LAB 1 :

Scan Web Server for Vulnerabilities

NIKTO TOOLS

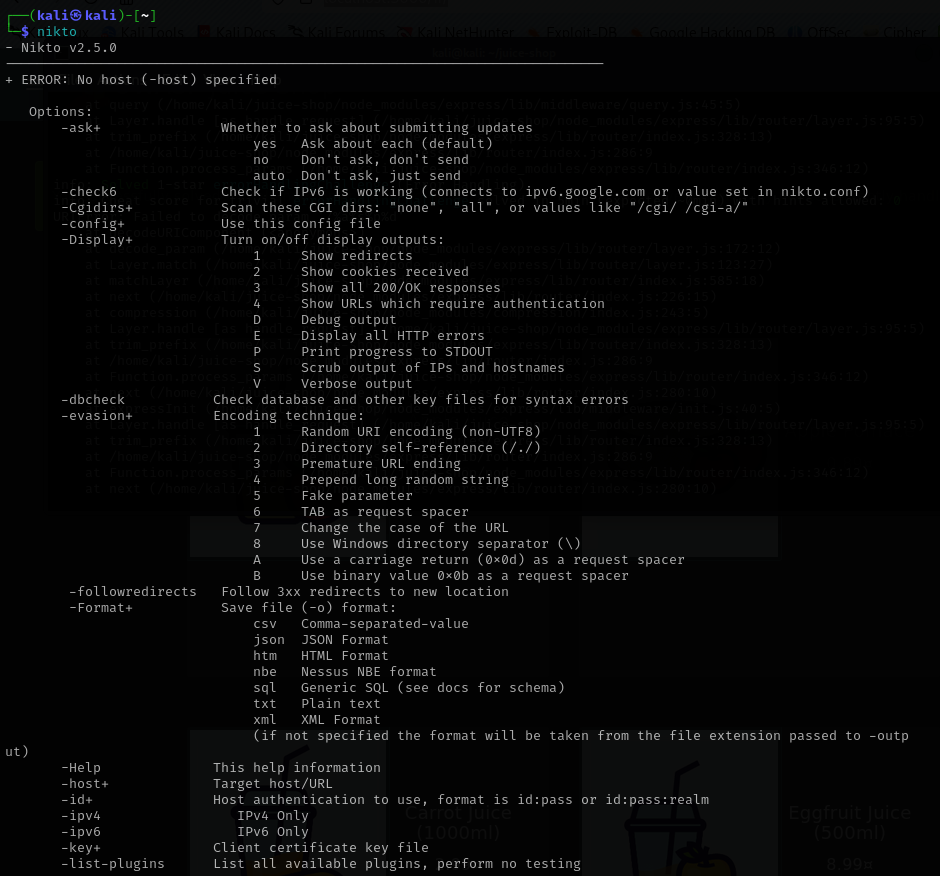
**Scans for:**

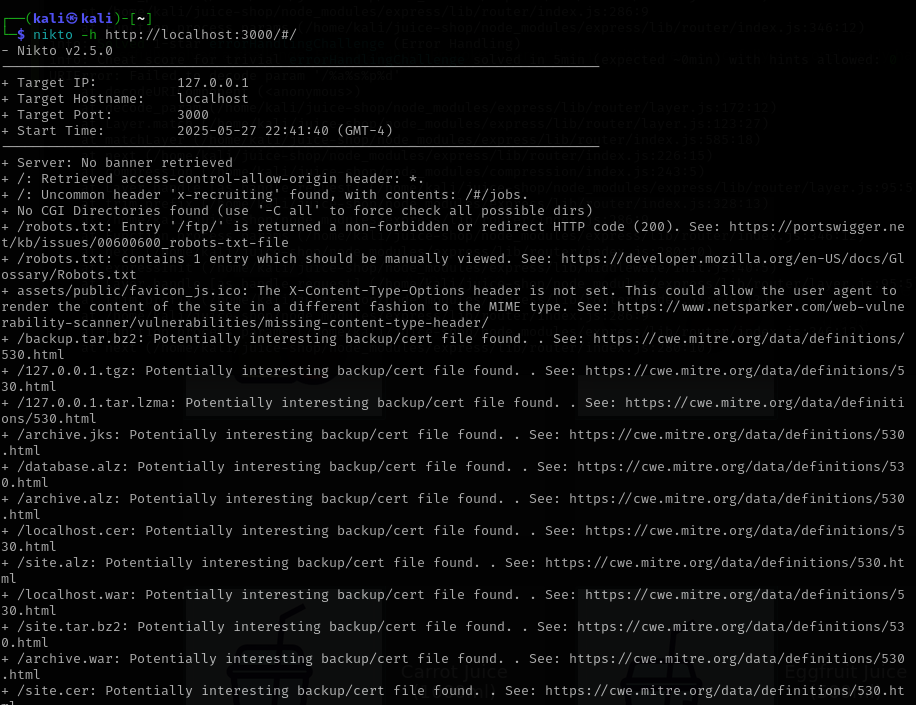
* Hidden directories
* Dangerous files
* Known vulnerabilities

Function nikto: Nikto is a web server vulnerability scanner and it focuses only on web servers (HTTP/HTTPS), not general network or port scanning like Nmap.

1. Track ip address for website

USING SYNTAX: nikto -h < https:// example website>





NOW WE GET TARGET IP ADDRESS: 127.0.0.1

**Function:**  
Scans a website for vulnerabilities (e.g., outdated software, misconfigured security headers, default files).

**Misconfigurations / Misunderstandings:**

* ❌ Using < https://example.com> with angle brackets or spaces:  
  ✅ **Correct:** nikto -h https://example.com
* ❌ Thinking it hides your identity — **Nikto does not anonymize you.** Use it from a safe environment or via proxy.
