Web Application Security Audit of Covidcare Integration

Test URL

https://covid19-staging.nhp.gov.in/covidcare/login

Level-1 Report

09th FEB 2022

AAA Technologies P. Ltd

278-280, F Wing, Solaris-1,
Saki Vihar Road, Opp. L & T Gate No. 6,
Powai, Andheri (East),
Mumbai 400 072, INDIA
Tel: + 91 22 28573815 / 16
Fax: + 91 22 40152501
info@aaatechnologies.co.in
www.aaatechnologies.co.in

Web Application Security Test Report For Covidcare Integration

Document Version Control			
Data Classification		CLASSIFIED	
Client Name		NIHFW/CHI	
Document Title		Web Application Security Test Report	
Author		Shashank Jain	
Version	**Date of Issue**	**Issued by**	**Change Description**
1.0	09-02-2022	AAA Technologies	Initial Issue

Web Application Security Test Report For Covidcare Integration

Web Application Security Test Report

Presented by:

Shashank Jain

Application Testing Conducted On:

04-02-2022 to 09-02-2022

Web Application Security Test Report For Covidcare Integration

Table of Contents

Web Application Security Test Report For
Covidcare Integration

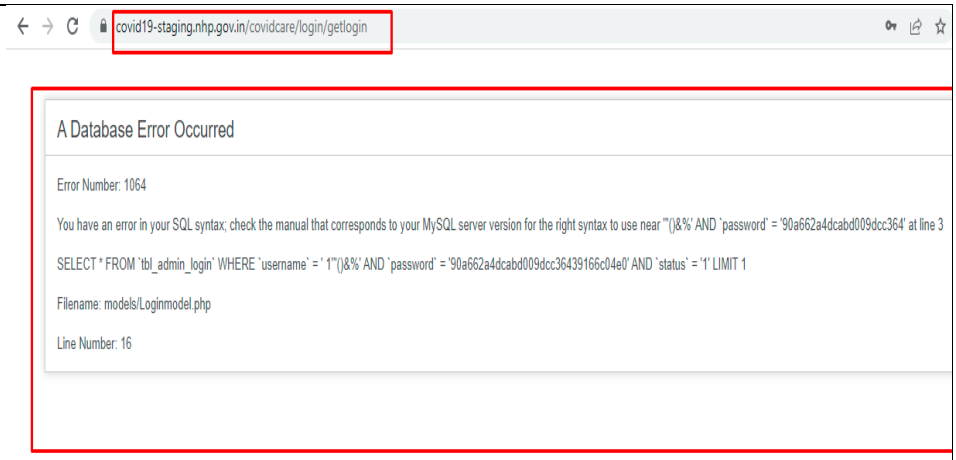
High

Web Application Security Test Report For Covidcare Integration

1. SQL Injection

1) Vulnerability Title: SQL Injection	
Risk	**High**
Abstract	It was observed that the application is vulnerable to SQL Injection.
Ease of Exploitation	Medium
Impact	An attacker may execute arbitrary SQL statements on the vulnerable system, this may compromise the integrity of your database and/or expose sensitive information.
Recommendations	The input data should be validated for special characters both in value fields and in URL. It is mandatory to implement server-side validations for every input vector in the application i.e. GET as well as POST parameters. Use parameterized queries in the application so that the all supplied parameters are treated as data, rather than potentially executable queries. Validation at the client side and server end is mandatory and application must trap all errors and give customized error message to the user. It is recommended to filter out all the following characters at user input: [1] | (pipe sign) [2] & (ampersand sign) [3] ; (semicolon sign) [4] $ (dollar sign) [5] % (percent sign) [6] @ (at sign) [7] ' (single apostrophe) [8] " (quotation mark) [9] \' (backslash-escaped apostrophe) [10] \" (backslash-escaped quotation mark) [11] <> (triangular parenthesis) [12] () (parenthesis) [13] + (plus sign) [14] CR (Carriage return, ASCII 0x0d) [15] LF (Line feed, ASCII 0x0a) [16] , (comma sign) [17] \ (backslash)

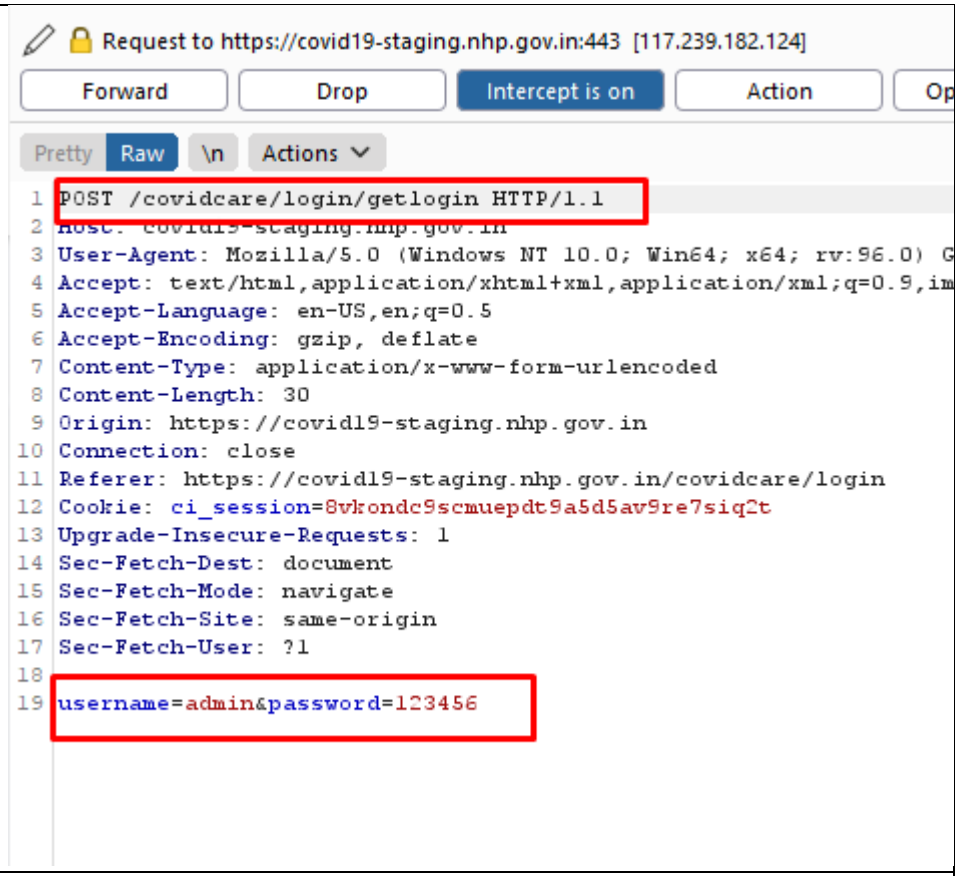
Web Application Security Test Report For Covidcare Integration

Snapshot	
Affected URLs	**Throughout the application**

2. Password is travelling in the clear text

2) Vulnerability Title: Password is travelling in the clear text	
Risk	**High**
Abstract	It was observed that the user password is travel in clear text in the application.
Ease of Exploitation	Easy
Impact	A third party may be able to Sniff the user credentials.
Recommendations	It is recommended to use SHA-256 with salt for more secured application. a) Salted SHA-256 technique in, authentication or login module b) SHA-256 hash technique in, change password and reset password modules. The pre-requisite to this is that the backend database stores a SHA-256 hash of the password. (SHA-256 hash is a cryptographic technique in which the actual value can never be recovered.). Here is how the salted SHA-256 technique works: When a client requests for the login page, the server generates a random number, the salt, and sends it to the client along with the page. A JavaScript code on the client computes the SHA-256 hash of the password entered by the user. It then concatenates the salt to the hash and re-computes the SHA-256 hash. This result is then sent to the server. The server picks the hash of the password from its database, concatenates the salt and computes the SHA-256 hash. If the user entered the correct password these two hashes should match. The server compares the two and if they match, the user is authenticated.

Web Application Security Test Report For Covidcare Integration

Snapshot	
Affected URLs	https://covid19-staging.nhp.gov.in/covidcare/login

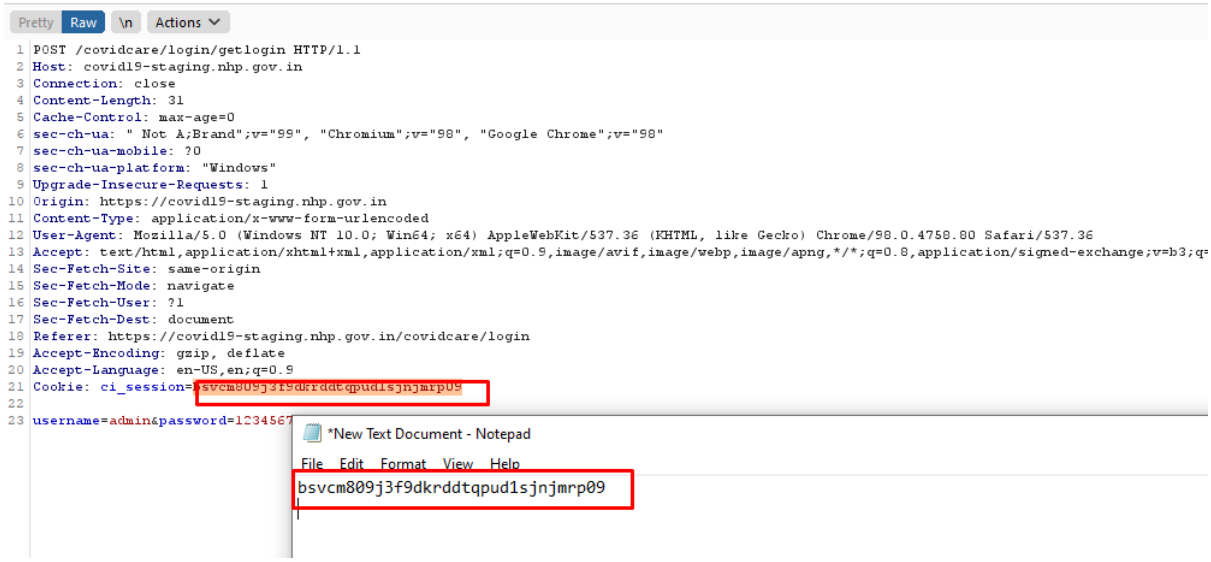
3. Session Fixation is possible in the application

3) Vulnerability Title: Session Fixation is possible in the application	
Risk	**High**
Abstract	It was observed that the application does not re-initialize the session ID stored in the cookie field after login. This allows an attacker to steal the session ID assigned after login, and then simultaneously use the stolen session ID of that user to access restricted pages in the application while the user is logged on. Application is vulnerable to Session Fixation attack.
Ease of Exploitation	Easy
Impact	When implemented successfully, attackers assume the identity of the compromised user, enjoying the same access to resources as the compromised user. Identity theft, Information theft, stealing sensitive data are some of the common impacts of session fixation.
Recommendations	Add a new cookie that randomly changes for each login attempt. Generate different session id before and after authentication. Also every request after the successful authentication should be associated with an extra session identifier cookie which also randomly changes and expires when user logs out from the application or closes the browser.
Snapshot	-
Affected URLs	**throughout the application**

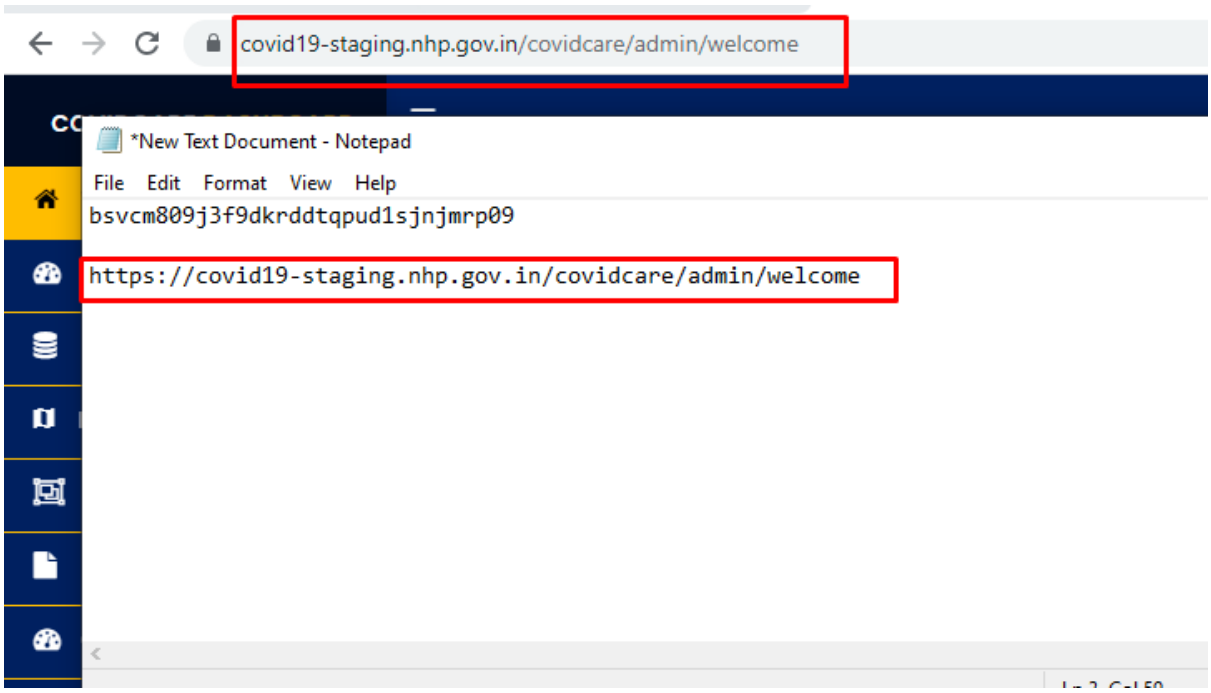
Web Application Security Test Report For Covidcare Integration

How Test was performed:

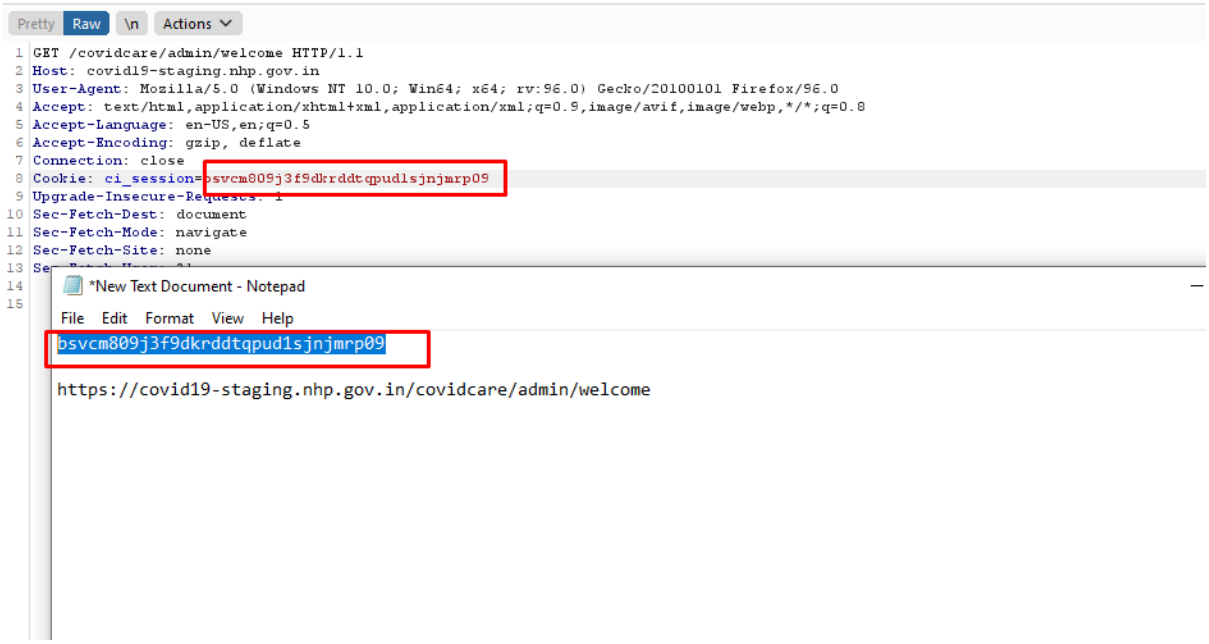
Step#1: Provide invalid credentials and click on 'Login' button and intercept the request. In the capture request, Copy the entire cookie value and paste it into the note pad as shown below:



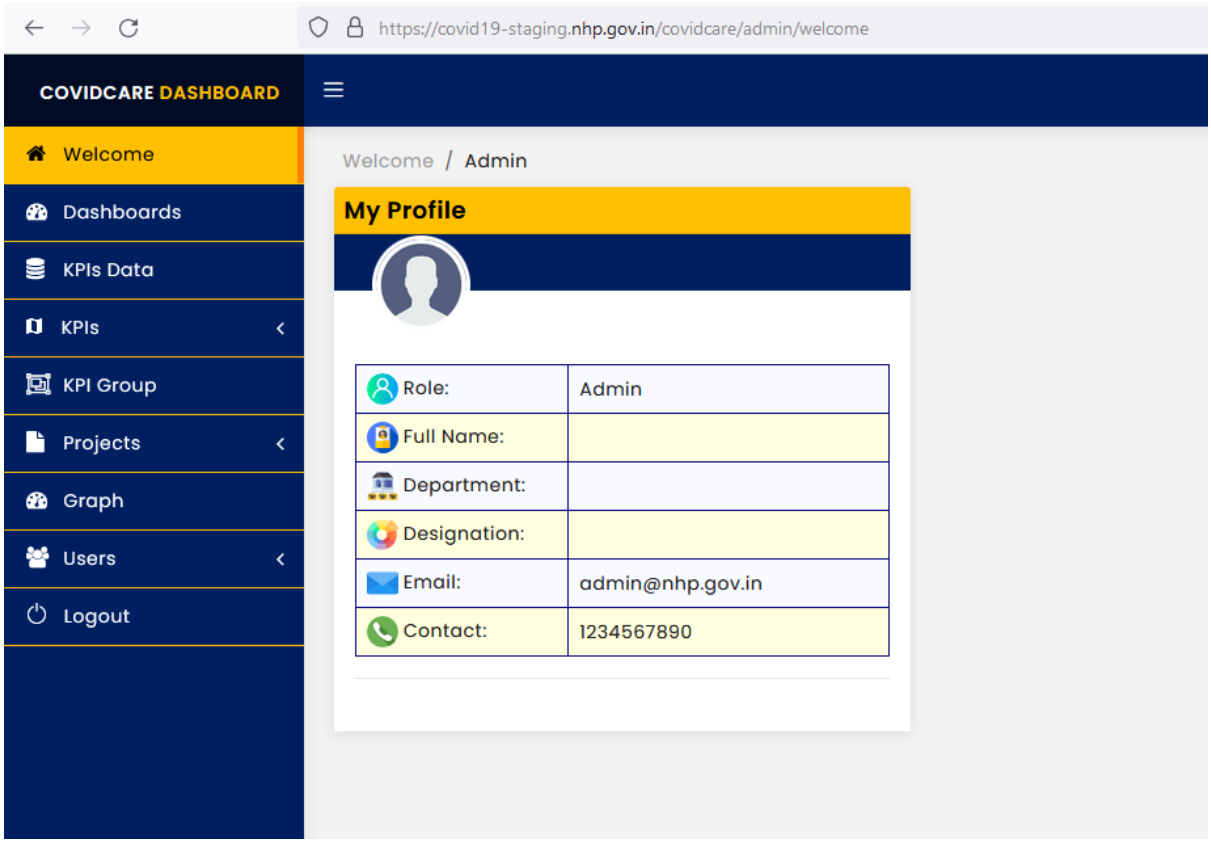
Step#2: Login with valid credentials and copy the authenticated URL and paste it into the notepad.



Step#3: Open another browser. Paste the copied URL into the address bar and intercept the request. In the captured request, replace the current cookie value with previously copied value and forward the modified request to the server as shown below.



Step#4: In the following snapshot, it is clearly seen that session fixation is possible in the application as authenticated page is accessible through modified pre-authenticated cookie value.



4. Session Hijacking is possible in the application

4) Vulnerability Title: Session Hijacking is possible in the application	
Risk	**High**
Abstract	Session hijacking is possible in the application
Ease of Exploitation	Medium
Impact	This test is to check whether the cookie can be reused in another computer during the login phase. The server maintains the user state between any client and itself by exchanging the cookie for each request-response between them. If this unique identifier is not removed from the state table of the web application server when user tries to login the same account from another computer then any request subsequent to the user login will be happily serviced by the web application, (Session Hijacking).
Recommendations	1. Add a new cookie that randomly changes for each login attempt. Generate different session id before and after authentication. Also every request after the successful authentication should be associated with an extra auth cookie: It is possible to view the sensitive information by fetching the page from the cache option of the browser. session identifier cookie which also randomly changes and expires when user logs out from the application or closes the browser. For example; Cookie: AUTHCookie=69BK7F0D8KL; ASP.NET_SessionId= JNHG7H0LKJ57CF4; Cookie: AUTHCookie=5A0KN5F9ER5; ASP.NET_SessionId= JNHG7H0LKJ57CF4; Cookie: AUTHCookie=CG4K8L3T5H; ASP.NET_SessionId= JNHG7H0LKJ57CF4; Cookie: AUTHCookie=L6D3G0JA3S; ASP.NET_SessionId= JNHG7H0LKJ57CF4. 2. SSL should be implemented.
Snapshot	-----
Affected URLs	**Throughout the application**

Web Application Security Test Report For Covidcare Integration

How Test was performed

Step:1#: In the authenticated Session, Capture the request, copy the Cookie value in notepad and Url as well. After that open the different browser and paste the url there with capturing the request change the new cookie value to old Cookie value.

```
Pretty Raw \n Actions
1 POST /covidcare/login/getlogin HTTP/1.1
2 Host: covid19-staging.nhp.gov.in
3 Connection: close
4 Content-Length: 30
5 Cache-Control: max-age=0
6 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="98", "Google Chrome";v="98"
7 sec-ch-ua-mobile: ?0
8 sec-ch-ua-platform: "Windows"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://covid19-staging.nhp.gov.in
11 Content-Type: application/x-www-form-urlencoded
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.47
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,ap
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://covid19-staging.nhp.gov.in/covidcare/login
19 Accept-Encoding: gzip, deflate
20 Accept-Language: en-US,en;q=0.9
21 Cookie: ci_session=3gl222trbk2ao8lj6ula8hcmrcobm6s2
22
23 username=admin&password=123456
```

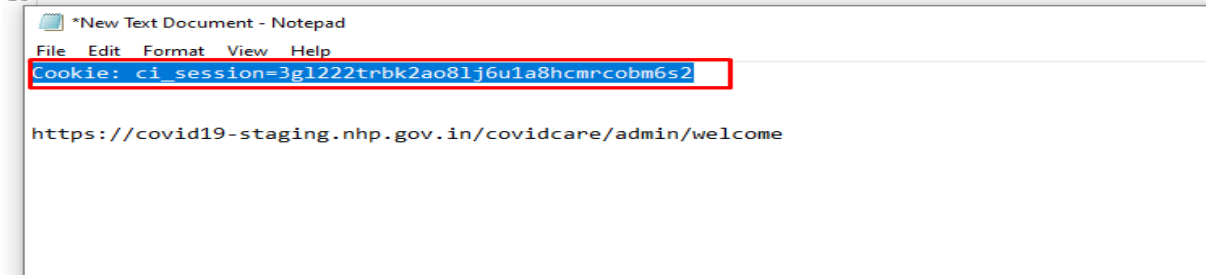

*New Text Document - Notepad

File Edit Format View Help

```
Cookie: ci_session=3gl222trbk2ao8lj6u1a8hcmrcobm6s2

https://covid19-staging.nhp.gov.in/covidcare/admin/welcome
```

```
Pretty Raw \n Actions
1 GET /covidcare/admin/welcome HTTP/1.1
2 Host: covid19-staging.nhp.gov.in
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:96.0) Gecko/20100101 Firefox/96.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Cookie: ci_session=3gl222trbk2ao8lj6ula8hcmrcobm6s2
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: none
13 Sec-Fetch-User: ?1
14
15
```



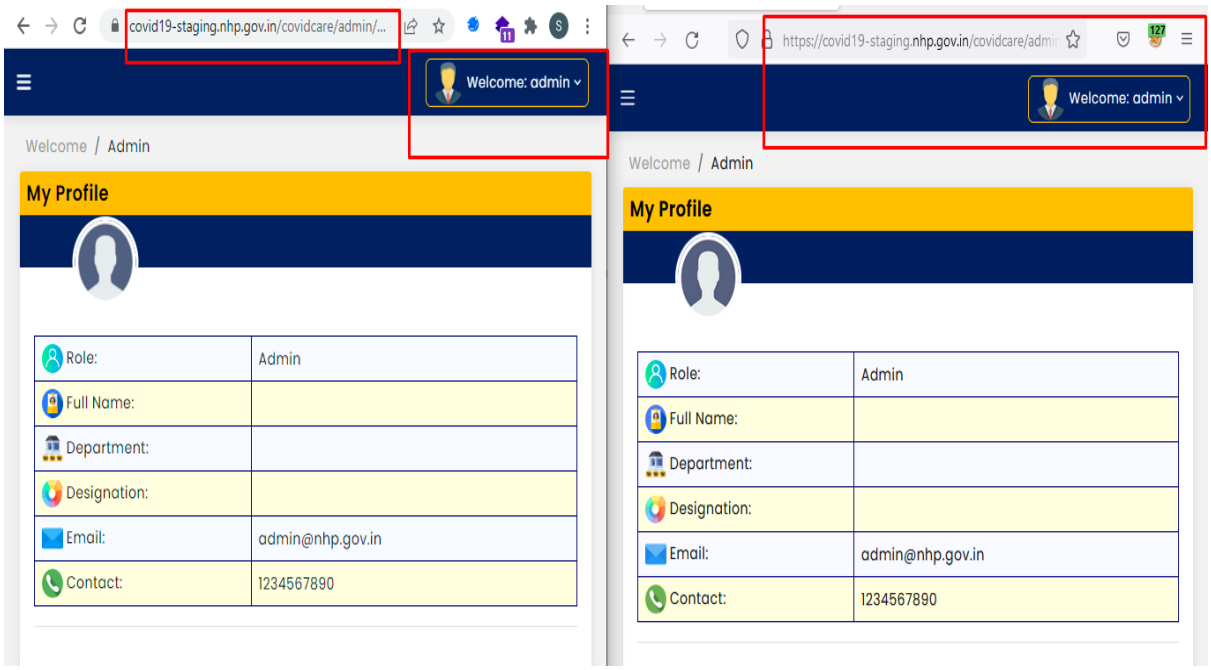
*New Text Document - Notepad

File Edit Format View Help

```
Cookie: ci_session=3gl222trbk2ao8lj6u1a8hcmrcobm6s2

https://covid19-staging.nhp.gov.in/covidcare/admin/welcome
```

Step:2#: As per below snapshot, it is clearly shown that Session hijacking is possible in the application



