

Bug-Hunting-Tips/Tricks

Mar 24, 2019

Here I will write all the tips tricks i got from twitter and even advices which when i got from others and daily update this blog post

Bug Hunting Tips 1 [By Jason Haddix] :→

Bounty Tip

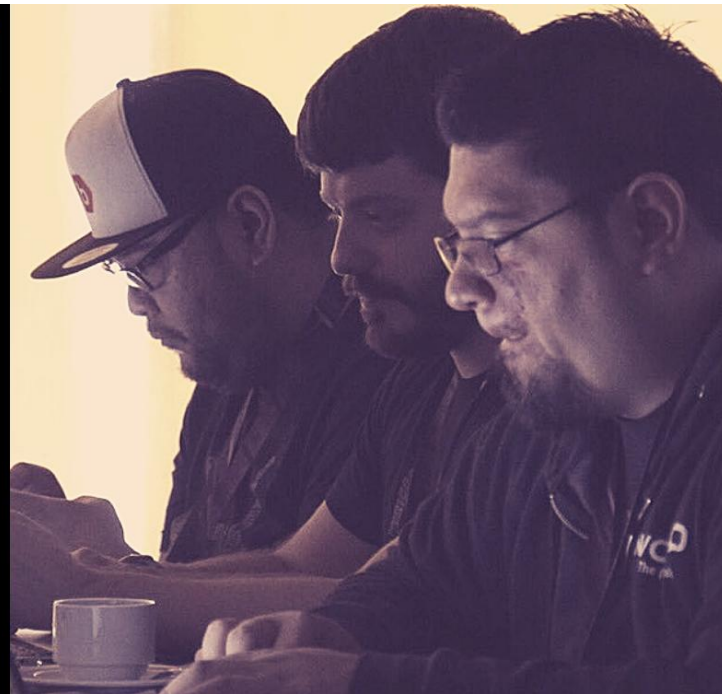
01

Functions that handle paths, links, and domains as part of a parameter or input are a high priority to bounty hunters. They are often subject to SSRF, Open Redirect, Path Traversal, or LFI.

`https://someapp.com/src?redir=something.com`

`https://someapp.com/src?site=http://something.com`

`https://someapp.com/src?dir=/path/to/something`



=====

Bug Hunting Tip 2 [By Jason Haddix] :→

Bounty Tip

02

"Content discovery" is trying to guess sensitive paths and files that might exist in the application but are not linked anywhere.

Often when doing this part of web assessment you will run into 401 Not Authorized responses. It is beneficial to recursively brute force that path. Often resources past that path have not had the same access controls applied. In addition, 401 replies should be investigated with waybackmachine (<https://archive.org/web/>) to see if they ever did not have authentication applied and to garner clues about the application pathing.

```
https://someapp.com/admin/          401
https://someapp.com/admin/dashboard/ 401
https://someapp.com/admin/dashboard/members 200
```

```
3] 301 - 146B - /models/emssql -> https
4] 301 - 146B - /models/fclicksql -> ht
8] 301 - 146B - /models/javatosql -> ht
2] 301 - 146B - /models/nukesql -> http
8] 301 - 146B - /models/settings_sql ->
10] 301 - 146B - /models/swissql -> http
1] 302 - 132B - /models/BLOCKS -> /mode
5] 302 - 136B - /models/ViewModels -> /
3] 301 - 146B - /models/backups_mysql ->
8] 301 - 146B - /models/cache_sql -> ht
12] 301 - 146B - /models/errormysql -> H
13] 301 - 146B - /models/ez_sql -> https
13] 301 - 146B - /models/ezsql -> https:
2] 301 - 146B - /models/htmlsql -> http
7] 301 - 146B - /models/kaybasql -> htt
0] 301 - 146B - /models/linkssql -> htt
5] 301 - 146B - /models/php-mysql -> ht
6] 301 - 146B - /models/plsql -> https:
5] 301 - 146B - /models/rpsql -> https:
5] 301 - 146B - /models/testmysql -> ht
5] 301 - 146B - /models/testsql -> http
16] Starting: controllers/
17] 301 - 146B - /controllers/.pgsql ->
17] 301 - 146B - /controllers/.mysql ->
18] 301 - 146B - /controllers/sql -> htt
0] 301 - 146B - /controllers/mysql -> H
1] 301 - 146B - /controllers/_sql -> ht
2] 301 - 146B - /controllers/_mysql ->
7] 301 - 146B - /controllers/error_mysql
0] 301 - 146B - /controllers/.psql -> H
6] 301 - 146B - /controllers/websql ->
1] 301 - 146B - /controllers/db_mysql ->
5] 301 - 146B - /controllers/ntunnel_mysq
5] 301 - 146B - /controllers/phpmysql ->
Last request to: return product
```

Bug Hunting Tips 3 [By Jason Haddix] :→

Check out @BountyBurp - <https://github.com/wagiro/BurpBounty/> and @egarme's port of LinkFinder (by @gerben_javado)

<https://github.com/ghsec/BBProfiles/blob/master/PASSIVE-EndpointsExtractor.bb>

https://www.youtube.com/watch?v=ELftJwkY_e0 <https://github.com/GerbenJavado/LinkFinder>

I'm gonna run out of good pics of myself!

Bounty Tip

03

Creating your own Burp Suite active and passive checks has never been easier!

