Bug Bounty

- Categories:

40/20	20/10	10/2	5 w HOF/1
P1: critical,	P2: high,	P3: medium,	P4: low.
1000+.	300-500	150-250	50-150

- Bounties:

Hall of fame, Swag: vouchers or goodies.
bounty:
appreciation letter.

- Bugcrowd, Hackerone, Intigrity, linked in, twitter.

- Bugcrowd - Researcher Portal.

- Pentester, Security Auditor.

- Report format:

$$\left.\begin{array}{l} \text{i) Vulnerability name} \\ \text{ii) vulnerability Desc} \\ \text{iii) Impact} \\ \text{iv) POC} \\ \text{v) Solution.} \end{array}\right\}\ \text{dmarc hackerone}$$

Bug 1 : Dmarc or SPF

Checking if the email domain of a particular company is secure. Just enter the email domain

mx-toolbox / mx-lookup

output: If dmarc policy enable : Secure

Solution:

On finding vulnerabilities: (on bugcrowd)	
In scope:	Out Scope:
Inside the Scope of my doing. It is what I have been told	vice - versa.

Bug crowd. Com / taximony to find the category of a bug.

Bug 2: SPF

kitterman . Com .

If SPF record not found, then ~~not~~ vulnerable. Else Secure

Bug 3: Session does not expire after password update. (P4).

Suppose my account is open on two devices. I change my password in one of the devices. If the session does not log out, then vulnerable.

✓ Session does not expire after password reset.

- forgot password.
- log into your account ~~with no~~
- Reset password using password reset / forgot.
- If session does not expire, then vulnerable.

Session does not expire after delete account

- Open same account into two diff browsers or devices
- Delete account from one
- Session does not expire after delete acc.

Wappalyzer: To determine what tools are used to build and host website.

Bug 4: Session Hijacking

If I login my credentials get stored in cookies temporarily.
If they don't get deleted after session end, then the third person can log in without my credentials.

Extension: Cookie Editor to attack.

- log in to your account
- export cookies
- log out
- Import the same cookies.
- Reload.
- If logs in, then vulnerable else secure

Bug 5: Account Deletion

Whenever you log into your account, check for primary ~~account~~ actions.

If there is no user side confirmation, then site can be vulnerable.

Eg: Delete account.

Bug 6 : Geolocation image data not stripped.

- If I click photo, location appears on photo or photo details.
- If I upload that photo anywhere and location is visible then vulnerable.

Impact : My location can be disclosed, and theft can occur if I am out somewhere at my home.

steps to reproduce :- Find sites where image can be uploaded.
- upload image
- Copy address of image
- If location visible, vulnerable.

check on : exif. regex. info / exif. cgi.

Bug 7: DNS misconfiguration.

Cond : Enter ping localhost . domain . com.

~~If reply then not vulnerable.~~

occrp.org

If reply then copy IP to browser
If works then vulnerable

Bug : 8 : Clickjacking

Basic : If website opens in iframe, then the website is vulnerable for clickjacking.

Impact : Attacker can make user click the buttons or links of their use or advantage or for the loss of user.

This goes for higher impact because account can be manipulated by attacker without user knowing it.

[illegible] :

Account login :

- login Account
- Select the URL for login or any big option.
- make the link for Delete account for iframe..

Solution :

Bug 9: DOS

long password DOS attack

DOS: Denial of service. (Single user)

user sends a request continuously and the load on server increases.
i.e. 1 user 10000 requests send karta, if so many user then server down.

DDOS: ~~Distributed~~ Distributed Denial of service
(Multiple User).

Solution: rate limit.
How, limit requests.
eg: 5 requests per unit time mostly seconds.

If set password has limit then secure else vulnerable.
This is because more characters in password use more memory.