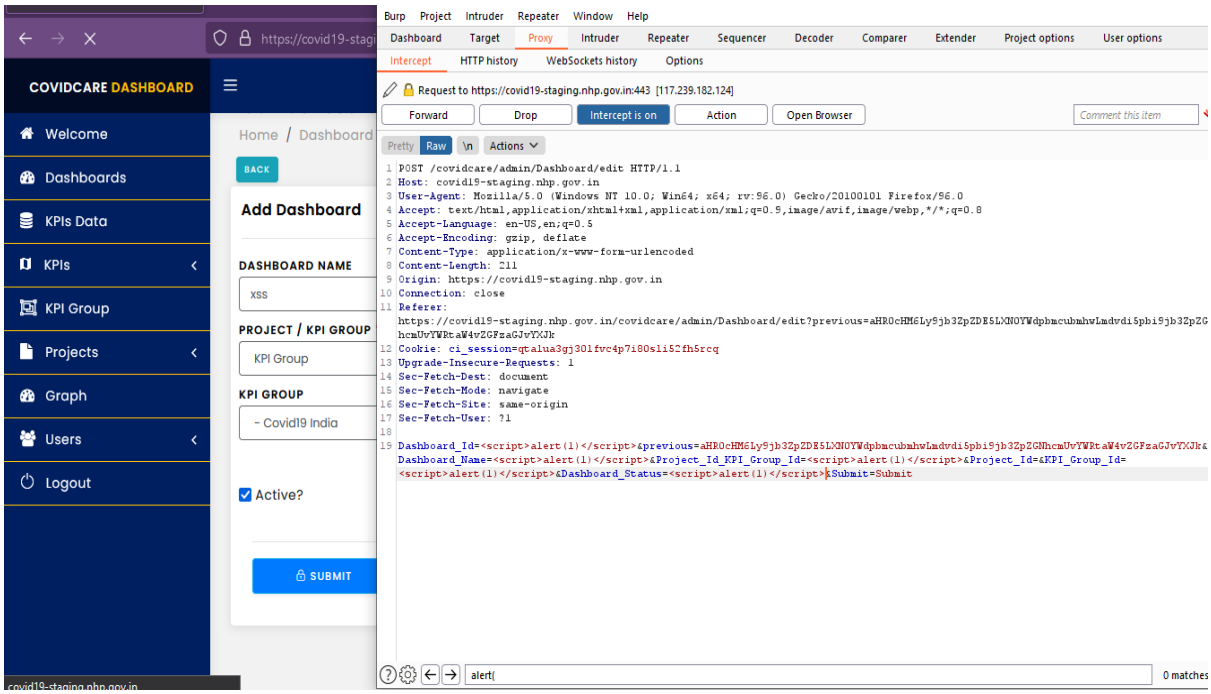
5. Stored Cross Site Scripting

5) Vulnerability Title: Stored Cross-Site Scripting	
Risk	**High**
Abstract	It was observed that there was a Stored Cross site scripting vulnerability found in the given application
Ease of Exploitation	Easy
Impact	Malicious users may inject JavaScript, VBScript, ActiveX, HTML or Flash into a vulnerable application to fool a user in order to gather data from them. An attacker can steal the session cookie and take over the account, impersonating the user every time the page is opened.
Recommendations	1. The input data should be validated for special characters both in value fields and in URL. Application should not save scripts in the database. Validation at the server end is mandatory. Note: Fix this vulnerability throughout the application. It is recommended to filter out all the following characters at user input: [1] | (pipe sign) [2] & (ampersand sign) [3] ; (semicolon sign) [4] $ (dollar sign) [5] % (percent sign) [6] @ (at sign) [7] ' (single apostrophe) [8] " (quotation mark) [9] \' (backslash-escaped apostrophe) [10] \" (backslash-escaped quotation mark) [11] e<> (triangular parenthesis) [12] () (parenthesis) [13] + (plus sign) [14] CR (Carriage return, ASCII 0x0d) [15] LF (Line feed, ASCII 0x0a) [16] , (comma sign) [17] \ (backslash) 2. Scripts should ideally not be included from untrusted domains. Applications that rely on static third-party scripts should consider using

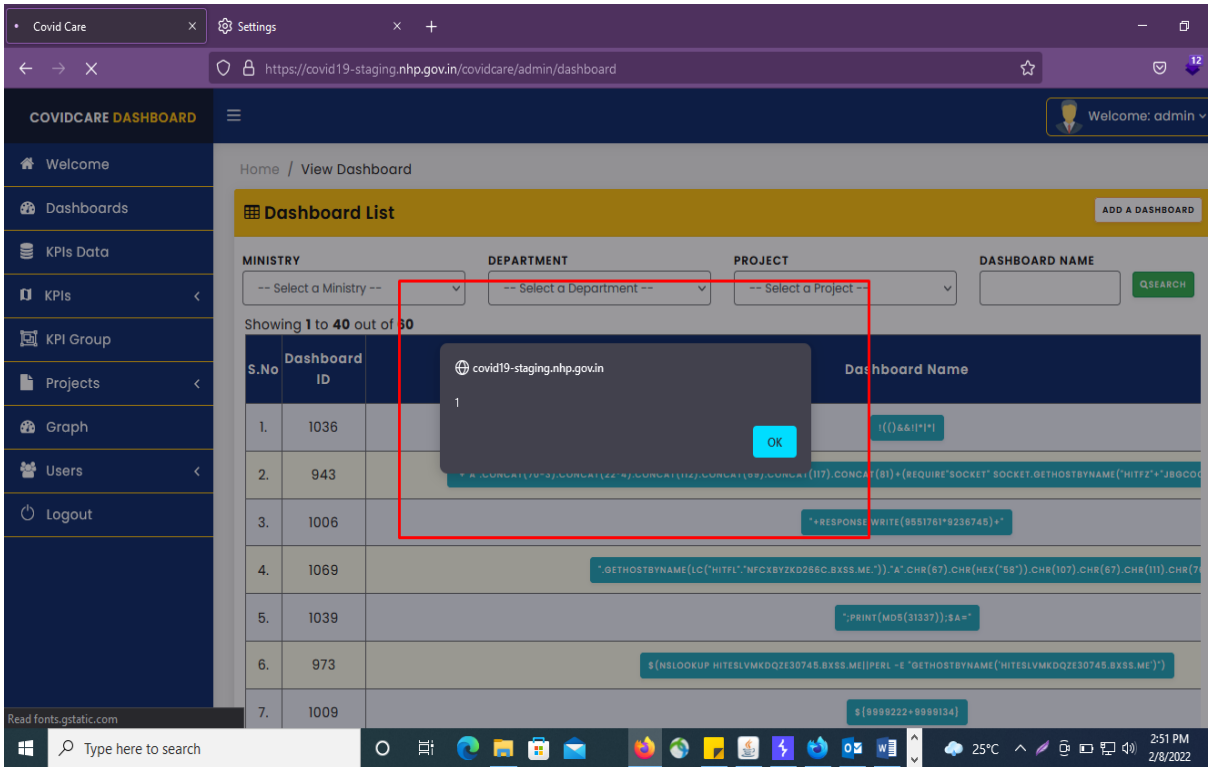
	Sub resource Integrity to make browsers verify them, or copying the contents of these scripts onto their own domain and including them from there. If that is not possible (e.g. for licensing reasons) then consider reimplementing the script's functionality within application code.
Snapshot	
Affected URLs	**throughout the application**

How Test was performed:

Step#1: Open the URL, Login with the valid credential, go to Dashboard – Add Dashboard name and add Script in the input field then save the details as shown below:



Step#2: It is clearly seen that the cross-site scripting attack is executed successfully as we see the popup in the screen.



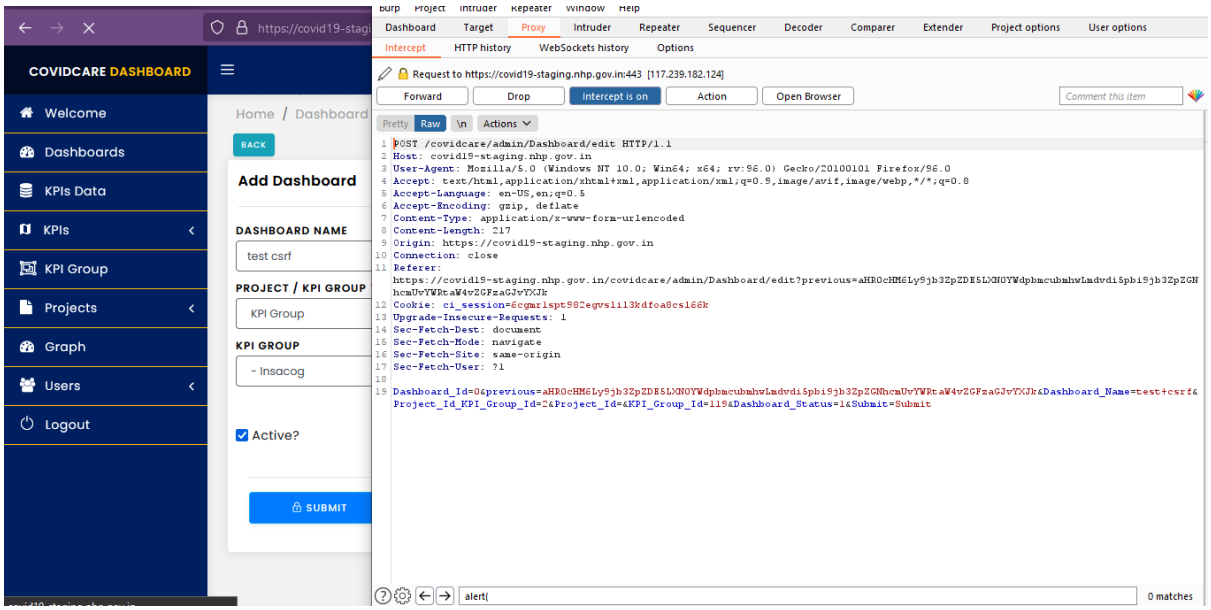
6. Cross-Site Request Forgery

6) Vulnerability Title: Cross-Site Request Forgery	
Risk	**High**
Abstract	Cross Site Request Forgery (CSRF) attack is possible which forces a logged-on victim's browser to send a request to a vulnerable web application, which then performs the chosen action on behalf of the victim.
Ease of Exploitation	Hard
Impact	It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user
Recommendations	Here the problem is - sharing of Cookie values across different instances of the same browser for the same host page. The application should implement the following techniques to prevent the CSRF attack: Insert custom random tokens into every form (page) – 1. Such that to validate each request a random token is generated for that request on the server side with the server response to the previous request. 2. These tokens will not be shared across different instances of the same browser accessing the same host page. Thus, these tokens will not be automatically submitted by the browser. 3. Validate the submitted token at the server end. 4. If the request doesn't contain the token or if the submitted token is incorrect then don't address the request. 5. Token value should change on each and every request. 6. Token value should be implemented on Page body. 7. Token value should be alphanumeric and minimum 32 characters. 8. Token should also be implemented on logout button. For example: <form action= "FundTransfer.aspx" method= "POST"> <input type = "hidden" name="csrf-token" value="O435d6a57t6g6hAFs39D8sd123456"> ……. </form>

	It is recommended the token value should change at each page load event and should be validated on the server side before addressing the request. Also, make sure that server does not address the request if the CSRF token contains previously used values.
Snapshot	-
Affected URLs	**throughout the application**

How Test was performed:

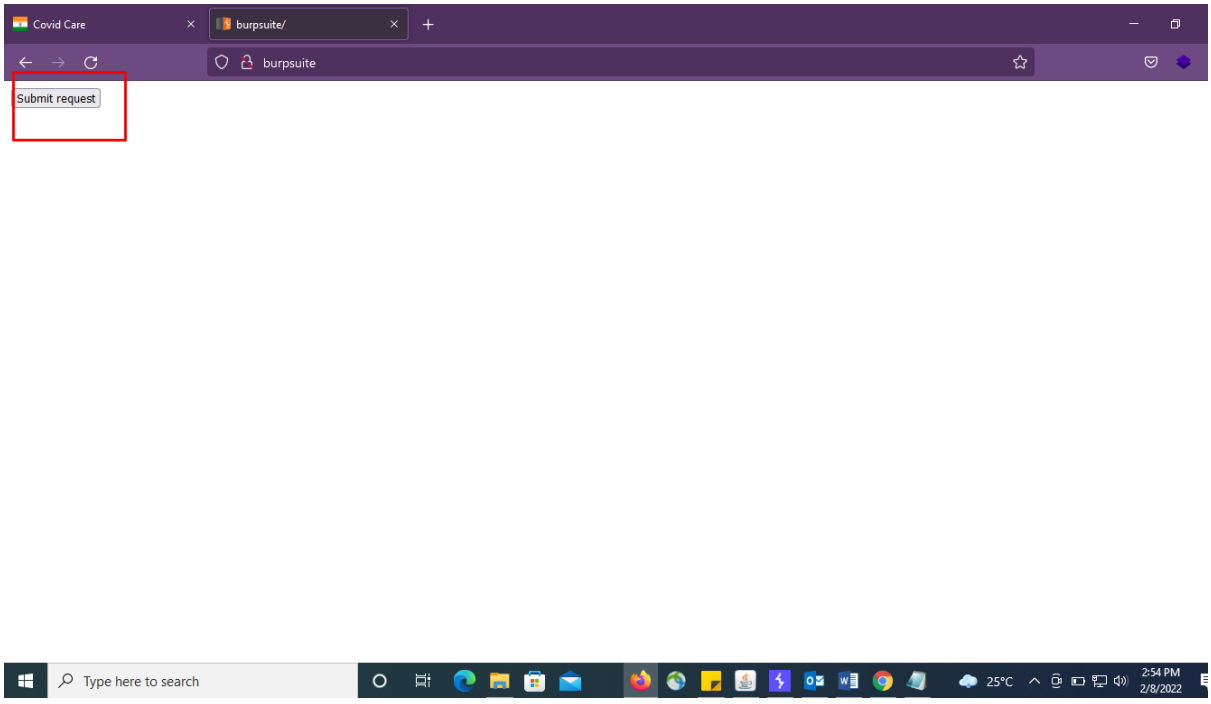
Step#1: Open the URL and Login with admin role and go to Add content then fill the required details and intercept the request as shown below:



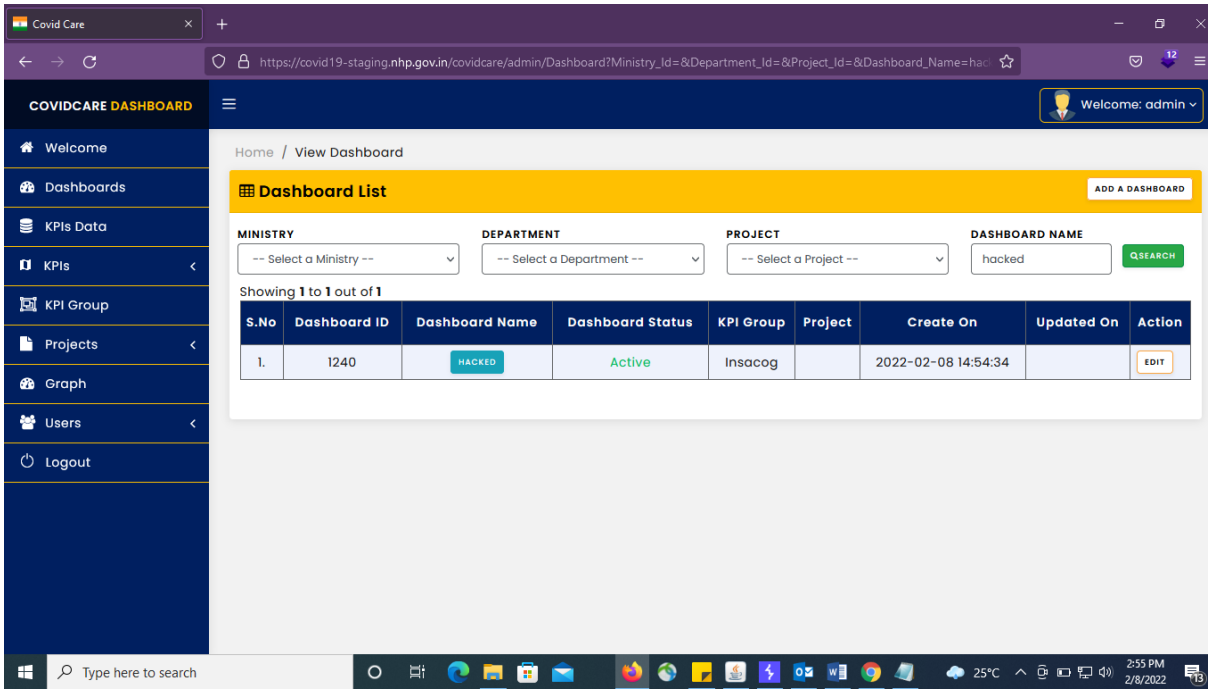
Step#2: Use the copied parameters for crafting a CSRF page and change the value of parameter Name “Hacked” save the file with .html as shown below:

```
<body>
<script>history.pushState('', '', '/')</script>
  <form action="https://covid19-staging.nhp.gov.in/covidcare/admin/Dashboard/edit" method="POST">
    <input type="hidden" name="Dashboard&#95;Id" value="0" />
    <input type="hidden" name="previous" value="aHR0cHM6Ly9jb3ZpZDE5LXN0YWdpbmcubmhwLmdvdi5pbi9jb3ZpZGNhcmUvYWRtaW4vZGFzaGJvYXJk" />
    <input type="hidden" name="Dashboard&#95;Name" value="hacked" />
    <input type="hidden" name="Project&#95;Id&#95;KPI&#95;Group&#95;Id" value="2" />
    <input type="hidden" name="Project&#95;Id" value="" />
    <input type="hidden" name="KPI&#95;Group&#95;Id" value="119" />
    <input type="hidden" name="Dashboard&#95;Status" value="1" />
    <input type="hidden" name="Submit" value="Submit" />
    <input type="submit" value="Submit request" />
  </form>
</body>
</html>
```

