Recon for bounty:

1- Introduction

2- Subdomain enumeration from tools

1-subfinder -d target.com -all -o subdomains.txt

2-subfinder -d target.com -silent -o subdomains.txt

3-python3 sublist3r.py -d target.com -o subs.txt

4-python3 sublist3r.py -d example.com -o subdomains.txt -t 20 =>

{cat subdomains.txt | dnsx -silent -resp -o resolved.txt

httpx -l resolved.txt -status-code -title -o live_web.txt}

5-amass enum -d example.com -passive -o amass_passive.txt

6-amass enum -d example.com -active -brute -src -ip -o amass_active.txt

7-amass enum -d example.com -dir ./amass_output -o amass_out.txt

8-amass viz -d example.com -i amass_output/amass.db -o graph.html

{cat amass_out.txt | dnsx -silent -a -resp-only -o resolved.txt

httpx -l resolved.txt -status-code -title -o live_web.txt}

9-assetfinder --subs-only example.com > assetfinder.txt

10-assetfinder example.com | tee assetfinder.txt

11-assetfinder --subs-only example.com > assetfinder.txt(

amass enum -passive -d example.com -o amass.txt

subfinder -d example.com -silent -o subfinder.txt

cat assetfinder.txt amass.txt subfinder.txt | sort -u > all_subs.txt

cat all_subs.txt | dnsx -silent -a -resp-only -o resolved.txt

)

12-python3 knockpy.py example.com --wordlist /path/to/wordlist.txt -o knock_results.txt

13-assetfinder --subs-only example.com | anew subs.txt

```
(
        assetfinder --subs-only example.com | anew all_subs.txt
        subfinder -d example.com -silent | anew all_subs.txt
        amass enum -passive -d example.com -o - | anew all_subs.txt


        cat all_subs.txt | dnsx -silent -a -o resolved.txt
        httpx -l resolved.txt -status-code -title -o live_web.txt
)
14-cat subdomains.txt | dnsgen - > perms.txt
15-dnsgen --wordlist perms.txt subdomains.txt > output-dnsgen.txt
```

3- Subdomain enumeration from websites

www.virustotal.com

www.censys.com

4- Filtering live subdomains

```
gau example.com | grep '\.js$'
gau example.com | grep '\.php$'
gau example.com | grep '\.aspx$'
gau example.com | anew urls.txt
gau example.com | grep '=' | httpx -mc 200 | anew live_params.txt
subfinder -d example.com | gau | httpx -mc 200

gospider -s "https://example.com" -o output -c 10 -d 2
gospider -s "https://example.com" -o output -c 10 -d 2 --other-source --include-subs
gospider -s "https://example.com" -o output -c 10 -d 2 --other-source --include-subs
```

5- URL extraction from the internet

Discover subdomains, then crawl them all:

subfinder -d example.com -silent > subs.txt

gospider -S subs.txt -o gospider_out -c 8 -d 2 --js --other-source --include-subs -q

gospider -s "https://example.com" -q | httpx -silent -mc 200 | anew live_urls.txt

***(This mixes historical URLs and live crawling for maximum surface discovery.)**

subfinder -d example.com -silent > subs.txt

gau example.com | anew gau_urls.txt

gospider -S subs.txt -o gospider_out -c 10 -d 2 --js --other-source -q | httpx -silent -title -mc 200 > live.txt

6- Finding parameters

python3 paramspider.py -d example.com -o example_params.txt

python3 paramspider.py -d example.com -s

Multiple domains from file:

python3 paramspider.py -l domains.txt -o all_params.txt

Filter only URLs with = (query params), check live with httpx:

python3 paramspider.py -d example.com -s | grep '=' | httpx -silent -mc 200 > live_params.txt

ParamSpider → Dalfox (automated XSS scanning):

python3 paramspider.py -d example.com -s | dalfox pipe -b -o dalfox_report.txt

Arjun finding parameters:

Test GET + POST + JSON:

```
python3 arjun.py -u "https://api.example.com/endpoint" --get --post --json
```

Scan many URLs from file (parallel):

```
python3 arjun.py --urls targets.txt --get -t 20 -o results.json
```

Use custom wordlist and proxy through Burp:

```
python3 arjun.py -u "https://example.com/path" --get --wordlist ./params.txt --proxy 127.0.0.1:8080
```