* ❌ Running it without permission — it's **legally considered an attack** if done without authorization.

Suitable tool to perform a scan and list open ports and running services.  
 Nmap Functions:

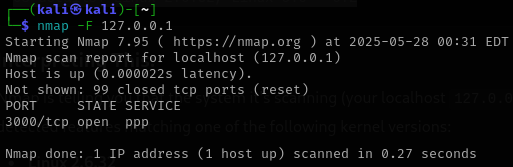
* Open ports
* Running services and their versions
* Operating systems (with OS detection)
* Vulnerabilities (with NSE scripts)

1. Scan Top 1000 TCP Ports:

Function:

|  |  |
| --- | --- |
|  | Tells Nmap to perform a "Fast scan", which scans only the top 1000 TCP ports (instead of all 65,535 ports). These are ports most commonly used by services like HTTP, HTTPS, FTP, SSH, etc. |

SYNTAX: nmap -F <target-ip>

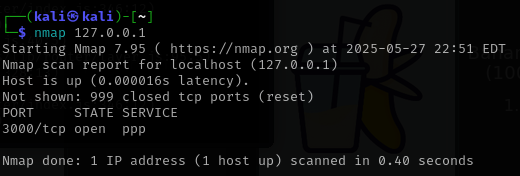


**Misconfigurations / Misunderstandings:**

* ❌ Thinking -F scans all ports — it scans **only the top 1000**, not all 65535 ports.
* ❌ Using hostname instead of IP without resolving DNS — can cause failure if no DNS.  
  ✅ Use actual IP or ensure DNS resolution works.

