Gobuster — Professional Cheat Sheet

One-page professional reference for **gobuster** — fast directory/file, DNS subdomain and vhost discovery, and S3 bucket enumeration using wordlists. Practical flags, examples, tuning and OPSEC tips for web reconnaissance and red-team engagements.

---

1) At-a-glance

- **Tool:** `gobuster` — a Go-based brute-force utility (highly parallel). Common modes: `dir` (directory/file discovery), `dns` (subdomain discovery), `vhost` (virtual host discovery), `fuzz` (HTTP fuzzing), `s3` (S3 bucket enumeration).
- **Primary uses:** Discover hidden directories/files, enumerate subdomains, find unlinked vhosts, and test web server responses. Fast and simple to integrate with wordlists.
- **Note:** Gobuster is noisy — use only on authorized targets.

---

2) Install

```
# Kali/apt
sudo apt update && sudo apt install gobuster

# Go install (latest)
GO111MODULE=on go install github.com/OJ/gobuster/v3@latest
# binary will be in $GOPATH/bin or $HOME/go/bin
```

---

3) Common modes

- `dir` — directory and file brute forcing (HTTP GET requests)
- `dns` — DNS subdomain bruteforce using wordlist (requires DNS server option)
- `vhost` — discover name-based virtual hosts via Host header
- `fuzz` — custom HTTP fuzzing on a single path (placeholder `FUZZ` in URL)
- `s3` — S3 bucket enumeration using wordlist (checks existence and ACL)

---

4) Core flags (most used)

- `-u, --url <url>` : target URL (for dir/fuzz/vhost/s3)
- `-w, --wordlist <file>` : wordlist file
- `-t, --threads <N>` : concurrent workers (default ~10)

- `-x, --extensions <exts>` : comma-separated extensions to try (e.g., `php,html,txt`)
- `-s, --statuscodes <codes>` : show only these HTTP status codes (e.g., `200,204,301,302,307,401,403`)
- `-e, --expanded` : expand results (show redirects and lengths)
- `-o, --output <file>` : write output to file
- `-r, --recursive` : recursively scan discovered directories (dir mode)
- `-q, --quiet` : suppress banner/extra output
- `-a, --useragent <UA>` : set custom User-Agent
- `-H, --header <header>` : supply custom headers (repeatable)
- `-k, --no-tls` / `--no-tls` : disable TLS verification (when connecting to HTTPS with invalid certs)
- `-p, --proxy <url>` : use HTTP proxy (e.g., `http://127.0.0.1:8080` for Burp)
- `-l, --wildcard` : automatically detect wildcard responses (helps reduce false positives)
- `--no-status` : hide status codes in results
- `--delay <ms>` : add delay between requests to reduce noise
- `--timeout <s>` : HTTP request timeout
- Mode-specific: `-d, --dns` server, `--vhost` additional domain, `--s3` region flags depending on version.

---

5) Practical examples

5.1 Directory brute force (basic)

```
gobuster dir -u https://example.com -w /usr/share/wordlists/dirb/common.txt -t
50 -o gob_dir.txt
```

5.2 Try common extensions and show redirects

```
gobuster dir -u https://site -w words.txt -x php,html,js -s 200,301,302 -e -t 40
```

5.3 Recursive directory scan

```
gobuster dir -u https://app -w small.txt -r -t 30 -o recursive.txt
```

5.4 DNS subdomain enumeration using Cloudflare resolver

```
gobuster dns -d example.com -w subdomains.txt -t 50 -o dns.txt -s  A,AAAA -r 1.
1.1.1:53
```

5.5 VHost discovery (Host header brute force)

```
gobuster vhost -u https://10.0.0.5 -w vhosts.txt -t 40 -o vhost.txt -H "Host: FUZZ.example.com"
# many builds accept: gobuster vhost -u https://10.0.0.5 -w vhosts.txt -t 40
```

5.6 Fuzzing a parameter/path (`FUZZ` placeholder)

```
gobuster fuzz -u https://example.com/FUZZ -w payloads.txt -t 40 -H "X-Api-Key: 123"
```

5.7 S3 bucket enumeration

```
gobuster s3 -w buckets.txt -u https://s3.amazonaws.com -t 40 -o s3.txt
```

---

6) Tuning, false positives & heuristics

- **Wildcard detection:** use `-l` (wildcard mode) or test manually for wildcard responses to avoid false positives where the server returns 200 for any path.
- **Use `-s` to filter** results by interesting status codes (e.g., `200,301,302,401,403`).
- **Check response lengths:** different content lengths help identify valid results when status codes are the same. `-e` shows lengths.
- **Follow redirects carefully:** many web apps redirect non-existent resources to a single page — filter those with `-s` and `-l`.
- **Adjust `--delay` and lower `-t`** when working against production or WAF-protected apps.

---

7) Wordlist advice

- Prefer targeted wordlists (per app, language/framework). Start with small lists to reduce noise, then expand.
- Combine `common` + `big` lists incrementally. Use `ffuf` / `dirsearch` or wordlist generators offline to craft prioritized lists.
- Remove duplicates and sort by probability. Use `cewl` to create custom wordlists from target sites.

---

8) Integration & automation

- Pipe results into tools or scripts: parse `gob_dir.txt` for discovered endpoints and enqueue for Burp/ffuf/nikto.
- Use `--proxy` to route through Burp for manual verification of discovered entries.

- Schedule quick scans after discovery phases (nmap) to validate web paths found by crawling.

---

9) Common troubleshooting

- **Too many false positives:** enable wildcard detection `-l`, filter status codes with `-s`, or check response lengths with `-e`.
- **SSL issues:** use `--no-tls` or specify correct SNI/Host header via `-H`.
- **DNS mode not resolving:** supply `-r` DNS server or ensure network allows DNS lookups.
- **High CPU/IO:** reduce `-t` or add `--delay`.

---

10) OPSEC & ethics

- Gobuster is noisy — always have written permission and clearly scoped targets.
- Avoid running large scans with high concurrency on production without coordination.
- Store outputs securely and redact sensitive data in reports.

---

11) Quick one-liners

```
# Fast directory scan (common list)
gobuster dir -u https://target -w /usr/share/wordlists/dirb/common.txt -t 50 -s
200,204,301,302,307,403 -e -o dir.txt

# Subdomain brute force using 1.1.1.1
gobuster dns -d example.com -w ~/wordlists/subdomains.txt -r 1.1.1.1:53 -t 40 -
o subdomains.txt

# VHost discovery
gobuster vhost -u https://10.0.0.5 -w ~/wordlists/vhosts.txt -t 30 -o vhost.txt

# Fuzz API endpoints
gobuster fuzz -u "https://api.example.com/v1/FUZZ" -w payloads.txt -t 30 -H
"Authorization: Bearer TOKEN" -o fuzz.txt
```

This cheat sheet is for authorized web reconnaissance and penetration testing. Use responsibly and within scope.