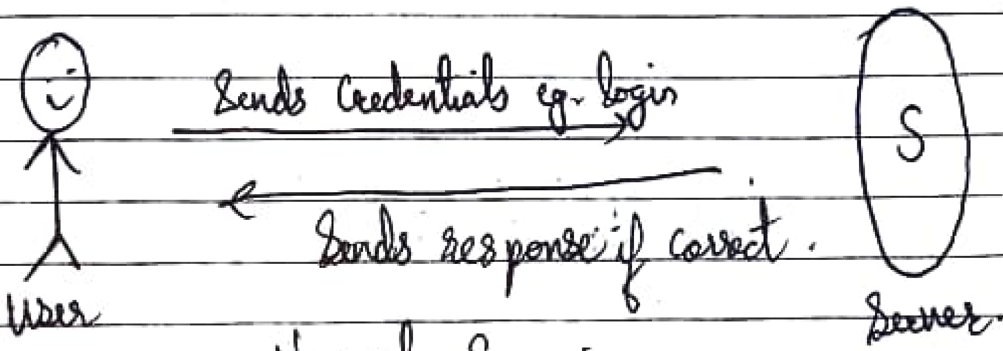
steps:
i) Enter 1000 char password.
ii) Check time taken to sign up.
If more then report.

1st condition is timetaken.
2nd is :

Enter 1000 char password
If does not login, still vulnerable.

Bug 10: CORS

Cross origin resource sharing. [P3, P4]



Normal Scenario

If man in the ~~mo~~ middle asks for the access to the server, and server gives it then CORS vulnerable.

CORS: Manually [redacted] and Burp-Suite.

Steps to reproduce: CORS mostly found on wordpress site.

i) write domain / wp-json
If large amount of data is visible then CORS vulnerable

This does not include man in the middle.

Report:

i) ~~Vul Name~~: In Command prompt.

SS attach

```
curl "https://www.bluescape.com/wp-json" -I.

curl "https://www.bluescape.com/wp-json/" -I -H
    Origin: ~~https://hacktify.in~~
            https://any website domain.
```

If Access - Control - Allow - Credentials : true
then vulnerable

then Copy the website domain in
CORS code in GET.

Run that code in browser, click.
if exploits then you get the access by CORS.
and makes the site ~~vul~~ vulnerable.

Solution: Remove garbage data.

[illegible] . .

Bug 11 : Information Disclosure.

Steps:
i) Enter domain/wp-json/wp/v2/users.
If information visible then vulnerable.

If sensitive info leak by any purpose, still vulnerable for information disclosure.

Bug 12 : Basic login testing

When signing up Enter ~~[illegible]~~ number in name
if accepted, vulnerable

Bug 13 : Broken Authentication

steps: click on forget password twice.
change password on one link.
if the other link expires then safe else vulnerable..

steps: ask for forgot password link
~~change password~~.
login to account change password, then logout
again use the forgot password link and try resetting password.
If password resets, then vulnerable.

Bug 14 : HSTS

HTTP strict transport security.

Steps : open HSTS preload
Enter domain.
If no HSTS, stage -1 - vulnerable.

open Cmd.
Enter curl - I http:// Domain
if status code 302, then safe
if status code 200, then vulnerable

Bug 15 : Web Cache Deception.

P4, rare.

There is load balancer between client and server to decrease load ~~between~~ on the server.

client . . . . . . load balancer . . . . . Server

→ Sends request → checks if any data is available.
if yes
← replies
if not
→ sent for server to processor . . .

Steps to reproduce :

i) Enter name.anyextension at new page

eg: www.google.com/ Sanket.css or Sanket.html

In normal cases, an error message is shown. ie. file not found.

But if vulnerable, then same page loads again.

ii) Enter anyextension at existing page.

eg: www.google.com/about css or about.htm

If same page loads then vulnerable.

iii) Enter .extension at existing page.
eg: www.google.com/about.css

Stage 2

Copy the URL and paste it onto new browser
if the page loads, ~~then vulnerable~~ and details visible, then vul.

Impact: I was logged in on chrome still the page loads on firefox without any credentials.

~~Section - 2~~

Found on forgot password, confirmation email, invite ~~other users,~~

Bug 16: No Rate limit → P4.