Check out the extension "**Burp Bounty**" which allows you to create simple scanner checks. Two of my favorites are:

1) The regex that is from **LinkFinder** by **Gerben Jervado** which is used to parse full urls, absolute urls, dotted, relative with slash, and relative without slash urls from **javascript**. Using a scanner rule brings this info in closer to Burp and makes scanning a whole set of domains faster.

2) Creating a regex rule for alerting when Burp sees **AWS access keys**:

```
(?<![A-Z0-9])[A-Z0-9]{20}(?![A-Z0-9])  
(?<![A-Za-z0-9/+])[A-Za-z0-9/+]{40}(?![A-Za-z0-9/+])
```



TABLE I
REGULAR EXPRESSIONS FOR API KEYS OF VARIOUS SERVICE
PROVIDERS [7].

Service Provider	Client ID	Secret Key
Amazon AWS	AKIA[0-9A-Z]{16}	[0-9a-zA-Z/+]{40}
Bitly	[0-9a-zA-Z_]{5,31}	R_[0-9a-f]{32}
Facebook	[0-9]{13,17}	[0-9a-f]{32}
Flickr	[0-9a-f]{32}	[0-9a-f]{16}
Foursquare	[0-9A-Z]{48}	[0-9A-Z]{48}
LinkedIn	[0-9a-z]{12}	[0-9a-zA-Z]{16}
Twitter	[0-9a-zA-Z]{18,25}	[0-9a-zA-Z]{35,44}

Don't Forget ->

from https://people.eecs.berkeley.edu/~rohanpadhye/files/key_leaks-msr15.pdf

=====

Recon Tips

Credit -> <https://twitter.com/stokfredrik/status/1109733020540567555>

1. By @stokfredrik

I always look up asn and ipranges on ripe, if it's a small scope or if I just want a feeling for it I run it threw domained/aquatone and sort the sizes of the screen shots to look for anomalies, (link: <https://github.com/cak/domained>) <https://github.com/cak/domained> (link: <https://github.com/michenriksen/aquatone>) <https://github.com/michenriksen/aquatone>

2. By @imhaxormad :-> My Recon tips :->

1. Wayback machine plugin in burp gives you some sweet endpoints, I had managed to find an RCE via an abandoned test page for a pentest client.
2. Look the source code, sometimes you might find an open redirection out of nowhere!
3. Run Gobuster+dirb+Burp spider

3. By @D0rkerDevil

1. sniper tool(everything u need)
2. gitrob or similar tools which does the github recon jobs
3. shodan and censys

4. By @berg0x00

1. To find other domains that the target own I use the reverse whois lookup over at (link: <http://viewdns.info>) viewdns.info.
2. I also look at what other domains are on the TLS certificates that have been created for the target domains.
3. Use wayback and (link: <http://commoncrawl.org>) commoncrawl.org

5. By @bit3c0de

1. Google dork - for indexed pages and endpoints
2. Dirsearch and gobuster - path/file discovery especially when i run into a 401(not authorized)
3. Arjun - discover hidden get/post parameters in urls
4. Wayback machine - especially for decommissioned pages but live endpoints
5. Also aquatone and amass. Gave a short write about them here :-> <https://sylarsec.com/2019/01/11/100-ways-to-discover-part-1/>

6. By @0xINT3 -> What I do:->

1. Find all subdomains using tools such as knockpy, sublist3r, amass, aquatone and sort them uniquely in a file.
 2. Pick a subdomain of your interest and run a burp spider in it with wayback plugin.
 3. On that subdomain run a directory fuzzing, and find entry points.
7. By @hossams01284251 :-> My recon tips is :->
1. focus on the web server bugs as the web application because most of people dont focus on web server
 2. allways dig deeper as much as you can to find juicy bugs
 3. dont depend on the tools 100% allways when work manually for more bugs

8. By @S4R1N

Tools : subfinder, massdns + dirsearch / eyewitness Start by finding interesting subdomain, then spider / bruteforce it (i use dirsearch across multiple subdomains). From there start digging more into host for interesting functionality or files.

1. By @JarPhish :->

1. Know your target machine, open port. Find it with wappalyzer/builtwith and nmap/masscan
2. Find subdomains, directory, path & parameters with amass, gobuster, Arjun, JSparser
3. Find more with google dorks, exploitdb, github.

2. By @mufeedvh ->

Some Recon Tips:

1. Google Dorks: search for common parameters and directories.
2. Collect parameters as you surf the target, these parameters may respond in other pages too that you can chain them to get an awesome bug (mainly in account settings).
3. Check "robots.txt".

3. By @JavoPagano

1. Spider to find all end points posible
2. Dirb it can give you some interesting pages
3. Sub domain search can be helpfull too

4. By @kh4st3x

Run your tools using VPS servers instead of local for turbo speed

=====

Tweet by Tom Joseph

[https://twitter.com/bot_infosec/status/1114035613655654400]

Does Anyone know some good vulnerability based automation scanners?

1. SQLMAP => SQL Injection
2. Commix => Command Injection
3. TPLMAP => SSTI

What's more??

Others reply ->

1. By Abhinav [@0xINT3]

Gitrob: GitHub reconnaissance Knockpy: Subdomain lister SAML Raider: SAML (burp plugin) XXEinjector:
XXE xsshunter: Blind XSS

I'll add more. :)

2. By Rakesh Mane @RakeshMane10

xcat => XPath Injection

3. By Arif Khan

- XSSStrike by @s0md3v for XSS.
- Photon, Spiderfoot for #OSINT
- Arjun for HTTP parameter discovery
- Aquatone for Recon

4. By lx43 Kadimus for LFI

5. XSSer = XSS

