INFOSEC DIARIES – AN INTRODUCTION TO SECURITY

Chronicles of my Infosec story telling

JANUARY 11, 2019 / SYLAR / 1 COMMENT

100 ways to discover (part 1)

As security researchers and pentesters know, Information Gathering has been overlooked by some, and not given the proper attention it deserves. Nevertheless, it remains to be a vital phase in the pentesting process.

This blog post will give different tools to do basic recon in a pentest engagement since no one only relies on one tool. More advanced recon techniques will be covered in part 2 of this blog.

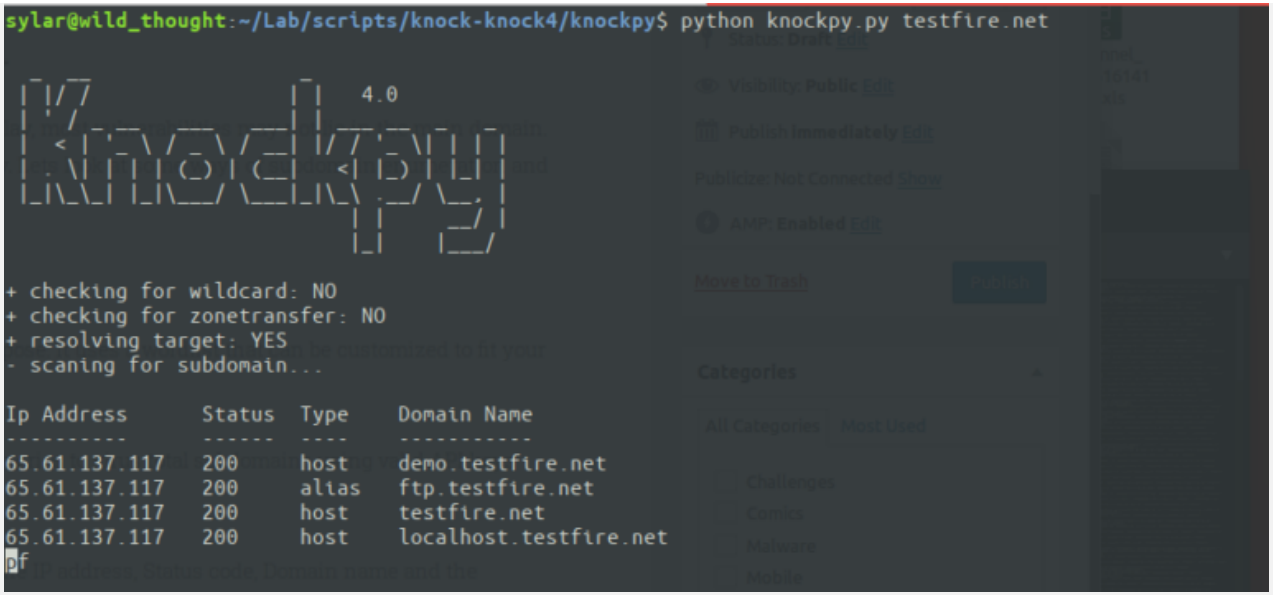
The Tools-set Under Different Categories

Subject 1. Sub domain Enumeration.

In most cases, say in a bug bounty play, most vulnerabilities may not lie in the main domain. Sub domain hunting comes in handy. Lets look at some ways of sub domain enumeration and discovery.

a. Knockpy

Knockpy is a handy tool for this purpose. It uses a wordlist that can be customized to fit your target attack.



b. Sublist3r

As a tool mentioned by pentesters and bug bounty hunters all over the internet, this is a must try.