1. FOR BASIC SCAN PORT

SYNTAX: nmap <target ip address>



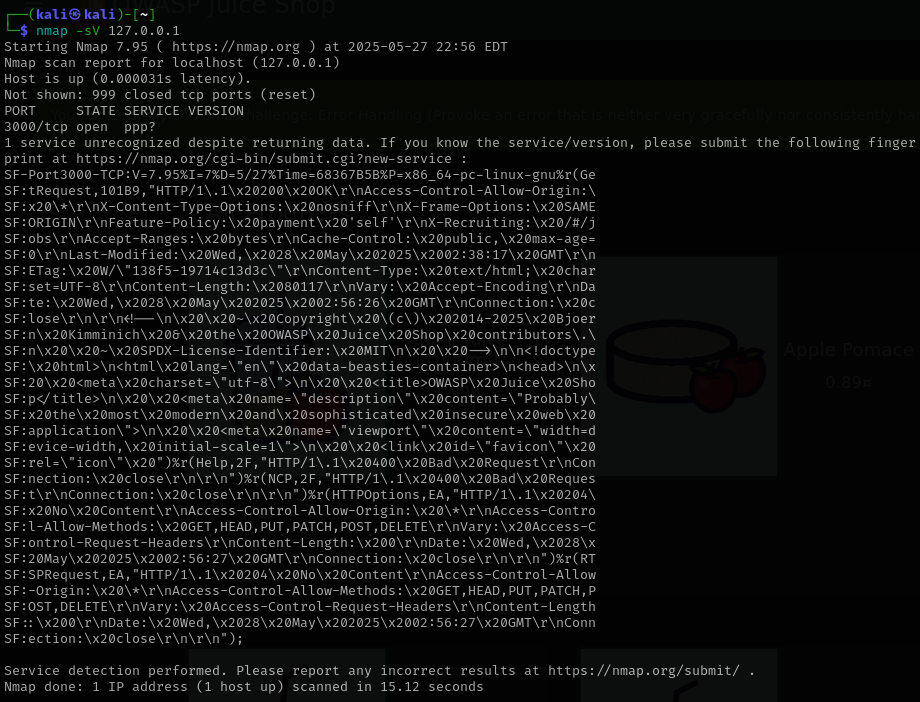
We will get port which is: 3000

**Misconfigurations / Misunderstandings:**

* ❌ Assuming it checks UDP — **this only checks TCP by default**.
* ❌ Believing it's stealthy — it's easily detectable by firewalls/IDS.

1. Scan for Open Ports and Service Versions:

SYNTAX: nmap -sV <target ip address>



The service running in port 3000 which is OWASP Juice Shop, a vulnerable web app used for training. Nmap couldn’t fingerprint the exact version, but from the copyright year (2014-2025) and headers, it is likely a recent version, possibly the latest version.

**Misconfigurations / Misunderstandings:**

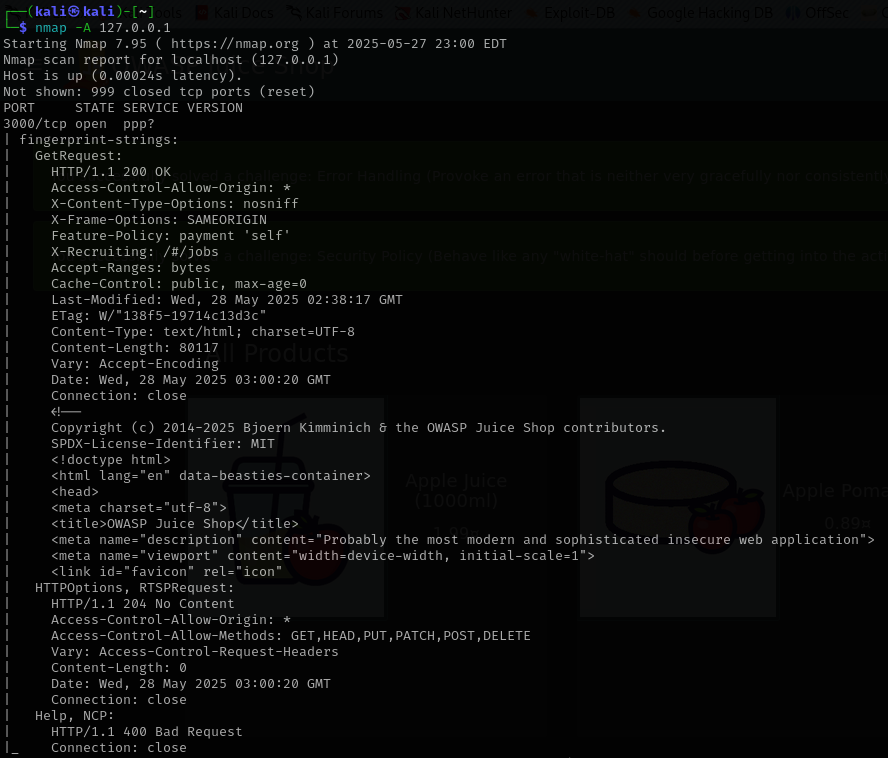
* ❌ Using -sV on a firewalled host — many firewalls block version detection, resulting in incomplete output.
* ❌ Forgetting sudo (on Linux) — might block low ports (like 22, 80).
* ❌ Thinking it's passive — **this is an active scan** and may be detected.