Feed arjun results to Dalfox for quick XSS checks:

```
python3 arjun.py -u "https://example.com/ep" --get -s | dalfox pipe -o dalfox_report.txt
```

```
{
    gau example.com | sort -u > urls.txt
    python3 arjun.py --urls urls.txt --get -t 20 -o arjun_results.json
    gau example.com | grep '=' | sort -u > candidate_urls.txt
    python3 arjun.py --urls candidate_urls.txt --get -t 10 -o arjun_params.json


    gau example.com | grep '=' | sort -u > candidate_urls.txt
    python3 arjun.py --urls candidate_urls.txt --get -t 10 -o arjun_params.json
}
```

7- Finding URL from past

www.wayback.com

```
echo "example.com" | waybackurls | sort -u > wayback_urls.txt
echo "example.com" | waybackurls | grep '=' | sort -u > params_from_wayback.txt
echo "example.com" | waybackurls | sort -u | httpx -silent -mc 200 > live_wayback.txt
```

8- Sorting urls

gf tools:

echo "example.com" | waybackurls | gf xss | httpx -silent -mc 200 | anew xss_candidates.txt

echo "example.com" | waybackurls | gf redirect | httpx -silent -follow-redirects -mc 200

echo "example.com" | waybackurls | gf xss | sed -n '1,50p'

9- Footprinting websites

cat urls.txt | qsreplace 'FUZZ'

cat urls.txt | qsreplace '"><script>alert(1)</script>' | while read url; do curl -s -L "$url" | grep "<script>alert(1)</script>" && echo "$url vulnerable"; done

gau example.com | qsreplace 'FUZZ' | httpx -silent -status-code

whatweb example.com

whatweb -v example.com

whatweb -a 3 example.com

whatweb -i targets.txt

gau target.com | whatweb -i /dev/stdin --log-json=whatweb.json

www.netcraft.com

www.securityheaders.com

www.dnsdumpster.com

www.whois.com

www.mxtoolbox.com

www.osintframework.com

11- Browser addons for recon:

Wappalyzer Extension

retire.js Extension

knoxx community Extension

12- WAF identification:

 # basic

wafw00f https://target.example

 # verbose

wafw00f -v https://target.example

 # find all possible matches (don't stop at first)

wafw00f -a https://target.example

 # output to file (if supported) or redirect stdout

wafw00f -a https://target.example > waf_result.txt

13- Subdomain takeover:

{git clone https://github.com/<author>/HostileSubBruteforcer.git

cd HostileSubBruteforcer

python3 -m venv venv

source venv/bin/activate

pip install -r requirements.txt

python3 hostile_sub_bruteforcer.py -d example.com -w wordlist.txt -o results.txt}

```
{
        git clone https://github.com/r3curs1v3-pr0xy/sub404.git
        cd sub404

        python3 -m venv venv
        source venv/bin/activate
        pip install -r requirements.txt

        python3 sub404.py -h
        python3 sub404.py -d example.com
        python3 sub404.py -f /path/to/subdomains.txt
        python3 sub404.py -f subdomains.txt -o results.txt

        subfinder + sublist3r

        subfinder -d example.com -silent > subfinder.txt
        sublist3r -d example.com -o sublist3r.txt
        cat subfinder.txt sublist3r.txt | sort -u > subdomains.txt
        python3 sub404.py -f subdomains.txt -o sub404_results.txt
}

{
        git clone https://github.com/haccer/subjack.git
        cd subjack
        go build -o subjack main.go

        ./subjack -w subdomains.txt
        ./subjack -w subdomains.txt -t 100 -timeout 30 -o results.txt
        ./subjack -w subdomains.txt -ssl
```

./subjack -w subdomains.txt -o results.json

./subjack -d sub.example.com

./subjack -w subdomains.txt -a

./subjack -w subdomains.txt -v

}

14- Fuzzing(Content-Discovery):

dirb http://example.com /usr/share/wordlists/dirb/common.txt

dirb http://example.com /usr/share/wordlists/dirb/common.txt -X .php,.html,.bak

dirb http://example.com /usr/share/wordlists/dirb/common.txt -p http://127.0.0.1:8080

ffuf -u <URL_with_FUZZ> -w <wordlist> [options]

ffuf -u https://example.com/FUZZ -w /usr/share/seclists/Discovery/Web-Content/common.txt -t 50 -c -mc 200,301,302

ffuf -u https://example.com/FUZZ -w /usr/share/seclists/Discovery/Web-Content/common.txt -t 80 -o ffuf_results.json -of json

ffuf -u https://example.com -H "Host: FUZZ" -w vhosts.txt -fs 4242

ffuf -u 'https://example.com/script.php?FUZZ=test' -w params.txt -fs 4242

ffuf -u https://example.com/login -X POST -d 'username=USER&password=PASS' -w users.txt:USER -w passwords.txt:PASS -mc 200

ffuf -u https://example.com/FUZZ -w common.txt -recursion -recursion-depth 2 -t 50

