



Web-Application Penetration Testing Report

For

Website URLs:

<https://rcb.res.in/>

26 September 2025

Confidential Report, not to be circulated or reproduced without appropriate authorization.

Contact Us:

Application Security Audit Division
(Cyber Security Group)
National Informatics Centre
A – Block, CGO Complex, Lodhi Road
New Delhi – 110003



011-2430-5129
011-2430-5210
011-2430-5142
011-2430-5971

Contributions:

	Name	Role
1.	Ms. Sunita Maurya Mr. Sudhir Kumar Verma	Reviewer
2.	Mr. Anil Kumar Jha	HOD
3.	Mr. Rajesh Mishra	HOG

Key Findings

- 1. Denial of Service Attack**
- 2. File Upload Functionality in Public Module**
- 3. Insecure Transmission of Password**
- 4. Host Header Injection**
- 5. Robots.txt File Accessible**
- 6. Internal Restricted File Accessible**
- 7. Clipboard Data Stealing Attack**
- 8. No Implementation of Forgot Password Functionality**
- 9. Missing Security Response Header**
- 10. Cross Origin Resource Sharing**
- 11. Email Harvesting**

1. Denial of Service Attack

Incident URL:

- <https://rcb.res.in/user/register>

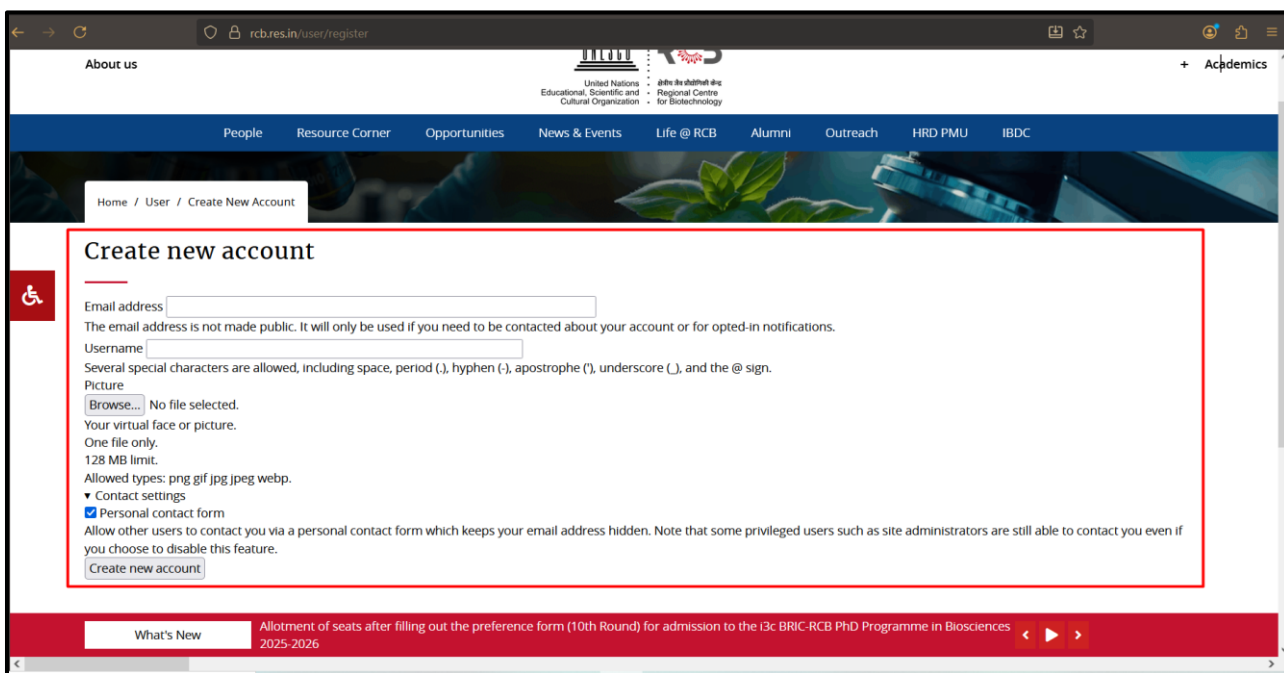
Description: An attacker over the internet may launch an automated denial of service attack against the application as there is no CAPTCHA implemented.

Impact: An attacker over the internet can launch an automated denial of service attack against the application as there is no CAPTCHA implemented. Since the attacker can cause unavailability of the resources for genuine user.

Severity: High

How to Test:

Step I: An attacker navigates to the application and it is observed that CAPTCHA is not implemented as shown below. As a result, an attacker may run automated tool to perform a denial of service attack which may cause unavailability of resources to the users.



Recommendation(s):

1. The application may restrict such attacks by implementing a CAPTCHA to ensure that no automated attack can be run.

2. File Upload Functionality in Public Module

Incident URL:

- <https://rcb.res.in/user/register>

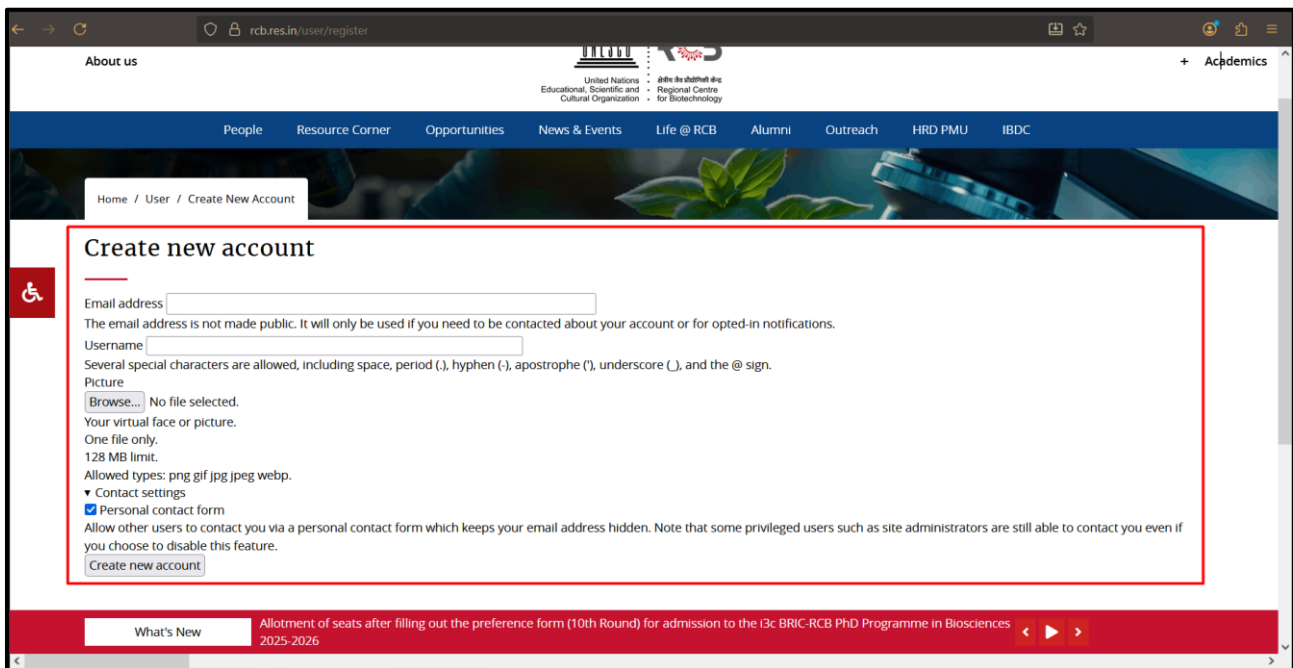
Description: A file upload functionality is present in the public page of the application.