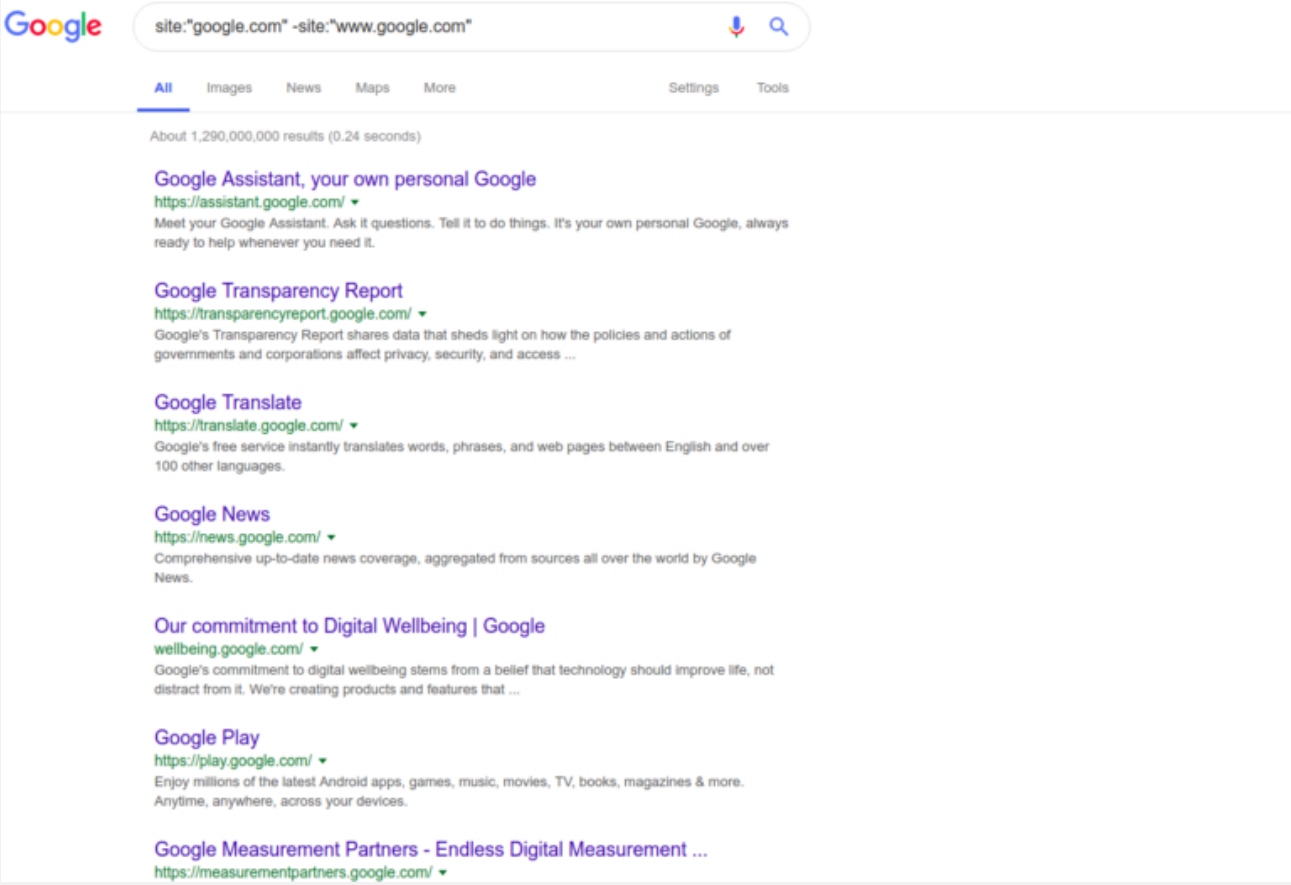
Sublist3r relies purely on OSINT techniques. It crawls different search engines including Google, Baidu, Yahoo, Ask etc. Sub domain enumeration also possible via DNSdumpster, Netcraft, Virus total among others.

```
# Coded By Ahmed Aboul-Ela - @aboul3la

[-] Enumerating subdomains now for testfire.net
[-] Searching now in Baidu..
[-] Searching now in Yahoo..
[-] Searching now in Google..
[-] Searching now in Bing..
[-] Searching now in Ask..
[-] Searching now in Netcraft..
[-] Searching now in DNSdumpster..
[-] Searching now in Virustotal..
[-] Searching now in ThreatCrowd..
[-] Searching now in SSL Certificates..
[-] Searching now in PassiveDNS..
[-] Total Unique Subdomains Found: 17
www.testfire.net
altoro.testfire.net
computerserviceandsales.cn srchttpdemo.testfire.net
hkcastte.com srchttpdemo.testfire.net
superkeychain.com srchttpdemo.testfire.net
demo.testfire.net
demo2.testfire.net
domain2.testfire.net
evil.testfire.net
ftp.testfire.net
httpdemo.testfire.net
localhost.testfire.net
wellpoint.jobs.net srchttpdemo.testfire.net
wellpoint.se srchttpdemo.testfire.net
testfire.net:8080
www.testfire.net:8080
demo.testfire.net:8080
```

c. Google dorks

Google as the most popular search engine caches all sorts of websites. This makes it a good tool to find sub domains visited. We just need to know how to ask. Using ‘google.com’ as an example, we can easily do this, exposing the sub domains.



Some good scripts also exist that automate google dorking. Here are 2:

- **GoogD0rker**

This one automatically launches a series of queries against the specified target. Great OSINT tool. The tool is able to find documents, login pages, backdoors, files by extension, pastebin posts, subdomains etc.

Download it here.

```
https://demo.testfire.net/index.jsp?content=business.htm
http://www.testfire.net:8080/index.jsp?content=business_deposit.htm
Finding FIND DOCUMENTS BY EXTENSION for testfire.net

https://demo.testfire.net/pr/communityannualreport.pdf
Finding FIND APACHE STRUTS RCE for testfire.net

Finding PASTEBIN POSTS FOR DOMAIN for testfire.net

https://pastebin.com/6rqvKgk5
https://pastebin.com/CZ37pwBy
https://pastebin.com/u/NeroHaxor1337
https://pastebin.com/NWgcKVxQ
https://pastebin.com/HgpnM6e6
https://pastebin.com/rQzc9eEh
https://pastebin.com/8RYLCsrT
http://pastebin.com/iXLL3mbF
https://pastebin.com/yaitE8Xd
Finding EMPLOYEES ON LINKEDIN for testfire.net
```

- **GooHak**

Similar to the above. Find it here.