15- Port scanning:

sudo nmap -sS -p 1-1000 -T4 -v 192.168.1.0/24

sudo nmap -sU -p 53,161 -T3 10.10.10.5

sudo nmap -A -p 22,80,443 example.com

nmap -Pn --script vuln -p 80,443 10.0.0.1

sudo nmap -sS -sV --script=http-enum -p80,8080 192.168.56.101 -oA myscan

sudo nmap -sS -p1-1000 --max-rate 200 --randomize-hosts 192.168.0.0/24 -oA quickscan

sudo nmap -sV --version-light --script=http-enum --script-trace example.com

sudo nmap -Pn --scan-delay 250ms --host-timeout 20s --exclude 10.0.0.5 10.0.0.0/24

sudo nmap --top-ports 100 10.0.0.1

nmap --script=default 10.0.0.1

nmap --script=default 10.0.0.1

nmap --script=http-enum -p 80 10.0.0.1

nmap --script=vuln 10.0.0.1

sudo nmap -sS -sV -p 22,80,443 -T4 -oA myscan 192.168.1.0/24

nmap -Pn --script=http-enum -p80,443 example.com -oN webscan.txt

sudo nmap -sU -p 53,161 -T3 10.10.10.5 -oA udp_scan

sudo nmap --script vulscan --script-args vulscandb = exploitdb.csv -sV (ip address)

sudo masscan [target] -p[ports] --rate=[pps] -oJ [output.json]

sudo masscan 127.0.0.1 -p1-65535 --rate=1000 -oJ ./masscan_local.json

sudo masscan 192.168.1.0/24 -p22,80,443 --rate=1000 -oJ ./lan_scan.json

sudo masscan 203.0.113.5 -p80,443,8080 --rate=500 -oJ ./host_scan.json

****

Project Discovery :

subfinder -d example.com -o subs.txt -silent

naabu -host example.com -ports 80,443 -o open_ports.txt

naabu -list subs.txt -o naabu_results.txt

naabu -list subs.txt -top-ports 100 -rate 1000 -o open.txt

```
naabu -list subs.txt -o naabu_hosts.txt
cat naabu_hosts.txt | httpx -silent -ports -o live_http.txt

httpx -l subs.txt -silent -status-code -title -o httpx_results.txt
httpx -l subs.txt -silent -status-code -mc 200 -o ok_200.txt
httpx -l subs.txt -silent -tls-probe -o tls_info.txt

subfinder -d example.com -silent -o subs.txt
httpx -l subs.txt -silent -o live.txt

nuclei -l live.txt -t cves/ -o nuclei_findings.txt -c 50
nuclei -u https://example.com -t cves/ -o nuclei_out.txt
nuclei -l live.txt -t default-logins/ -o default_logins.txt
nuclei -l live.txt -t ~/nuclei-templates/ -c 50 -o nuclei_all.txt
nuclei -l live.txt -severity high,critical -o nuclei_high.txt
nuclei -l live.txt -json -o nuclei.json

katana -u https://example.com -o katana_urls.txt
katana -u https://example.com -o - | sort -u > urls.txt
httpx -l urls.txt -silent -o live_from_katana.txt

subfinder -d example.com -silent -o subs.txt
httpx -l subs.txt -silent -o live.txt
naabu -list live.txt -o ports.txt
nuclei -l live.txt -t ~/nuclei-templates/ -c 40 -o findings.txt

subfinder -d example.com -silent -oJ | jq -r '.[].host' > subs.txt
httpx -l subs.txt -silent -status-code -o live.txt
nuclei -l live.txt -json -o nuclei.json -c 50
```

****

16- Visual recon:

gowitness single https://example.com

gowitness single https://example.com --output ./screenshots/example.png

17-Google Dorking:

site:example.com — restrict results to a site or domain.

filetype:pdf — find files of a certain type (pdf, docx, xls, pptx, txt).

intitle:keyword — pages with keyword in the title.

allintitle:word1 word2 — pages whose titles contain all words.

inurl:keyword — pages with keyword in the URL.

allinurl:word1 word2 — URL contains all words.

intext:keyword — pages with keyword in the page text.

"exact phrase" — match exact phrase (use quotes).

-term — exclude results containing term.

