The Beginners’ Guide to Bug Bounty

Whoami

I am Deependrasingh kushvaha. Many of you know me as Cheatdroit. I started my Bug bounty journey on 7 September 2020. After 6 months on 11 feb, I got my first bounty on Bugcrowd of 250$. I am a student of IT Engineering. Before starting this I am already aware of Linux, Burpsuite, and Metasploit. So I am curious to know more about Bug bounty and the course of Vikash chaudhary sir really help me understand website Vulnerabilities. This is the duration of the pandemic where I learn all this. Infosec and bugbounty community also help me a lot.

Twitter

Linkedin

Tools N CMD

Go-lang

1. I feel the most important and difficult part is to install Go. Almost all tools like gf, ffuf, Subjack, Subover are written in Go so we have to install it first.

```
$ wget https://golang.org/dl/go1.16.3.linux-amd64.tar.gz
```

2. Extract the archive you downloaded into /usr/local, creating a Go tree in /usr/local/go.

For example, run the following as root or through `sudo`:

```
$ tar -C /usr/local -xzf go1.16.3.linux-amd64.tar.gz
```

3. Add /usr/local/go/bin to the `PATH` environment variable.

```
$ export PATH=$PATH:/usr/local/go/bin
$ echo 'export PATH=$PATH:/usr/local/go/bin' >> ~/.bashrc
```

4. Verify that you've installed Go by opening a command prompt and typing the following command:

```
$ go version
```

gf-tool

After installing and verifying go.

Install gf-tool

```
$ go get -u github.com/tomnomnom/gf
$ export PATH=$PATH:/root/go/bin/
```

Also add this path variable to ~/.bashrc

change username to your machine's name

You must find "gf-completion.bash" manually in my case I found this in pkg/mod/github.com

This is very important when you install gf pattern

```
$ echo 'export PATH=$PATH:/home/username/go/bin/' >> ~/.bashrc
$ echo 'source /home/username/go/pkg/mod/github.com/tomnomnom/gf@v0.0.0-20
$ echo 'export GOPATH=/home/cheatdroit/go' >> ~/.bashrc
```

Now you have copy the regex (gf-pattern)

AsI told you you have to find it manually.

You have this either in src folder or in pkg folder check both

```
$ cp -r $GOPATH/pkg/mod/github.com/tomnomnom/gf@v0.0.0-20200618134122-dcd4c3
```

If you want then you can also add more gf pattern by shiv chouhan

```
$ git clone https://github.com/1ndianl33t/Gf-Patterns
$ cd Gf-Patterns
$ mv ~/Gf-Patterns/*.json ~/.gf
```

Subdomain

Assetfinder

```
$ go get -u github.com/tomnomnom/assetfinder
$ assetfinder --subs-only hackerone.com >> subdomain.txt
```

Sublist3r + httpx

Installation of sublist3r

```
$ sudo apt-get install sublist3r
$ sublist3r -d hackerone.com
```

Installation of httpx from source

```
$ GO111MODULE=on go get -v github.com/projectdiscovery/httpx/cmd/httpx
```

Installation of httpx from Github

```
$ git clone https://github.com/projectdiscovery/httpx.git; cd httpx/cmd/http
```

some command of httpx

```
$ httpx -l hosts.txt -silent
$ httpx -l hosts.txt -title -content-length -status-code -silent
```

Use both and find live subdomain

```
$ subfinder -d hackerone.com -silent | httpx -title -content-length -status-
```

httprobe

Installation

```
go get -u github.com/tomnomnom/httprobe
```

```
cat recon/example/domains.txt
example.com
example.edu
example.net
cat recon/example/domains.txt | httprobe
http://example.com
http://example.net
http://example.edu
https://example.com
https://example.edu
https://example.net
```

Amass

Installation

```
$ sudo apt-get install amass
```

Usage

```
$ amass enum -d hackerone.com
```

Dirsearch

Installation

```
$ git clone https://github.com/maurosoria/dirsearch.git
$ cd dirsearch
$ python3 dirsearch.py -u <URL> -e <EXTENSIONS>
```

---

usage

```
$ python3 dirsearch.py -e php,htm,js,bak,zip,tgz,txt -u https://target -t 30
```

Sqlmap

Installation

```
$ git clone --depth 1 https://github.com/sqlmapproject/sqlmap.git sqlmap-dev
```

Simple HTTP GET based test

```
$ python sqlmap.py -u 'http://mytestsite.com/page.php?id=5'
```

Getting blocked by the Web Application Firewall — WAF

```
$ python sqlmap.py -u "http://mytestsite.com/page.php?id=5" --random-agent
```

Retrieve the Database Tables

```
$ python sqlmap.py -u 'http://mytestsite.com/page.php?id=5' --tables
```

Dump the data

```
$ python sqlmap.py -u 'http://mytestsite.com/page.php?id=5' --tables
```

Crawl a website with SQLmap and auto-exploit

```
$ sqlmap -u "http://example.com/" --crawl=1 --random-agent --batch --forms -
```

Nuclei

Installation

```
GO111MODULE=on go get -v github.com/projectdiscovery/nuclei/v2/cmd/nuclei
```

Download Templates

```
nuclei -update-templates
```

Usage

here -l (list of urls) and -t (templates)

```
nuclei -l target_urls.txt -t cves/ -c 50 -v
```

Subzy

Installation

```
go get -u -v github.com/lukasikic/subzy
go install -v github.com/lukasikic/subzy
```