d. Amass

An OWASP tool for sub domain discovery that uses multiple sources to do this. More info can be found on their git page.

```
xpl0r3r@xpl0r3r-ThinkPad-T460s:~$ amass -src -ip -d testfire.net
[Forward DNS]     testfire.net,65.61.137.117
[CertSpotter]     demo.testfire.net,65.61.137.117
[ThreatCrowd]     localhost.testfire.net,65.61.137.117
[ThreatCrowd]     evil.testfire.net,65.61.137.117
[VirusTotal]      altoro.testfire.net,65.61.137.117
[ThreatCrowd]     demo2.testfire.net,65.61.137.117
[ThreatCrowd]     www.testfire.net,65.61.137.117
[ThreatCrowd]     ftp.testfire.net,65.61.137.117
Average DNS names processed: 193/sec

OWASP Amass v2.8.8                                  https://github.com/OWASP/Amass
-------------------------------------------------------------------------------
8 names discovered - dns: 1, cert: 1, scrape: 6
-------------------------------------------------------------------------------
```

e. Curl one liner

This is a cool script i found on twitter from Ben Sadeghipour's tweet. Its pretty simple and uses *archive.org* to scrape the sub domains.

```
xpl0r3r@xpl0r3r-ThinkPad-T460s:~$ curl -s "http://web.archive.org/cdx/search/cdx?url=*.testfire.
put=text&fl=original&collapse=urlkey" |sort| sed -e 's_https*://__' -e "s/\/.*//" -e 's/:.*//' -
\.//' | uniq
demo2.testfire.net
demo.testfire.net
evil.testfire.net
ftp.testfire.net
localhost.testfire.net
testfire.net
testsite.testfire.net
testfire.net
```

==>"curl -s "http://web.archive.org/cdx/search/cdx?url=.testfire.net/*&output=text&fl=original&collapse=urlkey" |sort| sed -e 's_https*://__' -e "s/\/.*//" -e 's/:.*//' -e 's/^www\.//' | uniq"<==*

f. Confirm live domains

During my hunts, i found out a number of the domains discovered from tools that do mass scraping do not resolve. In this case, i wrote a simple bash script that given a text file with all the valid sub domains, goes through them all and tries to resolve them and find out which ones don't. Download it here.

```
xpl0r3r@xpl0r3r-ThinkPad-T460s:~/Lab/scripts/bugbounty/misc$ ./url_exist.sh
-----------------------------------------------
Valid URL: demo2.testfire.net
-----------------------------------------------
-----------------------------------------------
Valid URL: demo.testfire.net
-----------------------------------------------
-----------------------------------------------
Valid URL: evil.testfire.net
-----------------------------------------------
-----------------------------------------------
Invalid URL: ldap.testfire.net
-----------------------------------------------
-----------------------------------------------
Valid URL: localhost.testfire.net
-----------------------------------------------
-----------------------------------------------
Valid URL: testfire.net
-----------------------------------------------
-----------------------------------------------
Invalid URL: good.testfire.net
-----------------------------------------------
-----------------------------------------------
Invalid URL: testsite.testfire.net
-----------------------------------------------
-----------------------------------------------
Valid URL: testfire.net
-----------------------------------------------
-----------------------------------------------
Invalid URL: admin.testfire.net
-----------------------------------------------
-----------------------------------------------
Invalid URL: logs.testfire.net
-----------------------------------------------
```

Subject 2. Web Server Fingerprinting.

culprit: HTTP Methods

a. Curl

Curl is a pretty powerful CLI tool. Despite being used by pentesters to exploit file inclusions (RFI, LFI), command injections, HTTP file uploads etc, it can also be

used to identify a HTTP methods allowed on the server. Some servers however have OPTIONS disabled, we can use HEAD instead.

```
sylar@wild_thought:~/Lab/scripts$ curl -i -X OPTIONS http://testfire.net
HTTP/1.1 200 OK
Allow: OPTIONS, TRACE, GET, HEAD, POST
Server: Microsoft-IIS/8.0
Public: OPTIONS, TRACE, GET, HEAD, POST
X-Powered-By: ASP.NET
Date: Sun, 17 Jun 2018 14:02:00 GMT
Content-Length: 0
```

Dangerous methods like TRACE and PUT should not be allowed. On exploitation of PUT, check out NMAP scripts, tools like burp and browser add-ons like poster.

b. NMAP

Weaponizing nmap scripts can come in handy.

```
sylar@wild_thought:~/Lab/scripts$ nmap -p80,443 --script=http-methods testfire.net

Starting Nmap 7.60 ( https://nmap.org ) at 2018-06-17 17:07 EAT
Nmap scan report for testfire.net (65.61.137.117)
Host is up (0.29s latency).

PORT    STATE SERVICE
80/tcp  open  http
| http-methods:
|   Supported Methods: OPTIONS TRACE GET HEAD POST
|_  Potentially risky methods: TRACE
443/tcp open  https
| http-methods:
|   Supported Methods: OPTIONS TRACE GET HEAD POST
|_  Potentially risky methods: TRACE

Nmap done: 1 IP address (1 host up) scanned in 3.77 seconds
```

c. My rudimentary curl script

I wrote this simple script to print out the response headers for a list of servers in a text file. However if the OPTIONS method is enabled on the server, we can get the list of allowed methods on the server. See it here. This includes a simple http(s) check using wget for the list of servers. As usual, this can be improved/modified.

```
================================================
===  Curl output for: twitter.com  ===
================================================
HTTP/2 405
allow: GET,HEAD
cache-control: no-cache
content-length: 0
date: Wed, 09 Jan 2019 14:28:02 GMT
server: tsa_o
set-cookie: personalization_id="v1_eZAjseYbDznmTTK9ofZUUQ=="; Max-Age=63072000; Expires=Fri, 8 Jan 2021 14:28
witter.com
set-cookie: guest_id=v1%3A154704408213517905; Max-Age=63072000; Expires=Fri, 8 Jan 2021 14:28:02 GMT; Path=/;
status: 405 Method Not Allowed
strict-transport-security: max-age=631138519
================================================


================================================
===  Curl output for: testfire.net  ===
================================================
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=D6092FCABC01BC34A2C6BB81216655CE; Path=/; HttpOnly
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 6490
Date: Wed, 09 Jan 2019 14:28:04 GMT
```

