



Web Application Security Audit Report: HSCL Website

Report Release Date	18.06.2026
Type of Audit	Application Security
Type of Audit Report	Initial Audit Report
Period	17.06.2026 – 18.06.2026

Report Prepared By:

AKS Information Technology Services Pvt. Ltd.

www.aksitservices.co.in

E-Mail: info@aksitservices.co.in

<Confidential>

Non-Disclosure Statement

This report is the sole property of Hooghly Cochin Shipyard Limited. All information obtained during the assessment is deemed privileged information and not for public dissemination. **AKS Information Technology Services Pvt. Ltd.** pledges its commitment that this information will remain strictly confidential. It will not be discussed or disclosed to any third party without the express written consent of Hooghly Cochin Shipyard Limited except required by the government regulator (Cert-In) or by the order of the Court.

Document Control

Document Preparation	
Document Title	Web Application Security Audit Report of HSCL Website
Document ID	--
Document Version	1.0
Prepared by	Ms. Manya Jain
Reviewed by	Ms. Surbhi Chhabra
Approved by	Mr. Darshan Deswal
Released by	Ms. Manya Jain
Release Date	18.06.2026

Document Change History		
Version	Date	Remarks / Reason of change
1.0	18.06.2026	Initial release

Document Distribution List			
Name	Organization	Designation	Email Id
Swarnab Dey	Hooghly Cochin Shipyard Ltd.	Project Assistant	it.support@hooghlycscl.com

Table of Contents

Table of Contents	4
Introduction.....	5
Engagement Scope.....	6
Details of the Auditing Team.....	7
Audit Activities and Timelines.....	8
Tools/ Software Used	9
Executive Summary.....	10
Detailed Findings	17
Observation.....	61
Appendix 'A'.....	64

Introduction

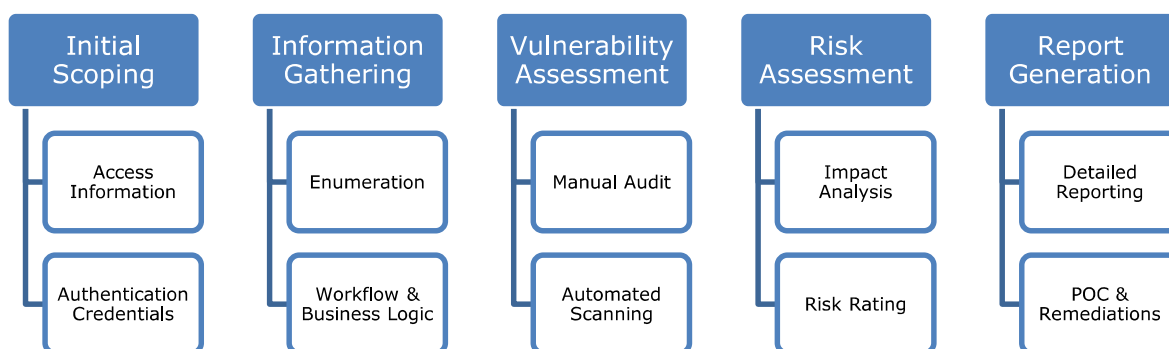
Objectives:

The key objective of this Web Application Security Audit was to identify whether any vulnerabilities exist in the Web Application and to exploit those that can be seen and compromised by malicious users. Additionally, the objective of this activity was to ensure the security of the network and web server from external threats through the Web Application.

Methodology & Standard:

Security Consultants at AKS IT Services Pvt. Ltd. Used the OWASP Web Application Security Testing Methodology for conducting the security audit of the in-scope Web Application.

The OWASP Web Application Methodology is based on the 'grey-box' approach. The testing model consists of the following phases:



Standard:

The Open Worldwide Application Security Project (OWASP) standard was used for conducting the initial level security audit of the HSCL Website Web Application. The assessment was aimed at identifying the vulnerabilities that are defined in the by following standards mentioned for conducting the final level security audit of the HSCL Website web application.

- CERT-In Guidelines: <https://www.cert-in.org.in/PDF/guidelinesgovtentities.pdf>
- NIC Guidelines: https://meity.gov.in/writereaddata/files/checklist_development.pdf
- OWASP standard: [OWASP Web Security Testing Guide | OWASP Foundation](#)
- Best Practices.

Engagement Scope

S. No	Asset Description	Criticality of Asset	Internal IP Address	URL	Public IP Address	Location	Hash Value of final audit report	Version
1.	NA	NA	NA	http://staging.hooghlycsl.in/en/	NA	Remote	NA	NA

The login mechanism was implemented on the following URL:

- http://staging.hooghlycsl.in/Admin/Login

The following User Roles were provided and were in the scope of audit.

- Admin

Details of the Auditing Team

S. No	Name	Designation	Email Id	Professional Qualifications/ Certifications	Whether the resource has been listed in the Snapshot information published on CERT-In's Web Application (Yes/No)
1	Manya Jain	Infosec Consultant	manya.jain@aksitservices.co.in	CEH	Yes
2	Surbhi Chhabra	Team Lead – Application Security	surbhi.chhabra@aksitservices.co.in	CEH	Yes
3	Darshan Deswal	Assistant Manager – Application Security	darshan.deswal@aksitservices.co.in	OSCP, OSCP, KLCP, CRTP, MCRTA, CAP, CCSP-AWS, CEH, ISO 27001:2022, ISMS LA, CMPEN	Yes

<Confidential>

Audit Activities and Timelines

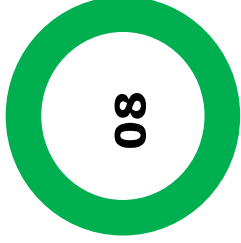
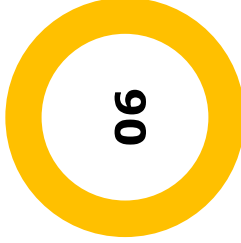
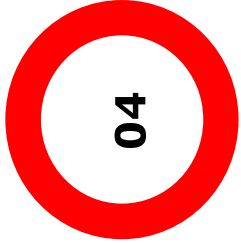
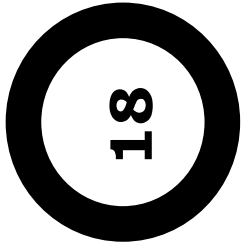
Audit Activity	Timelines
Phase I	
Auditor Assigned	20.05.2026
Audit Initiated	17.06.2026
Audit Report Preparation	18.06.2026
Initial Audit Report Published	18.06.2026

Tools/ Software Used

S. No	Name of Tool/Software used	Version of the tool/Software used	Open Source/Licensed
1	Burp Suite Professional	2025.3	Commercial
2	Acunetix Vulnerability Scanner	24.5	Commercial
3	DirSearch	0.4.3	Open Source
4	SQLMAP	1.8	Open Source

[Appendix 'A'](#): Description of the tools

Executive Summary



Total

High

Medium

Low

S. No	Affected Asset i.e. IP/URL/Application etc.	Observation/ Vulnerability title	CVE/ CWE	Severity	Recommendation	Reference	New or Repeat or closed observation
1.	http://staging.hooghlycsl.in/Admin/Login	Cleartext Submission of Password	CWE - 644	High	Use latest stable version of TLS to protect all sensitive communications passing between the client and the server. Implement HTTP Strict Transport Layer Security. If HTTP cookies are used for transmitting session tokens, then the secure flag should be set to prevent transmission over clear-text HTTP. If implementing HTTPS is not possible, implement salted SHA-256 or salted SHA-512 hashing algorithms on password fields, while using plain SHA-256 or SHA-512 hashing on new password fields.	Case I	New

<Confidential>

2.	http://staging.hooghlycsl.in/User/User	Cross Site Scripting (XSS)	CWE - 79	High	Input should be validated as strictly as possible on arrival, given the kind of content that it is expected to contain. For example, personal names should consist of alphabetical and a small range of typographical characters, and be relatively short; a year of birth should consist of exactly four numerals; email addresses should match a well-defined regular expression. Input which fails the validation should be rejected, not sanitized. User input should be HTML-encoded at any point where it is copied into application responses. All HTML metacharacters, including < > ' " ' and =, should be replaced with the corresponding HTML entities (&lt; &gt; etc). In cases where the application's functionality allows users to author content using a restricted subset of HTML tags and attributes (for example, blog comments which allow limited formatting and linking), it is necessary to parse the supplied HTML to validate that it does not use any dangerous syntax.	Case II	New
3.	http://staging.hooghlycsl.in/Admin/Login	Session Fixation	CWE - 384	High	Invalidate the existing session and assign a new session identifier on successful login attempts. For platforms such as ASP that do not generate new values for sessionid cookies, utilize a secondary cookie. In this approach, set a secondary cookie on the user's browser to a random	Case III	New

<Confidential>

					value and set a session variable to the same value. If the session variable and the cookie value ever don't match, invalidate the session, and force the user to log on again.		
4.	http://staging.hooghlycsl.in/Admin/ChangePassword	Improper Session Termination	CWE - 613	High	Delete the session variables and destroy the session when the user logs out of the application.	Case IV	New
5.	http://staging.hooghlycsl.in/	IIS Tilde Directory Enumeration	CWE - 548	Medium	Discarding web requests including the tilde "~" character.	Case V	New
6.	http://staging.hooghlycsl.in/en/add-feedback http://staging.hooghlycsl.in/User/User http://staging.hooghlycsl.in/Admin/Dashboard	Improper Error Handling	CWE - 755	Medium	Ensure that a customized error message is shown for any error that has occurred, which gives out very limited information. Disable or limit detailed error handling. In particular, do not display debug information to end users, stack traces, or path information. Application should make secure to prevent revealing of any kind of error and Hardening process should be carried out periodically.	Case VI	New
7.	http://staging.hooghlycsl.in/Admin/ChangePassword	Weak Password Policy	CWE - 521	Medium	Enforce strong password requirement Minimum Length: 8 characters At-least 1 uppercase, 1 lowercase, 1 numeric and 1 special characters. Restrictions against password reuse	Case VII	New