Step#3: Now, Call the crafted page in another tab of logged-in user’s session, click on ‘Submit request’ as shown below:



Step#4: It is clearly seen that authentication action is performed by calling CSRF page as the data has been updated successfully as shown below:



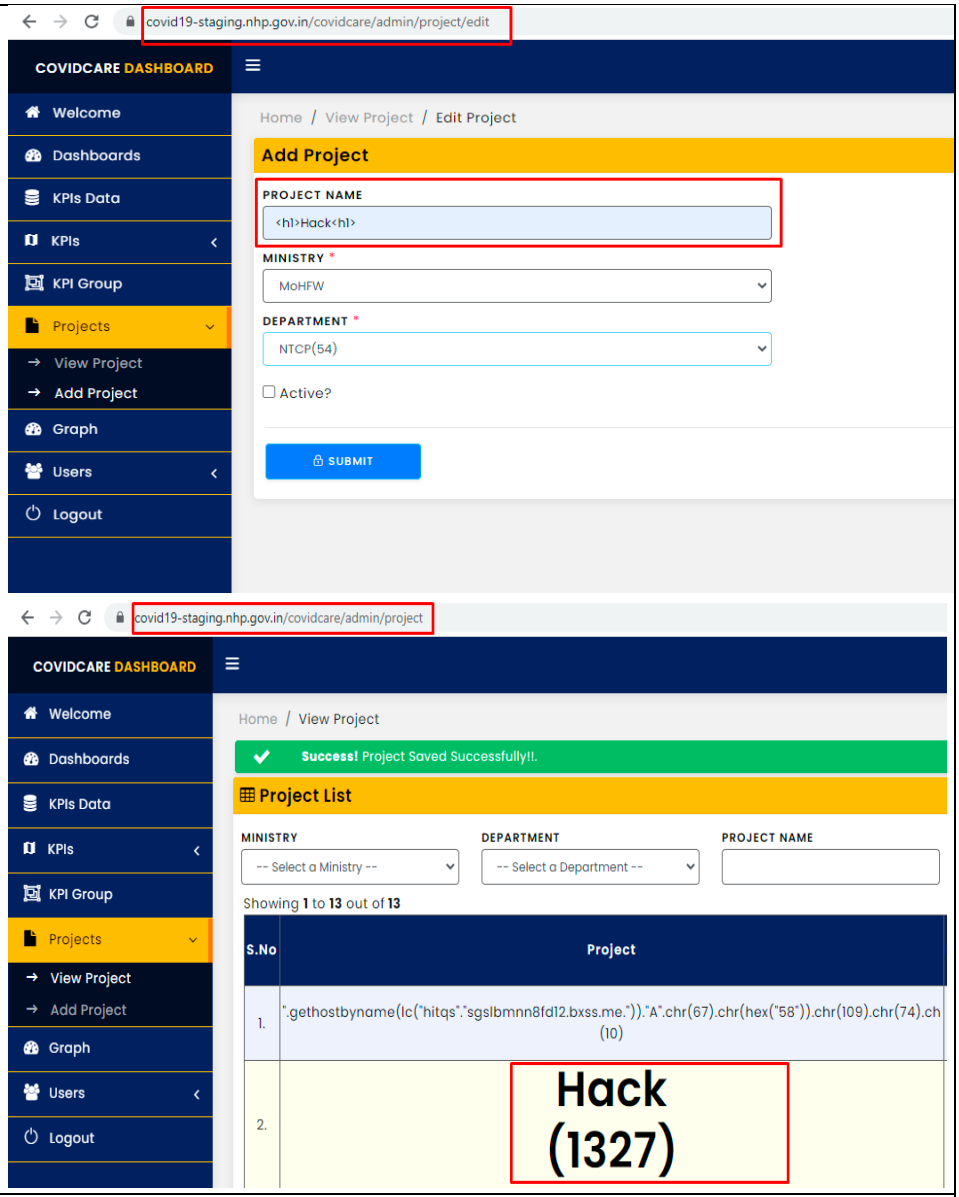
Web Application Security Test Report For
Covidcare Integration

Medium

7. HTML Injection attack is possible in the application

1) Vulnerability Title: HTML Injection attack is possible in the application	
Risk	**High**
Abstract	It was observed that HTML injection is possible in the application.
Ease of Exploitation	Easy
Impact	• An attacker can change displayed website's appearance. • To steal another person's identity
Recommendations	It is recommended to filter out all the following characters at user input: [1] | (pipe sign) [2] & (ampersand sign) [3] ; (semicolon sign) [4] $ (dollar sign) [5] % (percent sign) [6] @ (at sign) [7] ' (single apostrophe) [8] " (quotation mark) [9] \' (backslash-escaped apostrophe) [10] \" (backslash-escaped quotation mark) [11] e<> (triangular parenthesis) [12] () (parenthesis) [13] + (plus sign) [14] CR (Carriage return, ASCII 0x0d) [15] LF (Line feed, ASCII 0x0a) [16] , (comma sign) [17] \ (backslash)

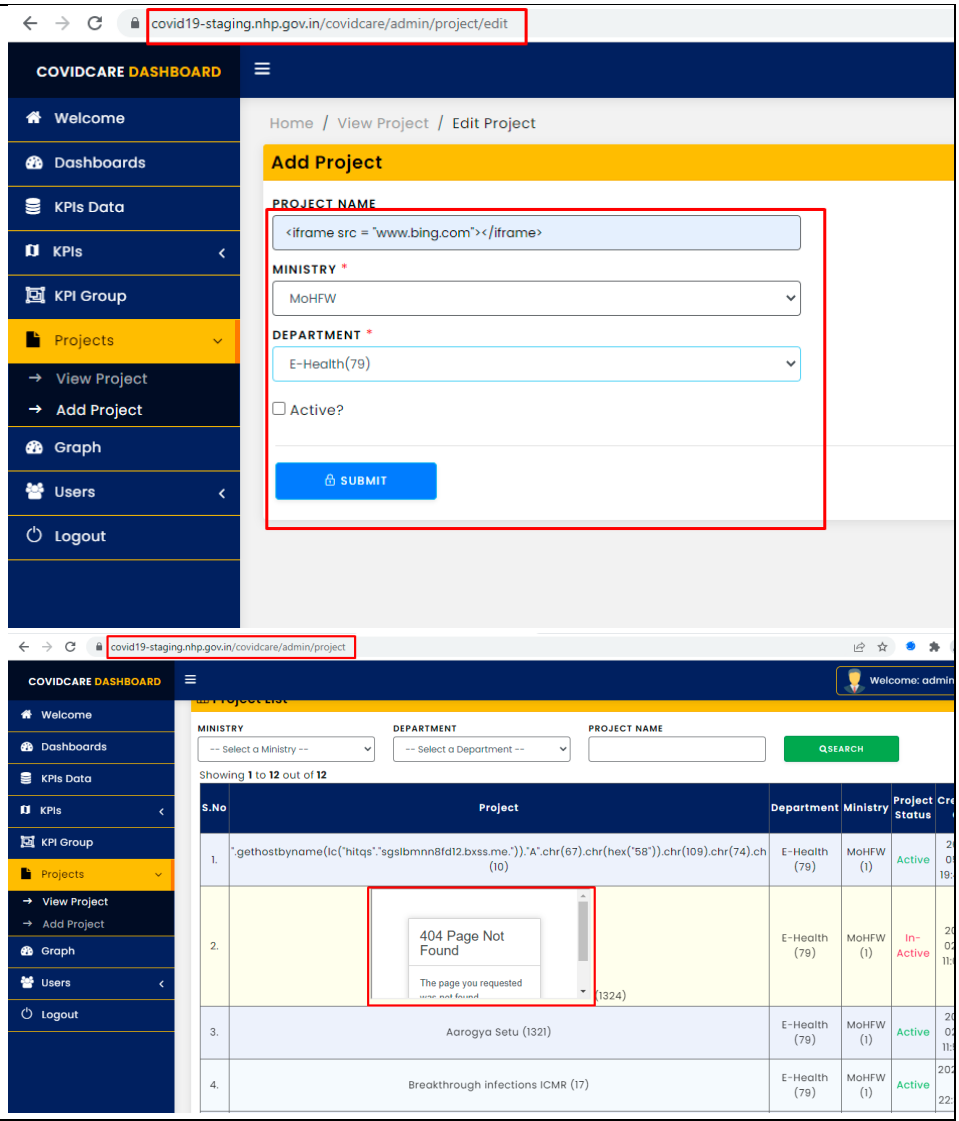
Web Application Security Test Report For Covidcare Integration

Snapshot	
Affected URLs	**throughout the application**

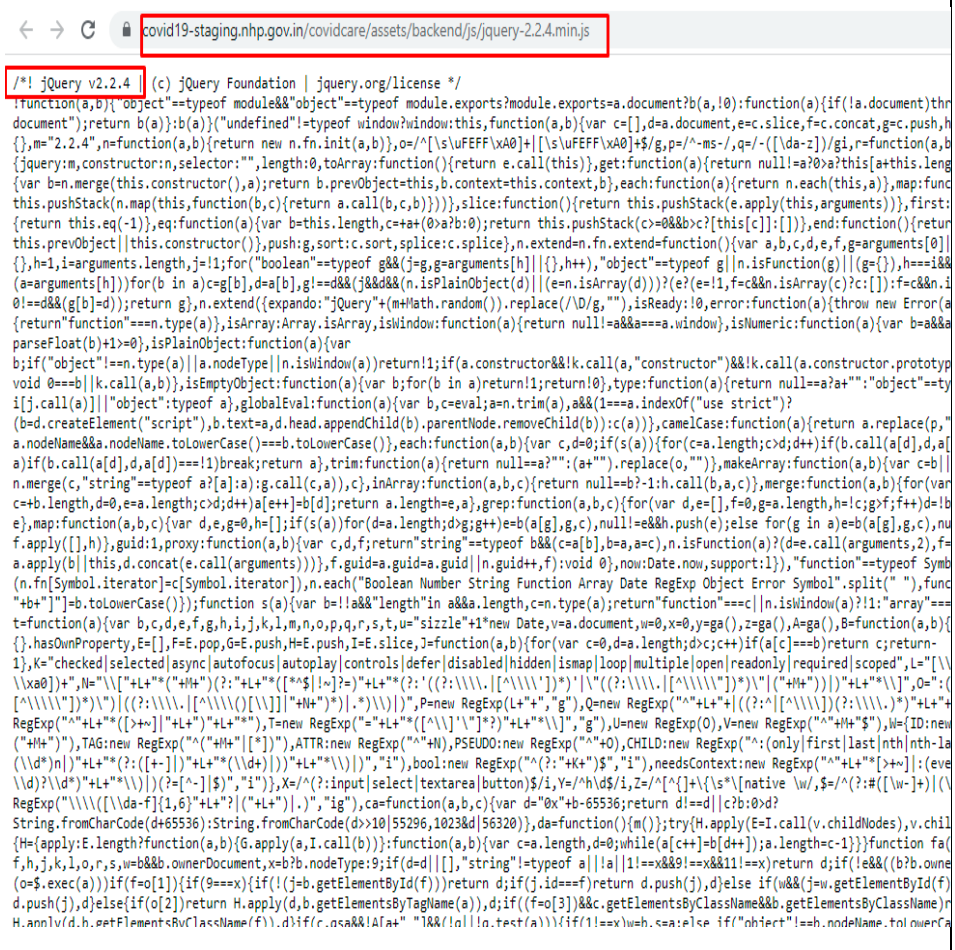
8. Iframe Injection attack is possible in the application

2) Vulnerability Title: Iframe Injection attack is possible in the application	
Risk	**High**
Abstract	It was observed that Iframe injection is possible in the application.
Ease of Exploitation	Easy
Impact	• An attacker can change displayed website's appearance. • To steal another person's identity
Recommendations	It is recommended to filter out all the following characters at user input: [1] | (pipe sign) [2] & (ampersand sign) [3] ; (semicolon sign) [4] $ (dollar sign) [5] % (percent sign) [6] @ (at sign) [7] ' (single apostrophe) [8] " (quotation mark) [9] \' (backslash-escaped apostrophe) [10] \" (backslash-escaped quotation mark) [11] e<> (triangular parenthesis) [12] () (parenthesis) [13] + (plus sign) [14] CR (Carriage return, ASCII 0x0d) [15] LF (Line feed, ASCII 0x0a) [16] , (comma sign) [17] \ (backslash)

