certificates issued for this domain.
* If the SSL cert covers other names (like vpn.demo.testfire.net, staging.demo.testfire.net), you discover **new subdomains** that may not be public but still exist.
* Next → try to open them in browser or scan them with Nmap.

**http.title:"Apache Tomcat"**  
Finds websites where the **page title** says "Apache Tomcat".  
Purpose: detect Tomcat admin panels exposed to the internet.

**Step 4: Check for Technology Pages**

Type:

http.title:"Apache Tomcat"

👉 What happens:

* If Tomcat admin console is exposed, you’ll find it.
* Next step → Test default creds (admin:admin) or weak creds.
* If it opens → that’s a serious finding.
* **product:"nginx" port:80**  
  → Finds all servers running **nginx** on port 80.  
  Purpose: know which software (and version) the site uses → check if vulnerable.

**Step 5: Search by Software/Version**

Type:

product:"Apache httpd" hostname:demo.testfire.net

👉 What happens:

* Shodan shows the **version** of Apache.
* Example: Apache/2.4.49
* Then you check CVE database → Apache 2.4.49 vulnerable to **CVE-2021-41773 Path Traversal**.
* Next step → confirm with manual exploit or PoC.
* **Step 6: Confirm with Nmap**
* Take the IP address and run:
* nmap -sV -p80,443,22 <IP>
* Confirms what Shodan said.  
  If Apache version is outdated → document as vulnerability.