Name & ID

Name - Shivani Yadav
Intern ID - 279

Tool Name

Gobuster & Whatweb

Gobuster :

History

Gobuster is a tool written in Go language, initially released in 2016, designed to brute-force URIs including directories and files as well as DNS subdomains on web servers. It was developed for fast enumeration using wordlists.

Description

Gobuster is a command-line tool used by penetration testers and ethical hackers to find hidden resources like directories, files, and virtual hosts on a target web server.

What Is This Tool About?

1. Hidden Resource Discovery: Gobuster is used to find hidden directories, files, subdomains, and virtual hosts on web servers by brute-forcing known wordlists.
2. High-Speed Enumeration: It leverages Go's concurrency model to perform fast, parallel HTTP requests, making it highly efficient for large-scale scans.
3. Versatile Recon Tool: Supports multiple modes (dir, dns, vhost) and can be tailored with filters, custom headers, and recursive scanning to adapt to various reconnaissance needs.

Key Characteristics / Features

1. Fast and lightweight
2. Written in Go (cross-platform)
3. Supports directory/file brute-forcing
4. DNS subdomain brute-forcing
5. Virtual host brute-forcing (vhost mode)
6. Wildcard DNS handling
7. Recursive brute-forcing
8. Custom headers and user agents
9. Output in JSON or plain text
10. HTTP status code filtering
11. Timeout and retry options
12. TLS/SSL support
13. Can run via WSL on Windows
14. Easily scriptable for automation
15. Integrates with wordlists like SecLists

Types / Modules Available

- dir – Directory and file scanning

- dns – Subdomain enumeration
- vhost – Virtual host detection

How Will This Tool Helps?

Gobuster helps discover:

- Unlisted admin panels
- Hidden API endpoints
- Sensitive files (e.g., `.git`, `config.php`)
- Subdomains not shown via DNS records
- Virtual host environments

Proof of Concept(PoC) Images

```
## Changes in 3.0

- New CLI options so modes are strictly separated (`-m` is now gone!)
- Performance Optimizations and better connection handling
- Ability to enumerate vhost names
- Option to supply custom HTTP headers

## Available Modes

- dir - the classic directory brute-forcing mode
- dns - DNS subdomain brute-forcing mode
- s3 - Enumerate open S3 buckets and look for existence and bucket listings
- vhost - virtual host brute-forcing mode (not the same as DNS!)

## Built-in Help

Help is built-in!

- `gobuster help` - outputs the top-level help.
- `gobuster help <mode>` - outputs the help specific to that mode.
```

```
kali@kali:~$ gobuster dir -u http://testphp.vulnweb.com/login.php -w /usr/share/wordlists/dirb/common.txt -U test -P test --wildcard
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://testphp.vulnweb.com/login.php
[+] Threads:        10
[+] Wordlist:       /usr/share/wordlists/dirb/common.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Auth User:      test
[+] Timeout:        10s
===============================================================
2021/07/14 09:09:07 Starting gobuster
===============================================================
/admin.php (Status: 200)
Progress: 584 / 4615 (12.65%)
```

```
kali@kali:~$ gobuster dir -h
Uses directory/file brutceforcing mode

Usage:
  gobuster dir [flags]

Flags:
  -f, --addslash             Apped / to each request
  -c, --cookies string       Cookies to use for the requests
  -e, --expanded             Expanded mode, print full URLs
  -x, --extensions string    File extension(s) to search for
  -r, --followredirect       Follow redirects
  -H, --headers stringArray  Specify HTTP headers, -H 'Header1: val1' -H 'Header2: val2'
  -h, --help                 help for dir
  -l, --includelength        Include the length of the body in the output
  -k, --insecuressl          Skip SSL certificate verification
  -n, --nostatus             Don't print status codes
  -P, --password string      Password for Basic Auth
  -p, --proxy string         Proxy to use for requests [http(s)://host:port]
  -s, --statuscodes string   Positive status codes (will be overwritten with statuscodesblacklist if set) (default "200,204,301,302,307,4
01,403")
  -b, --statuscodesblacklist string   Negative status codes (will override statuscodes if set)
      --timeout duration     HTTP Timeout (default 10s)
  -u, --url string           The target URL
  -a, --useragent string     Set the User-Agent string (default "gobuster/3.0.1")
  -U, --username string      Username for Basic Auth
      --wildcard             Force continued operation when wildcard found

Global Flags:
  -z, --noprogress        Don't display progress
  -o, --output string     Output file to write results to (defaults to stdout)
  -q, --quiet             Don't print the banner and other noise
  -t, --threads int       Number of concurrent threads (default 10)
  -v, --verbose           Verbose output (errors)
  -w, --wordlist string   Path to the wordlist
kali@kali:~$
```

```
Parrot Terminal
File  Edit  View  Search  Terminal  Help
┌─[sterny@sterny]─[~]
└──╼ $gobuster -h
Usage:
  gobuster [command]

Available Commands:
  dir         Uses directory/file enumeration mode
  dns         Uses DNS subdomain enumeration mode
  fuzz        Uses fuzzing mode
  help        Help about any command
  s3          Uses aws bucket enumeration mode
  version     shows the current version
  vhost       Uses VHOST enumeration mode

Flags:
      --delay duration    Time each thread waits between requests (e.g. 1500ms)
  -h, --help              help for gobuster
      --no-error          Don't display errors
  -z, --no-progress       Don't display progress
  -o, --output string     Output file to write results to (defaults to stdout)
  -p, --pattern string    File containing replacement patterns
  -q, --quiet             Don't print the banner and other noise
  -t, --threads int       Number of concurrent threads (default 10)
  -v, --verbose           Verbose output (errors)
  -w, --wordlist string   Path to the wordlist
```

