

# Directory Enumeration notes

Eyad Islam El-Taher

Friday, October 3, 2025

## Subdomain Enumeration

- Knock Subdomain Scan

```
knockpy -d "domain.com" --recon --bruteforce
```

- subfinder Scan

```
subfinder -d someweb.com -o subf.txt -v
```

- assetfinder Scan

```
assetfinder -subs-only someweb.com > asset.txt
```

- Subdomain Finder

```
https://subdomainfinder.c99.nl/  
Save output to a file "subfinder.txt"
```

- Add all Enumerated/Collected subdomains from different tools in different files into one file with unique subdomains

```
cat subf.txt subfinder.txt asset.txt | sort -u > subdomains.txt
```

- To check the live subdomains and checking the status code of them

```
cat subdomains.txt | httpx -title -wc -sc -cl -ct -location -web-server  
-o alive-subdomains.txt
```

## Gobuster Directory Enumeration

Basic scan with common options

```
gobuster dir -u http://target.com/ -w common.txt -o output.txt
```

With extensions

```
gobuster dir -u http://target.com/ -w wordlist.txt -x php,txt,html
```

Specific status codes

```
gobuster dir -u http://target.com/ -w wordlist.txt -s 200,301,302,403
```

status-codes-blacklist

```
gobuster dir -u http://target.com/ -w wordlist.txt -b 404,403
```

Set threads (faster but more noisy)

```
gobuster dir -u http://target.com/ -w wordlist.txt -t 50
```

- **Advanced Example - Comprehensive Scan**

```
gobuster dir -u http://target.com/
-w /seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
-x php,html,txt,js,bak,old,json
-s 200,204,301,302,307,403,500
-b 400,404,403
-t 40
-r
-o gobuster-comprehensive.txt
--timeout 10s
--user-agent "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36"
```

- **Gobuster DNS command**

```
gobuster dns -q -r 8.8.8.8 -d example.com -w /Discovery/DNS/
subdomains-top1million-5000.txt -t 4 --delay 1s -o results.txt
```

NOTE → -r : Use custom DNS server (format server.com or server.com:port)

- **Virtual Host Discovery:**

```
gobuster vhost -u https://example.com -t 50
-w /wordlists/Discovery/DNS/subdomains-top1million-5000.txt
```

The vhost command discovers Virtual host names on target web servers. Virtual hosting is a technique for hosting multiple domain names on a single server.

Exposing hostnames on a server may reveal supplementary web content belonging to the target. Vhost checks if the subdomains exist by visiting the formed URL and cross-checking the IP address.

To brute-force virtual hosts, use the same wordlists as for DNS brute-forcing subdomains.

Similar to brute forcing subdomains eg. url = example.com, vhost looks for dev.example.com or beta.example.com etc.

## **Tips & Best Practices**

1. Start small, then scale up to larger wordlists
2. http:// and https:// versions of a URL
3. Scan both http:// and https:// versions of a URL
4. Use the -e flag to follow redirects if needed.
5. Continue to enumerate results to find as much information as possible. Run gobuster again with the results found and see what else appears. Keep digging to locate those hidden directories.

## **FFUF Directory Enumeration**

- **Simple Directory Discovery:**

```
ffuf -u http://target.com/FUZZ -w wordlist.txt
```

- **With Extensions:**

```
ffuf -u http://target.com/FUZZ -w wordlist.txt -e .php,.html,.txt
```

- **With Status Code Filtering:**

```
ffuf -u http://target.com/FUZZ -w wordlist.txt -mc 200,301,302,403
```

– Critical Filtering Options

– Filter by Response Size (Most Important):

```
Filter out common wildcard response sizes
ffuf -u http://target.com/FUZZ -w wordlist.txt -fs 1042,1245,184

Filter size ranges
ffuf -u http://target.com/FUZZ -w wordlist.txt -fs 0-100,1000-2000

Filter code
ffuf -u http://target.com/FUZZ -w wordlist.txt -fc 401,403
```

– Performance & Stealth

– Thread Control:

```
ffuf -u http://target.com/FUZZ -w wordlist.txt -t 50 # Faster
ffuf -u http://target.com/FUZZ -w wordlist.txt -t 10 # Stealthier
```

– Request Delay:

```
ffuf -u http://target.com/FUZZ -w wordlist.txt -p 0.1 # Fixed delay
ffuf -u http://target.com/FUZZ -w wordlist.txt -p 0.1-0.5 # Random
```

– Rate Limiting:

```
ffuf -u http://target.com/FUZZ -w wordlist.txt -rate 10
```

– Headers & Authentication

– Custom Headers:

```
ffuf -u http://target.com/FUZZ -w wordlist.txt
-H "User-Agent: Mozilla/5.0"
-H "Authorization: Bearer token123"
-H "X-API-Key: value"
```

– Host Header Fuzzing:

```
ffuf -w subdomains.txt -u https://target.com/ -H "Host: FUZZ" -mc 200
```

– Advanced Scanning Techniques

– Recursive Scanning:

```
ffuf -u http://target.com/FUZZ -w wordlist.txt -recursion
-recursion-depth 2
```