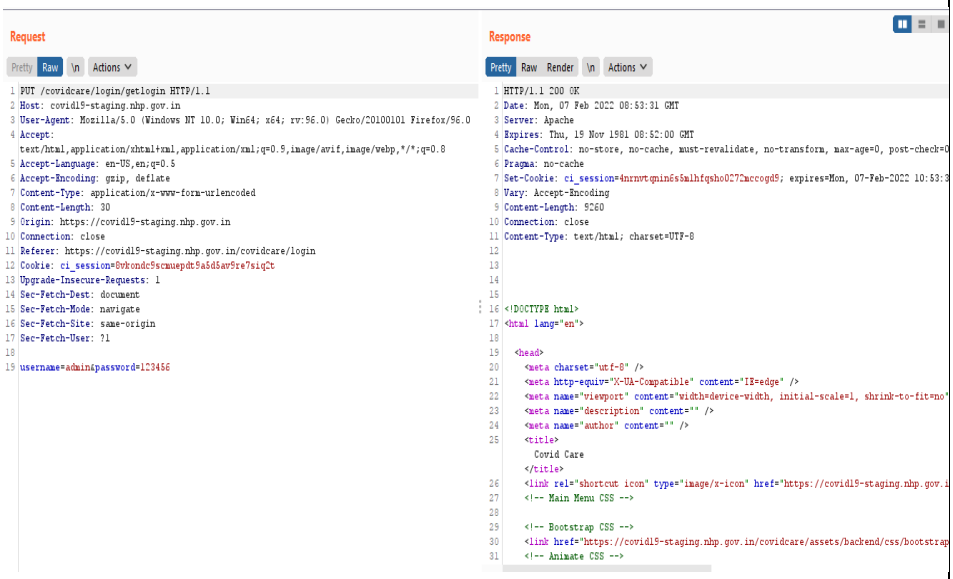
Web Application Security Test Report For Covidcare Integration

Snapshot	
Affected URLs	**throughout the application**

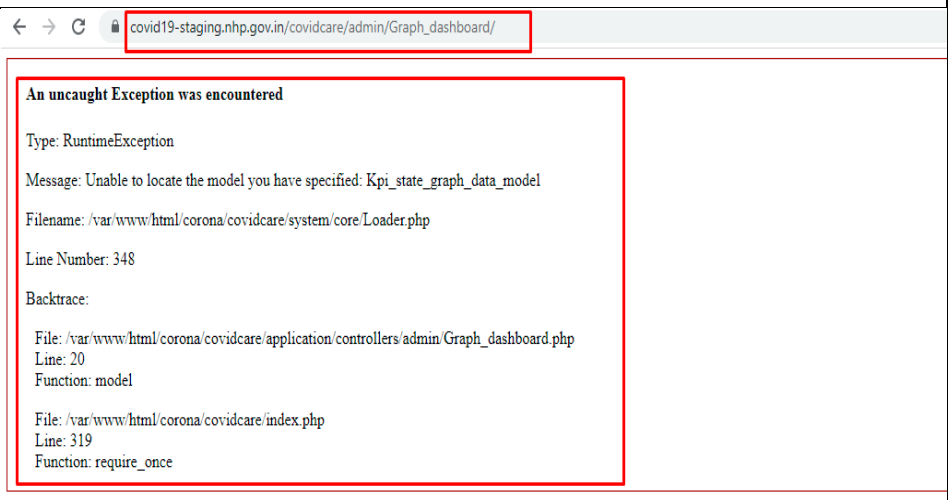
9. Old & Vulnerable version of jQuery is used in the application

3) Vulnerability Title: Old & Vulnerable version of JQuery used	
Risk	**Medium**
Abstract	It was observed that this page is using an older version of jQuery that is vulnerable to a Cross Site Scripting vulnerability
Ease of Exploitation	Medium
Impact	An attacker can steal the cookies as well as the user session Id.
Recommendations	It is recommended to update to latest version of jQuery.
Snapshot	
Affected URLs	**throughout the application**

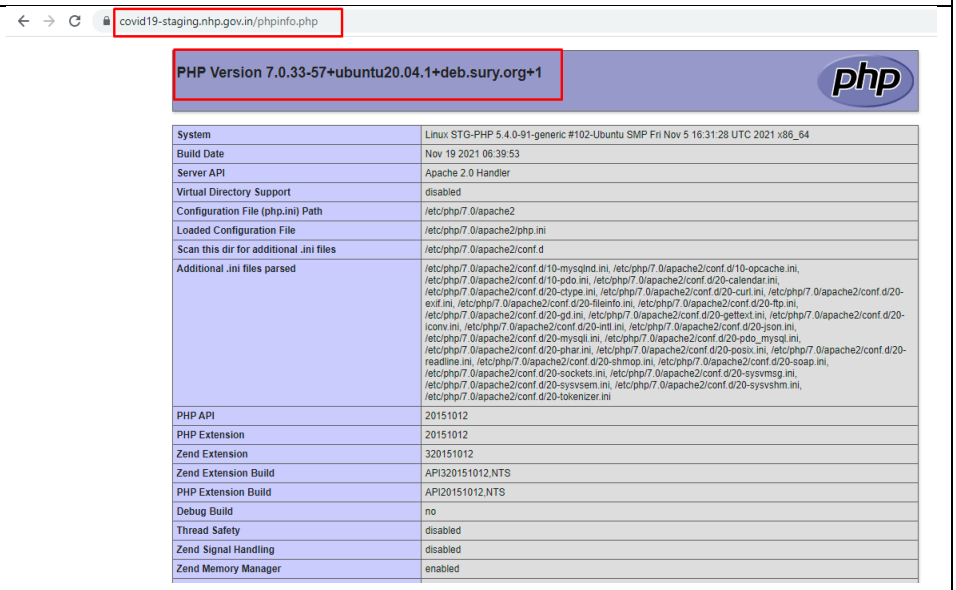
10. Dangerous HTTP Method is used in the application

4) Vulnerability Title: Dangerous HTTP Method used	
Risk	**Medium**
Abstract	It was observed that Http methods (PUT, DELETE, TRACE/TRACK, OPTIONS) are enabled on this web server.
Ease of Exploitation	Medium
Impact	It was observed that using these methods may expose sensitive information that may help malicious user to prepare more advanced attacks
Recommendations	It is recommended to disable http dangerous methods on the web server
Snapshot	
Affected URLs	**throughout the application**

11. Error Message on Page

5) Vulnerability Title: Error Message on Page	
Risk	Medium
Abstract	Application error or warning messages may expose sensitive information about an application's internal workings to an attacker.
Ease of Exploitation	Easy
Impact	Error messages may disclose sensitive information which can be used to escalate attacks.
Recommendations	1. Verify that this page is disclosing error or warning messages and properly configure the application to log errors to a file instead of displaying the error to the user. 2. Use customized error message.
Snapshot	
Affected URLs	**throughout the application**

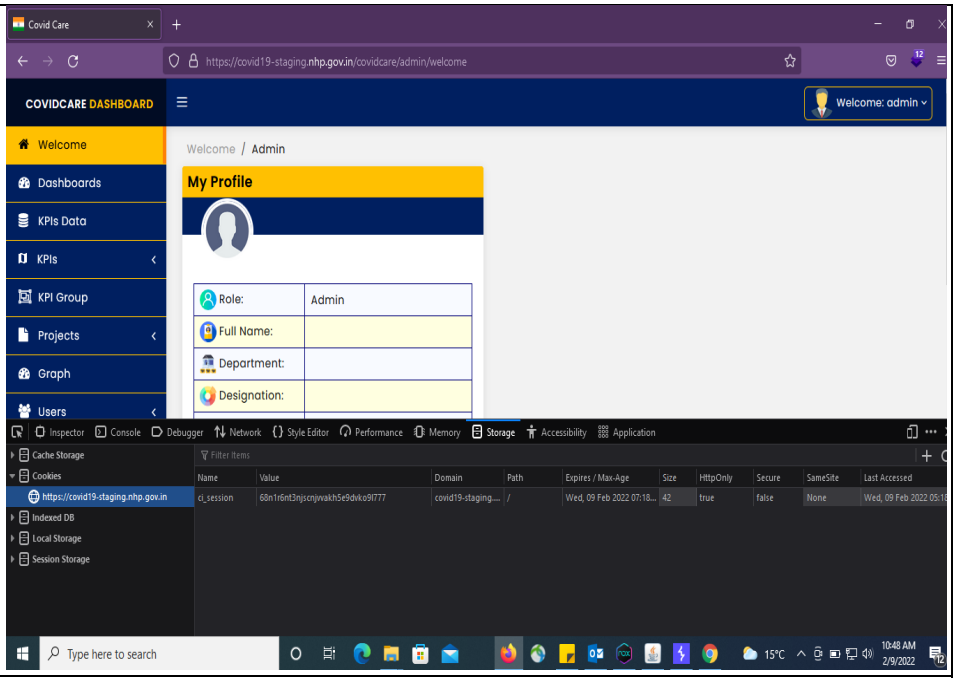
12. Older version of PHP is used in the application

6) Vulnerability Title: Older version of PHP is used in the application	
Risk	**Medium**
Abstract	It was observed that Older version of PHP is used in the application.
Ease of Exploitation	Easy
Impact	PHP before has buffer overflow, remote code execution, xmlrpc etc. like vulnerabilities.
Recommendations	It is recommended to use latest or stable version.
Snapshot	
Affected URLs	**throughout the application**

13. Session Cookie without Secure and Same Site Flag Set

7) Vulnerability Title: Session cookie without secure and Same Site flag Set	
Risk	**Medium**
Abstract	It was observed that session cookie is without secure & same site flag set.
Ease of Exploitation	Medium
Impact	If the secure flag is set on a cookie, then browsers will not submit the cookie in any requests that use an unencrypted HTTP connection, thereby preventing the cookie from being trivially intercepted by an attacker monitoring network traffic. If the secure flag is not set, then the cookie will be transmitted in clear-text if the user visits any HTTP URLs within the cookie's scope. An attacker may be able to induce this event by feeding a user suitable links, either directly or via another web site. Without the SameSite flag, the application may be vulnerable to cross site request forgery (CSRF) and cross origin information leakage attacks since the browser will send cookies across origins. An attacker can use these attacks to trick a user into performing an action or into leaking sensitive data.
Recommendations	It is recommended that Configure the application to set the Secure flag. Configure the application to set the SameSite flag. As this is a relatively new cookie attribute, the flag may need to be set programmatically by code on the server side. Set the attribute as such: Set-Cookie: CookieName=CookieValue; SameSite=Strict/Lax; The SameSite attribute can be set with two values: Strict – The cookie is not sent with any cross-site requests, even if the user follows a link to a third party site. Lax – The cookie is sent with a top-level GET request, such as a user following a link to a third party site.

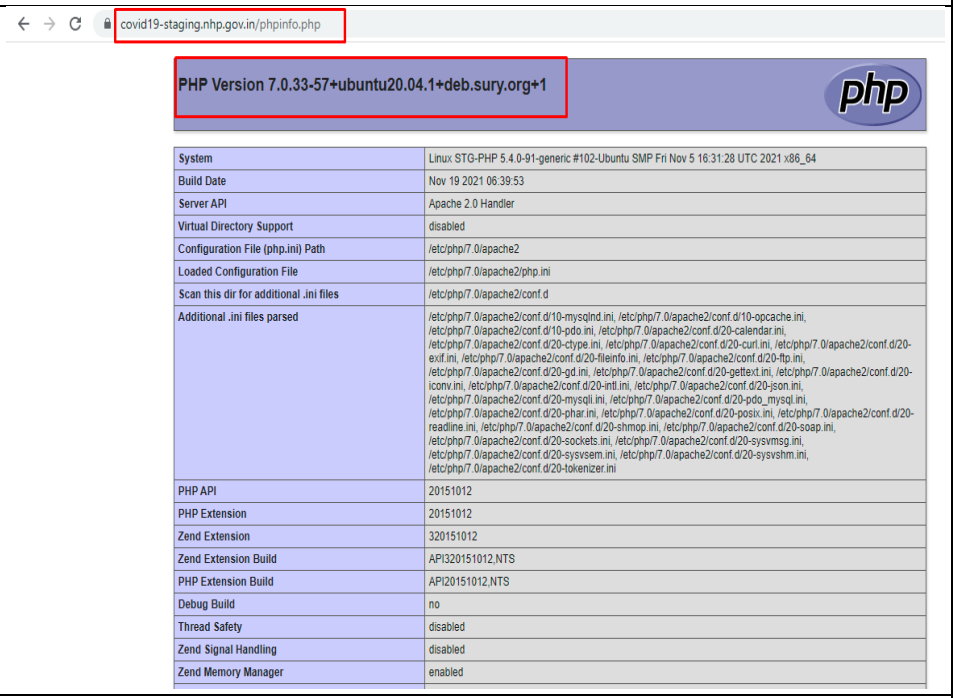
Web Application Security Test Report For Covidcare Integration

Snapshot	
Affected Site	**Throughout the application**

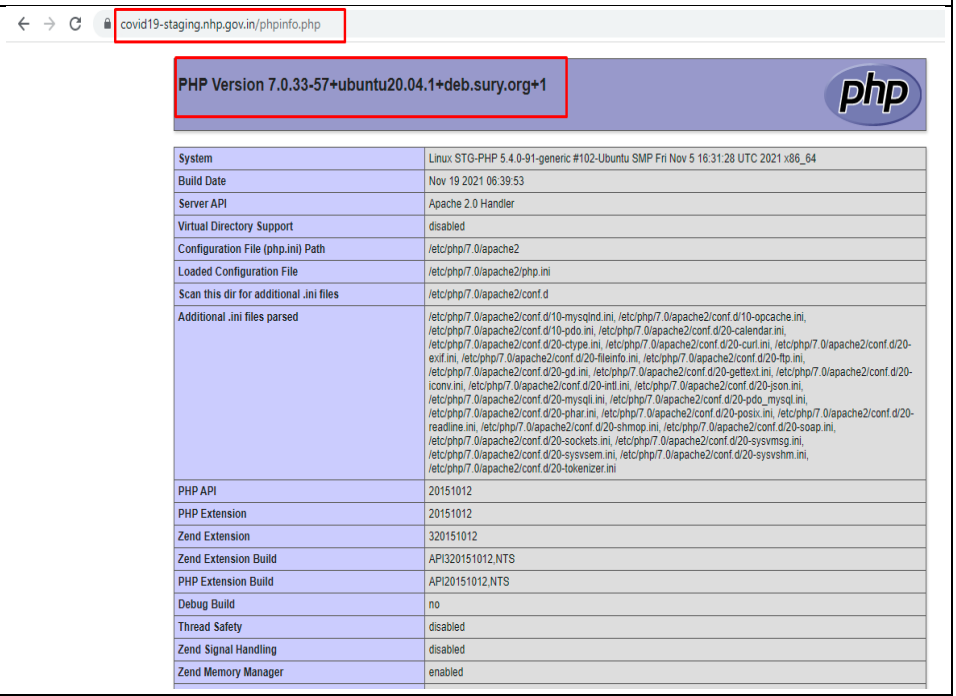
14. Session Timeout is not defined

8) Vulnerability Title: Session Timeout is not defined	
Risk	**Medium**
Abstract	It was observed that even if the Browser is logged in and idle it does not logout the session automatically
Ease of Exploitation	Easy
Impact	It is possible to access authenticated pages.
Recommendations	Application should terminate a session if there is no activity from the user-side for a fixed period of time, e.g., 15 minutes.
Snapshot	-
Affected URLs	**throughout the application**