15 Liner Summary

1. Fast directory brute-forcing tool
2. Written in Go (cross-platform)
3. Supports directories, DNS, VHost modes
4. High-performance scanning
5. Easy integration with SecLists
6. CLI-based with rich options
7. Supports recursion
8. Flexible output formats
9. Supports HTTP/S
10. Timeout, retries configurable
11. Easily runs on Linux or WSL
12. Good for web reconnaissance
13. Helps find misconfigured files
14. Compatible with automation scripts
15. Popular in CTFs and real-world pentests

Time to Use / Best Case Scenario

- During web application reconnaissance
- After identifying a target server

- Before running exploits
- When scanning black-box targets
- To validate hidden resource access

When to Use During Investigation

- Early recon stage of penetration testing
- Identifying admin interfaces
- Detecting backup files
- Scanning for file upload points
- Mapping subdomains for pivoting

Best Person to Use This Tool & Required Skills:

Best User: Penetration Tester / Red Team Operator
Skills Required:

- Knowledge of HTTP protocol
- Wordlist usage and brute-force logic
- Understanding of server response codes
- Comfort with CLI tools and scripting

Flaws / Suggestions to Improve:

- No GUI (CLI only)
- Limited real-time status visualization
- Lacks built-in rate-limiting
- Requires good wordlist tuning
- Could benefit from HTML report generation

Good About the Tool:

- Extremely fast and reliable
- Lightweight and cross-platform
- Versatile for different scan types
- Simple to use with powerful features
- Great community support and documentation

WhatWeb:

History

WhatWeb was introduced as a passive and active reconnaissance tool that identifies web technologies used on websites. Developed in Ruby, it has become a staple in web fingerprinting since its release.

Description

WhatWeb scans a website and reveals the technologies it is built on, such as CMS (WordPress, Joomla), web servers (Apache, nginx), analytics platforms, frameworks, and more.

What Is This Tool About?

1. Technology Fingerprinting: WhatWeb scans websites to identify the underlying technologies—such as CMS platforms, web servers, JavaScript libraries, and analytics tools.
2. Pattern-Matching & Banner Grabbing: It uses signatures, HTTP headers, HTML content, and regex-based plugins to detect tech stacks accurately.
3. Recon & Vulnerability Insight: Helps assess the website's potential attack surface by revealing outdated or vulnerable components used in its construction.