Also called Brute force attack, email bomb

Suppose I know anyone's mail Id. I go onto flipkart, enter email Id and then click on forgot password 1000s of times. The mail address receives so many emails so there is not rate limit for a particular action.

Impact:

Bussiness loss: Someone has to pay for these emails. obvio flipkart

User loss: I won't recommend anyone to use flipkart.

~~Cost~~ Customer Satisfaction lost.

Solution: Implement rate limit.
as in add limit to do any particular action per unit time.

Brute force attack:
Ransomware
DDOS
DOS
Cryptography.

} Imp Que for all things.

OWASP Top 10

Steps to reproduce: (In Mozilla)

i) Enter email on forgot password.
ii) Switch on Burp Suite.
iii) Switch Intercept on
iv) ~~Submit~~ Click on Submit forgot pass.
v) Search for request with email.
vi) Send to intruder
vii) Clear intruder and add "$$" in front of user agent.
viii) Select payload type → brute force.
ix) Select rate 16.
x) Start attack.
xi) Check if you have 16 emails.
if yes vulnerable..

Bug 17: Captcha Bypass.

steps to reproduce:

i) On sign up form you will find captcha
ii) fill the form, solve captcha, click on Sign up.
iii) At this time keep Burp on with intercept on
iv) when you capture the request, delete the captcha response.
v) forward, forward.
vi) intercept off
vii) click on reload. If account gets created then vulnerable.

Impact:

Creation of fake accounts.

Bug: 18 : Authentication Bypass → P1

Also called as server misconfig.
Authentication: verification of credentials.

in short OTP bypass

what we do: we try login using wrong OTP by manipulating the request.

OTP is used to verify actual human being. This is because login credentials can verify themselves, but OTP is the only way to verify actual human.

steps to Reproduce:

i) Sign up with the required credentials.
ii) you see an OTP screen saying "Enter OTP".
iii) Switch on Burp Suite. Intercept on.
iv) Enter random OTP and click on submit
v) Find appropriate request in Burp by clicking forward.
vi) Right click → Do intercept → Response to this

vii) Again click forward until you find status for OTP.

viii) Change the requests in positive.
Such as: Status: true / 1.
Incorrect OTP → Correct OTP
Invalid → Valid.

ix) Click on forward until you get blank burp page

x) Intercept off →

xi) Reload page, you should be logged in.

Impact: fake account can be created without auth.

How to find OTP page on BURP: check for status code 200.

Bug 19: Parameter Tampering

mostly found on E-Commerce websites.
Tampering with the basic parameters such as price or quantity, which may lead to business loss.

Try changing price and quantity parameter to perform the attack.

<u>steps to reproduce:</u>

i) login into your account.
ii) Add some items to cart.
iii) Turn on Burp Suite and make intercept on.
iv) Find the request by clicking forward until you find parameters.
v) try changing the price or quantity.
vi) Intercept off
vii) Check if the parameters are changed on actual website.
viii) If the parameters are changed, then vulnerable

<u>Solution</u>: Add specific things for the price doesn't change.

<u>Bug 20: File upload</u> → P4 / P3

P3 if the file can bee opened

If the upload input field goes beyond the listed parameters, then it is vulnerable.
For eg: A pdf is required and I upload mp4.
<u>Four ways</u>: i) Content Spoofing
ii) direct upload
iii) double extension.
iv) Content spoofing and double extension.

i) direct upload:

a) try uploading file with a diff. extension.

ii) Content Spoofing:

a) checking the content of the file to satisfy the requirements.
b) Switch on Burp with Intercept on
c) Capture the upload request.
d) How to see if the request is valid - contents are shown
e) Search Content type: octet stream. change to valid extensions.
f) forward req.
g) check if the file gets uploaded.

iii) double extension

a) Same as above
b) Change the ~~f~~ file extension in Burp.
c) add valid extension.
d) forward req

iv) Content Spoofing

a) Same as above
b) change content type to valid extension.
c) ~~clear~~ ~~Add~~ extension as well.
d) forward req.