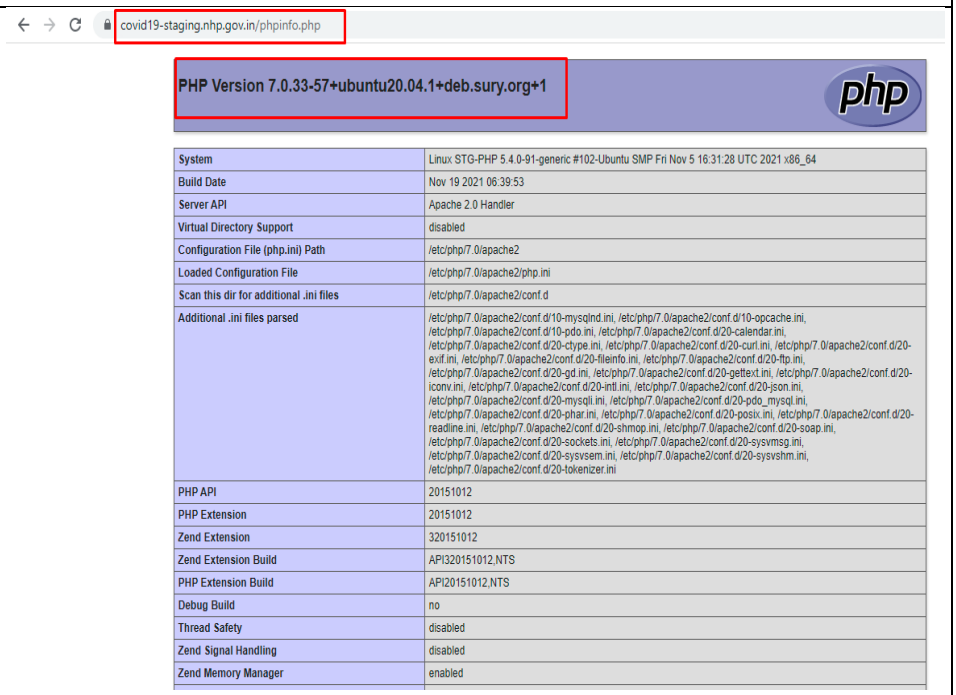
15. PHP allow_url_fopen enabled

9) Vulnerability Title: PHP allow_url_fopen enabled	
Risk	Medium
Abstract	The PHP configuration directive allow_url_fopen is enabled. When enabled, this directive allows data retrieval from remote locations (web site or FTP server). A large number of code injection vulnerabilities reported in PHP-based web applications are caused by the combination of enabling allow_url_fopen and bad input filtering.
Ease of Exploitation	Easy
Impact	Application dependant - possible remote file inclusion.
Recommendations	You can disable allow_url_fopen from either php.ini (for PHP versions newer than 4.3.4) or .htaccess (for PHP versions up to 4.3.4). php.ini allow_url_fopen = 'off' .htaccess php_flag allow_url_fopen off
Snapshot	
Affected URLs	**throughout the application**

16. PHP open_basedir is not set

10) Vulnerability Title: PHP open_basedir is not set	
Risk	Medium
Abstract	The open_basedir configuration directive will limit the files that can be opened by PHP to the specified directory-tree. When a script tries to open a file with, for example, fopen() or gzopen(), the location of the file is checked. When the file is outside the specified directory-tree, PHP will refuse to open it. open_basedir is a good protection against remote file inclusion vulnerabilities.
Ease of Exploitation	Easy
Impact	Application dependant - possible remote file inclusion.
Recommendations	You can set open_basedir from php.ini php.ini open_basedir = your_application_directory
Snapshot	
Affected URLs	**throughout the application**

17. PHP info page Found

11) Vulnerability Title: PHP info page Found	
Risk	Medium
Abstract	PHPinfo page has been found in this directory. The PHPinfo page outputs a large amount of information about the current state of PHP. This includes information about PHP compilation options and extensions, the PHP version, server information and environment (if compiled as a module), the PHP environment, OS version information, paths, master and local values of configuration options, HTTP headers, and the PHP License.
Ease of Exploitation	Easy
Impact	This file may expose sensitive information that may help an malicious user to prepare more advanced attacks.
Recommendations	Remove the file from production systems.
Snapshot	
Affected URLs	**throughout the application**

System	Linux STG-PHP 5.4.0-91-generic #102-Ubuntu SMP Fri Nov 5 16:31:28 UTC 2021 x86_64
Build Date	Nov 19 2021 06:39:53
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.0/apache2
Loaded Configuration File	/etc/php/7.0/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.0/apache2/conf.d
Additional .ini files parsed	/etc/php/7.0/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.0/apache2/conf.d/10-opcache.ini, /etc/php/7.0/apache2/conf.d/10-pdo.ini, /etc/php/7.0/apache2/conf.d/20-calendar.ini, /etc/php/7.0/apache2/conf.d/20-ctype.ini, /etc/php/7.0/apache2/conf.d/20-curl.ini, /etc/php/7.0/apache2/conf.d/20-exif.ini, /etc/php/7.0/apache2/conf.d/20-fileinfo.ini, /etc/php/7.0/apache2/conf.d/20-ftp.ini, /etc/php/7.0/apache2/conf.d/20-gd.ini, /etc/php/7.0/apache2/conf.d/20-gettext.ini, /etc/php/7.0/apache2/conf.d/20-iconv.ini, /etc/php/7.0/apache2/conf.d/20-intl.ini, /etc/php/7.0/apache2/conf.d/20-json.ini, /etc/php/7.0/apache2/conf.d/20-mysqli.ini, /etc/php/7.0/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.0/apache2/conf.d/20-phar.ini, /etc/php/7.0/apache2/conf.d/20-posix.ini, /etc/php/7.0/apache2/conf.d/20-readline.ini, /etc/php/7.0/apache2/conf.d/20-shmop.ini, /etc/php/7.0/apache2/conf.d/20-soap.ini, /etc/php/7.0/apache2/conf.d/20-sockets.ini, /etc/php/7.0/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.0/apache2/conf.d/20-sysvsem.ini, /etc/php/7.0/apache2/conf.d/20-sysvshm.ini, /etc/php/7.0/apache2/conf.d/20-tokenizer.ini
PHP API	20151012
PHP Extension	20151012
Zend Extension	320151012
Zend Extension Build	API320151012,NTS
PHP Extension Build	API20151012,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled

18. Audit Log History is not properly maintained in the application

12) Vulnerability Title: Audit trail is not properly maintained in the application	
Risk	**Medium**
Abstract	It was observed that log record system is not properly implemented to check last login details and IP of the user.
Ease of Exploitation	Easy
Impact	It is difficult to keep track of logged in users in case of any incident theft/fraud.
Recommendations	It is recommended to implement log recording system which monitor login history records An Audit trail should be incorporated in the application admin module, where all user activities have to be logged. Following points should be considered: • Audits are to be generated at the time of resource access and by the same routines accessing the resource • Information to be logged including the following: IP of the originating client, Date, Time, username if any in addition to other details to be logged in the web server. • These IP, date, time, session details, user details (NO password), referrer, process id to be logged in application logs. • To create audit logs, use auto numbering so that every logged entry has a log number, which is not editable. Then if one audit entry is deleted a gap in the numbering sequence will appear. Log entries are to be hashed/signed so that changes to audit log can be detected. • Audit trails to answer the following • Logging of Authentication Process. Success and failed attempts. • Logging Authentication details and changes. • Software error and failures logged • Should not be possible to retrieve confidential authentication information from these logs (including passwords) • Is it possible to uniquely identify both client host and user from these logs? • What level of information is logged by the application (read/write access, modification data, and copy/paste data)? • Are log files time sequential and can they positively identify the time of action?
Snapshot	-

Web Application Security Test Report For Covidcare Integration

Affected URLs	**throughout the application**

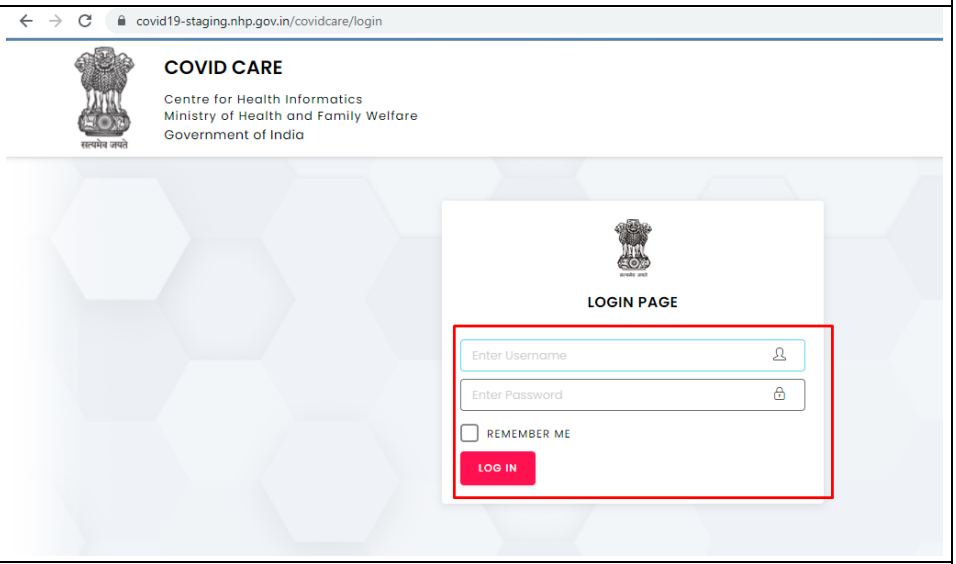
19. Back Button Enabled

3) Vulnerability Title: Back button is enabled	
Risk	**Low**
Abstract	Back button is enabled in the application.
Ease of Exploitation	Easy
Impact	It is possible to access authenticated pages through back button of the browser.
Recommendations	Restriction on accessing the application through back button to be implemented properly. After logout, all session values to be expired & application to be redirected on login page where the back button has to be disabled properly.
Affected URLs	**throughout the application**

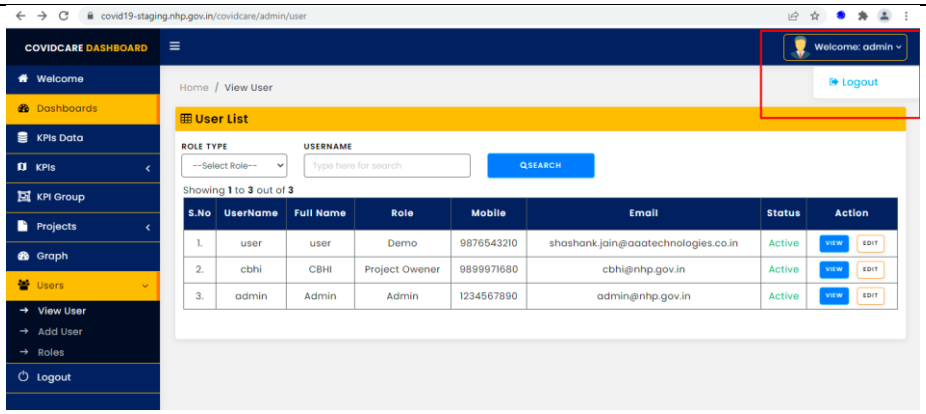
Web Application Security Test Report For Covidcare Integration

Low

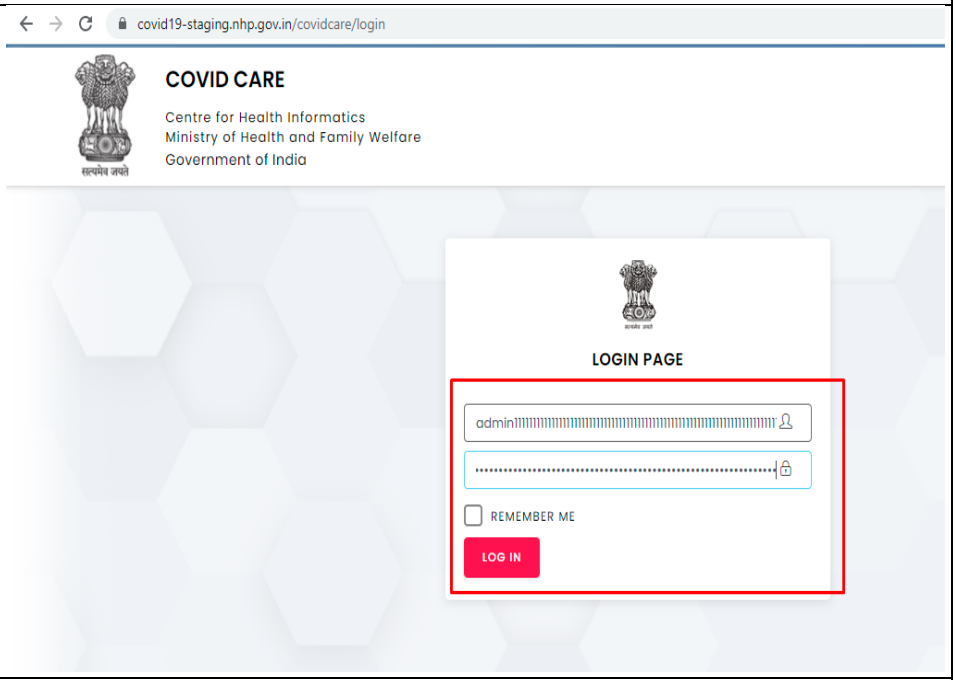
20. Forgot Password is not implemented in the application.