Key Characteristics / Features

1. Identifies CMS, web servers, frameworks

2. Plugin-based architecture

3. Passive and aggressive modes
4. Detects JavaScript libraries and analytics tools
5. Customizable output formats
6. Supports HTTP headers, status codes
7. Can integrate proxies and cookies
8. Easily scriptable
9. CLI-based with rich options
10. Works with IP or domain
11. Can be run on Kali or WSL
12. Plugin contributions from the community
13. Multiple fingerprinting techniques
14. Plugin updates regularly
15. Batch scanning supported

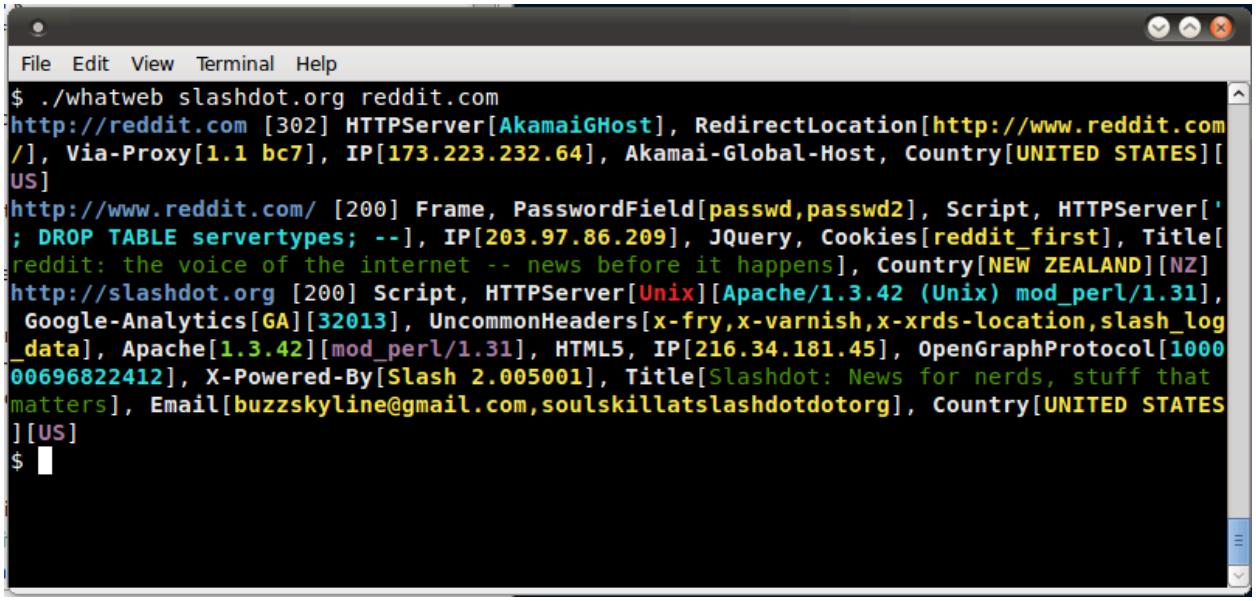
Types / Modules Available

- CMS detection module
- JavaScript library detection
- Web server identification
- Aggressive mode plugins
- SSL certificate fingerprinting

How Will This Tool Help?

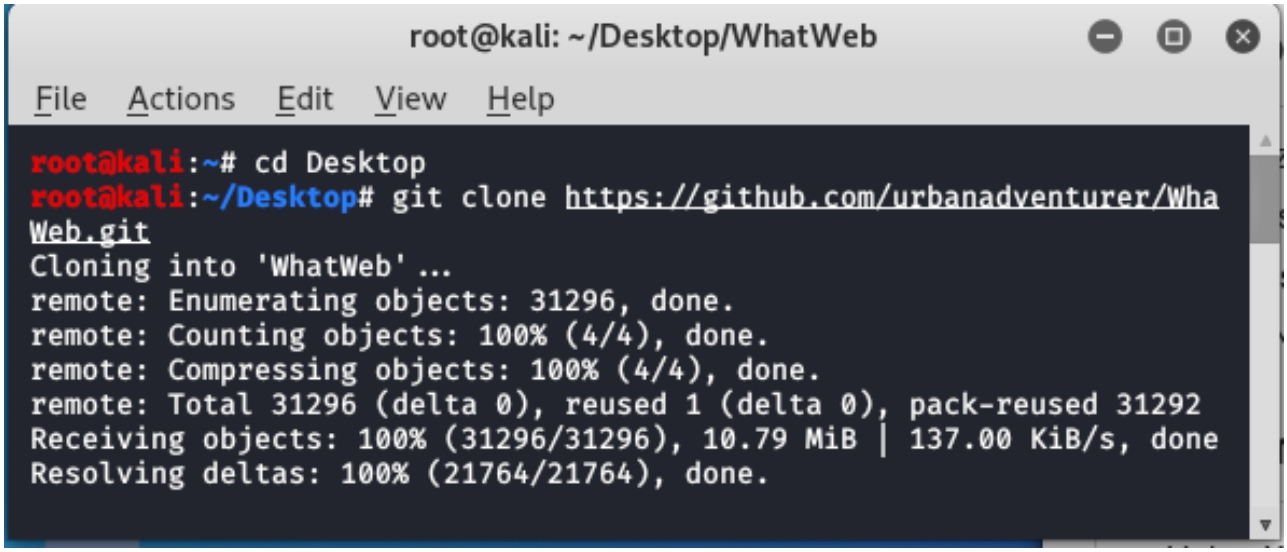
- Maps technology stack of target
- Identifies vulnerable CMS versions
- Helps plan exploits based on tech
- Aids passive reconnaissance
- Speeds up vulnerability identification