Impact: A file upload functionality is present in the public page of the application as a result, an attacker may upload a malicious executable file into the application using file upload in public page.

Severity: High

How to Test:

Step I: An attacker navigates to the registration page of the application and observes that file upload option is present in the public page.



Recommendation(s):

1. The application should not allow files to be uploaded in the public page.
2. The application should keep track of who is uploading files and when and log all file upload activity to help detect potential security incidents.

3. Insecure Transmission of Password

Incident URL:

- <https://rcb.res.in/user/login>

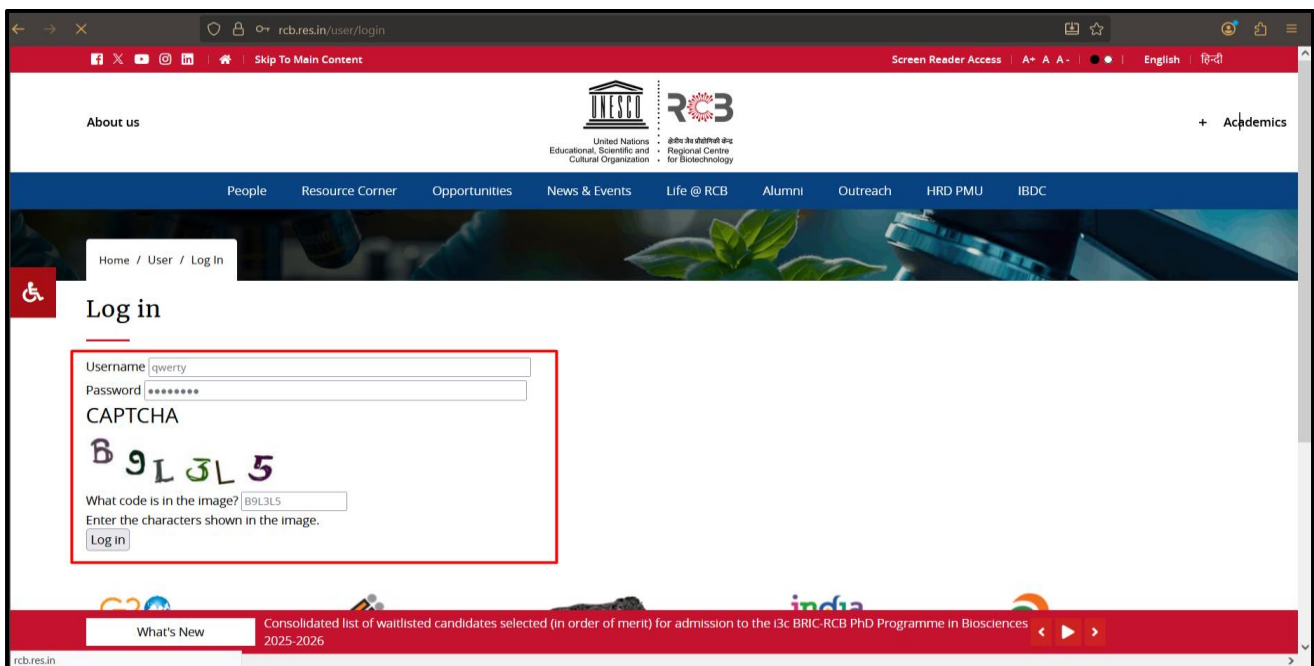
Description: The application transmits passwords in plaintext from client to server.

Impact: The application transmits passwords in plaintext from client to server. This exposes sensitive authentication data to interception by attackers can lead to credential theft, unauthorized access, and potential compromise of user accounts.

Severity: Medium

How to Test:

Step I: An attacker navigates to the 'Login page' and try to login with Credentials as shown below:

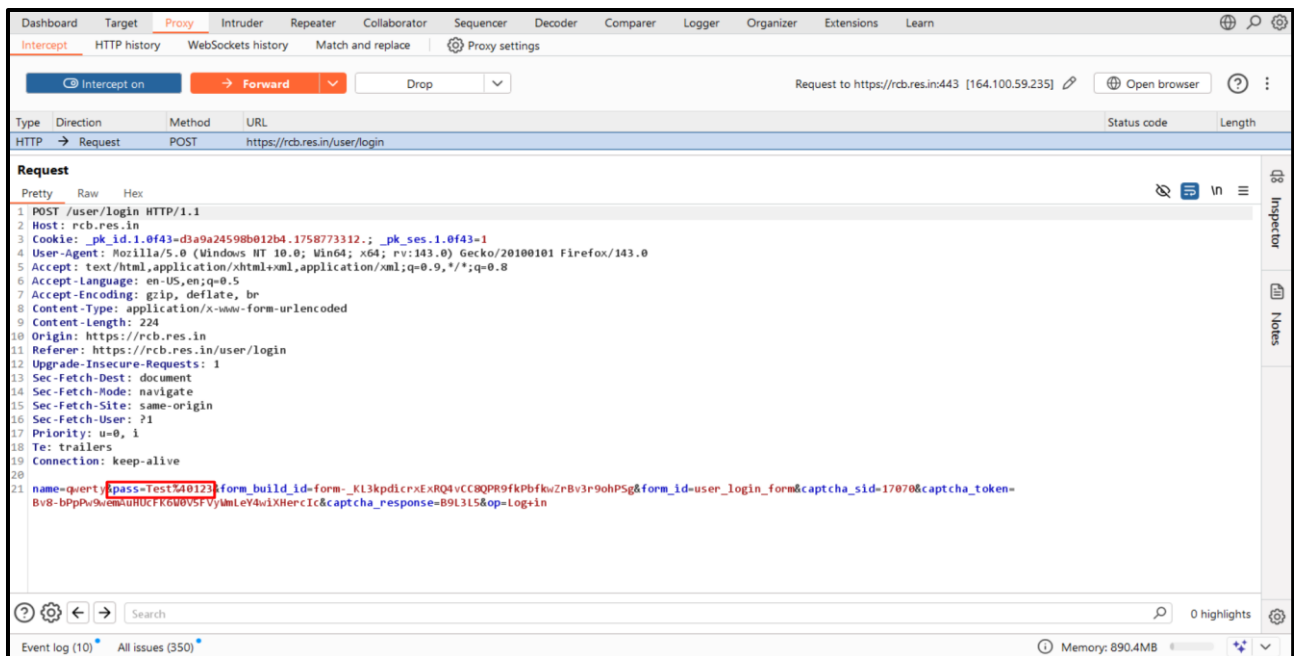


The screenshot shows the login page of the RCB (Regional Centre for Biotechnology) website. The URL in the browser is rcb.res.in/user/login. The page has a red header with navigation links and a blue sidebar. The main content area is titled 'Log in' and contains a form with the following fields:

- Username:
- Password:
- CAPTCHA: 
- What code is in the image?
- Enter the characters shown in the image.
- Log in button

The login form is highlighted with a red rectangular box. At the bottom of the page, there is a red footer with the text 'What's New' and 'Consolidated list of waitlisted candidates selected (in order of merit) for admission to the I3c BRIC-RCB PhD Programme in Biosciences 2025-2026'.

Step II: An attacker captures the intermediate request using http intercepting tool and observes that password travels in clear text from client to server as shown below:



Recommendation (s):

It is recommended to implement the following:
For Login Page:

1. The application should implement the '**Salted SHA256 hashing technique**' of the password for login page.

4. Host Header Injection

Incident URL:

- <https://rcb.res.in/>

Description: The application is accepting invalid host's name; this is vulnerable to the Host Header Injection attack.

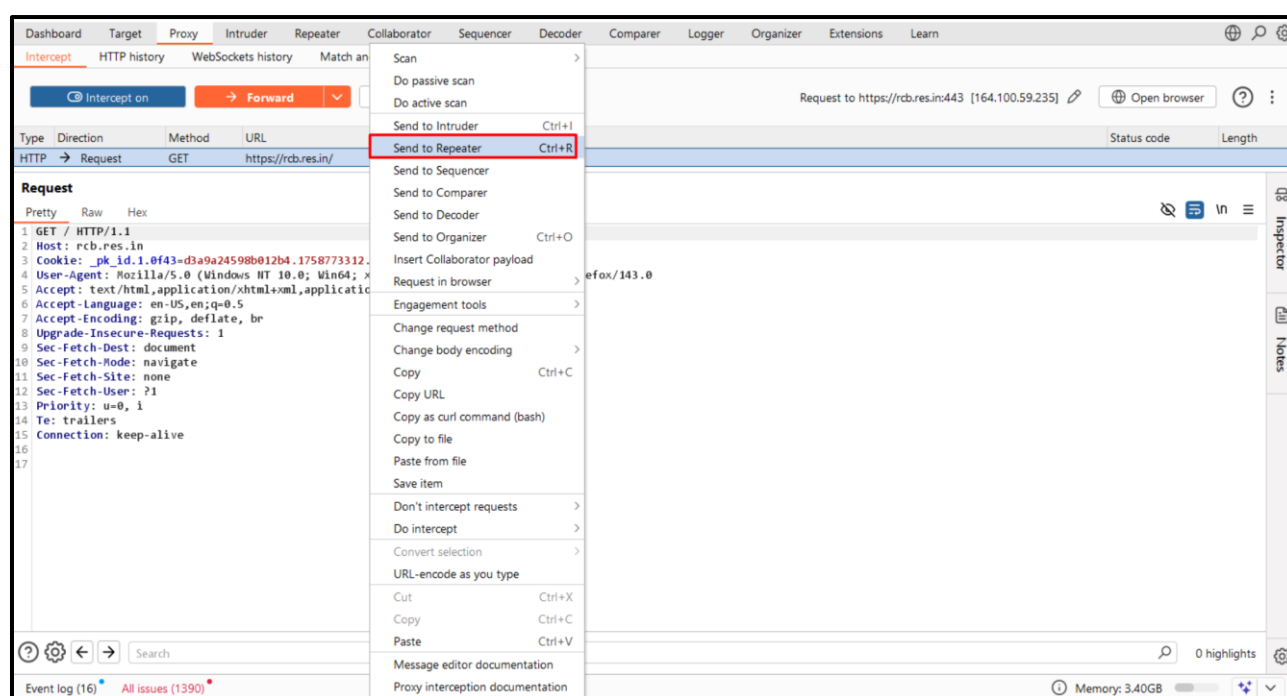
Impact: The application is accepting invalid host, the attacker can tamper the HTTP host header which may lead to web cache poisoning and password reset poisoning attacks.

Severity: Medium

How to Test:

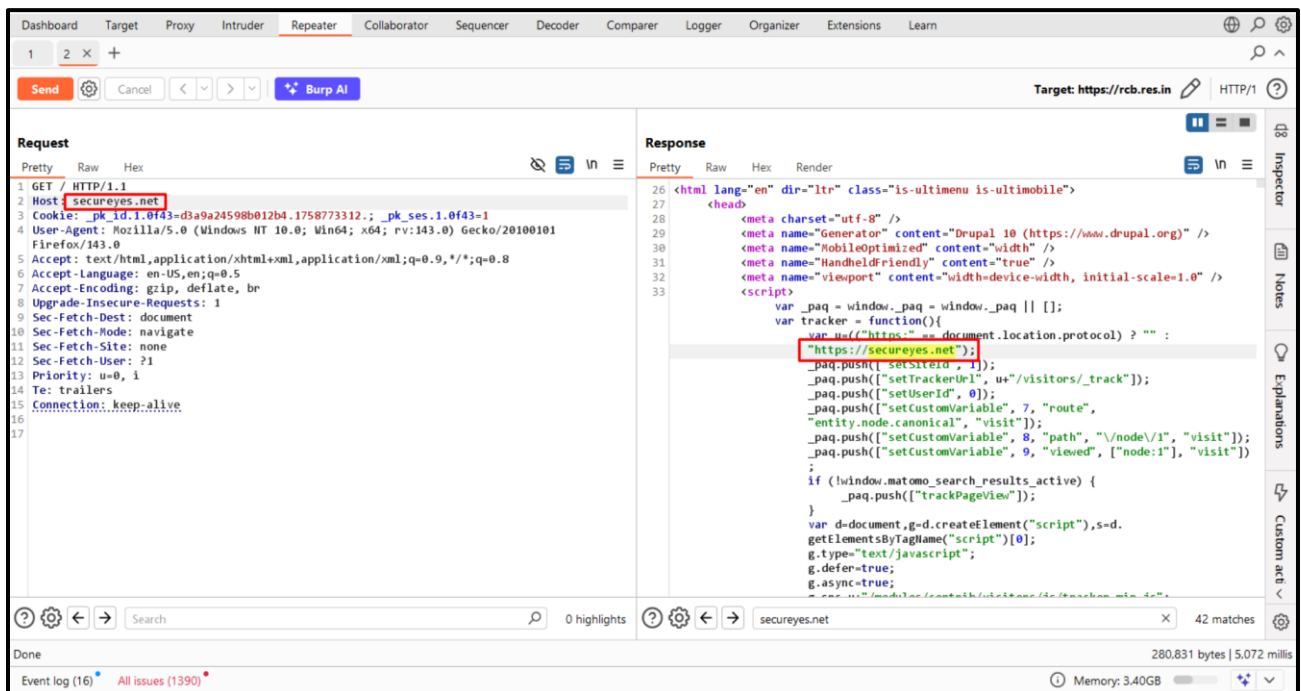
Step I: A malicious user navigates to the application and captures the request using an HTTP interceptor and the request is sent to Repeater as shown below:

Original Request:



Step II: From the below screenshot it is observed that the Host name is changed from 'rcb.res.in' to 'secureyes.net' and the manipulated host was reflected in the response as shown below:

Modified Request:



Recommendation(s):

1. Reject any request that doesn't match the target domain.
2. Validate the Host header to ensure that the request originates from that target host or not.
3. Whitelisting the trusted domains during the initial setup of the application and mapping domains received in Host header of each request with it.

5. Robots.txt File Accessible

Incident URL:

- <https://rcb.res.in/robots.txt>

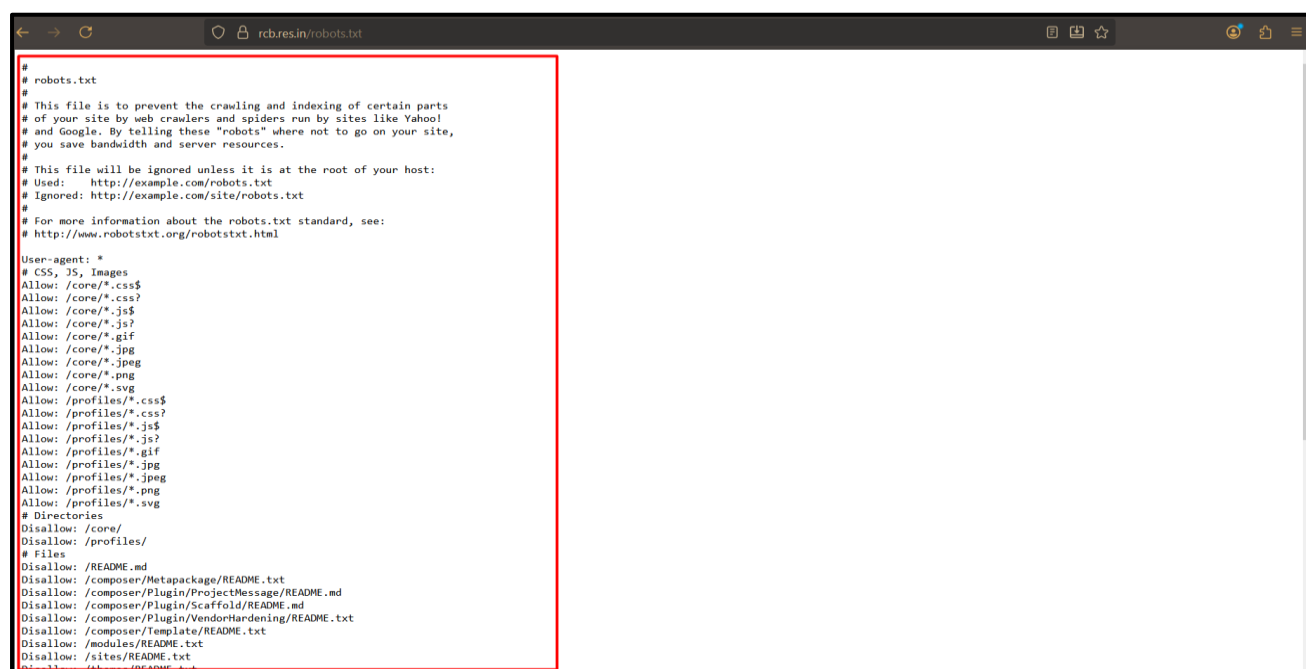
Description: Web server's robots.txt page is directly accessible over the internet.

Impact: The web server's 'robots.txt' file is accessible over the internet that discloses restricted pages of the server. This information helps the attackers to perform targeted attacks on remote server.