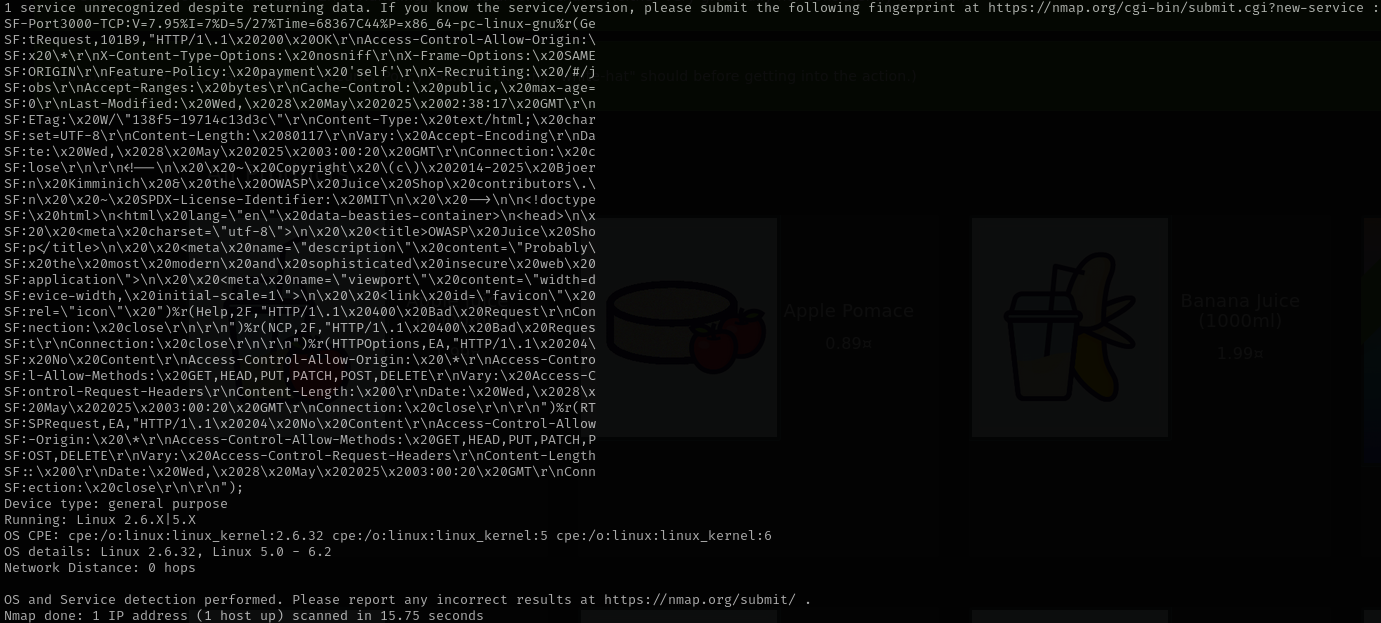
1. Aggressive Scan (OS detection + services + scripts):

Function:

* Identify the OS (e.g., Ubuntu, Windows Server)
* Show open ports and service versions
* Run scripts (like checking for CVEs or weak SSL)
* Show how your packets reach the target (traceroute)

SYNTAX: nmap -A <target-ip>







CAUTIOUS:

* Aggressive scanning is noisy — it can be detected by firewalls or intrusion detection systems (IDS).
* Only use this scan on authorized systems

**Misconfigurations / Misunderstandings:**

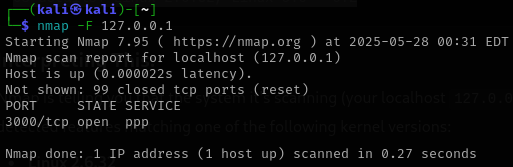
* ❌ Using -sV on a firewalled host — many firewalls block version detection, resulting in incomplete output.
* ❌ Forgetting sudo (on Linux) — might block low ports (like 22, 80).
* ❌ Thinking it's passive — **this is an active scan** and may be detected.

1. Scan Top 1000 TCP Ports:

Function:

|  |  |
| --- | --- |
|  | Tells Nmap to perform a "Fast scan", which scans only the top 1000 TCP ports (instead of all 65,535 ports). These are ports most commonly used by services like HTTP, HTTPS, FTP, SSH, etc. |

SYNTAX: nmap -F <target-ip>



**Misconfigurations / Misunderstandings:**

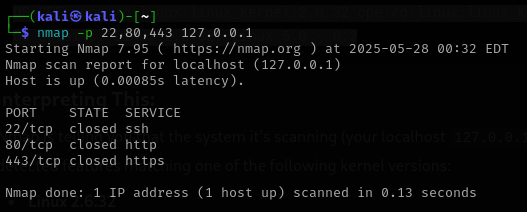
* ❌ Using -A on production servers — **it is loud and easily detected**, may even crash weak services.
* ❌ Forgetting root privileges — OS detection might fail.
* ❌ Using without permission — **highly intrusive**, can be logged.

1. Scan Specific Ports:

|  |
| --- |
|  |

|  |
| --- |
| Specifies which ports to scan instead of the default top 1000 TCP ports. You can list individual ports, ranges, or combinations. |

SYNTAX: nmap -p 22,80,443 <target-ip>



Misconfigurations / Misunderstandings:

❌ Typo in port list (e.g., spaces or wrong delimiter):

✅ Use: -p 22,80,443 not -p 22 80 443

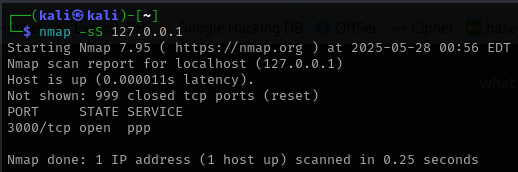
❌ Assuming services will always run on default ports — they might be moved to custom ports.

**nmap -sS -sV -O <target-ip>**

Powerful Nmap scan that combines several useful options to gather detailed information about the target system.

| **Option** | **Description** |
| --- | --- |
| -sS | **TCP SYN scan** (also known as a "stealth scan") – It sends SYN packets and analyzes responses without completing the full TCP handshake. It's faster and less likely to be logged by the target. |
| -sV | **Service version detection** – Attempts to determine the version of the services running on open ports (e.g., Apache 2.4.29, OpenSSH 7.6). |
| -O | **OS detection** – Tries to identify the operating system of the target by analyzing TCP/IP stack behavior. |