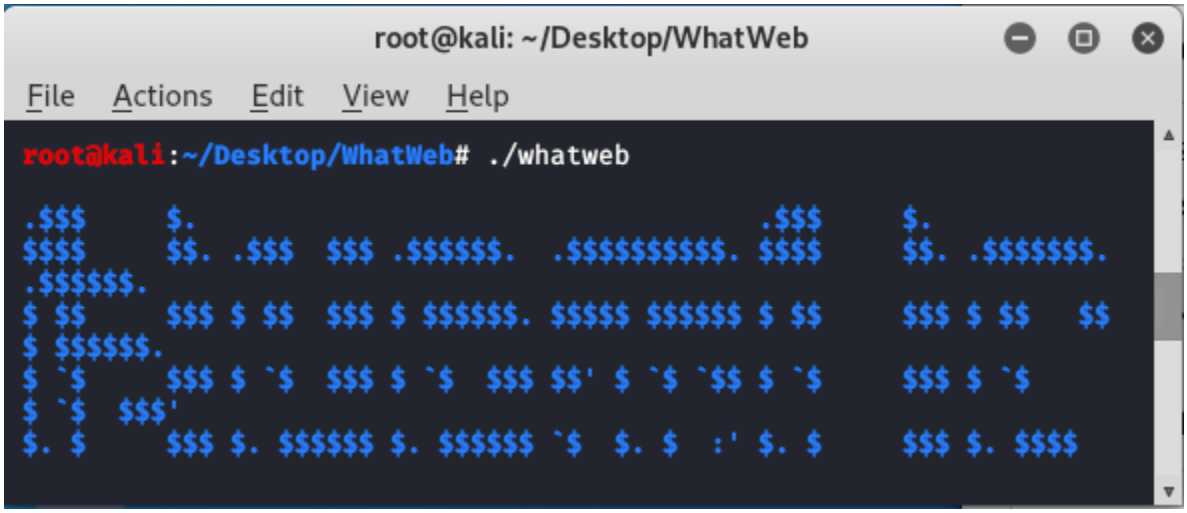
Proof of Concept(PoC) Images



```
root@kali:~# whatweb

.$$$     $.                                    .$$$     $.
$$$$     $$. .$$$  $$$ .$$$$$$.  .$$$$$$$$$$. $$$$     $$. .$$$$$$$. .$$$$$$.
$ $$     $$$ $ $$  $$$ $ $$$$$$. $$$$$ $$$$$$ $ $$     $$$ $ $$   $$ $ $$$$$$.
$ `$     $$$ $ `$  $$$ $ `$  $$$ $$' $ `$ `$$ $ `$     $$$ $ `$      $ `$  $$$'
$. $     $$$ $. $$$$$$ $. $$$$$$ `$  $. $  :' $. $     $$$ $. $$$$   $. $$$$$.
$::$  .  $$$ $::$  $$$ $::$  $$$     $::$     $::$  .  $$$ $::$      $::$  $$$$
$;;$ $$$ $$$ $;;$  $$$ $;;$  $$$     $;;$     $;;$ $$$ $$$ $;;$      $;;$  $$$$
$$$$$$ $$$$$ $$$$  $$$ $$$$  $$$     $$$$     $$$$$$ $$$$$ $$$$$$$$$ $$$$$$$$$'

WhatWeb - Next generation web scanner version 0.4.9.
Developed by Andrew Horton aka urbanadventurer and Brendan Coles.
Homepage: http://www.morningstarsecurity.com/research/whatweb

Usage: whatweb [options] <URLs>

TARGET SELECTION:
  <TARGETs>                       Enter URLs, hostnames, IP adddresses, or
                                  nmap-format IP ranges.
  --input-file=FILE, -i           Read targets from a file.
```

```
LHN@kali:/tools/WhatWeb$ ./whatweb

.$$$     $.                                    .$$$     $.
$$$$     $$. .$$$  $$$ .$$$$$$.  .$$$$$$$$$$. $$$$     $$. .$$$$$$$. .$$$$$$.
$ $$     $$$ $ $$  $$$ $ $$$$$$. $$$$$ $$$$$$ $ $$     $$$ $ $$   $$ $ $$$$$$.
$ `$     $$$ $ `$  $$$ $ `$  $$$ $$' $ `$ `$$ $ `$     $$$ $ `$      $ `$  $$$'
$. $     $$$ $. $$$$$$ $. $$$$$$ `$  $. $  :' $. $     $$$ $. $$$$   $. $$$$$.
$::$  .  $$$ $::$  $$$ $::$  $$$     $::$     $::$  .  $$$ $::$      $::$  $$$$
$;;$ $$$ $$$ $;;$  $$$ $;;$  $$$     $;;$     $;;$ $$$ $$$ $;;$      $;;$  $$$$
$$$$$$ $$$$$ $$$$  $$$ $$$$  $$$     $$$$     $$$$$$ $$$$$ $$$$$$$$$ $$$$$$$$$'

WhatWeb - Next generation web scanner version 0.5.0.
Developed by Andrew Horton (urbanadventurer) and Brendan Coles (bcoles)
Homepage: https://www.morningstarsecurity.com/research/whatweb

Usage: whatweb [options] <URLs>

  <TARGETs>                       Enter URLs, hostnames, IP addresses, filenames or
                                  IP ranges in CIDR, x.x.x-x, or x.x.x.x-x.x.x.x
                                  format.
  --input-file=FILE, -i           Read targets from a file.

  --aggression, -a=LEVEL          Set the aggression level. Default: 1.
  1. Stealthy                     Makes one HTTP request per target and also
                                  follows redirects.
  3. Aggressive                   If a level 1 plugin is matched, additional
                                  requests will be made.

  --list-plugins, -l              List all plugins.
  --info-plugins, -I=[SEARCH]     List all plugins with detailed information.
                                  Optionally search with a keyword.

  --verbose, -v                   Verbose output includes plugin descriptions.

Note: This is the short usage help. For the complete usage help use -h or --help.

LHN@kali:/tools/WhatWeb$
```