N/B. netcat, nikto can also be used for this.

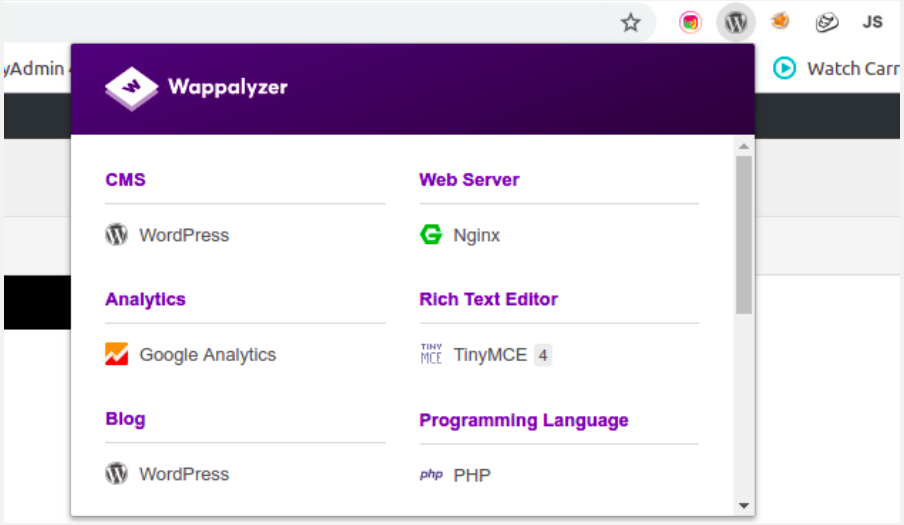
culprit: Application Mapping

In an attempt to attack an application, we have to understand its working, architecture and underlying technologies.

Identifying technology used

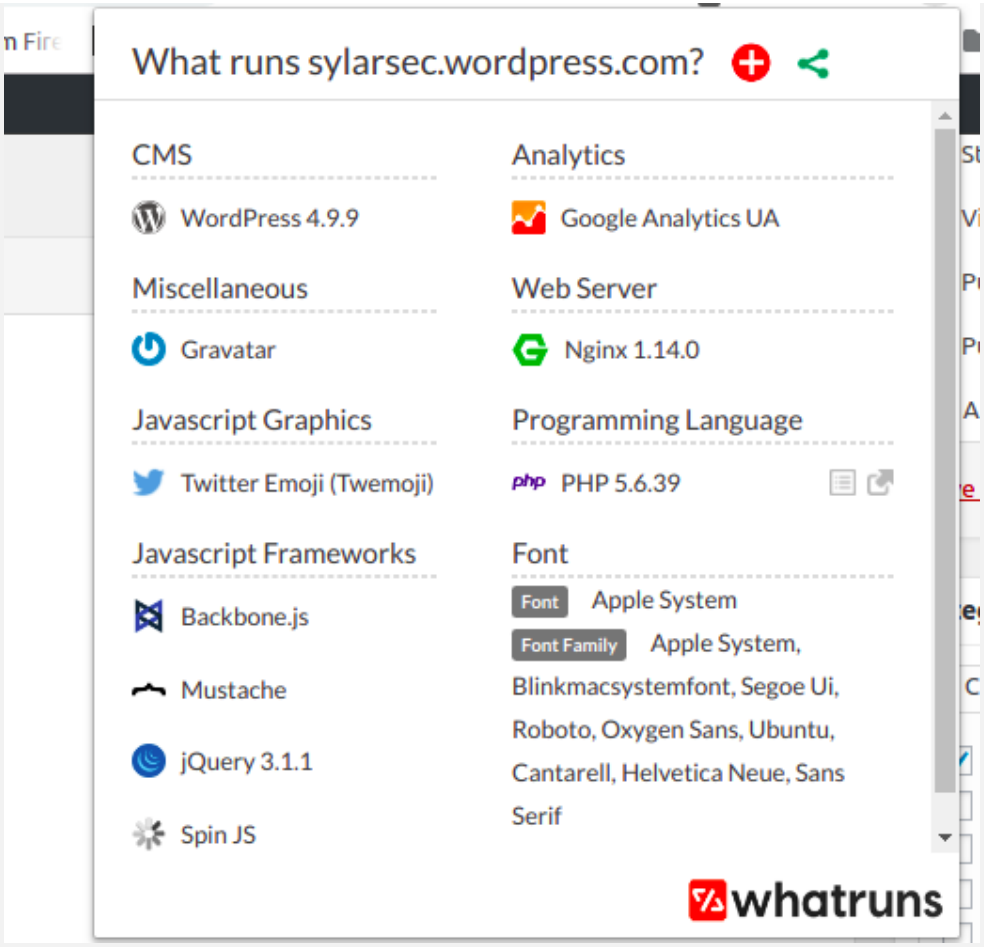
- **Wappalyzer**

This is a browser extension that identifies an applications underlying technologies. This may include the language used, development frameworks, CMS, analytics frameworks etc. It runs on both firefox and google chrome.