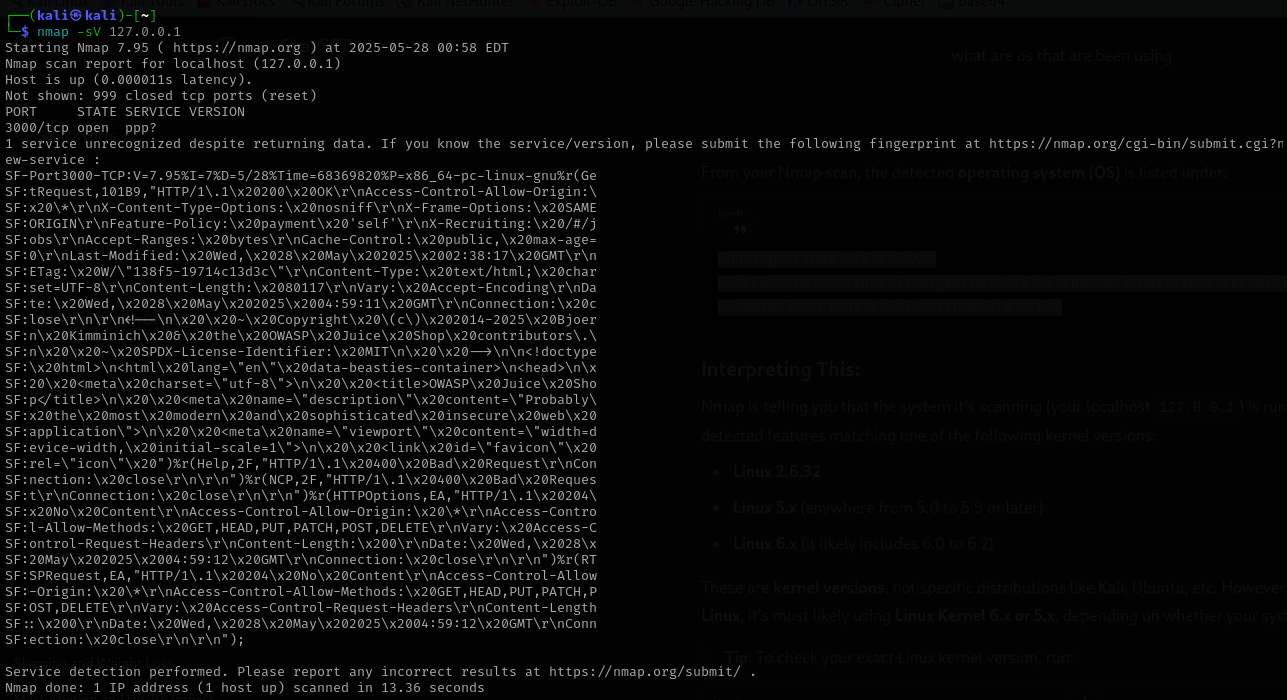
Example for command -sS:



**Misconfigurations / Misunderstandings:**

* ❌ Requires root/admin — may fail if not run with elevated privileges.
* ❌ Not invisible — **IDS/IPS can still detect** SYN scans.
* ❌ Doesn’t check service version or OS by default.

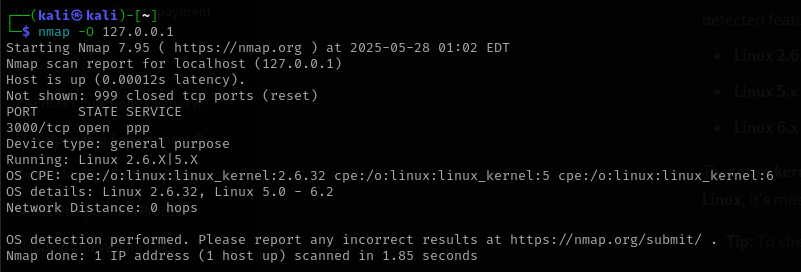
Example for command -sV:



**Misconfigurations / Misunderstandings:**

* ❌ Used against hardened hosts — might result in inaccurate or no version info.
* ❌ Believing it's always accurate — results can be spoofed by admins.

Example for command -O:



**Misconfigurations / Misunderstandings:**

* ❌ Needs root — will not work properly without elevated privileges.
* ❌ Can fail if the host blocks fingerprint packets — **not always reliable**.
* ❌ Might be mistaken for a DoS attempt by some firewalls.