1) Vulnerability Title: Forgot password is not available for the users.	
Risk	**Low**
Abstract	It was observed that Forgot password is not implemented on the login page.
Ease of Exploitation	Easy
Impact	User is not able to retrieve their password.
Recommendations	Users may be required to retrieve their password. Users should be provided with a “forgot password” option through which user will retrieve their password whenever required. Forgot password should be enabled with the user’s email address. However, if the password retrieval is internal in the application then it is recommended to implement a hyperlink on login page resulting to a static page containing a message. “Please contact your site administrator at mail_id[at]domain[dot]com”. Please note that the email address in the message should not be a hyperlink.
Snapshot	
Affected URLs	**throughout the application**

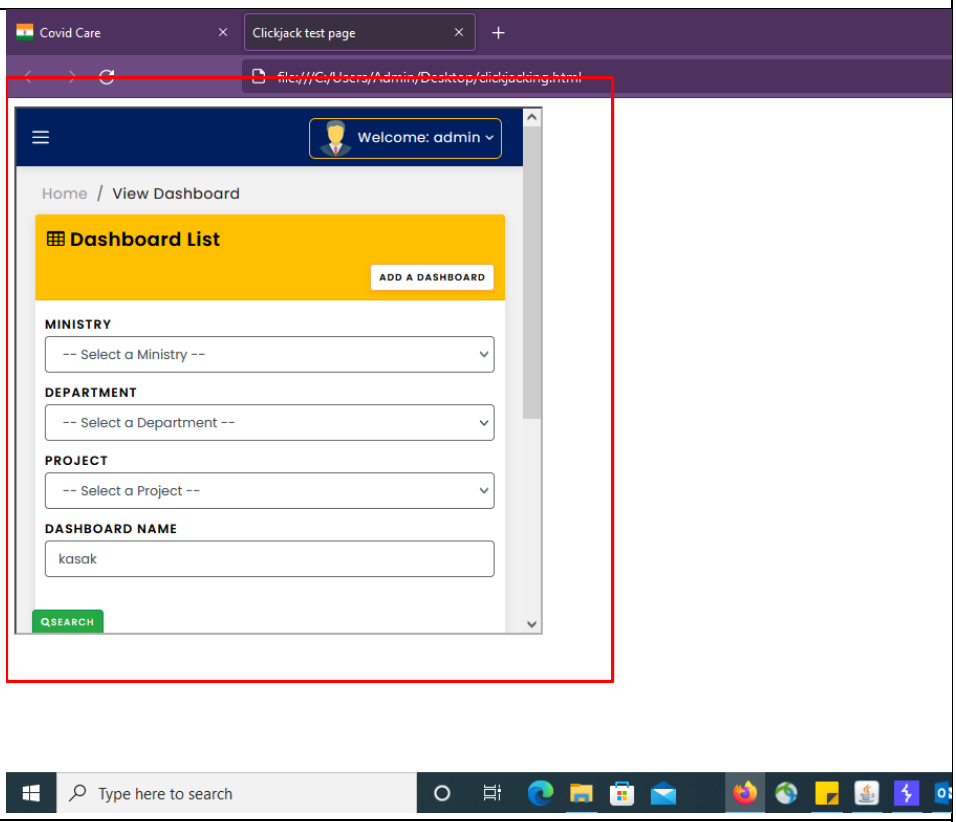
21. Change Password is not implemented in the application.

2) Vulnerability Title: Change password is not available for the users.	
Risk	**Low**
Abstract	It was observed that Change password is not implemented in the application.
Ease of Exploitation	Easy
Impact	User is not able to change their password.
Recommendations	Users may be required to change their password. Users should be provided with a “Change Password” module through which user will change their password whenever required. There are the following conditions should be implemented for Change Password module: 1.The password between client and server must be passed in SHA-256 hash technique. 2.Passwords should have restrictions that require a minimum size (8-15 characters) and complexity for the password. Complexity typically requires the use of minimum combinations of alphabetic, numeric, and non-alphanumeric characters in a user?s password (e.g. at least one special character ($,@,#,&), one upper case letter, one lower case letter and one number like Test@123). 3.Users should be prevented from reusing their current or previous 3 passwords. Password history should ideally be 3.
Snapshot	
Affected URLs	**throughout the application**

22. Max. Length of Input Fields is not defined in the application

3) Vulnerability Title: Max. length of input field is not defined in the application	
Risk	**Low**
Abstract	It was observed that max. Length for input fields is not defined.
Ease of Exploitation	Easy
Impact	This may lead to a buffer overflow attack.
Recommendations	Length restriction for every input field should be defined at client as well as at server end.
Snapshot	
Affected URLs	**throughout the application**

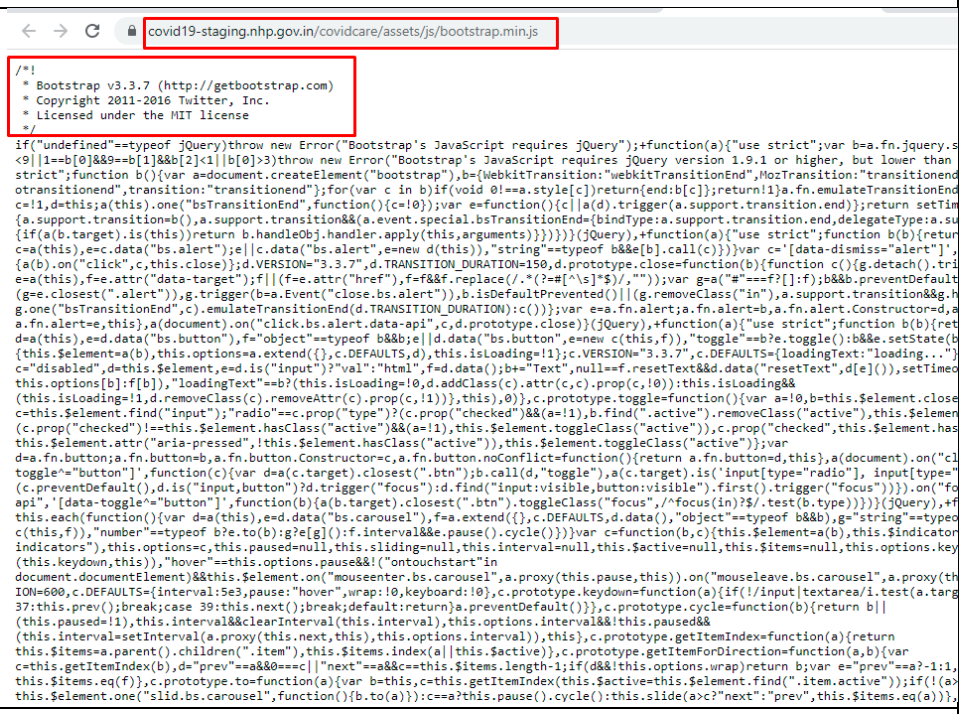
23. X-Frame header is not implemented in the application

4) Vulnerability Title: X-frame header is not implemented in the application	
Risk	**Low**
Abstract	It was observed that x-frame header is not implemented in the application.
Ease of Exploitation	Easy
Impact	Missing of X-frame header may lead to click jacking attack.
Recommendations	It is recommended to implement X-frame header. To effectively prevent framing attacks, the application should return a response header with the name X-Frame-Options and the value DENY to prevent framing altogether, or the value SAMEORIGIN to allow framing only by pages on the same origin as the response itself. Note that the SAMEORIGIN header can be partially bypassed if the application itself can be made to frame untrusted websites.
Snapshot	
Affected URLs	**throughout the application**

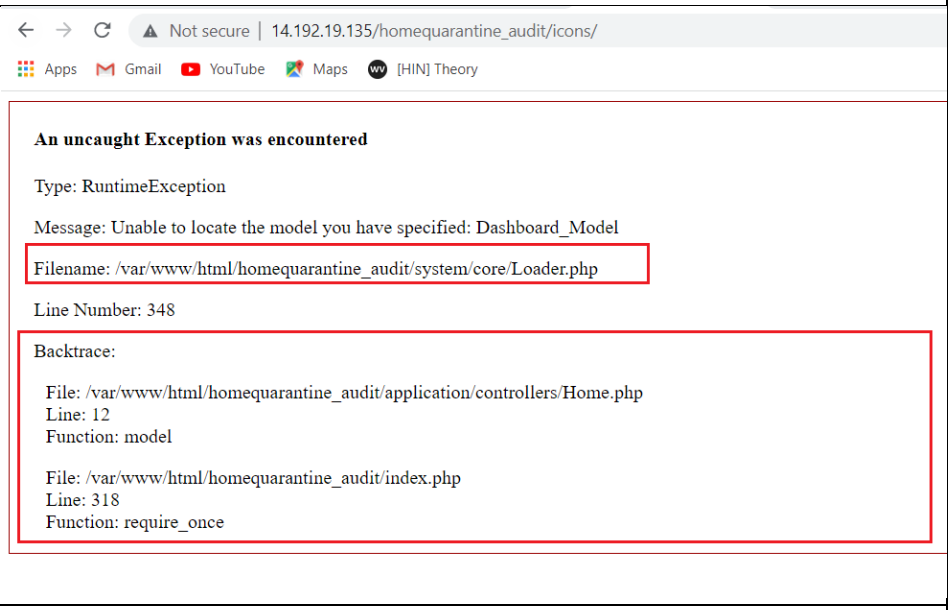
24. Account Lockout policy not implemented in the application

5) Vulnerability Title: Account Lockout policy not implemented in the application	
Risk	**Low**
Abstract	It was observed that This login page doesn't have any protection against password-guessing attacks (brute force attacks).
Ease of Exploitation	Easy
Impact	An attacker may attempt to discover a weak password by systematically trying every possible combination of letters, numbers, and symbols until it discovers the one correct combination that works.
Recommendations	It's recommended to implement some type of account lockout after a defined number of incorrect password attempts.
Snapshot	-
Affected URLs	**throughout the application**

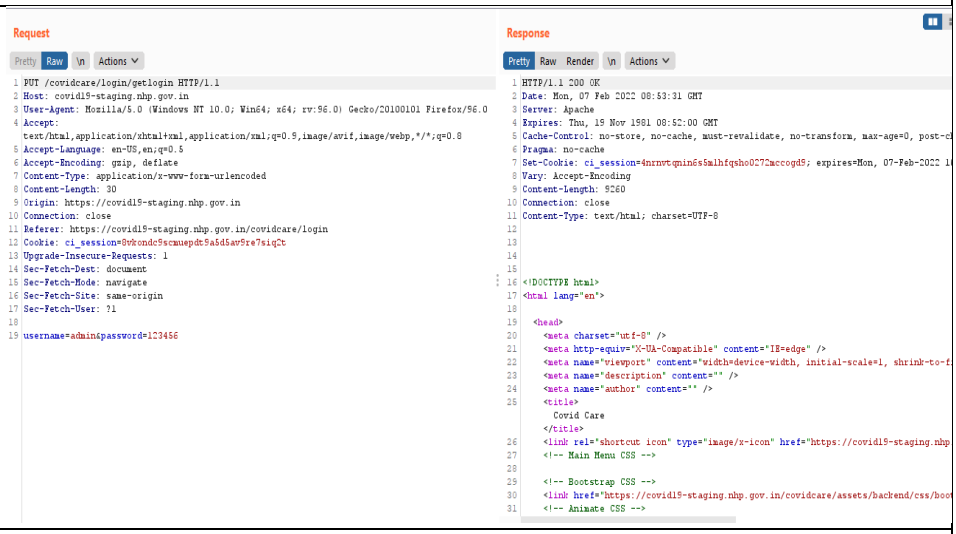
25. Old & Vulnerable version of Bootstrap in the application

6) Vulnerability Title: Old & Vulnerable bootstrap version	
Risk	**Low**
Abstract	It is observed that old & vulnerable bootstrap used
Ease of Exploitation	Easy
Impact	An attacker can steal the cookies as well as the user session id and also Perform XSS
Recommendations	It is recommended that use only Updated Bootstrap version in the application
Snapshot	
Affected URLs	**Throughout the application**

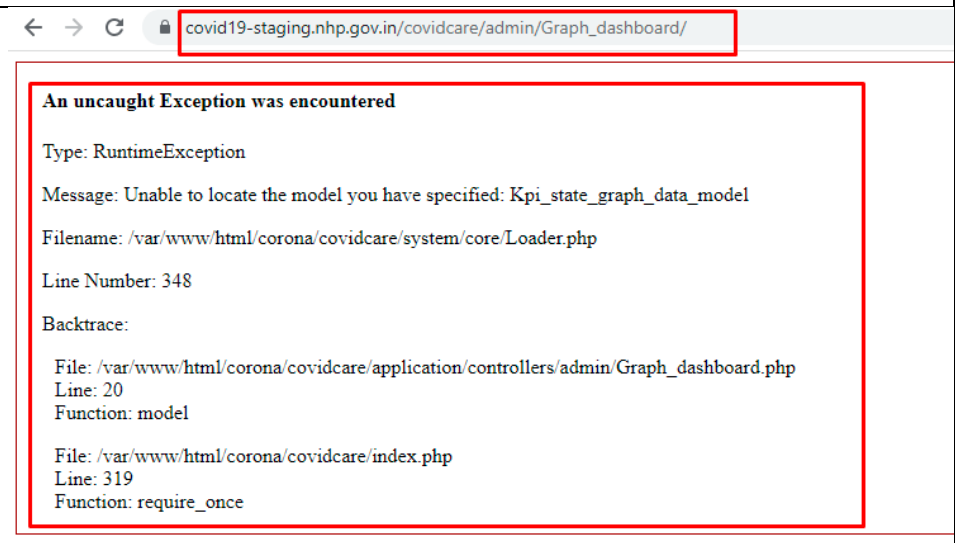
26. Source code disclosure in the application.

7) Vulnerability Title: Sensitive information disclosed	
Risk	**Low**
Abstract	It is observed that sensitive information is disclosed in the application
Ease of Exploitation	Easy
Impact	It may lead to perform multiple vulnerabilities by attacker
Recommendations	It is recommended that sensitive information should not disclosed in the application
Snapshot	
Affected URLs	----

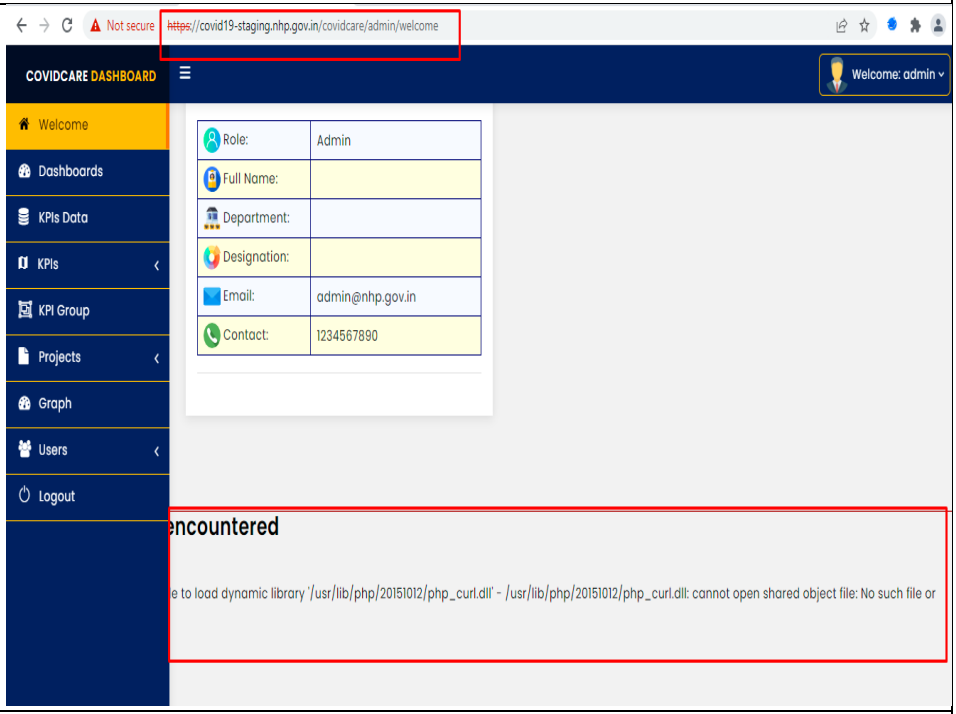
27. Security headers are not implemented in the application.