- **Whatruns**

Another browser plugin that works the same way. Just a bit more aggressive.



- **WafW00f (Firewall discovery)**

So we want to actively interact with the target. However different probes might get blocked by a possible security solution like a WAF. If so we can identify the WAF in

use by using sandrogauci ‘s tool, WafW00f that can be found here.

```
xpl0r3r@xpl0r3r-ThinkPad-T460s:~$ wafw00f testfire.net

                                  ^     ^
       _   __  _   ____ _   __  _    _   ____
      ///7/ /.' \ / __////7/ /,' \ ,' \ / __/
     | V V // o // _/ | V V // 0 // 0 // _/
     |_n_,'/_n_//_/   |_n_,' \_,' \_,'/_/
                                <
                                ...'

    WAFW00F - Web Application Firewall Detection Tool

    By Sandro Gauci && Wendel G. Henrique

Checking http://testfire.net
The site http://testfire.net is behind a ModSecurity (OWASP CRS)
Number of requests: 11
```

Content discovery

- **dirb**

Very comprehensive directory/file bruteforce tool that uses a custom word list to find the directories or files that exists. This happens to be my favorite.

```
---- Scanning URL: https://testfire.net/ ----
+ https://testfire.net/admin (CODE:302|SIZE:0)
+ https://testfire.net/aux (CODE:200|SIZE:0)
+ https://testfire.net/bank (CODE:302|SIZE:0)
+ https://testfire.net/com1 (CODE:200|SIZE:0)
+ https://testfire.net/com2 (CODE:200|SIZE:0)
+ https://testfire.net/com3 (CODE:200|SIZE:0)
+ https://testfire.net/con (CODE:200|SIZE:0)
+ https://testfire.net/docs (CODE:302|SIZE:0)
+ https://testfire.net/images (CODE:302|SIZE:0)
+ https://testfire.net/lpt1 (CODE:200|SIZE:0)
+ https://testfire.net/lpt2 (CODE:200|SIZE:0)
+ https://testfire.net/manager (CODE:302|SIZE:0)
+ https://testfire.net/nul (CODE:200|SIZE:0)
+ https://testfire.net/pr (CODE:302|SIZE:0)
+ https://testfire.net/prn (CODE:200|SIZE:0)
+ https://testfire.net/static (CODE:302|SIZE:0)
+ https://testfire.net/util (CODE:302|SIZE:0)
```