Usage

```
subzy -targets list.txt
subzy -target test.google.com
subzy -target test.google.com,https://test.yahoo.com
```

ffuf

Installation

```
$ go get -u github.com/ffuf/ffuf
```

Fuzzing

```
$ ffuf -w /path/to/wordlist -u https://target/FUZZ
```

GET parameter fuzzing

GET parameter name fuzzing is very similar to directory discovery, and works by defining the `FUZZ` keyword as a part of the URL. This also assumes an response size of 4242 bytes for invalid GET parameter name.

```
$ ffuf -w /path/to/paramnames.txt -u https://target/script.php?FUZZ=test_val
```

If the parameter name is known, the values can be fuzzed the same way. This example assumes a wrong parameter value returning HTTP response code 401.

```
$ ffuf -w /path/to/values.txt -u https://target/script.php?valid_name=FUZZ -
```

Post data fuzzing

```
$ffuf -w /path/to/postdata.txt -X POST -d "username=admin\&password=FUZZ"

```

Checklist For Web

Checklist

Recon

1. Find subdomain(assetfinder+sublist3r)
2. Check CNAME Records of those subdomains, Check for SubD takeover.
3. Use masscan for Port scanning(nmap)
4. Find live subdomain (httprobe + httpx)
5. Use nuclei temples (vuln+ expo and all)
6. Use Google dork
7. Do Github dork
8. Do shodan dork

On Webapp

1. Check for CORS Misconfiguration
2. Check for email header Injection on reset password function.
3. Check for SMTP and host header Injection
4. Check for I-Frame (for clickjacking)
5. Check for Improper access control and parameter Tampering.
6. Check for Session management.
7. Check for Burp History for finding endpoint
8. Use Arjun for finding hidden endpoints.
9. Check for CSRF.
10. Check for SSRF Parameters.
11. Check For XSS and SSTI.
12. Check Cryptography in Reset password Token.
13. Check for Unicode Injection In Email Parameter.
14. Check for Bypassing Rate Limit.
15. Direcctory Brute Force
16. Check For HTTP Request smuggling.
17. Check For Open Redirect Through waybackurls.
18. Check For social-sign bypass
19. Check For state Parameter in social sign-In & Check whether it's possible to cause DOS using Multiple cookies.
20. File-upload , CSRF, XSS, SSRF, RCE, LFI, XXE

21. Buffer overflow

Dorks

Github Dork

Github dork for critical files

- filename:manifest.xml
- filename:travis.yml
- filename:vim_settings.xml
- filename:database
- filename:prod.exs NOT prod.secret.exs
- filename:prod.secret.exs
- filename:.npmrc _auth
- filename:.dockercfg auth
- filename:WebServers.xml
- filename:.bash_history
- filename:sftp-config.json
- filename:sftp.json path:.vscode
- filename:secrets.yml password
- filename:.esmtprc password
- filename:passwd path:etc
- filename:dbeaver-data-sources.xml
- path:sites databases password
- filename:config.php dbpasswd
- filename:prod.secret.exs
- filename:configuration.php JConfig password
- filename:.sh_history
- shodan_api_key language:python
- filename:shadow path:etc
- JEKYLL_GITHUB_TOKEN
- filename:proftpdpasswd
- filename:.pgpass
- filename:idea14.key
- filename:hub oauth_token
- HEROKU_API_KEY language:json
- HEROKU_API_KEY language:shell

- SF_USERNAME salesforce
- filename:.bash_profile aws
- extension:json api.forecast.io
- filename:.env MAIL_HOST=smtp.gmail.com
- filename:wp-config.php
- extension:sql mysql dump
- filename:credentials aws_access_key_id
- filename:id_rsa or filename:id_dsa

GitHub Dorks for Finding Languages

- api_key
- “api keys”
- authorization_bearer:
- oauth
- auth
- authentication
- client_secret
- api_token:
- “api token”
- client_id
- password
- user_password
- user_pass
- passcode
- client_secret
- secret
- password hash
- OTP
- user auth
- jenkins

Github Dorks for finding usernames

- user:name (user:admin)

- org:name (org:google type:users)
- in:login (in:login)
- in:name (in:name)
- fullname:firstname lastname (fullname:)
- in:email (data in:email)
- GitHub Dorks for Finding Information using Dates
- created:<2012–04–05
- created:>=2011–06–12
- created:2016–02–07 location:iceland
- created:2011–04–06..2013–01–14 in:username

GitHub Dorks for Finding Information using Extension

- extension:pem private
- extension:ppk private
- extension:sql mysql dump
- extension:sql mysql dump password
- extension:json [api.forecast.io] (http://api.forecast.io/)
- extension:json [mongolab.com] (http://mongolab.com/)
- extension:yaml [mongolab.com] (http://mongolab.com/)
- [WFClient] Password= extension:ica
- extension:avastlic “[support.avast.com] (http://support.avast.com/)”
- extension:json googleusercontent client_secret

Shodan Dorks

- Big IP shodan Search:-

`http.title:"BIG-IP&reg;-Redirect" org:Org`

- CVE 2020-3452

`http.html_hash:-628873716 “set-cookie: webvpn;”`

- CVE CVE-2019-11510

`http.html:/dana-na/`

`My all time fav`

ssl:target.* 200

```
Ssl.cert.subject.CN:"target.*" 200
```

Google dorks

```
inurl:example.com intitle:"index of"
inurl:example.com intitle:"index of /" "*key.pem"
inurl:example.com ext:log
inurl:example.com intitle:"index of" ext:sql|xls|xml|json|csv
inurl:example.com "MYSQL_ROOT_PASSWORD:" ext:env OR ext:yml -git
inurl:example.com intitle:"index of" "config.db"
inurl:example.com allintext:"API_SECRET*" ext:env | ext:yml
inurl:example.com intext:admin ext:sql inurl:admin
inurl:example.com allintext:username,password filetype:log
site:example.com "-----BEGIN RSA PRIVATE KEY-----" inurl:id_rsa
```