15 Liner Summary

1. Web fingerprinting tool

2. Identifies technologies used

3. Detects CMS/frameworks
4. Shows server headers and versions
5. CLI-based and plugin-driven
6. Works passively or actively
7. JSON/CSV output supported
8. Custom scripts integration
9. Detects analytics/tracking tools
10. Good for reconnaissance phase
11. Plugin-rich architecture
12. Lightweight and fast
13. Batch scanning available
14. Proxy and header support
15. Easy to run on WSL or Kali

Time to Use / Best Case Scenarios:

- Initial web recon in pentest
- Scanning competitor tech stack
- Identifying vulnerable tech
- Running on large-scale scans
- When web app is black-box

When to Use During Investigation:

- Before exploiting web applications
- For tech stack analysis
- During OSINT gathering
- For CMS/plugin vulnerability matching

Best Person to Use This Tool & Required Skills:

Best User: OSINT Analyst / Penetration Tester
Skills Required:

- Basic HTTP/web tech knowledge
- CLI navigation
- Familiarity with web reconnaissance
- Ruby familiarity (for plugin tweaking)

Flaws / Suggestions to Improve:

- Limited GUI interface
- Aggressive mode may be noisy
- Plugin accuracy varies
- Some plugins outdated
- No built-in vulnerability database linking

Good About the Tool:

- **Fast tech detection**
- **Wide plugin support**
- **Lightweight and easy to run**
- **Useful for web intel gathering**
- **Excellent for passive recon**