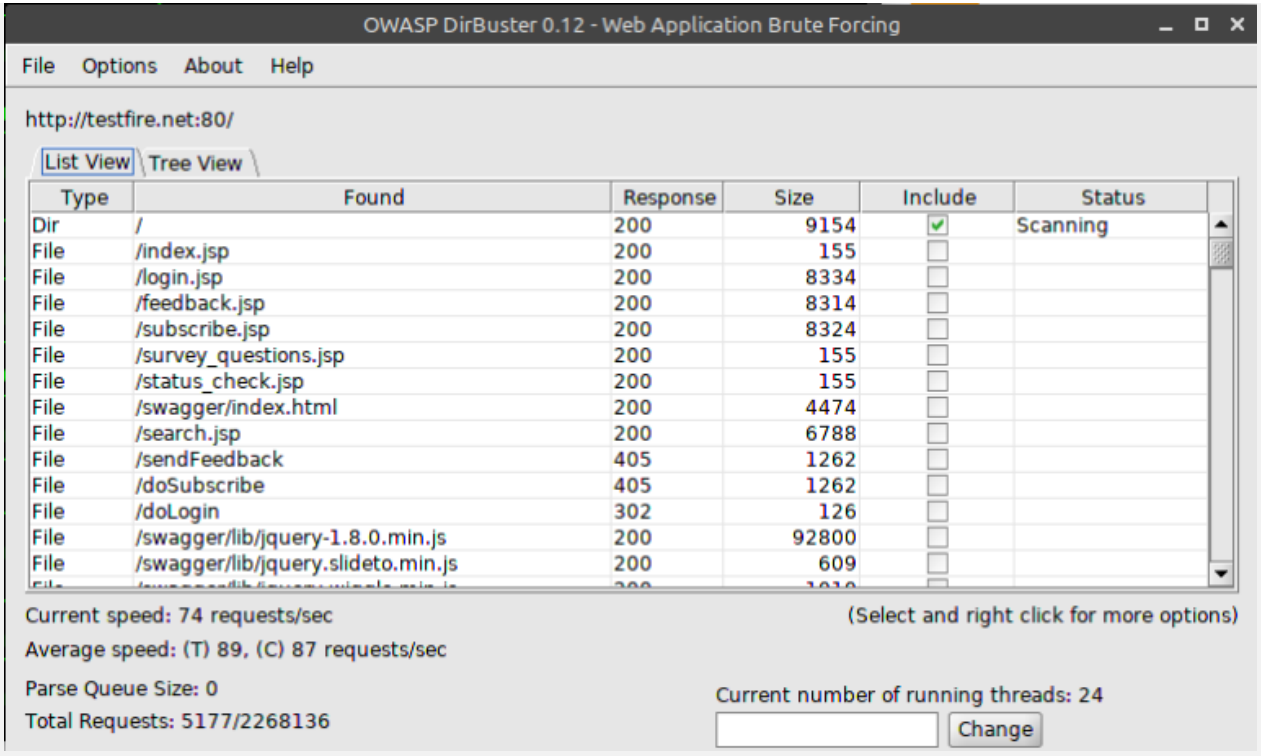
- **dirsearch**

Similar to dirb but with some fancy colors for easier status identification. Searches for both files and directories as well. This has the ability to specify extensions. e.g php, txt, rar, zip etc.

```
[18:29:46] 302 -    0B  - /admin/user_count.txt  ->  /login.jsp
[18:29:46] 302 -    0B  - /admin/web/  ->  /login.jsp
[18:31:29] 302 -    0B  - /docs  ->  /docs/
[18:31:29] 200 -   19KB - /docs/
[18:32:09] 302 -    0B  - /images  ->  /images/
[18:32:13] 200 -    9KB - /index.jsp
[18:32:34] 200 -    8KB - /login.jsp
[18:32:42] 302 -    0B  - /manager  ->  /manager/
[18:32:42] 302 -    0B  - /manager/  ->  /manager/html
[18:32:42] 401 -    2KB - /manager/html
[18:32:43] 401 -    2KB - /manager/html/
[18:33:29] 302 -    0B  - /pr  ->  /pr/
[18:34:06] 302 -    0B  - /static  ->  /static/
```

- **dir buster**

Another one by OWASP. With a cool looking GUI, it does file and content discovery with an option to specify custom word list. Also comes with a cool set of word lists. Can be found here



- **nikto**

From banner grabbing, header analysis, light default directory/file discovery, nikto is pretty handy. Also offers some suggestions and advisory info for why the discovered issues are dangerous.

```
xpl0r3r@xpl0r3r-ThinkPad-T460s:~$ nikto -h http://testfire.net
- Nikto v2.1.5
---------------------------------------------------------------------------
+ Target IP:          65.61.137.117
+ Target Hostname:    testfire.net
+ Target Port:        80
+ Start Time:         2019-01-10 23:26:02 (GMT3)
---------------------------------------------------------------------------
+ Server: Apache-Coyote/1.1
+ The anti-clickjacking X-Frame-Options header is not present.
+ Cookie JSESSIONID created without the httponly flag
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Allowed HTTP Methods: GET, HEAD, POST, PUT, DELETE, OPTIONS
+ OSVDB-397: HTTP method ('Allow' Header): 'PUT' method could allow clients to save
+ OSVDB-5646: HTTP method ('Allow' Header): 'DELETE' may allow clients to remove fil
+ DEBUG HTTP verb may show server debugging information. See http://msdn.microsoft.c
+ Server leaks inodes via ETags, header found with file /lpt9, fields: 0xW/0 0x0
+ /manager/html: Default Tomcat Manager interface found
+ 6544 items checked: 0 error(s) and 8 item(s) reported on remote host
```

- **Aquatone**

It gives a visual representation of the websites listed on a text file. This helps easily map out the best attack surface. For example, makes it easy to find login pages without manually visiting the pages. It takes screenshots of the pages and saves them to a folder. Also includes headers.

It also has other modules i.e. scan, discovery, gather, takeover that will be discussed on part 2 of this blog.

```
xpl0r3r@xpl0r3r-ThinkPad-T460s:~/Lab/scripts/Aquatone$ cat hosts.txt | ./aquatone
aquatone v1.4.3 started at 2019-01-10T00:42:36+03:00

An older version of Chromium is installed. Screenshotting of HTTPS URLs might be unreliable.

Targets    : 8
Threads    : 4
Ports      : 80, 443, 8000, 8080, 8443
Output dir : .

https://localhost.testfire.net: 200 OK
https://ftp.testfire.net: 200 OK
https://evil.testfire.net: 200 OK
https://testfire.net: 200 OK
https://demo2.testfire.net: 200 OK
https://www.testfire.net: 200 OK
https://demo.testfire.net: 200 OK
https://altoro.testfire.net: 200 OK
https://evil.testfire.net: screenshot successful
https://localhost.testfire.net: screenshot successful
https://ftp.testfire.net: screenshot successful
https://testfire.net: screenshot successful
https://demo.testfire.net: screenshot successful
https://demo2.testfire.net: screenshot successful
https://www.testfire.net: screenshot successful
https://altoro.testfire.net: screenshot successful
```

- **burp intruder**

Burp’s intruder also serves as a multipurpose tool. In this context it can be used to bruteforce files, directories, GET params etc while observing status codes as well as content length.