<Confidential>

					Restrictions against using common passwords Restrictions against using contextual string in the password (e.g., user id, app name).		
8.	http://staging.hooghlycsl.in/User/User List	Improper Cache Control	CWE-525	Medium	Implement the following response headers on the web pages containing sensitive information: Cache-Control: no-cache, no-store Pragma: no-cache.	Case VIII	New
9.	http://staging.hooghlycsl.in/en/	Insecure HTTP Methods Enabled	CWE-749	Medium	Unnecessary HTTP methods like TRACE, DEBUG, PUT and DELETE should be disabled.	Case IX	New
10.	http://staging.hooghlycsl.in/en/	Lack of Security Header	CWE-644	Medium	Implement security headers such as X-XSS-Protection, Content-Security-Policy, Referrer Policy, X-Content-Type-Options, Permission Policy and Strict-transport-layer-protection. To prevent untrusted JavaScript execution, replace allowedlisted resources in script-based directives with a secure, random nonce of at least 8 characters 'nonce-RANDOM'. Mitigate style-based data exfiltration by avoiding 'unsafe-inline', data: URLs, and global wildcards in style directives. Use a secure, random nonce of at least 8 characters 'nonce-RANDOM' in the relevant directive.	Case X	New
11.	http://staging.hooghlycsl.in/User/User	Improper Input Validation	CWE-20	Low	All only the required characters for all the parameters as per requirement by whitelisting of those fields with not allowing all special and meta	Case XI	New

<Confidential>

					characters. Data type validators available natively in web application frameworks (such as Django Validators, Apache Commons Validators etc.). Minimum and maximum value range check for numerical parameters and dates, minimum and maximum length check for strings. Array of allowed values for small sets of string parameters (e.g., days of week).		
12.	http://staging.hooghlycsl.in/Admin/Login http://staging.hooghlycsl.in/en/add-feedback	Weak Captcha	CWE-804	Low	Use a minimum 6-character alphanumeric CAPTCHA (mix of upper/lowercase letters and numbers). Implement case-sensitive and randomized CAPTCHA strings. Add distortion or background noise to prevent OCR attacks. Validate CAPTCHA server-side to avoid bypass. Expire CAPTCHA tokens after a short time or single use. Limit failed attempts and enforce rate limiting	Case XII	New
13.	http://staging.hooghlycsl.in/Admin/Dashboard	Improper Session Timeout	CWE-613	Low	Application should automatically log out the user and destroy the session after 20-30 mins of inactivity.	Case XIII	New
14.	http://staging.hooghlycsl.in/Admin/ChangePassword	Vulnerable and Outdated Components		Low	Upgrade the components used to the latest version. If upgradation is not possible,	Case XIV	New

<Confidential>

					download the security patches and hide the version details.		
15.	http://staging.hooghlycsl.in/en/key-personnel	Email Harvesting	CWE-200	Low	The application should properly customize the email addresses while posting on the website as: 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.	Case XV	New
16.	http://staging.hooghlycsl.in/en/	Server Banner Disclosure	CWE-200	Low	Hide Server version details in the response	Case XVI	New
17.	http://staging.hooghlycsl.in/ http://staging.hooghlycsl.in/Admin/Login	SSL/TLS Not Implemented	CWE-311	Low	Obtain and install a valid SSL/TLS certificate from a trusted Certificate Authority (CA). Configure the server to enforce HTTPS for all communication. Redirect all HTTP traffic to HTTPS to prevent unencrypted access. Implement HTTP Strict Transport Security (HSTS) headers to protect against downgrade attacks. Regularly monitor, update, and renew SSL/TLS certificates before expiration. Disable insecure protocols (e.g., SSLv2, SSLv3, TLS 1.0/1.1) and use	Case XVII	New

					strong cipher suites		
18.	http://staging.hooghlycsli.in/Admin/Login	Security Logging and Monitoring		Low	Information to be logged includes the following: IP address of the originating Source, Date, Time, Username (No Password), session details, Referrer, Process id, URL, User Agent, Countries if any in addition to other details to be logged in the website. Logging of Authentication Process which includes number of successful and failed login attempts. To create audit logs, use auto numbering so that every logged entry has an un-editable log number. Then if one audit entry is deleted a gap in the numbering sequence will appear. Report of the website logs to be generated weekly by the administrator to keep track of the website activities	Case XVIII	New

Detailed Findings

Case I

i. Affected Asset i.e. IP/URL/Application etc.

<http://staging.hooghlycs1.in/Admin/Login>

ii. Observation/ Vulnerability title

Cleartext Submission of Password

iii. Detailed observation / Vulnerable point

Applications can unintentionally leak information about their configuration, internal workings, or violate privacy through a variety of application problems. Attackers use this weakness to steal sensitive data or conduct more serious attacks.

iv. CVE/CWE

CWE-644

v. Severity

High

vi. Recommendation

Use latest stable version of TLS to protect all sensitive communications passing between the client and the server.

Implement HTTP Strict Transport Layer Security.

If HTTP cookies are used for transmitting session tokens, then the secure flag should be set to prevent transmission over clear-text HTTP.

If implementing HTTPS is not possible, implement salted SHA-256 or salted SHA-512 hashing algorithms on password fields, while using plain SHA-256 or SHA-512 hashing on new password fields.

vii. Reference

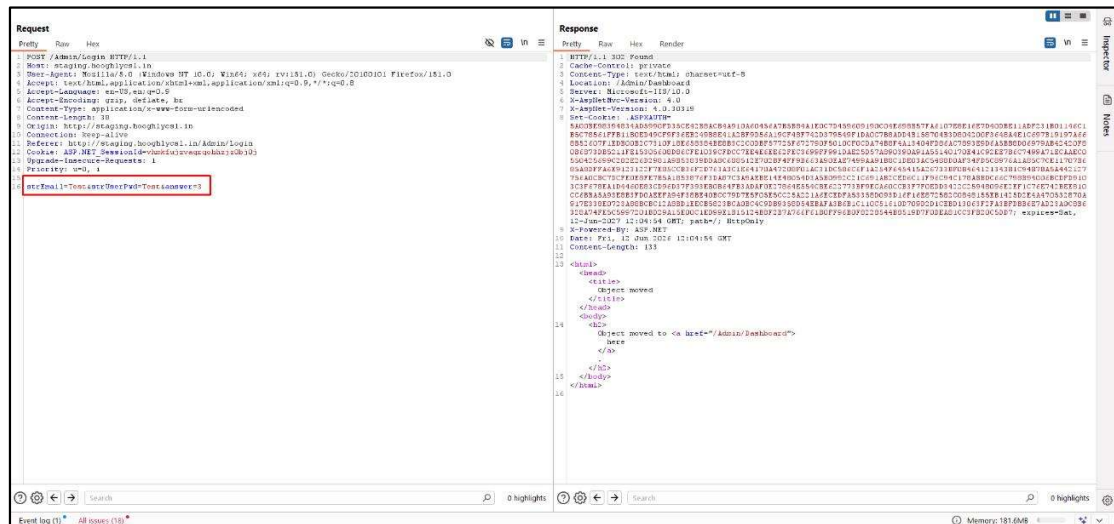
<https://cwe.mitre.org/data/definitions/644.html>

viii. New or Repeat observation

New

ix. References to evidence / Proof of Concept

Step I: Observe that the password is transmitted in cleartext within the request as shown in snapshot below:



Note: Kindly, patch this vulnerability throughout the application.

Case II

i. **Affected Asset i.e. IP/URL/Application etc.**

<http://staging.hooghlycs1.in/User/User>

ii. **Observation/ Vulnerability title**

Cross Site Scripting (XSS)

iii. **Detailed observation / Vulnerable point**

The application does not neutralize or incorrectly neutralizes user-controllable input before it is placed in output that is used as a web page that is served to other users.

iv. **CVE/CWE**

CWE - 79

v. **Severity**

High

vi. **Recommendation**

Input should be validated as strictly as possible on arrival, given the kind of content that it is expected to contain. For example, personal names should consist of alphabetical and a small range of typographical characters, and be relatively short; a year of birth should consist of exactly four numerals; email addresses should match a well-defined regular expression. Input which fails the validation should be rejected, not sanitized.

User input should be HTML-encoded at any point where it is copied into application responses. All HTML metacharacters, including < > " ' and =, should be replaced with the corresponding HTML entities (< > etc).

In cases where the application's functionality allows users to author content using a restricted subset of HTML tags and attributes (for example, blog comments which allow limited formatting and linking), it is necessary to parse the supplied HTML to validate that it does not use any dangerous syntax.

vii. **Reference**

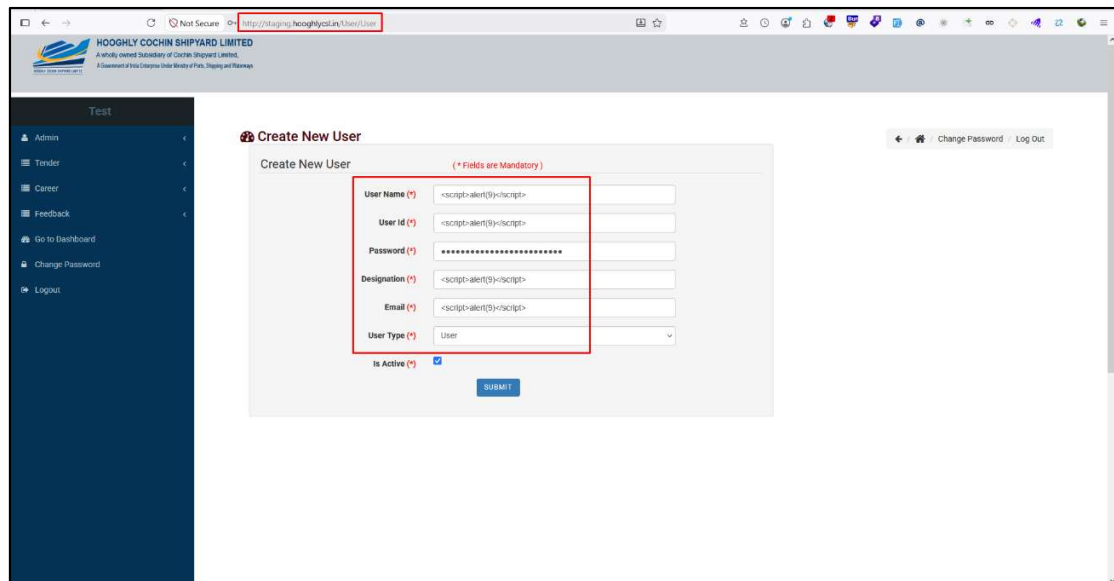
<https://cwe.mitre.org/data/definitions/79.html>