Severity: Low

How to Test:

Step I: An attacker navigates to the application at the URL: <https://rcb.res.in/robots.txt> and observes that robots.txt file is accessible.



```
# robots.txt
#
# This file is to prevent the crawling and indexing of certain parts
# of your site by web crawlers and spiders run by sites like Yahoo!
# and Google. By telling these "robots" where not to go on your site,
# you save bandwidth and server resources.
#
# This file will be ignored unless it is at the root of your host:
# Used: http://example.com/robots.txt
# Ignored: http://example.com/site/robots.txt
#
# For more information about the robots.txt standard, see:
# http://www.robotstxt.org/robotstxt.html
User-agent: *
# CSS, JS, Images
Allow: /core/*.css$
Allow: /core/*.css?
Allow: /core/*.js$
Allow: /core/*.js?
Allow: /core/*.gif
Allow: /core/*.jpg
Allow: /core/*.jpeg
Allow: /core/*.png
Allow: /core/*.svg
Allow: /profiles/*.css$
Allow: /profiles/*.css?
Allow: /profiles/*.js$
Allow: /profiles/*.js?
Allow: /profiles/*.gif
Allow: /profiles/*.jpg
Allow: /profiles/*.jpeg
Allow: /profiles/*.png
Allow: /profiles/*.svg
# Directories
Disallow: /core/
Disallow: /profiles/
# Files
Disallow: /README.md
Disallow: /composer/Metapackage/README.txt
Disallow: /composer/Plugin/ProjectMessage/README.md
Disallow: /composer/Plugin/Scaffold/README.md
Disallow: /composer/Plugin/VendorHardening/README.txt
Disallow: /composer/Template/README.txt
Disallow: /modules/README.txt
Disallow: /sites/README.txt
Disallow: /themes/README.txt
```

Recommendation(s):

1. Internal restricted files like robots.txt contain generic information based on the pages that the administrator does not want a web crawler to index, so, this should be restricted from public access

6. Internal Restricted File Accessible

Incident URL:

- <https://rcb.res.in/core/LICENSE.txt>

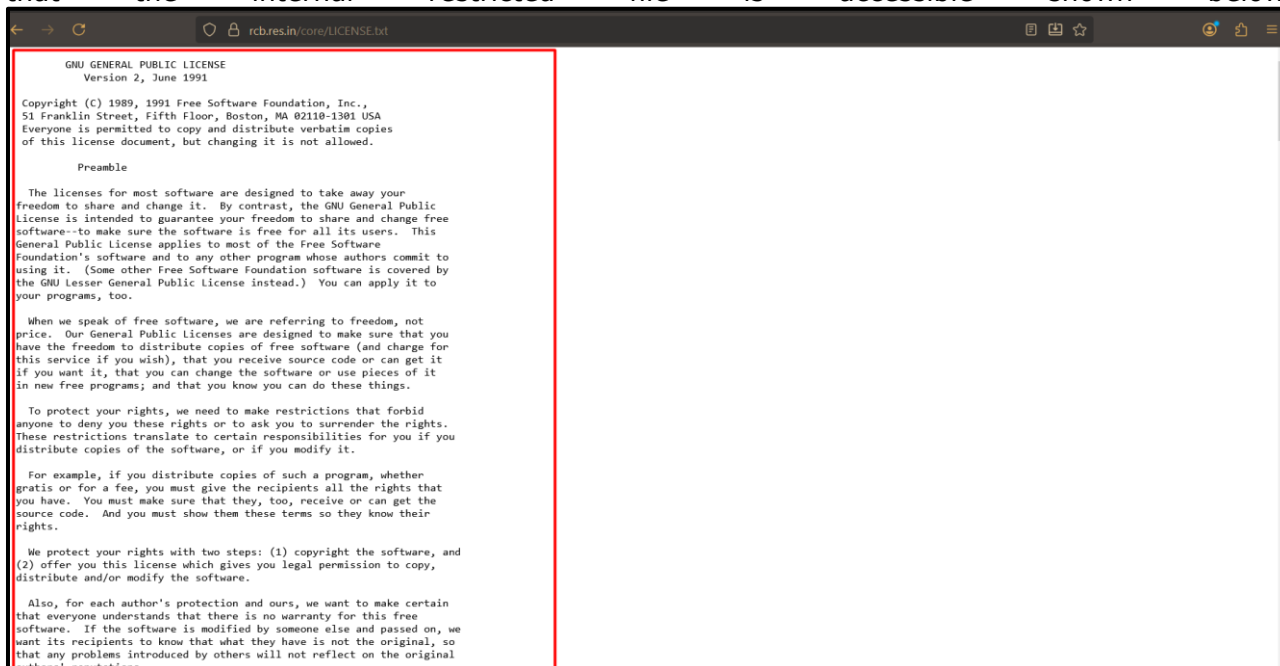
Description: Application internal restricted files can be directly accessible over the internet.

Impact: The attacker can access the internal restricted file which may allow an attacker to use the information present there to carry out further attacks on the application server.

Severity: Low

How to Test:

Step I: An attacker modifies the application URL while trying to call restricted file and observes that the internal restricted file is accessible shown below:



Recommendation(s):

1. It is recommended that access to internal restricted files must be restricted from public access.

7. Clipboard Data Stealing Attack

Incident URL:

- <https://rcb.res.in/user/login>

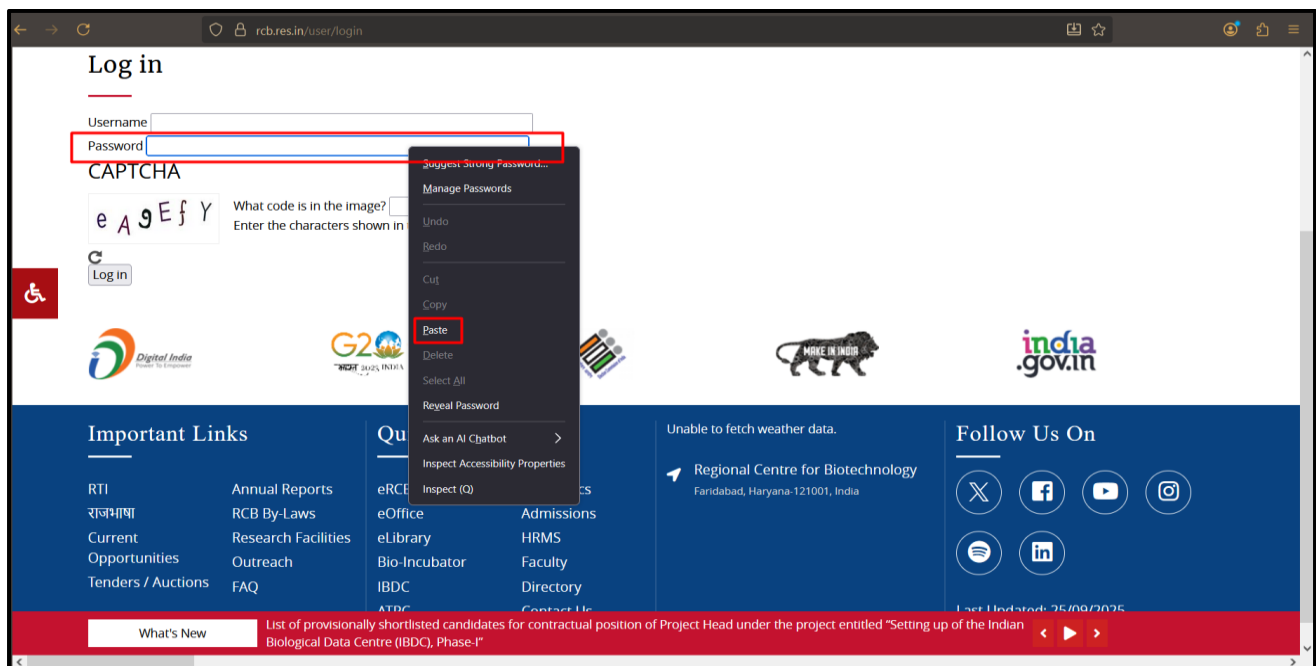
Description: An application has enabled copy and paste feature on input fields of the 'Login Page'.