Request	Payload	Status	Error	Timeout	Length	Comment
0		200			9167	
1	connect	200			9167	
2	contacts	200			9167	
3	container	200			9167	
4	content	200			6658	
5	content1	200			9167	
6	contents	200			9167	
7	contentType	200			9167	
8	decode	200			9167	
9	decoded	200			9167	
10	decomposition	200			9167	
11	decrypt_key	200			9167	
12	def	200			9167	

Request | Response

Raw | Params | Headers | Hex

```
GET /index.jsp?content= HTTP/1.1
Host: demo.testfire.net
```

- **The perfect wordlist for the job**

All these tools wont give a heavy punch without a good set of word lists. From my research i discovered seclist. Probably as comprehensive as it gets. Coupled with different usernames, passwords, URLs, payloads etc. It earns the ‘ultimate wordlist’ title.

Discovery	Merge pull request #262 from g0tmi1k/websphere	4 days ago
Fuzzing	Merge pull request #237 from s0md3v/patch-1	3 days ago
IOCs	Fix #259 - Recover from bad merge	4 days ago
Miscellaneous	Fix #259 - Recover from bad merge	4 days ago
Passwords	Merge pull request #258 from henry701/patch-1	4 days ago
Pattern-Matching	Fix #259 - Recover from bad merge	4 days ago
Payloads	Fix #259 - Recover from bad merge	4 days ago
Usernames	Fix #259 - Recover from bad merge	4 days ago
Web-Shells	Fix #259 - Recover from bad merge	4 days ago
.gitignore	Quick move about	7 months ago
CONTRIBUTING.md	Update CONTRIBUTING.md	7 months ago
CONTRIBUTORS.md	Sort out README	3 months ago
LICENSE	Create LICENSE	7 months ago
README.md	Typo	4 days ago

N/B: Discovering ‘hidden’ GET/POST parameters.

During pentesting or bug bounty hunting, the best way to attack a page is on inputs. Hence parameters are really important. If we cant find them on the first look, its possible to try and find the ‘*hidden*‘ parameters using different tools.

- **Arjun**

One tool by UltimateHackers comes to mind. Arjun is a script that helps bruteforce these parameters using a word list that can be customized.

```
xpl0r3r@xpl0r3r-ThinkPad-T460s:~/Lab/scripts/Arjun$ python3 arjun.py -u http://testfire.net/index.jsp --get
    _
   /_| _ '
  (  |/ /(//) v1.0
      _/
[~] Analysing the content of the webpage
[~] Now lets see how target deals with a non-existent parameter
[!] Reflections: 0
[!] Response Code: 200
[!] Content Length: 8985
[~] Parsing webpage for potenial parameters
[+] Heuristic found a potenial parameter: query
[+] Prioritizing it
[!] Content Length Multiplier: 0
[+] Found a valid parameter: content
[!] Reason: Different number of reflections
[!] Progress: 3370/3370
[!] Scan Completed
```

- **Parameth**

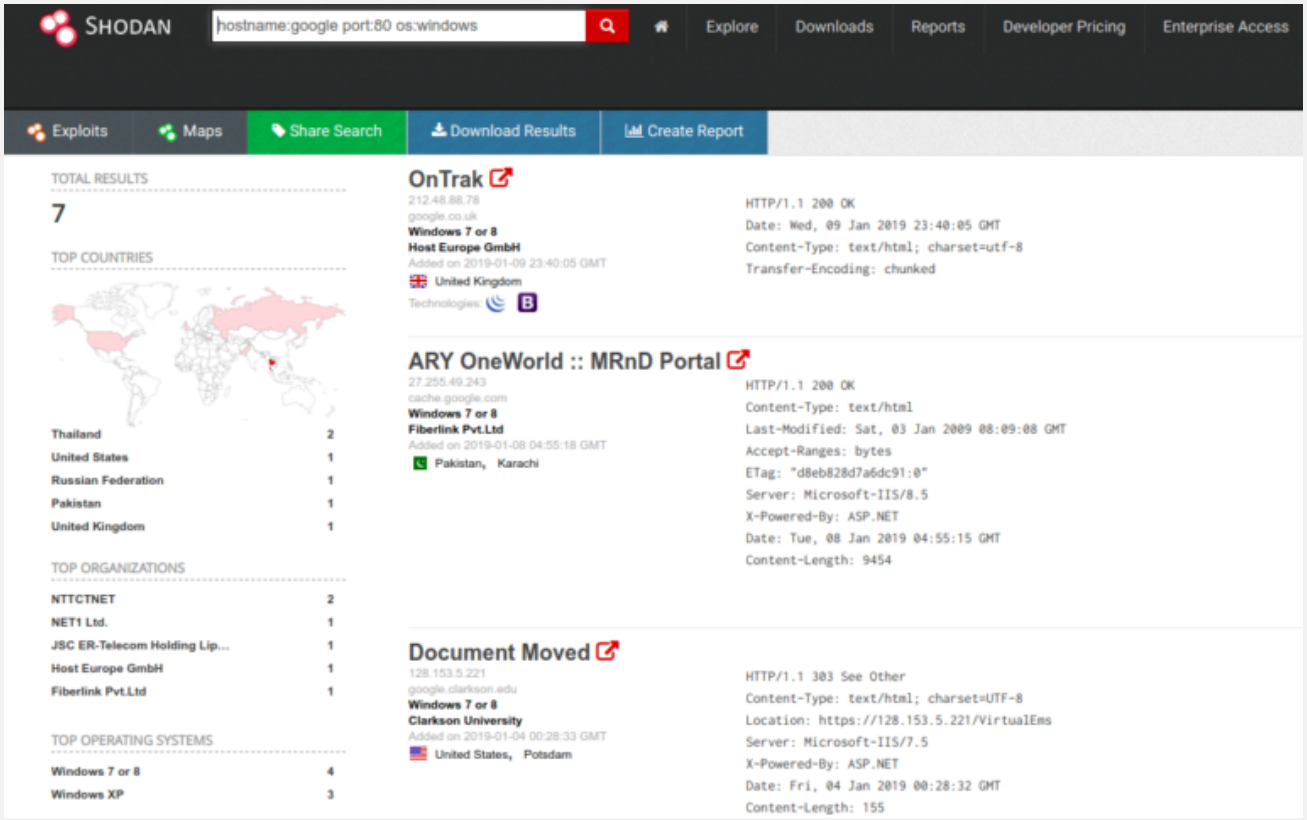
This one worked for me a while back. Does the same. But i still prefer Arjun.