viii. New or Repeat observation

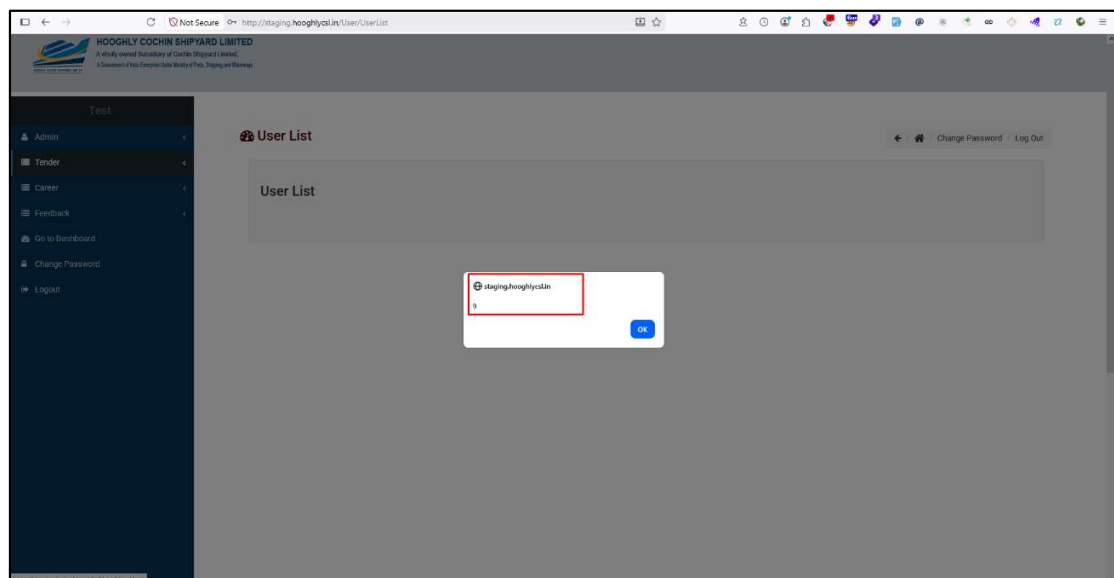
New

ix. References to evidence / Proof of Concept

Step I: Visit the URL, enter the XSS payload in required fields as shown in snapshot.



Step II: Observe that new user created successfully and now visit User List functionality, XSS payload executed simultaneously as shown in snapshots below.



<Confidential>

HOOGHLY COCHIN SHIPYARD LIMITED
A Wholly owned Subsidiary of Cochin Shipyard Limited,
A Government of India Enterprise under Ministry of Ports, Shipping and Maritime

Test

Admin
Tender
Career
Feedback
Go to Dashboard
Change Password
Logout

User List

Change Password / Log Out

Show 10 entries

Search:

User Name	User ID	User Type	Status	Edit
Admin	admin@gmail.com	Admin	Active	EDIT
		User	Active	EDIT
Career	career@gmail.com	User	Active	EDIT
DFS	admindfs@gmail.com	Admin	Active	EDIT
Tender	tender@gmail.com	User	Active	EDIT
Test	Test	Admin	Active	EDIT
Test01	Test01	User	Active	EDIT

Showing 1 to 7 of 7 entries

Previous 1 Next

Note: Kindly, patch this vulnerability throughout the application.

Case III

i. Affected Asset i.e. IP/URL/Application etc.

<http://staging.hooghlycs1.in/Admin/Login>

ii. Observation/ Vulnerability title

Session Fixation

iii. Detailed observation / Vulnerable point

The application does not assign a new session identifier after a successful authentication.

iv. CVE/CWE

CWE-384

v. Severity

High

vi. Recommendation

Invalidate the existing session and assign a new session identifier on successful login attempts.

For platforms such as ASP that do not generate new values for sessionid cookies, utilize a secondary cookie. In this approach, set a secondary cookie on the user's browser to a random value and set a session variable to the same value. If the session variable and the cookie value ever don't match, invalidate the session, and force the user to log on again.

vii. Reference

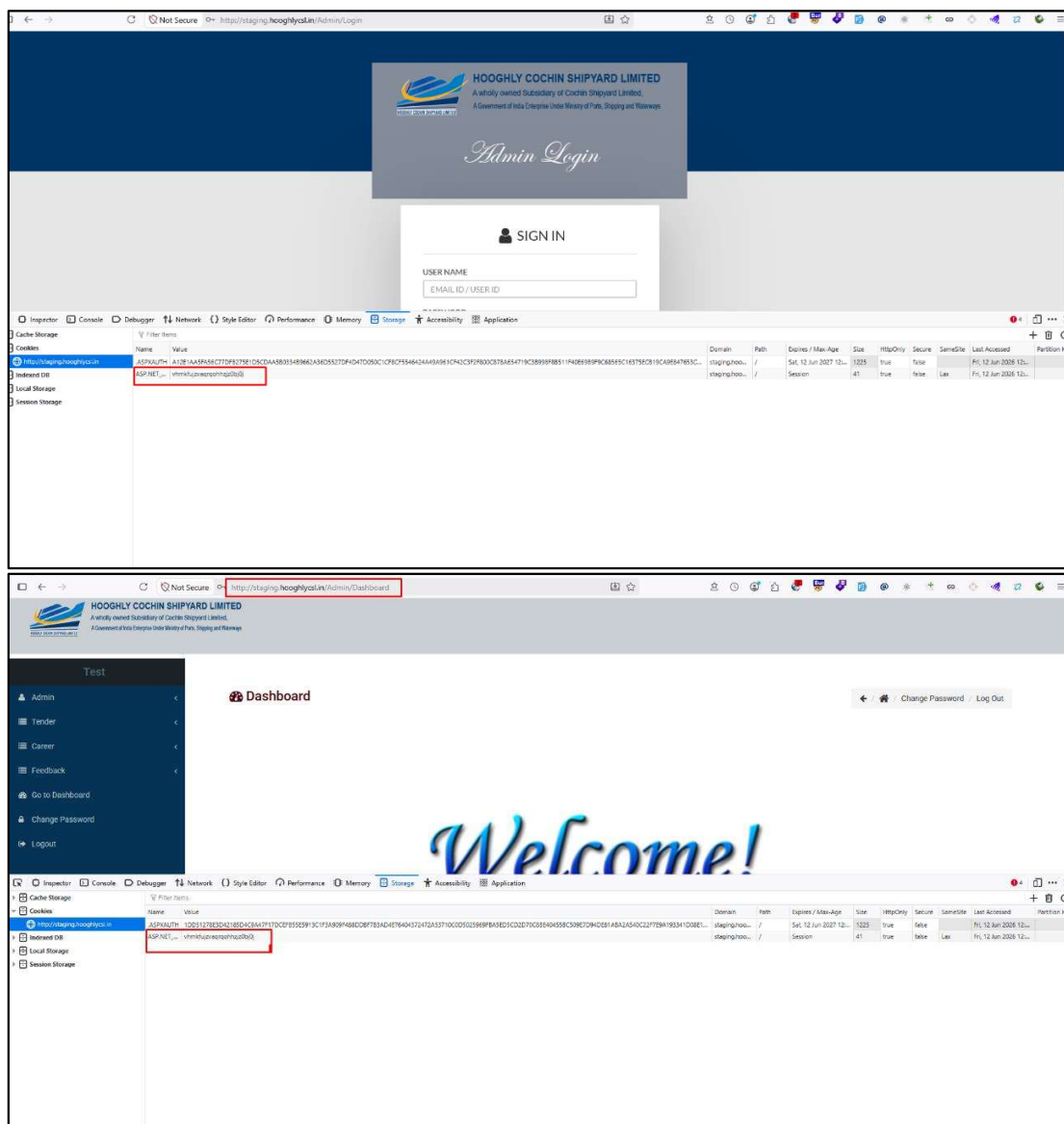
<https://cwe.mitre.org/data/definitions/384.html>

viii. New or Repeat observation

New

ix. References to evidence / Proof of Concept

Step I: Observe that session value is fixed pre-login and post-login as shown in snapshots below:



Note: Kindly, patch this vulnerability throughout the application.

Case IV

i. Affected Asset i.e. IP/URL/Application etc.