In any case, if the file gets uploaded then vulnerable.

P5 → P2.

Bug 21: Cross Site request forgery

what we do: Attacker sends the malicious script to the client. If client clicks, then the account settings can be changed.

~~Forgot~~ For eg: My facebook login credentials gets changed to the attackers desired credentials.

works on: Name, Password ~~OTP~~, Email, Cart, address, username phone number.

Steps to Reproduce:

i) login to ~~on~~ your account → victim side
ii) login to your account → attacker side.
iii) click on edit parameter. on attacker
iv) ~~click~~ Switch on Burp, Intercept on
v) Capture request onclick save button.
vi) Click on forward until you find your valid request.
vii) Right Click → Engagement tools → CSRF POC.
viii) Copy report.
ix) paste it into notepad and save with HTML extension.
x) open the HTML file and click on Submit request.
xi) The parameters get changed as per the attacker.
xii) If ~~the~~ it doesn't work then change the method from ~~post~~ ~~go~~ post to get in HTML file.
xiii) Try again.

If the user/victim is logged out then the parameters will get changed on the log in of user.

Solution: implement csrf token / auth token.

Bug 22 : Authentication token bypass of csrf

what we do : we change the ~~a~~ CSRF token value to bypass the authentication.

Can be done by 4 steps :

i) Remove value.

Steps :

a) Switch on Burp.
b) Enter login credentials, ~~intercept off~~
c) hit Sign in or log in
d) ~~make~~ Search for valid request in burp.
e) remove authenticity token value.
f) Intercept off
g) If logs in then vulnerable.

ii) ~~Change~~ alter value

a) Same as above
b) ~~change~~ alter the value of token by keeping the token length same.
c) Intercept off
d) If logs in then vulnerable..

iii) Change value

a) Same as above
b) change the complete value of token with previously used token.
c) If logs in then ~~to~~ vulnerable.

iv) Replace the value of token with another user.

a) Same as above.
b) login with 2 diff. users.
c) The token generated with A's account copy it and replace it with B's account token.
d) If logs in then vulnerable.

other way: when you find the valid request, change request method.

Bug 23: CSRF login

Mostly no impact.

steps: i) Enter login credentials.
ii) Capture the req in Burp.
iii) right click → Engagement tools → CSRF POC.
iv) Copy HTML
v) paste in notepad, Save in .HTML extension
vi) Open the file in another browser
vii) If logs in then vulnerable.

P4.

Bug 24: Password token leaked via third party.

what we do:

Capture the password request. check if any token is present for password.
If present, then check who is the host.
If host is third party, then vulnerable

Steps to Reproduce:

i) Ask for forgot password link.
ii) open the link.
iii) Capture reset password request in Burp Suite.
iv) ~~check the host. if~~
iv) find the correct request.
v) Referar: site which request is captured
Host: third party
password token avaliable
vi) forward until you satisfy all 3 cases.
vii) Copy the complete request and paste in Browser
viii) If ~~we are~~ password changed then vulnerable.

Bug ~~15~~ 25: Account lockout.

what ~~to~~ we do:

we try to enter a password by using ~~the~~ random passwords. Basically it should stop the user from doing so after any no. of times. If doesn't lock account then vulnerable.

steps to reproduce:

i) Open a website, click on login.
ii) Enter wrong credentials and capture login request in Burp.
iii) find right request and send to intruder.
iv) clear $ and add around password.
v) ~~start~~ change payload.
vi) start attack.
vii) If still password can bee entered then vulnerable.
viii) Sometimes only in one browser gets locked. Check with another browser
ix) Sometimes only IP gets blocked then try with different network.

Bug 26: Password reset poison. P2

Hacker might send his page where he can capture the new credentials of user

i) ~~f~~ Ask for forgot password link
ii)
iii) Capture "Change pass" request
iv) check for the valid request.
v) Send to repeater
vi) Change host to Bing.com
vii) Go.
viii) Intercept off
ix) check mail

classmate
Date
Page

Bug 27: Host header.