culprit: Other search engines

While google might be the most popular search engine, its not the only one. And if you’re gonna be finding vulnerabilities, then you most likely need these 2 as well…

- **shodan (shodan.io)**

Hands down the ultimate IoT search engine. Same as Google, it uses ‘dorks’ for smarter searches and improve on what it finds. Its right here.



Some common dorks may include:

- ***country***: *find devices in a certain country*
- ***hostname***: *find devices matching the given hostname*
- ***port***: *find devices on given open ports*
- ***os***: *return results that match the given OS*
- ***before/after***: *find results within a given time frame*
- ***city***: *find devices in a certain city*

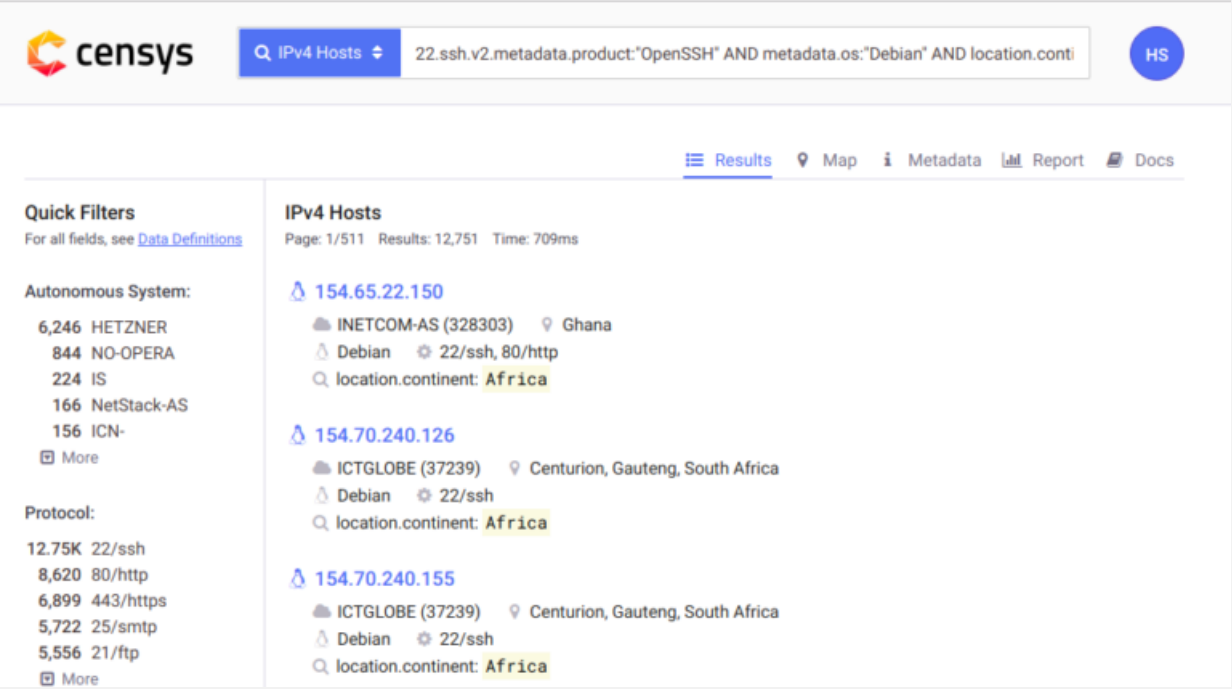
- **Censys (censys.io)**

Kinda like shodan, it compares to the fact that it can also search for devices accessible from the internet.

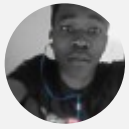
Lets find *debian* servers running *ssh* from *Africa* using the query

---

"22.ssh.v2.metadata.product:"OpenSSH" AND metadata.os:"Debian" AND location.continent:"Europe"



Have fun with the 100 ways of discovery. Part 2 coming soon.



Published by sylar

Guardians of the web

View all posts by sylar

Web

1 thought on “100 ways to discover (part 1)”

Felipe March 7, 2019 — 1:57 am

Your website has outstanding material. I bookmarked the website

REPLY

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Post Comment

PREVIOUS

Show me thy XSS abilities, polyglot!

NEXT

Vulnerability Assessment and Management with Archerysec

Search …