<http://staging.hooghlycs1.in/Admin/ChangePassword>

ii. Observation/ Vulnerability title

Improper Session Termination

iii. Detailed observation / Vulnerable point

The application does not destroy or invalidates the session after the user logs out.

iv. CVE/CWE

CWE-613

v. Severity

High

vi. Recommendation

Delete the session variables and destroy the session when the user logs out of the application.

vii. Reference

<https://cwe.mitre.org/data/definitions/613.html>

viii. New or Repeat observation

New

ix. References to evidence / Proof of Concept

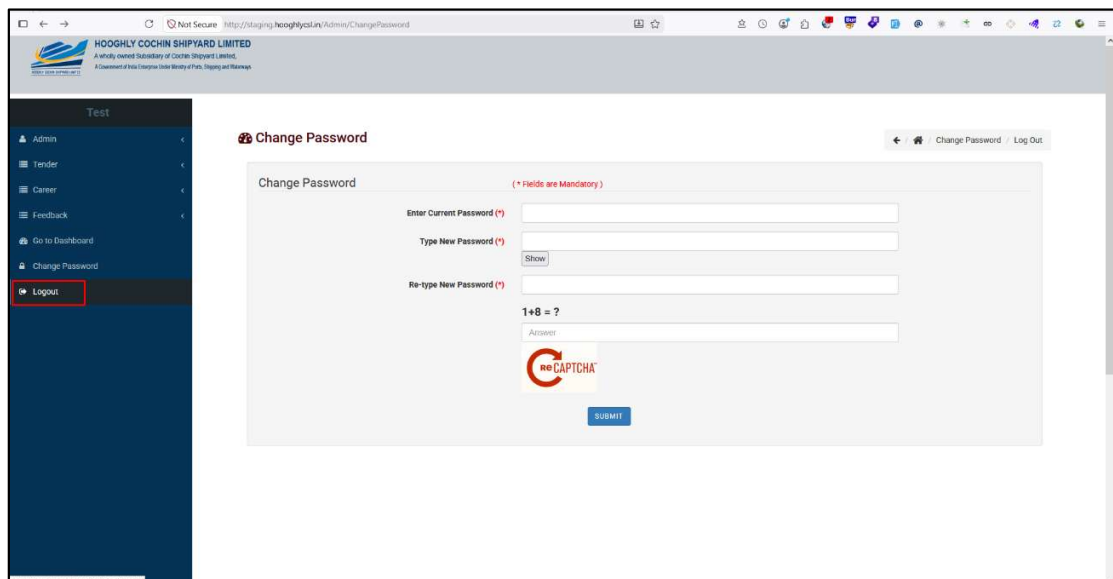
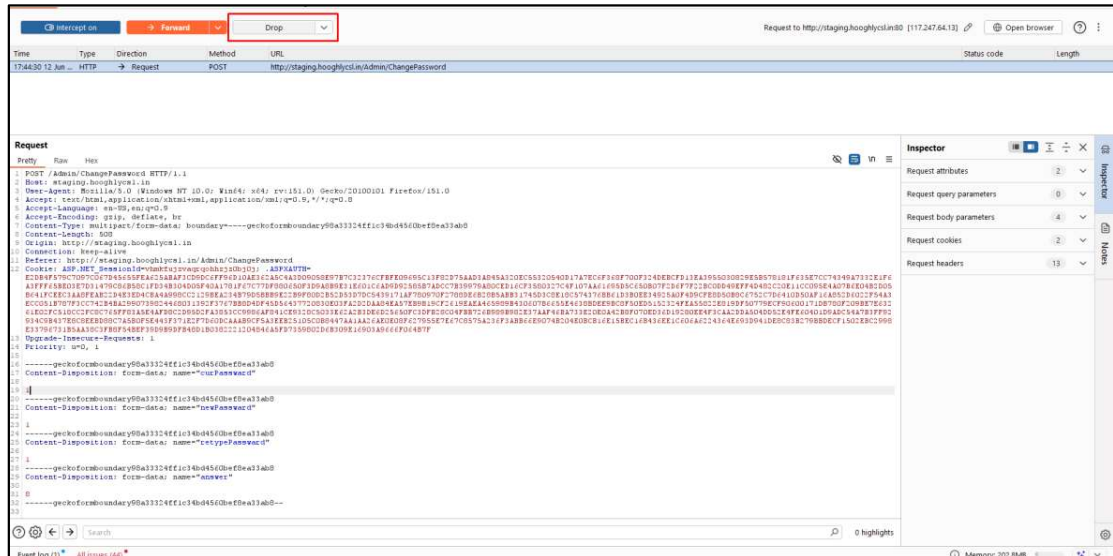
Step I: Visit Change Password functionality, and intercept the request in burp and send it to repeater as shown in snapshots below:

The top screenshot shows a web browser displaying the 'Change Password' form for 'HOOGHLY COCHIN SHIPYARD LIMITED'. The form includes fields for 'Enter Current Password (*)', 'Type New Password (*)', and 'Re-type New Password (*)', along with a CAPTCHA challenge. A red box highlights these input fields.

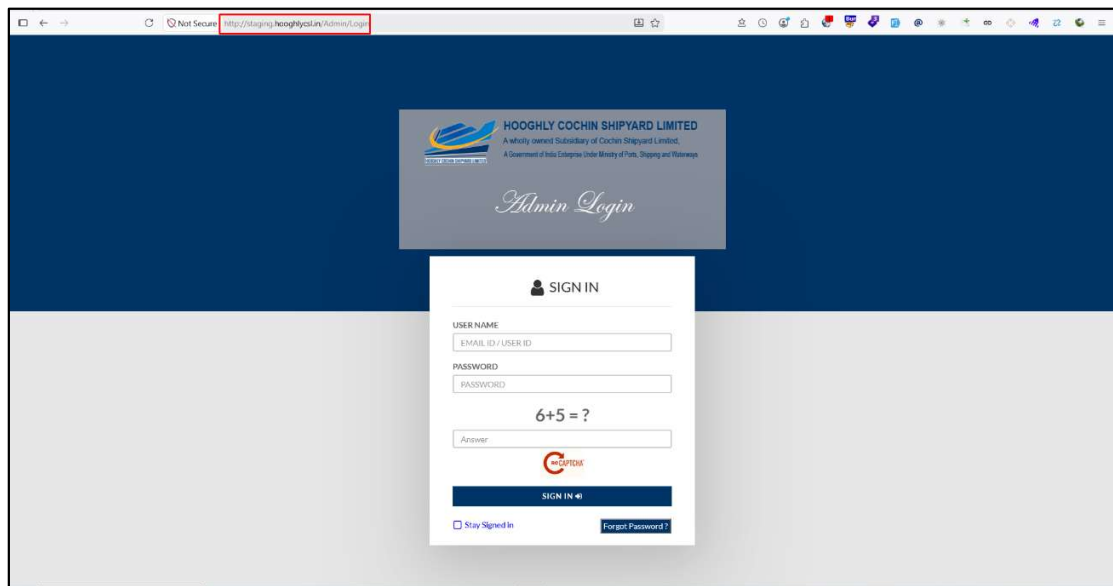
The bottom screenshot shows the Burp Suite interface. The 'Request' tab is active, displaying the intercepted HTTP POST request to 'http://staging.hooghlyco.in/Admin/ChangePassword'. The right-click context menu is open, and the 'Send to Repeater' option is highlighted.

<Confidential>

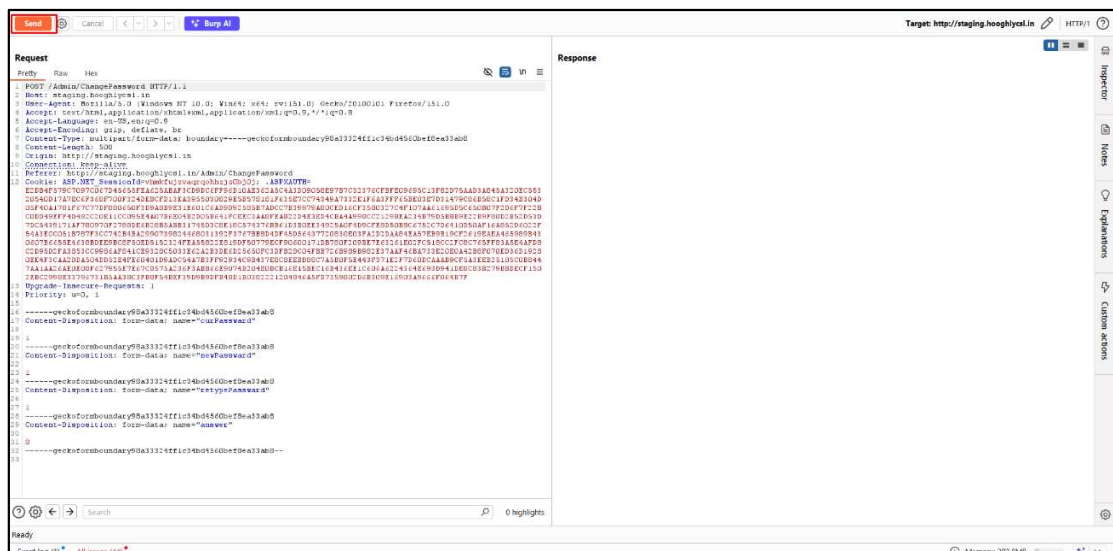
Step II: Drop the request and logout from the Admin login as shown in snapshots below:



<Confidential>



Step III: Send the previously intercepted request and observe that password changed successfully as shown in snapshots below:



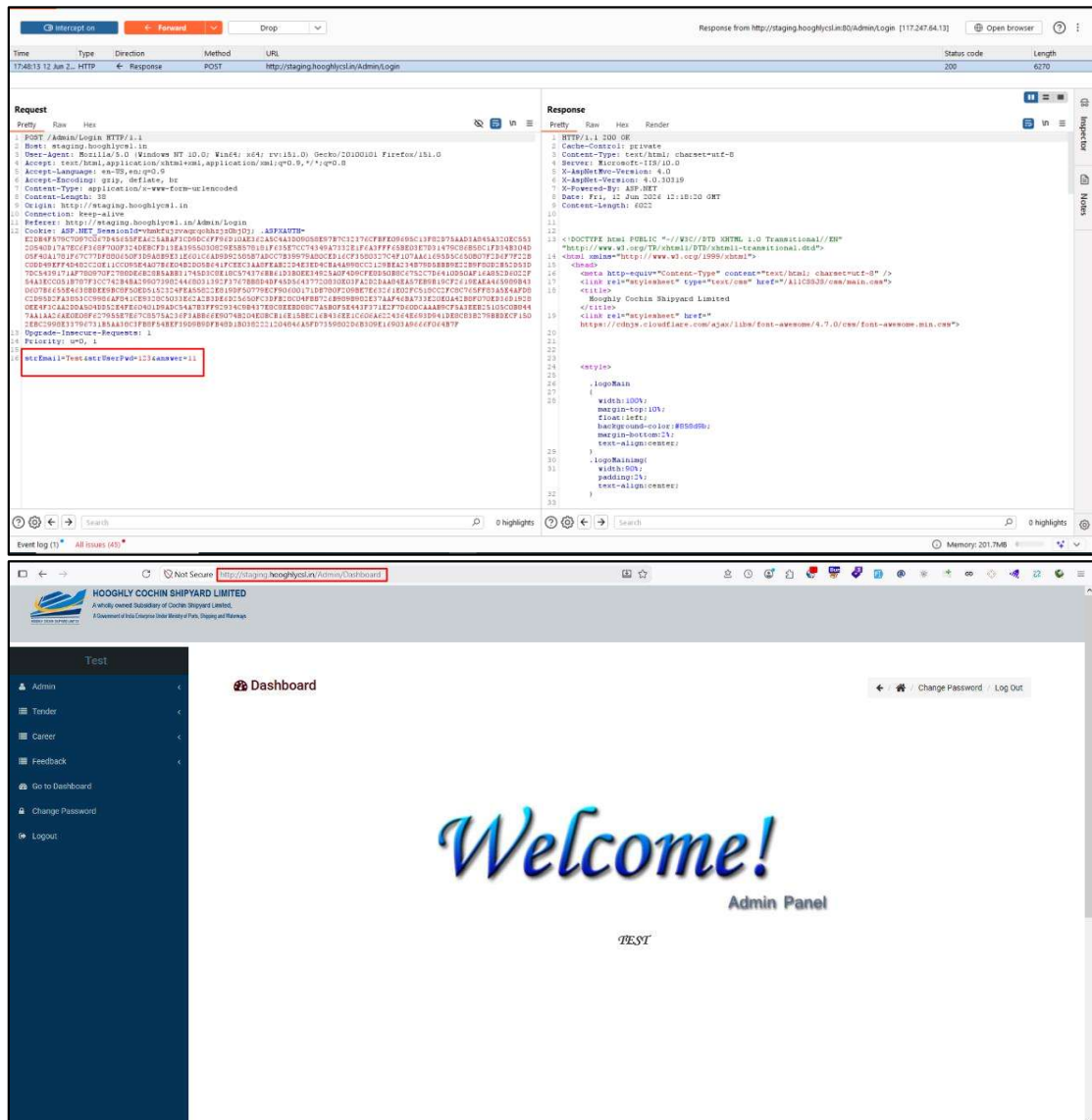
<Confidential>

The screenshot shows the Burp Suite interface with a captured HTTP request and response. The 'Request' tab is active, displaying a POST request to `http://staging.hooglycal.in/admin/ChangePassword`. The request body is a multipart/form-data payload. A red box highlights the `Content-Disposition: form-data; name='password'` line. The 'Response' tab is also visible, showing a 302 Found status and a `Location: /admin/login` header. The 'Follow redirector' button is highlighted in the top toolbar.

[illegible]

<Confidential>

Step IV: Now try to login with the updated credentials and observe that we are able to login with updated credentials as shown in snapshots below:



Note: Kindly, patch this vulnerability throughout the application.

Case V

i. Affected Asset i.e. IP/URL/Application etc.

<http://staging.hooghlycs1.in/>

ii. Observation/ Vulnerability title

IIS Tilde Directory Enumeration

iii. Detailed observation / Vulnerable point

This vulnerability is caused by the tilde character (~) with the old DOS 8.3 name convention (SFN) in a HTTP request. It allows a remote attacker to disclose file and folder names (that are not supposed to be accessible) under the web root. Attackers could find important files that are normally not accessible from the outside and gain intelligence about the application's infrastructure.

iv. CVE/CWE

CWE - 548

v. Severity

Medium

vi. Recommendation

Discarding web requests including the tilde "~" character.

vii. Reference

<https://cwe.mitre.org/data/definitions/548.html>

viii. New or Repeat observation

New

ix. References to evidence / Proof of Concept

Step I: Observe that IIS Tilde Directory Enumeration is there as shown in the snapshots below:

The top screenshot shows the Burp Suite 'Scan' results for the target URL `http://staging.hooghlycal.in/`. The scan results indicate a vulnerability in IIS Tilde Directory Enumeration. The results are as follows:

```

[*] Started a scan for URL "http://staging.hooghlycal.in/"
[*] Trying method "OPTIONS" with magic final part "~/.item"
[*] Trying method "POST" with magic final part "~/.item"
[*] Trying method "DEBUG" with magic final part "~/.item"
[*] Host "http://staging.hooghlycal.in/" is vulnerable!
[*] Used HTTP method "DEBUG"
[*] Built magic part "~/.item"
[*] Starting filename and directory enumeration on "http://staging.hooghlycal.in/"
[*] File: WEB-1.CON
[*] File: GLOBAL-1.ASA
[*] File: PHOQAG-1.CON
[*] Brute force completed in 87 seconds
[*] Total time elapsed: 95 seconds
[*] Requests sent: 376
[*] No directories found
[*] Identified files: 3
  L WEB-1.CON
  L Actual file name = WEB
  L GLOBAL-1.ASA
  L PHOQAG-1.CON
  
```

The bottom screenshot shows the 'Response' tab for a 404.0 error. The response is an 'HTTP Error 404.0 - Not Found'. The message states: 'The resource you are looking for has been removed, had its name changed, or is temporarily unavailable.' The 'Most likely causes' are listed as:

- The directory or file specified does not exist on the Web server.
- The URL contains a typographical error.
- A custom filter or module, such as URLScan, restricts access to the file.

The 'Things you can try' section suggests:

- Create the content on the Web server.
- Review the browser URL.
- Create a tracing rule to track failed requests for this HTTP status code and see which module is calling SetStatus. For more information about creating a tracing rule for failed requests, click [here](#).

The 'Detailed Error Information' section provides the following details:

Module	Notification	Handler	Requested URL	Physical Path	Logon Method	Logon User
IIS Web Core	MapRequestHandler	Integrated-4.0	http://staging.hooghlycal.in/80/~/.item?asperrorpath=/	C:\inetpub\wwwroot\hooghly_36032023\~/.item	Anonymous	Anonymous

The 'Error Code' is 0x00000005. The 'More Information' section states: 'This error means that the file or directory does not exist on the server. Create the file or directory and try the request again. [View more information.](#)'

Note: Kindly, patch this vulnerability throughout the application.

Case VI

i. **Affected Asset i.e. IP/URL/Application etc.**

<http://staging.hooghlycs1.in/en/add-feedback>

<http://staging.hooghlycs1.in/User/User>

<http://staging.hooghlycs1.in/Admin/Dashboard>

ii. **Observation/ Vulnerability title**

Improper Error Handling

iii. **Detailed observation / Vulnerable point**

Applications can unintentionally leak information about their configuration, internal workings, or violate privacy through a variety of application problems. Attackers use this weakness to steal sensitive data or conduct more serious attacks.

iv. **CVE/CWE**

CWE - 755

v. **Severity**

Medium

vi. **Recommendation**

Ensure that a customized error message is shown for any error that has occurred, which gives out very limited information.

Disable or limit detailed error handling. In particular, do not display debug information to end users, stack traces, or path information.

Application should make secure to prevent revealing of any kind of error and Hardening process should be carried out periodically.

vii. **Reference**

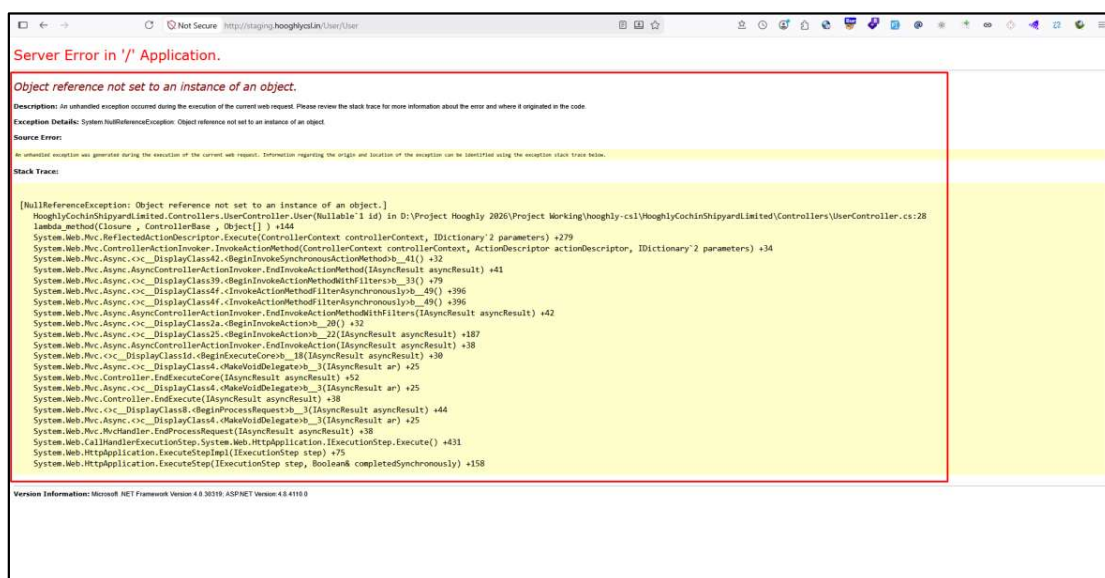
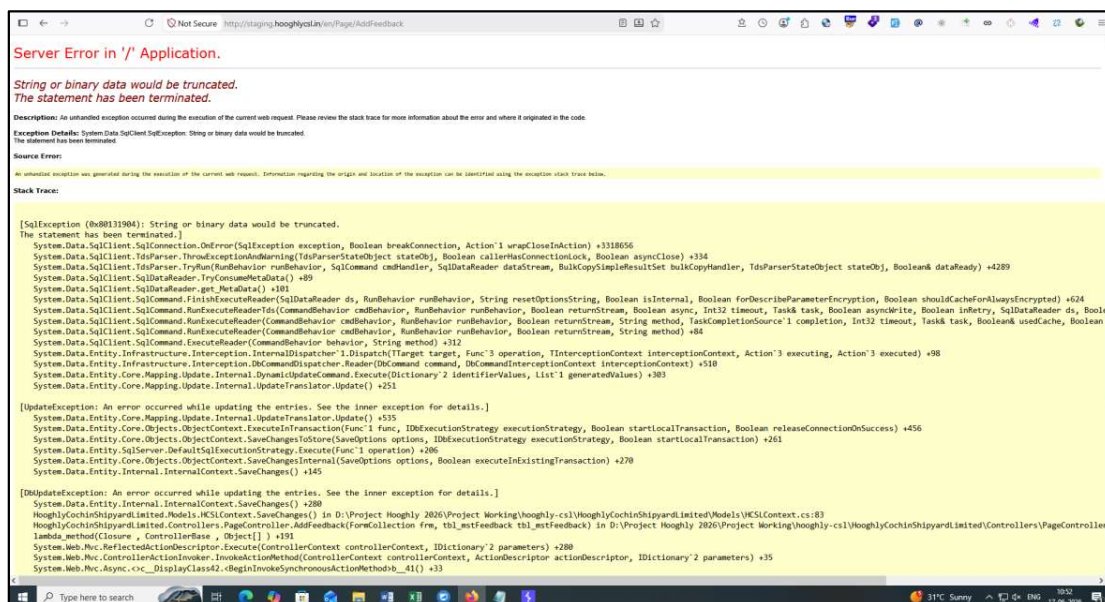
<https://cwe.mitre.org/data/definitions/755.html>

viii. **New or Repeat observation**

New

ix. References to evidence / Proof of Concept

Step I: Observe that errors are not handled properly as shown in snapshot below:



Case VII

i. Affected Asset i.e. IP/URL/Application etc.