OR — logical OR (capitalized).

related:example.com — find sites similar to the given site.

cache:example.com/page — view Google's cached version of a page.

link:example.com — (limited) pages that link to that URL.

Restrict domain + filetype:

site:university.edu filetype:pdf "research paper"

Exact phrase in title on a site:

site:gov intitle:"climate change report"

Find pages excluding a term:

"annual report" -site:example.com

Find pages with multiple words in URL:

allinurl: blog signup

Search for a phrase within a date range:

"market analysis" 2018..2020

Multiple domains:

site:example.com OR site:example.org "contact"

site:example.org filetype:pdf "white paper" after:2023-01-01

info:example.com — summary info that Google has about the page.

18- Tips for advvance google dorking:

www.pentest-tools.com

www.Dorksearch.com

19- Shodan dorking:

shodan search --fields ip_str,port,hostnames,org,location "hostname:{SUB}"

shodan download namava_play "hostname:namava.ir" --limit 10000

cat {OUT}_parsed.csv | awk -F, '{print $1}' | sort -u > unique_ips.txt

cat namava_play_parsed.csv | awk -F, '{print $1}' | sort -u > namava_unique_ips.txt

shodan search --fields ip_str,ssl.cert.subject,ssl.cert.alt_names "ssl.cert.subject.CN:{DOMAIN}"

shodan search --fields ip_str,ssl.cert.subject "ssl.cert.alt_names:{DOMAIN}"

shodan search --fields ip_str,port,product "port:80,443 product:nginx"

shodan search --fields ip_str,asn,org "hostname:{DOMAIN} country:IR"

20- Github dorking :

filename:.env "AWS_ACCESS_KEY_ID"

filename:.env "DATABASE_URL"

filename:.env "SECRET_KEY"

extension:env "PASSWORD"

filename:.git-credentials

filename:.netrc

filename:.npmrc "registry"

"BEGIN RSA PRIVATE KEY"

"-----BEGIN OPENSSH PRIVATE KEY-----"

"api_key" OR "API_KEY" OR "apiKey" in:file

"access_token" in:file

"authorization: Bearer" in:file

"ghp_" (GitHub personal access tokens)

"xoxb-" OR "xoxp-" (Slack tokens)

"AKIA" "AWS_SECRET_ACCESS_KEY" (AWS access keys patterns)

"google_api_key" OR "AIza" (Google API keys)

org:{ORG} "aws_secret_access_key"

"google-service-account" OR "serviceAccount" extension:json

"private_key" "client_email" extension:json (Google service account JSON)

"azure_client_id" OR "azure_secret"

"aws_access_key_id" "aws_secret_access_key" in:file

extension:tf "access_key" OR "secret_key" (Terraform files)

"https://hooks.slack.com/services/" in:file

"https://api.telegram.org/bot" in:file

"discordapp.com/api/webhooks" in:file

21- Vulnerability scanning :

nuclei -u https://target.com

nuclei -l targets.txt

nuclei -u https://target.com -t cves/ -t exposures/ -t misconfigurations/

nuclei -l targets.txt -o results.txt

nuclei -l targets.txt -s high,critical -o high_critical.txt

nuclei -l targets.txt -rl 50

nuclei -u https://target.com -tags xss,ssrf,sqli

nuclei -l targets.txt -silent

httpx -l subdomains.txt -status-code -silent | nuclei -t cves/ -o vuln.txt

wpscan --url https://example.com

wpscan --url https://example.com --enumerate u,p,t,vp,tt

wpscan --url https://example.com --enumerate ap

wpscan --url https://example.com --random-agent --disable-tls-checks --follow-redirection

wpscan --url https://example.com --usernames admin --passwords /path/to/passwords.txt --threads 10

22- Metasploit for scan :

{use auxiliary/scanner/portscan/tcp

set RHOSTS target.com

set PORTS 1-65535

set THREADS 20

run}

{

use auxiliary/scanner/http/vhosts

set RHOSTS target.com

set THREADS 30

 # optionally point to a wordlist if the module supports it:

set VHOSTS_FILE /path/to/vhosts_wordlist.txt

run

use auxiliary/scanner/http/http_version

set RHOSTS target.com

run

23- How to create tools for recon?

git clone https://github.com/Screetsec/Sudomy.git

cd Sudomy

pip3 install -r requirements.txt

./sudomy -d bugcrowd.com -dP -eP -rS -cF -pS -tO -gW --httpx --dnsprobe -aI webanalyze -sS

 # provide a file with one domain per line

./sudomy -l targets.txt

Hydra -L usernames.txt -P passwords.txt ftp:// ip address