**Tool to identify the web server and backend technologies using any suitable tools.**

ANSWER:

1. **WhatWeb (Best for Web Technology Fingerprinting)**

whatweb <target-url>

**Identifies:**

* Web server (e.g., Apache, Nginx, IIS)
* Backend technologies (e.g., PHP, Python, Ruby, ASP.NET)
* CMS (e.g., WordPress, Joomla)
* Frameworks, analytics tools, and more

1. Identify Web Server (e.g., Apache, Nginx, IIS)

SYNTAX: whatweb <http://example.com>

Output:

**Frameworks / Frontend Tools:**

* HTML5
* JQuery[2.2.4]

**Web App Title:**

* OWASP Juice Shop (which is a vulnerable web application for security testing)

**Script Type:**

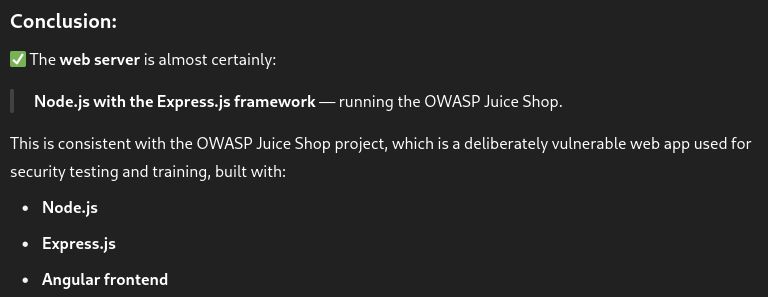
* Script[module]

**HTTP Headers:**

* Uncommon Headers including:
  + access-control-allow-origin
  + x-content-type-options
  + feature-policy
  + x-recruiting
* X-Frame-Options [SAMEORIGIN]

**IP:**

* [::1] (IPv6 loopback)



**Discover and list any accessible or hidden directories.**

1. **Gobuster – Fast and reliable directory brute-forcer**

gobuster dir -u http://<target-url> -w /usr/share/wordlists/dirb/common.txt

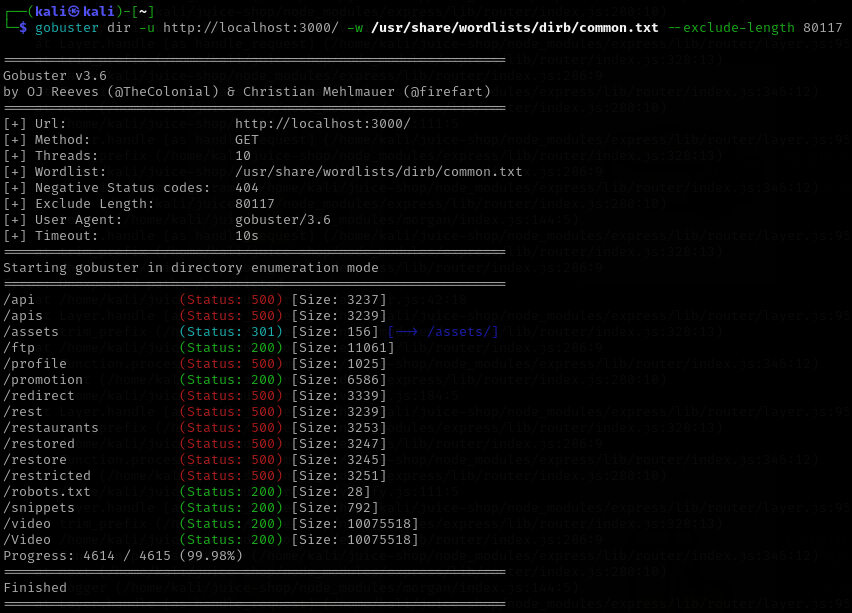
**What it does:**

* Performs a **directory brute-force attack** using a wordlist.
* Discovers:
  + Hidden or unlinked directories (/admin, /uploads, /backup, etc.)
  + Returns HTTP status codes (e.g., 200 OK, 403 Forbidden)

1. Performs a directory brute-force attack using a wordlist

Command: gobuster dir -u http://<target-url> -w /usr/share/wordlists/dirb/common.txt

(replace <target-url> )



The server returned a successful page (**status 200**) with the **same length** even for paths that don’t exist. That confused Gobuster.