– POST Request Fuzzing:

```
ffuf -u http://target.com/login -w passwords.txt -X POST
-d "username=admin&password=FUZZ"
-H "Content-Type: application/x-www-form-urlencoded" -mc 200 -fs 0
```

– Parameter Fuzzing:

```
ffuf -u http://target.com/page?param=FUZZ -w parameters.txt -mc 200
```

– Multi-wordlist Fuzzing (Clusterbomb):

```
ffuf -w usernames.txt:USER -w passwords.txt:PASS
-u http://target.com/login?user=USER&pass=PASS
-mode clusterbomb -mc 200
```

– Output Options

- **Save to File:**

```
ffuf -u http://target.com/FUZZ -w list.txt -o result.json -of json
ffuf -u http://target.com/FUZZ -w list.txt -o result.txt -of csv
```

- **Silent Mode:**

```
ffuf -u http://target.com/FUZZ -w wordlist.txt -s
```

## Practical Complete Examples

- **Comprehensive Directory Scan:**

```
ffuf -u http://target.com/FUZZ
-w /seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
-e .php,.html,.txt,.js,.bak,.old
-mc 200,301,302,403,500
-fs 0,1042,1245
-t 40
-p 0.1
-c
-o ffuf-complete.json
-of json
```

- **API Endpoint Discovery:**

```
ffuf -u http://target.com/api/FUZZ
-w /seclists/Discovery/Web-Content/api/common-api-endpoints.txt
-mc 200,201,204
-H "Content-Type: application/json"
-H "Authorization: Bearer token"
-fs 0
-o api-endpoints.json
```

- **Virtual Host Discovery:**

```
ffuf -w subdomains.txt
-u http://target.com
-H "Host: FUZZ.target.com"
-mc 200,301,302
-fs 0
-o vhosts.txt
```

## Best Practices

- **Always use filters:** Start with `-fs` to filter out wildcard responses
- **Use auto-calibration:** `-ac` helps with automatic filtering
- **Respect rate limits:** Use `-p` or `-rate` for production systems
- **Save your results:** Always use `-o` with appropriate format
- **Start small:** Test with small wordlists before comprehensive scans
- **Use recursion wisely:** `-recursion-depth` prevents infinite loops

## Common Response Size Filters

- **ASP.NET:** Often `-fs 184` for wildcard redirects
- **WordPress:** Common sizes `-fs 1042,1245`
- **Custom apps:** Use `-ac` to auto-detect sizes to filter

- Dirsearch Comprehensive Scan

```
dirsearch -u https://target.com/ -e php,html,js,txt,json,asp,aspx  
-w /usr/share/wordlists/dirb/common.txt -t 50 --recursive-depth 2  
-o dirsearch-results.txt
```

- Combine and Sort Results from Multiple Tools

```
cat gobuster-.txt ffuf-.txt dirb-*.txt dirsearch-results.txt | grep  
-Eo '(http|https)://[~/" ]+' | sort -u > all-directories.txt
```

- Validate Live Directories with Httpx

```
cat all-directories.txt | httpx -title -status-code -content-length  
-web-server -location -follow-redirects -o live-directories.txt
```

- Filter Interesting Findings

```
cat live-directories.txt | grep  
-E "(admin|login|dashboard|config|backup|api)" > interesting-paths.txt
```

## Important Notes

- Always check robots.txt for hidden directories: <https://target.com/robots.txt>
- Look for common backup files: .bak, .old, .txt, \_backup, \_old
- Test for directory traversal vulnerabilities during enumeration
- Use rate limiting (-delay in gobuster, -p in ffuf) to avoid overwhelming the target
- Always respect the target's robots.txt and terms of service

## Directory Enumeration Wordlists

- Top Recommended Wordlists

- General Purpose - Most Popular

```
/usr/share/seclists/Discovery/Web-Content/common.txt  
/usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt  
/usr/share/wordlists/dirb/common.txt
```

- Comprehensive Scanning

```
/usr/share/seclists/Discovery/Web-Content/directory-list-2.3-big.txt  
/usr/share/seclists/Discovery/Web-Content/raft-medium-directories.txt
```

- Quick & Fast Scans

```
/usr/share/seclists/Discovery/Web-Content/quickhits.txt  
/usr/share/seclists/Discovery/Web-Content/top-1000.txt
```

- API & Modern Web Apps

```
/usr/share/seclists/Discovery/Web-Content/api/  
/usr/share/seclists/Discovery/Web-Content/raft-small-words.txt  
/usr/share/seclists/Discovery/Web-Content/common-api-endpoints-mazen160.txt
```

- Technology Specific Wordlists

```
WordPress
/usr/share/seclists/Discovery/Web-Content/CMS/wp-plugins.fuzz.txt

Apache
/usr/share/seclists/Discovery/Web-Content/apache.txt

IIS
/usr/share/seclists/Discovery/Web-Content/iis.txt
```

## Best Practice Recommendations

- **Start Small:** Use `quickhits.txt` or `common.txt` first
- **Escalate:** Move to medium/big lists if initial scans find little
- **Be Specific:** Use technology-specific wordlists when you know the stack
- **Avoid Overkill:** Don't start with huge wordlists - they're slow and noisy