Impact: The application has enabled copy and paste feature on the password field of '**Login page**' page as a result a malicious user can steal the password as copy paste option is enabled thus compromising the victim's account.

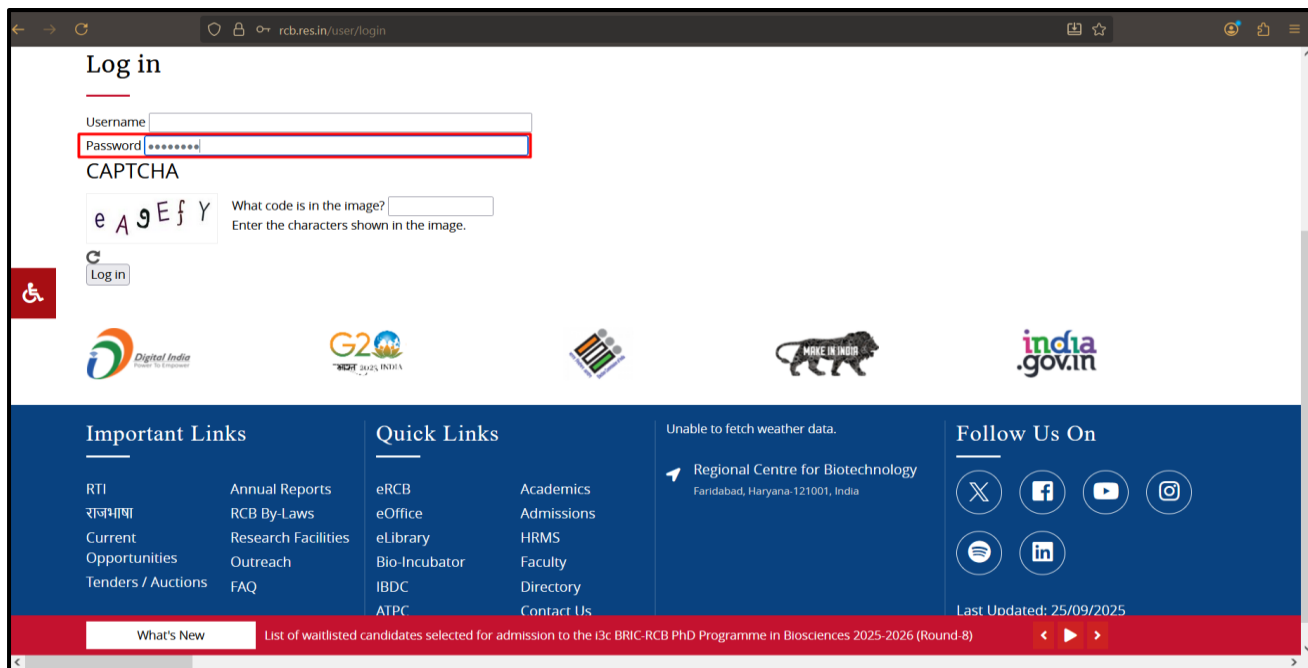
Severity: Low

How to Test:

Step I: An attacker navigates the Login Page of the application and try to paste the copied password as shown below as shown below:



Step II: From the screenshot below, it is observed that the Password has been successfully pasted in the input field as shown below:



The screenshot shows a web browser window at the URL `rcb.res.in/user/login`. The page is titled "Log in". It features a "Username" input field and a "Password" input field. The "Password" field is highlighted with a red border and contains a series of asterisks, indicating that a password has been entered. Below the password field is a CAPTCHA section with the text "What code is in the image?" and a small image of a grid of characters. A "Log in" button is located below the CAPTCHA. The page also includes a footer with various links and logos, including the Digital India logo, G20 India 2023 logo, and the India.gov.in logo.

Recommendation (s):

1. The application should disable the 'paste' option including the keyboard shortcuts of the same for the password field and for the other critical data fields.

8. No Implementation of forgot Password functionality.

Incident URL:

- <https://rcb.res.in/user/login>

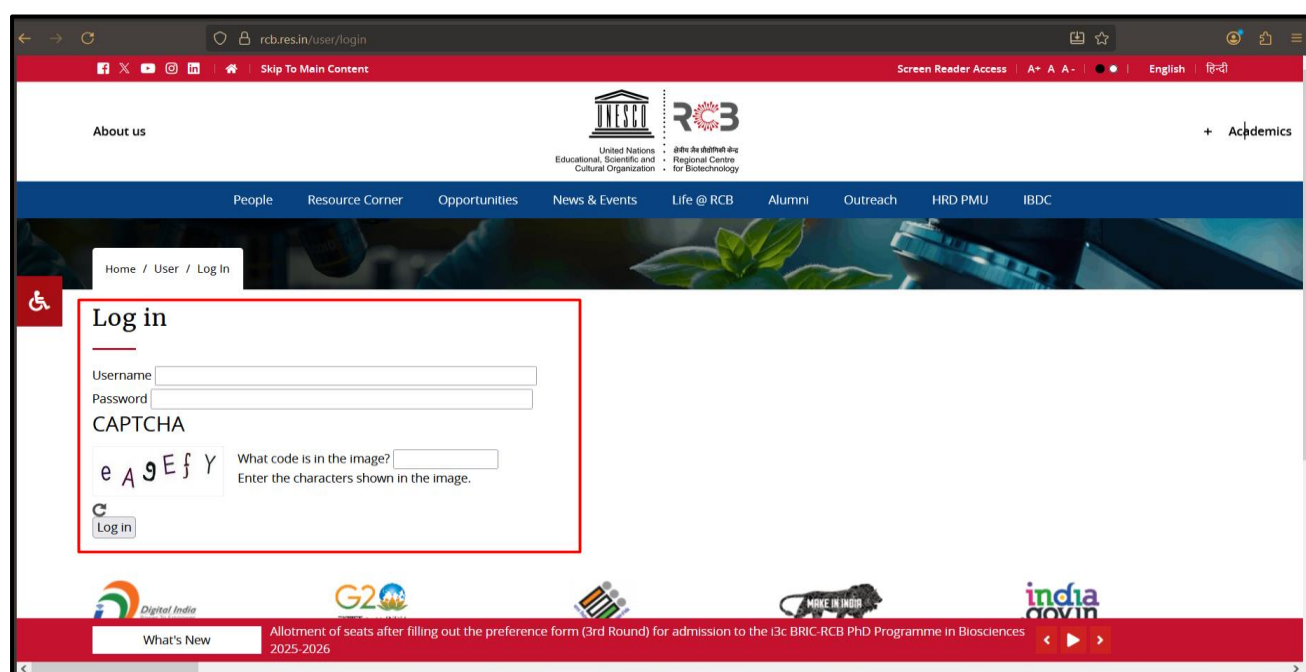
Description: The application has not implemented a forgot password page for its users.

Impact: The application has not implemented the forgot Password functionality for its users which causes inconvenience to the application users during credential theft as they cannot change password immediately during credential theft.

Severity: Low

How to Test:

Step I: A user navigates to the Login page of the application and observes that no forgot password page has been implemented as shown below:



Recommendation(s):

1. It is security best practice that the application should implement 'Forgot Password' page in the application.

9. Missing Security Response Header

Incident URL:

- <https://rcb.res.in/>

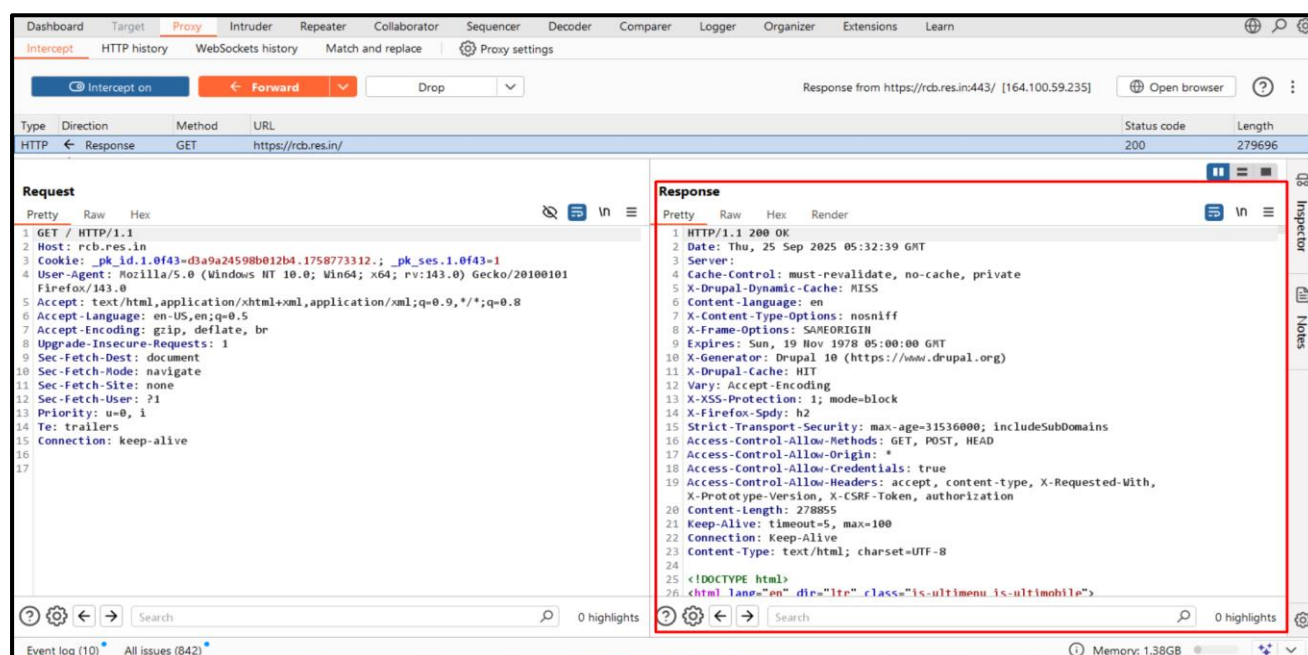
Description: The application has not implemented the Content Security Policy header in HTTP Response.

Impact: The application has not implemented Content-Security-Policy attacker may take advantage of this vulnerability and can perform man-in-the-middle attacks and weakness cookie hijacking protections.it permits the execution of JavaScript code from unsafe sources, thereby increasing the risk of XSS attacks.

Severity: Low

How to Test:

Step I: From the screenshot below, it is observed that the application has not implemented Content-Security-Policy as shown below:



Recommendation(s):

1. The application should implement Content-Security-Policy: script-src 'self', object-src 'none', default-src 'self' in response header.
2. It is recommended to remove 'unsafe-inline' unsafe-eval' from Content-Security-Policy response header.ab=4

10. Cross Origin Resource Sharing

Incident URL:

- <https://rcb.res.in/>

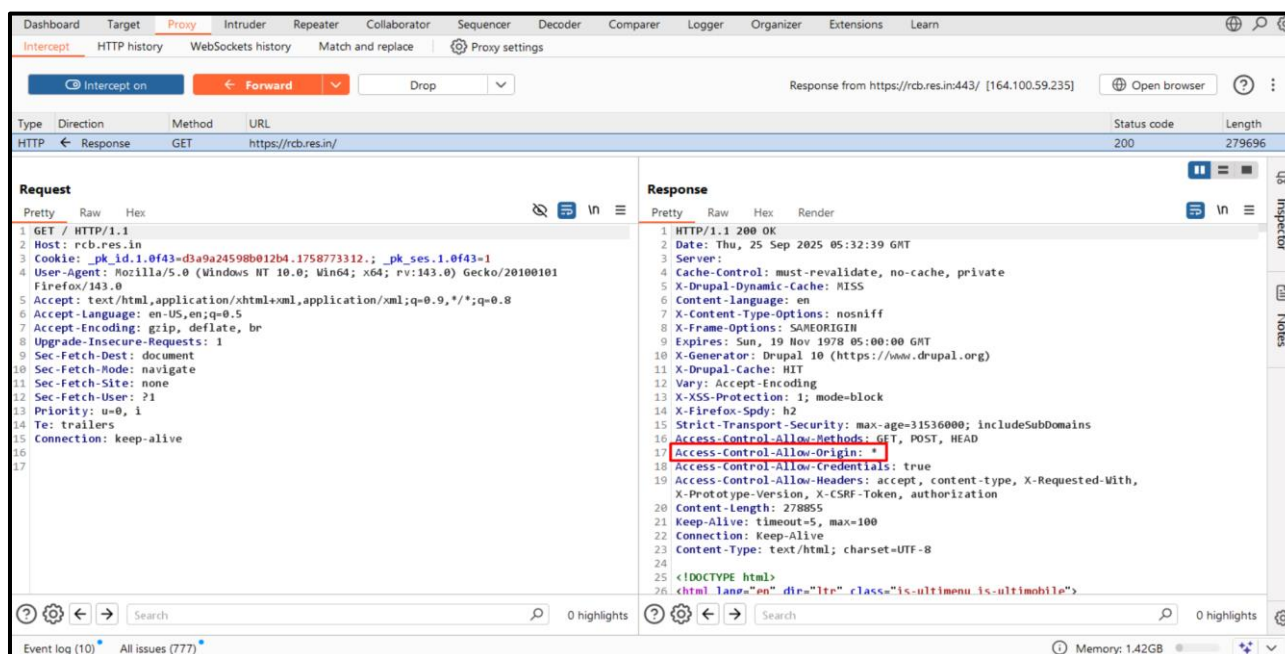
Description: Cross origin resource sharing policy in the application is set insecurely as it allows any domain URL to access some internal modules of the application.

Impact: The cross-origin resource sharing policy in the application is set insecurely as it allows any domain URL to access some internal modules of the application. Thus, it allows an attacker to misuse the insecure cross origin policy to launch attacks and can put any value into the origin request HTTP header to force web application to provide it the target resource content.

Severity: Low

How to Test:

Step I: From the below screenshot it can be observed that the application is throwing backend information in the error page An attacker navigates to the application, captures the response in HTTP interceptor, and it was observed that the value of response header 'Access-Control-Allow-Origin' is set to '*' as shown below:



Recommendation(s):

1. The application should not set the 'Access-Control-Allow-Origin' response header to '*'. Only the valid and appropriate domains should be allowed in the CORS policy.

11. Email Harvesting

Incident URL:

- <https://rcb.res.in/contact-us/>

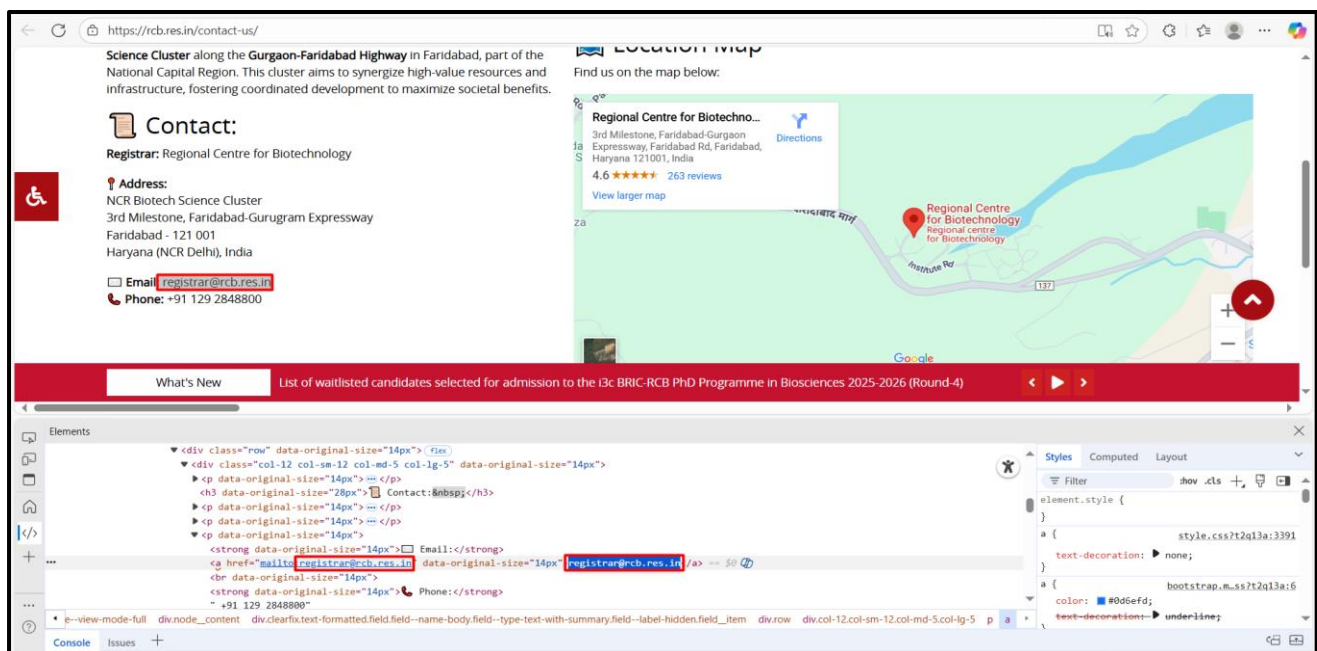
Description: The e-mail addresses present in the application are in text format thus leading to email harvesting attack.

Impact: The e-mail addresses present in the application are in text format which allows an attacker to automate and send multiple spam emails to the harvested email ids present in the application causing Email Flooding.

Severity: Low

How to Test:

Step I: An attacker navigates to the application, and it is observed that 'email Ids' are present in text format as shown below:



Recommendation(s):

1. The email addresses present in the application should be in the form of images so that the spammers cannot get email addresses present in the application using automated tools.
2. 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].
3. For example, xyz@gma.com should be written as xzyzy[at]gma[dot]com.
4. High privilege email addresses should not be posted on the website.
5. Email addresses should not be provided as a hyperlink, for example, mail to: xyz[at]gma[dot]com.