I fixed it by **excluding responses of length 80117** — which is what the app returned for fake paths.

| **Path** | **Status Code** | **Size** | **Notes** |
| --- | --- | --- | --- |
| /api | 500 | 3237 | Server error — might be a broken or internal endpoint |
| /apis | 500 | 3239 | Same as above, maybe different subroutes |
| /assets/ | 301 → 200 | 156 | Redirects to /assets/, usually static files (JS, CSS, images) |
| /ftp | 200 | 11061 | Something is here; could be a file listing or a page |
| /profile | 500 | 1025 | Internal error — maybe requires login/auth |
| /promotion | 200 | 6586 | This one works — a real page |
| /redirect | 500 | 3339 | Internal error |
| /rest | 500 | 3239 | Could be part of a REST API, but it's failing |
| /restaurants | 500 | 3253 | Might be for showing restaurant info, but broken |
| /restored | 500 | 3247 | Possibly an admin or backup function? |
| /restore | 500 | 3245 | Similar to above |
| /restricted | 500 | 3251 | Might need authentication — could be sensitive |
| /robots.txt | 200 | 28 | A text file showing which parts of the site are off-limits to bots |
| /snippets | 200 | 792 | Accessible page — may show code/text snippets |
| /video | 200 | 10MB+ | Large file — maybe an actual video |
| /Video | 200 | 10MB+ | Same as above but with capital V — case-sensitive |

**🔑 What can you do with this info?**

* Paths with **200** status (like /ftp, /promotion, /video, /snippets) are real and accessible — check them in the browser.
* Paths with **500** errors exist but the server has an issue — could be:
  + Misconfiguration
  + Missing authentication
  + Bugs
* /robots.txt might give hints about other hidden or sensitive directories.
* Explore these endpoints with a browser or tools like curl, Burp Suite, or Postman.

**💡 Tips:**

* Try accessing /ftp, /promotion, /snippets, and /robots.txt in your browser.
* Use tools like curl http://localhost:3000/api to test responses.
* The /restricted or /profile pages might require login — check for login pages or tokens.

Let me know if you want help exploring any of these!

| **❌ Misconfiguration** | **✅ Explanation** |
| --- | --- |
| whatweb < target-url > (with angle brackets) | Don’t include angle brackets. Correct syntax: whatweb http://example.com |
| Target URL without protocol (http:// or https://) | WhatWeb might not resolve it correctly. Always use full URL: http://example.com |
| Assuming it's stealthy | It sends requests just like a browser — **not stealthy** and can be logged. |
| Running on HTTPS sites without SSL support installed | Could fail if system lacks SSL libraries. |
| Not using --verbose or --aggression options | Might miss tech details. Use -v or -a 3 for deeper scans. |

**Inspect HTTP headers using browser developer tool**

Steps to Inspect HTTP Headers:

✅ Using Google Chrome (similar in Edge & Brave):

1. Open the Website
   * Go to the URL you want to inspect.
2. Open Developer Tools
   * Press F12 or
   * Right-click on the page > Inspect or
   * Use shortcut: Ctrl + Shift + I (Windows/Linux) or Cmd + Option + I (Mac)
3. Go to the "Network" Tab
   * Reload the page if nothing shows up (DevTools only logs traffic after it’s opened).
   * You'll see a list of all network requests made by the page.
4. Click on the First Entry (usually the main URL)
   * This is the request for the page itself (e.g., /, index.html)
5. Inspect Headers
   * In the right-hand panel, click the "Headers" tab.
   * You’ll see two main sections:
     + Request Headers – What your browser sent to the server
     + Response Headers – What the server sent back