<http://staging.hooghlycs1.in/Admin/ChangePassword>

ii. Observation/ Vulnerability title

Weak Password Policy

iii. Detailed observation / Vulnerable point

The application does not require that users should have strong passwords, which makes it easier for attackers to compromise user accounts.

iv. CVE/CWE

CWE-521

v. Severity

Medium

vi. Recommendation

Enforce strong password requirement

Minimum Length: 8 characters

At-least 1 uppercase, 1 lowercase, 1 numeric and 1 special characters.

Restrictions against password reuse

Restrictions against using common passwords

Restrictions against using contextual string in the password (e.g., user id, app name)

vii. Reference

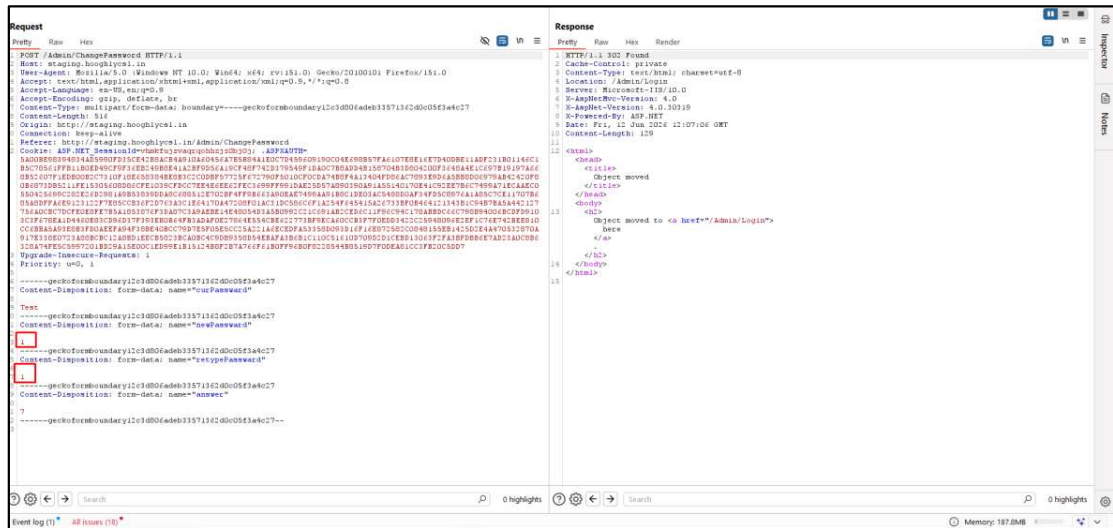
<https://cwe.mitre.org/data/definitions/521.html>

viii. New or Repeat observation

New

ix. References to evidence / Proof of Concept

Step I: Observe that weak password policy is implemented as shown in snapshots below:



Note: Kindly, patch this vulnerability throughout the application.

Case VIII

i. Affected Asset i.e. IP/URL/Application etc.

<http://staging.hooghlycs1.in/User/UserList>

ii. Observation/ Vulnerability title

Improper Cache Control

iii. Detailed observation / Vulnerable point

The web application does not use an appropriate caching policy and allows the web browser to cache post-authentication web pages.

iv. CVE/CWE

CWE-525

v. Severity

Medium

vi. Recommendation

Implement the following response headers on the web pages containing sensitive information:

Cache-Control: no-cache, no-store

Pragma: no-cache.

vii. Reference

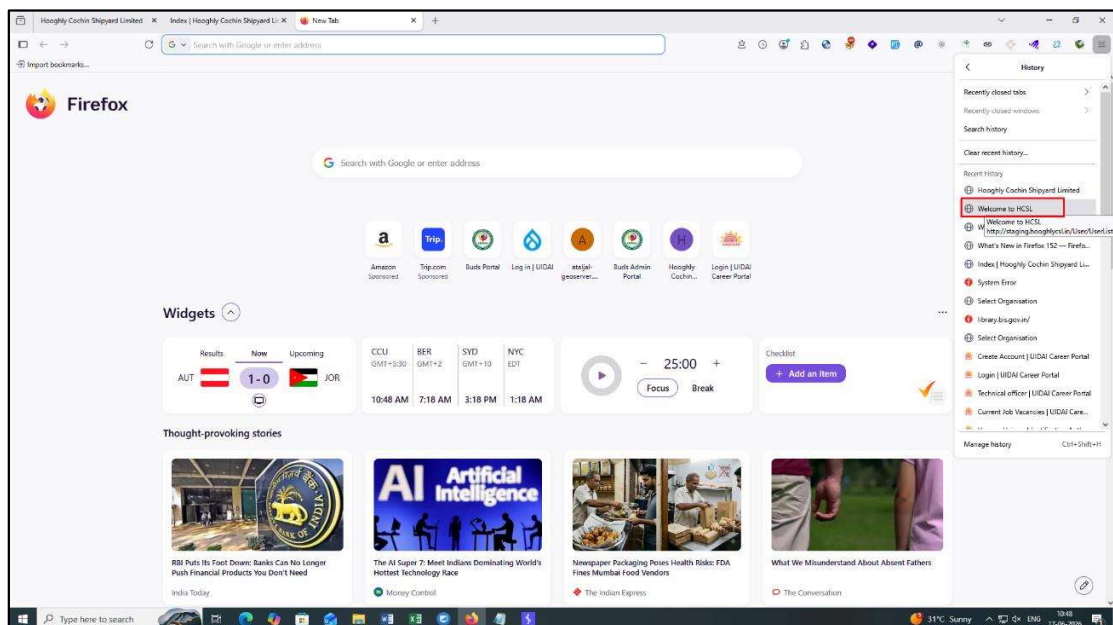
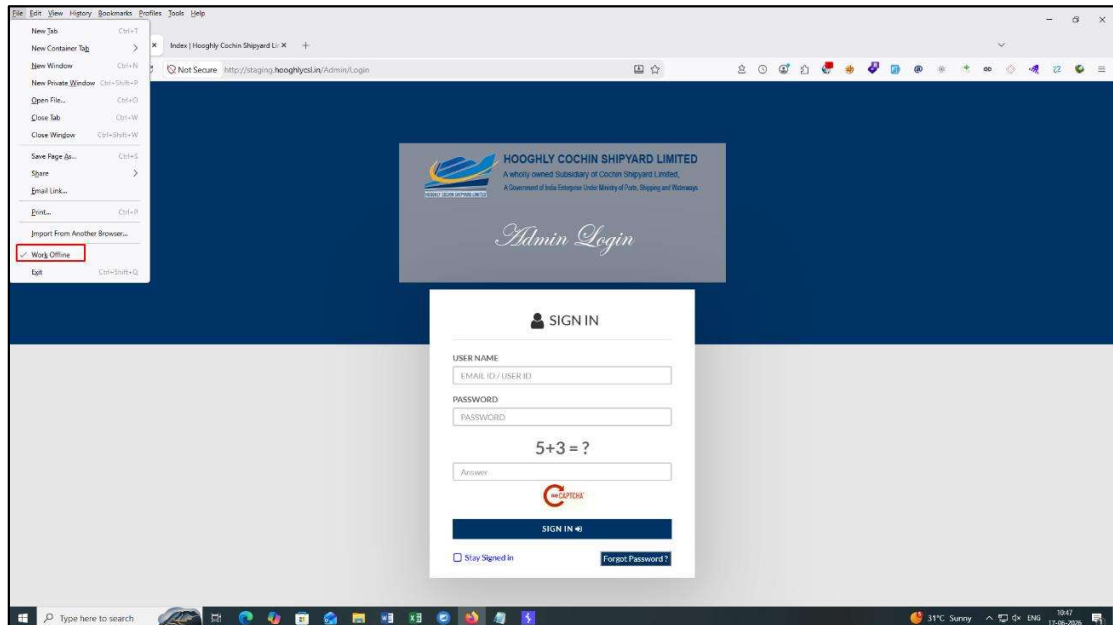
<https://cwe.mitre.org/data/definitions/525.html>

viii. New or Repeat observation

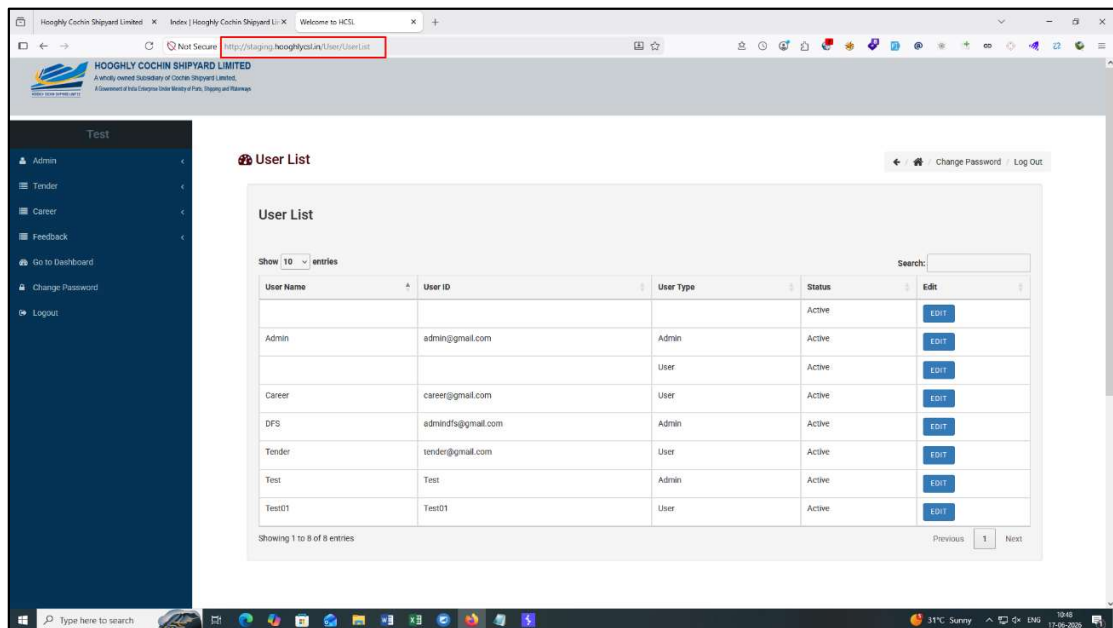
New

ix. References to evidence / Proof of Concept

Step I: After enabling Work Offline and opening a post-login page from the browsing history, it was observed that the page loaded successfully as shown in the snapshots below:



<Confidential>



Note: Kindly, patch this vulnerability throughout the application.

Case IX

i. Affected Asset i.e. IP/URL/Application etc.

<http://staging.hooghlycs1.in/en/>

ii. Observation/ Vulnerability title

Insecure HTTP Methods Enabled

iii. Detailed observation / Vulnerable point

Applications can unintentionally leak information about their configuration, internal workings, or violate privacy through a variety of application problems. Attackers use this weakness to steal sensitive data or conduct more serious attacks.

iv. CVE/CWE

CWE-749

v. Severity

Medium

vi. Recommendation

Unnecessary HTTP methods like TRACE, DEBUG, PUT and DELETE should be disabled.

vii. Reference

<https://cwe.mitre.org/data/definitions/749.html>

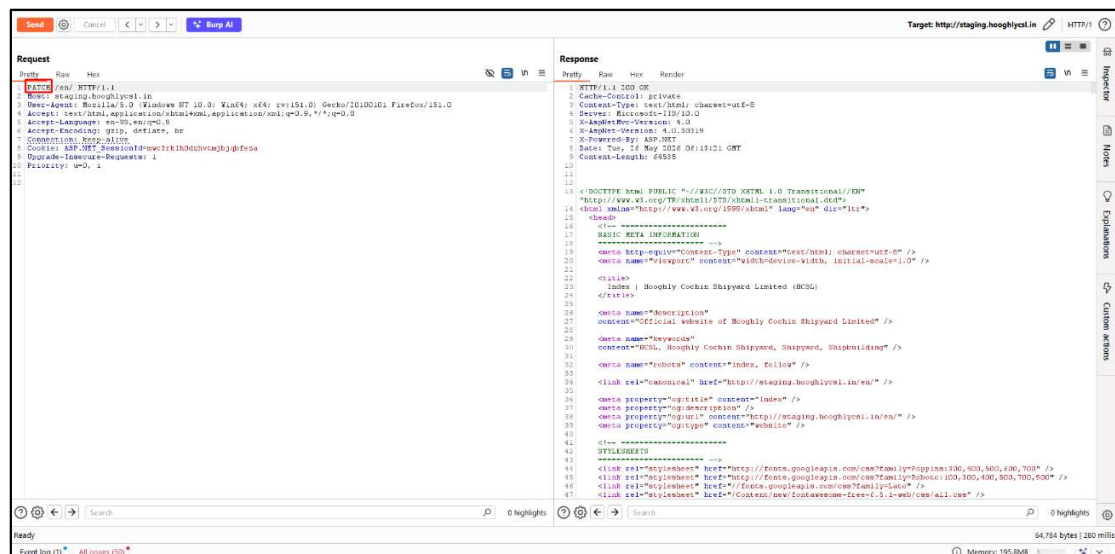
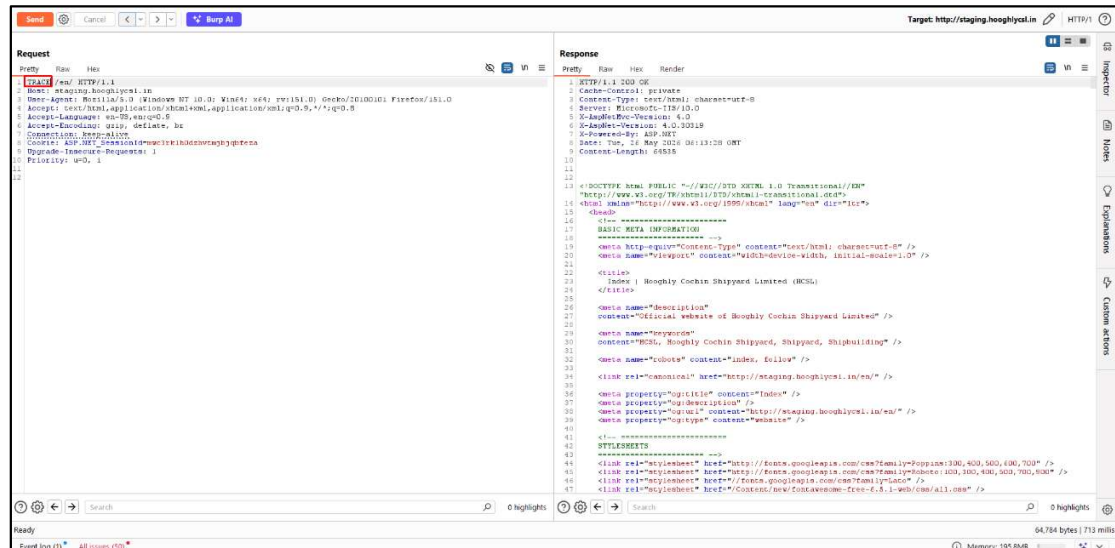
viii. New or Repeat observation

New

<Confidential>

ix. References to evidence / Proof of Concept

Step I: Observe that insecure HTTP Methods are enabled as shown in snapshots below:



<Confidential>

The screenshot displays the Burp Suite interface with a target URL of `http://staging.hooghlycal.in`. The Request tab shows a GET request to `/Admin/login HTTP/1.1` from the browser `86aging.hooghlycal.in`. The Response tab shows the resulting HTML document, which includes a `<title>` tag with the text "Hooghly Calin Shipyard Limited" and a `<link>` tag for a stylesheet. The response also contains a `<style>` block with CSS rules for a login form, including margins, widths, and positions.

Note: Kindly, patch this vulnerability throughout the application.

Case X

i. **Affected Asset i.e. IP/URL/Application etc.**

<http://staging.hooghlycs1.in/en/>

ii. **Observation/ Vulnerability title**

Lack of Security Header

iii. **Detailed observation / Vulnerable point**

Applications can unintentionally leak information about their configuration, internal workings, or violate privacy through a variety of application problems. Attackers use this weakness to steal sensitive data or conduct more serious attacks.

iv. **CVE/CWE**

CWE-644

v. **Severity**

Medium

vi. **Recommendation**

Implement security headers such as X-XSS-Protection, Content-Security-Policy, Referrer Policy, X-Content-Type-Options, Permission Policy and Strict-transport-layer-protection.

To prevent untrusted JavaScript execution, replace allowlisted resources in script-based directives with a secure, random nonce of at least 8 characters 'nonce-RANDOM'.

Mitigate style-based data exfiltration by avoiding 'unsafe-inline', data: URLs, and global wildcards in style directives. Use a secure, random nonce of at least 8 characters 'nonce-RANDOM' in the relevant directive.

vii. **Reference**

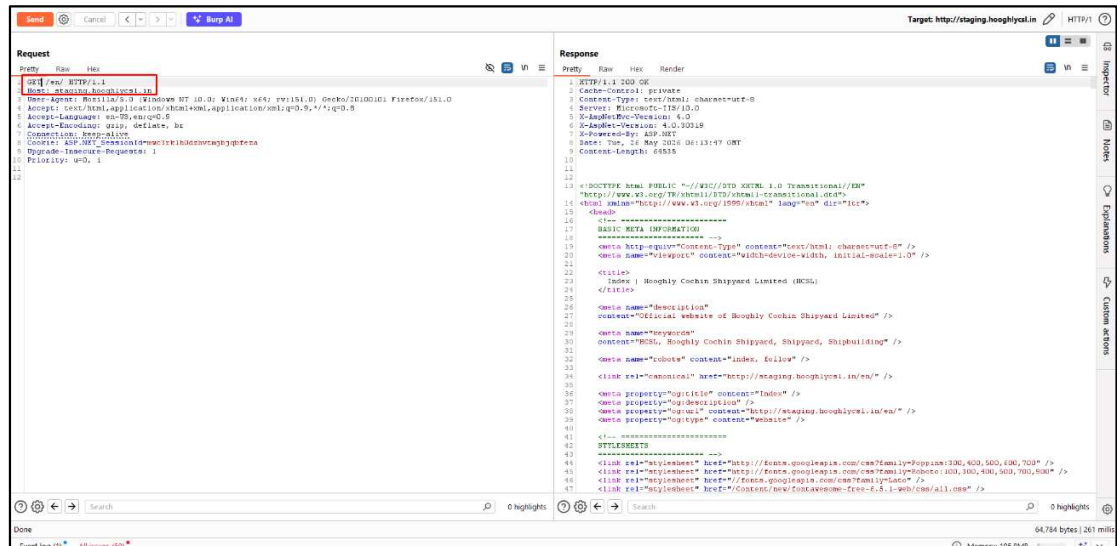
<https://cwe.mitre.org/data/definitions/644.html>

viii. **New or Repeat observation**

New

ix. References to evidence / Proof of Concept

Step I: Observe that security headers are not implemented as shown in snapshots below:



Note: Kindly, patch this vulnerability throughout the application.