Steps: i) Capture the load page req. in Burp.
ii) Search for valid request.
iii) Send req. to Spider.
iv) go to target
v) In left menu ~~& right click and~~ on accurate site, right click and Spider this host.
vi) Check for status codes (302, 200)
vii) Right Click → send to repeater
viii) In repeater change host to bing.com
ix) Hit go. Check response in browser
x) If redirects then vulnerable.

Generally P1.

Bug 28: IDOR (Insecure direct object ref).

what we do: We manipulate the ID in database and if the user data gets available due to this, then it is vulnerable..

Case 1: hard to find, easy to exploit

Case 2 easy to find, hard to exploit.

Case 11:

- Steps to reproduce:

i) Create two accounts, 1 for victim and other for attacker.
ii) Capture the Account details edit request
iii) You will see an id there
Keywords: id, usrid, user id, account.
iv) log in to other account and do the same
v) Replace the user Ids.
vi) Try ~~loading~~ loging in to first account.
vii) you ~~can~~ also change email and

P4.

Bug 29: URL redirection

Basically used for phishing attack.

i) Spider the website
ii) Try to find url parameters.
iii) If parameters found then play with it i.e. arrange it parameter wise.

Date
Page

Steps to reproduce:

i) Intercept on, Reload website
ii) Find the valid request
iii) Send request to spider
iv) Intercept off
v) Go to target.
vi) Right click → spider this host
vii) Find accurate parameters and Send to repeater.
viii) Change parameter content to bing.com
ix) hit go and search for the content.
x) If ~~avaliable th~~ availalele then open response in browser
xi) If redirects then vulnerable.

To search params:
Go to taskbar, burp → search
input parameter → inscope only and search only req.

P1, P2

Bug 30: Server Side req forgery

Steps: ~~Spider website~~
i) Reload website and capture req in Burp
ii) Send to spider
iii) Find correct link in left menu
iv) Right click → spider this site

v) Find the hidden parameters.
vi) If we get req, send to ~~intruder~~ Repeater
~~vii) [illegible]~~
vii) add burp collab in parameter
viii) Hit go.
ix) If server gets response then
vulnerable

exploits:

Case I:

i) Send para request to intruder.
ii) change parameter content
with file:/// test
iii) position $ file content $
iv) payloads load → payload list →
v) Start attack.
vi) In result, sort length descending order.
check top 10.
vii) Select request and check its response.
viii) If mass text in black then
report (LOL).

~~i)~~ Case II:

change content to ports.

classmate
Date
Page

Bug 31: LFI and RFI

what we do: We try to access root directory.

LFI → local file inclusion.

steps: i) Spider the website.
ii) Find parameters using Burp search
iii) Send to Repeater.
iv) you will see the parameter content in request.
v) add content = "etc/passwd".
vi) If no response or error response then add "../" prior to etc.
vii) Keep adding the dot dot slash until you get a correct response. (only until 6 times)

If you want to do ~~with~~ load all payloads.

i) Send to intruder.
ii) positions, load payload.
iii) check response

Bug 32: XSS → P1.

reflected → P2, P3
stored → P2, P1
blind → P_1, P_2.

reflected → in search bars and hidden parameters.

steps :
i) Find input field.
ii) add XSS payloads.
iii) If the content of payload is visible then vulnerable.

steps to do with Burp - Suite →

i) Spider the website.
ii) Send to spider
iii) Filter
iv) Show inscope only
v) you get the parameters.

Blind XSS :

chat, feedback, contact

XSS hunter → app. sign in and free payloads.

Copy the payload from XSS hunter.
If the owner opens the form, and XSS works, you recieve a mail.
If yes then vulnerable

Stored XSS: First name ~~and~~ last name, address and all common input fields..
Steps: Same vas reflected .

Bug 33 : SQL Injection.

Download Cyber fox.
Download hackbar XPI file.

SQL injection mostly on php.

Query php id pune.

Copy URL in hack bar.

url'.