8) Vulnerability Title: Security headers not implemented	
Risk	**Low**
Abstract	It is observed that security headers are not implemented in the application
Ease of Exploitation	Easy
Impact	It may lead to perform attacks like XSS, Clickjacking etc.
Recommendations	It is recommended that sensitive information should not disclosed in the application
Snapshot	
Affected URLs	----

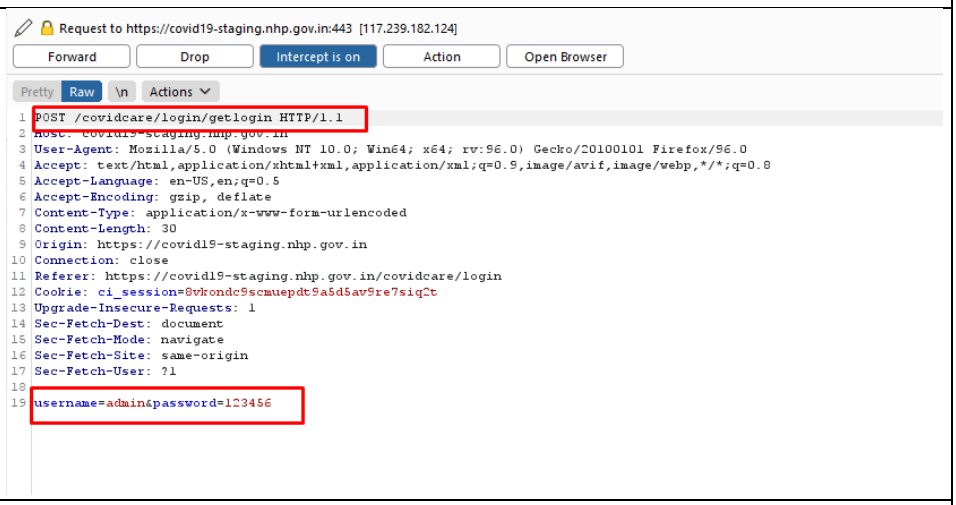
28. Functionality issues in the application

9) Vulnerability Title: Functionality issue	
Risk	**Low**
Abstract	It was observed that Functionality issues are in the application
Ease of Exploitation	Medium
Impact	Attacker can advantage of this and implement upload or store malicious script or vulnerable page in the application
Recommendations	It is recommended that application should be functional properly
Snapshot	
Affected URLs	**https://covid19-staging.nhp.gov.in/covidcare/admin/Graph_dashboard/**

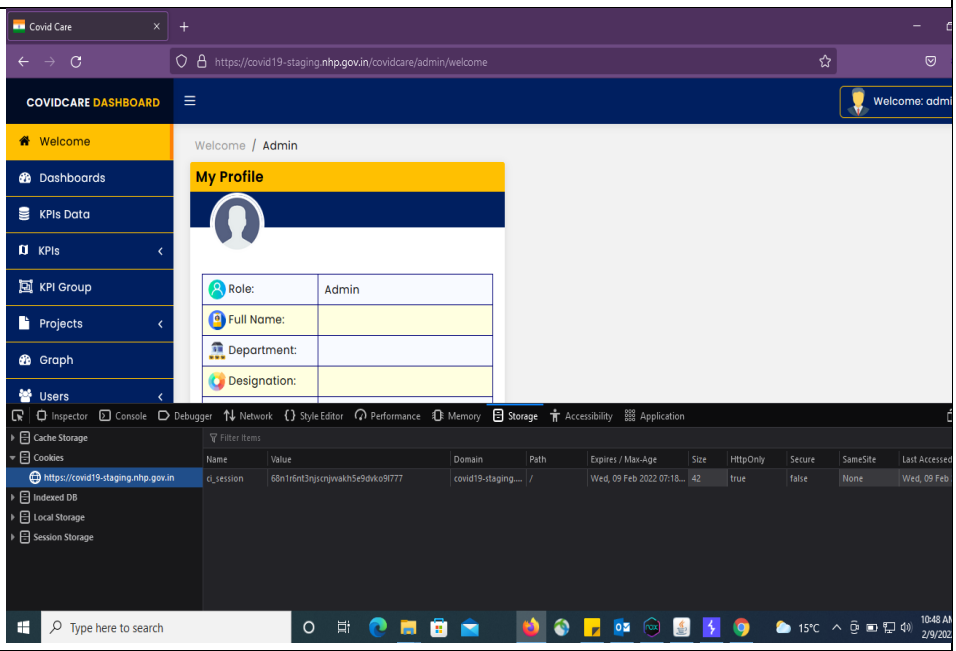
29. Sensitive information disclosed in the application.

10) Vulnerability Title: Sensitive information disclosed	
Risk	**Low**
Abstract	It is observed that sensitive information is disclosed in the application
Ease of Exploitation	Easy
Impact	It may lead to perform multiple vulnerabilities by attacker
Recommendations	It is recommended that sensitive information should not disclosed in the application
Snapshot	
Affected URLs	----

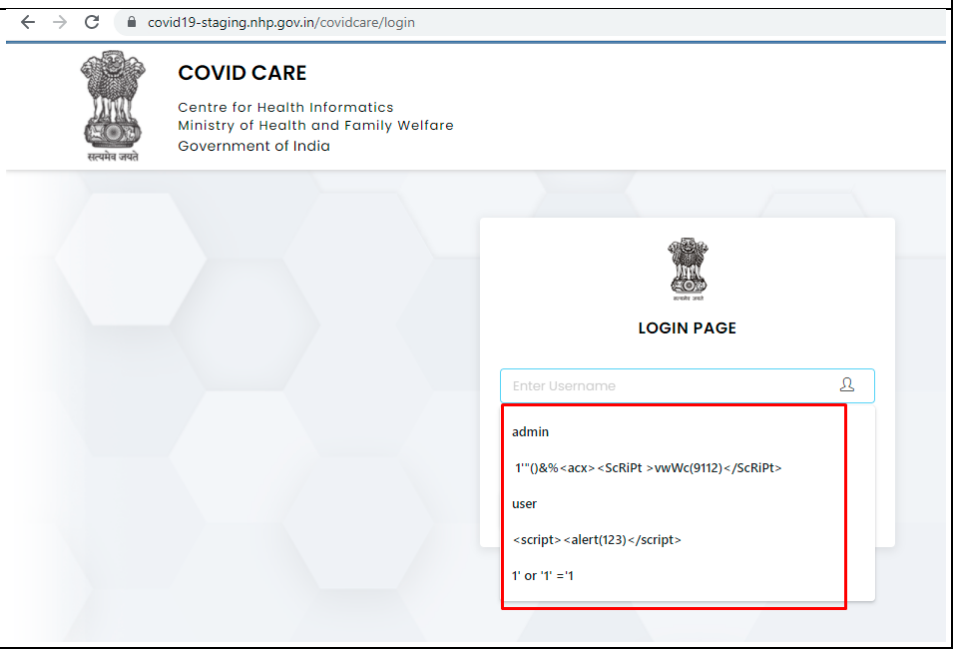
30. Password Complexity is not defined

11) Vulnerability Title: Password complexity is not defined	
Risk	**Low**
Abstract	It was observed that password complexity is not implemented in the application
Ease of Exploitation	Easy
Impact	The attacker can guess the password or the attacker may perform the simple brute force attack to find the password.
Recommendations	Passwords should have restrictions that require a minimum size (8-15 characters) and complexity for the password. Complexity typically requires the use of minimum combinations of alphabetic, numeric, and non-alphanumeric characters in a user's password (e.g. one special character ($,@,#,&),one upper case letter and one lower case letter and one number like Test@123).
Snapshot	
Affected URLs	**throughout the application**

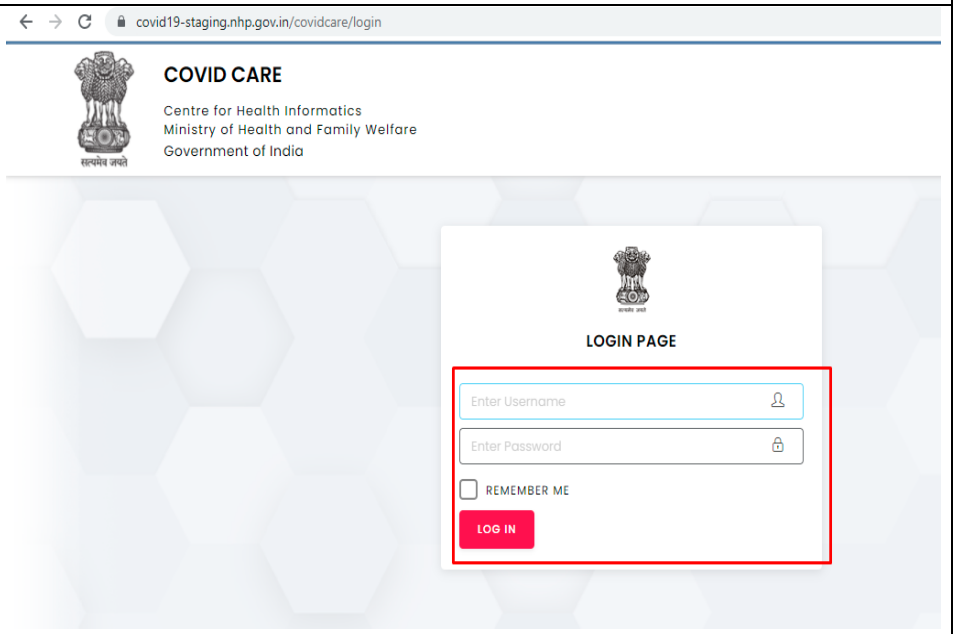
31. Path is set to Default in the application

12) Vulnerability Title: Path is set to Default in the application	
Risk	**Low**
Abstract	It was observed that path is set to default i.e. '/' in the application.
Ease of Exploitation	Easy
Impact	It is difficult to keep track of logged in users in case of any incident theft/fraud.
Recommendations	It is recommended to verify that that the path attribute, just as the Domain attribute, has not been set too loosely. Even if the Domain attribute has been configured as tight as possible, if the path is set to the root directory "/" then it can be vulnerable to less secure applications on the same server.
Snapshot	
Affected URLs	**throughout the application**

32. Auto Complete is Enabled

13) Vulnerability Title: Auto complete is enabled in the application	
Risk	**Low**
Abstract	It was observed that auto complete is possible in the application.
Ease of Exploitation	Easy
Impact	This could be lead to its compromise or re-usage without the user's content or approval.
Recommendations	Auto fill option should disable in the application.
Snapshot	
Affected URLs	**throughout the application**

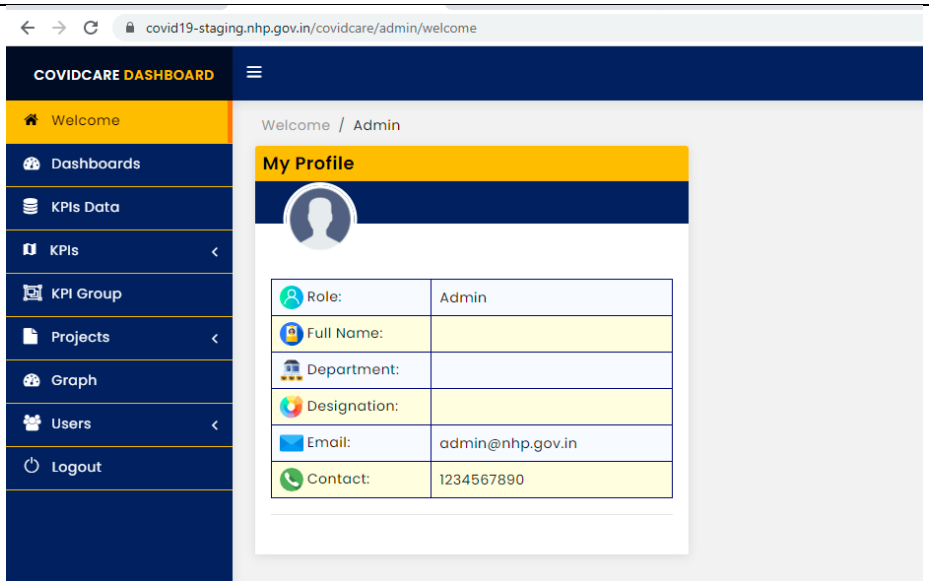
33. CAPTCHA is missing in the application.

14) Vulnerability Title: CAPTCHA is missing in the application	
Risk	**Low**
Abstract	It was observed that CAPTCHA is missing in the application.
Ease of Exploitation	Easy
Impact	The attacker may perform the DOS attack and harm the server.
Recommendations	CAPTCHA should follow the following condition: a) The combination of alphanumeric value. b) Combination of Upper case and lower-case letters. c) Case-Sensitive d) Its length should be minimum 6 characters. e) Should not be a third-party CAPTCHA: f) Should be Random and not follow a pattern. g) Example: Ab73jy, PT34h8, Hos3t3, nic23n etc.
Snapshot	
Affected URLs	**throughout the application**

34. Same user logged in in multiple browser.

15) Vulnerability Title: Same user logged in in multiple browser	
Risk	**Low**
Abstract	It is observed that Same user logged in multiple browser
Ease of Exploitation	Easy
Impact	The attacker may perform the DOS attack and harm the server.
Recommendations	It is recommended that Any user should allowed one login at a time on a single browser
Snapshot	-----
Affected URLs	**throughout the application**

35. Email Spamming in the application.

16) Vulnerability Title: Email Spamming in the application	
Risk	**Low**
Abstract	It is observed that Same user logged in multiple browser
Ease of Exploitation	Easy
Impact	E-mail flooding etc
Recommendations	Email addresses should be posted as an image not as a hyperlink. Alternatively, instead of @symbol, [at] should be used. Similarly the dot character (.) should be replaced by [dot]. So abc@nic.in should be written as abc[at]nic[dot]in.
Snapshot	
Affected URLs	**throughout the application**