Case XI

i. Affected Asset i.e. IP/URL/Application etc.

<http://staging.hooghlycs1.in/User/User>

ii. Observation/ Vulnerability title

Improper Input Validation

iii. Detailed observation / Vulnerable point

It is observed that the application does not validates user inputs, this can affect the control flow or data flow of a program.

iv. CVE/CWE

CWE - 20

v. Severity

Low

vi. Recommendation

All only the required characters for all the parameters as per requirement by whitelisting of those fields with not allowing all special and meta characters. Data type validators available natively in web application frameworks (such as Django Validators, Apache Commons Validators etc.). Minimum and maximum value range check for numerical parameters and dates, minimum and maximum length check for strings. Array of allowed values for small sets of string parameters (e.g., days of week).

vii. Reference

<https://cwe.mitre.org/data/definitions/20.html>

viii. New or Repeat observation

New

ix. References to evidence / Proof of Concept

Step I: Observe that inputs are not properly validated as shown in snapshot below:

The screenshot shows a web browser window with the URL <http://staging.hooghlycochin/User/User>. The page title is "HOOGHLY COCHIN SHIPYARD LIMITED". The left sidebar contains a "Test" menu with options: Admin, Tender, Career, Feedback, Go to Dashboard, Change Password, and Logout. The main content area is titled "Create New User" and includes a sub-header "Create New User" and a note "(* Fields are Mandatory)". The form contains the following fields:

- User Name (*)
- User Id (*)
- Password (*)
- Designation (*)
- Email (*)
- User Type (*)
- Is Active (*)

All input fields contain the payload `<script>alert(1)</script>`. A red box highlights the User Name, User Id, Designation, Email, and User Type fields. The "Is Active" checkbox is checked. A "SUBMIT" button is at the bottom right of the form.

Note: Kindly, patch this vulnerability throughout the application.

Case XII

i. Affected Asset i.e. IP/URL/Application etc.

<http://staging.hooghlycsl.in/Admin/Login>

<http://staging.hooghlycsl.in/en/add-feedback>

ii. Observation/ Vulnerability title

Weak Captcha

iii. Detailed observation / Vulnerable point

CAPTCHA ("Completely Automated Public Turing test to tell Computers and Humans Apart") is a type of challenge-response test used by many web applications to ensure that the response is not generated by a computer. CAPTCHA implementations are often vulnerable to various kinds of attacks even if the generated CAPTCHA is unbreakable. A web application should implement captcha where login, registration pages are there.

iv. CVE/CWE

CWE-804

v. Severity

Low

vi. Recommendation

Use a minimum 6-character alphanumeric CAPTCHA (mix of upper/lowercase letters and numbers).

Implement case-sensitive and randomized CAPTCHA strings.

Add distortion or background noise to prevent OCR attacks.

Validate CAPTCHA server-side to avoid bypass.

Expire CAPTCHA tokens after a short time or single use.

Limit failed attempts and enforce rate limiting.

vii. Reference

<https://cwe.mitre.org/data/definitions/804.html>

viii. New or Repeat observation

New

ix. References to evidence / Proof of Concept

Step I: Observe that weak captcha is implemented on login page as shown in snapshot below:

The first screenshot shows the 'Admin Login' page. The URL in the browser is <http://staging.hooghlycsli.in/Admin/Login>. The login form includes fields for 'USER NAME' (Email ID / User ID) and 'PASSWORD'. Below the password field is a CAPTCHA challenge: $4+8=?$. The user is prompted to enter the answer in the 'Answer' field. There is a 'SIGN IN' button and a 'Stay Signed In' checkbox. A 'Forgot Password?' link is also present.

The second screenshot shows the 'add-feedback' page. The URL in the browser is <http://staging.hooghlycsli.in/add-feedback>. The page has a header with navigation links: ABOUT US, TEAM, PRODUCTS & SERVICES, INVESTORS, Facilities, TENDER, Career Details, RELATED LINKS, and Contact Us. Below the header is a 'Permanent address' field. The main section is titled 'Feedback / Enquiry' and contains a text area for 'Enter your feedback or enquiry'. Below this is a 'Security Check' section with a CAPTCHA challenge: $6+2=?$. The user is prompted to enter the answer in the 'Enter answer' field. There are 'Submit' and 'Reset' buttons. At the bottom, there is a footer with 'HOOGHLY COCHIN SHIPYARD LIMITED' and various links.

Note: Kindly, patch this vulnerability throughout the application.

Case XIII

i. Affected Asset i.e. IP/URL/Application etc.

<http://staging.hooghlycs1.in/Admin/Dashboard>

ii. Observation/ Vulnerability title

Improper Session Timeout

iii. Detailed observation / Vulnerable point

The application does not automatically logs out the user after 20 mins of inactivity.

iv. CVE/CWE

CWE - 613

v. Severity

Low

vi. Recommendation

Application should automatically log out the user and destroy the session after 20-30 mins of inactivity.

vii. Reference

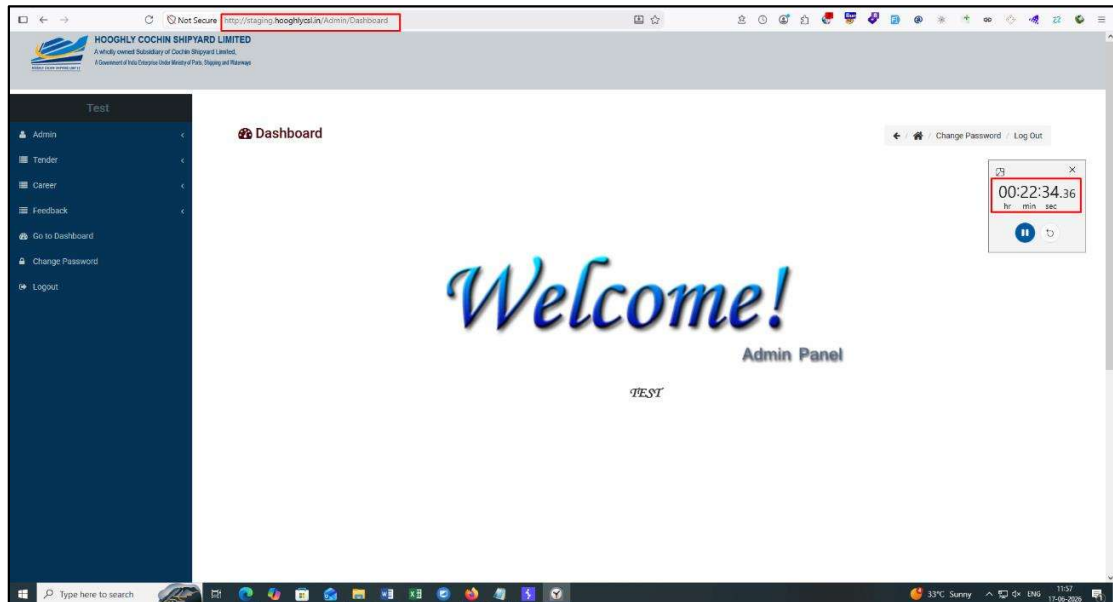
<https://cwe.mitre.org/data/definitions/613.html>

viii. New or Repeat observation

New

ix. References to evidence / Proof of Concept

Step I: It was observed that after 20 minutes of inactivity, the session remained active as shown in the snapshot below:



Note: Kindly, patch this vulnerability throughout the application.

Case XIV

i. Affected Asset i.e. IP/URL/Application etc.

<http://staging.hooghlycs1.in/Admin/ChangePassword>

ii. Observation/ Vulnerability title

Vulnerable and Outdated Components

iii. Detailed observation / Vulnerable point

Vulnerable components, such as libraries, frameworks, and other software modules almost always run with full privilege. So, if exploited, they can cause serious data loss or server takeover. Applications using these vulnerable components may undermine their defenses and enable a range of possible attacks and impacts.

iv. CVE/CWE

CWE-937

v. Severity

Low

vi. Recommendation

Upgrade the components used to the latest version.

If upgradation is not possible, download the security patches and hide the version details.

vii. Reference

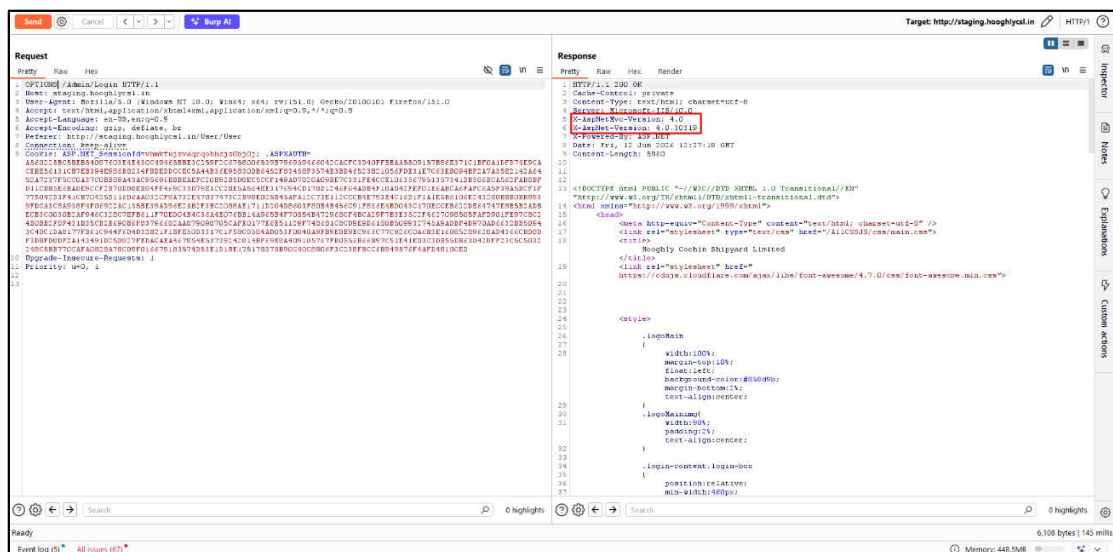
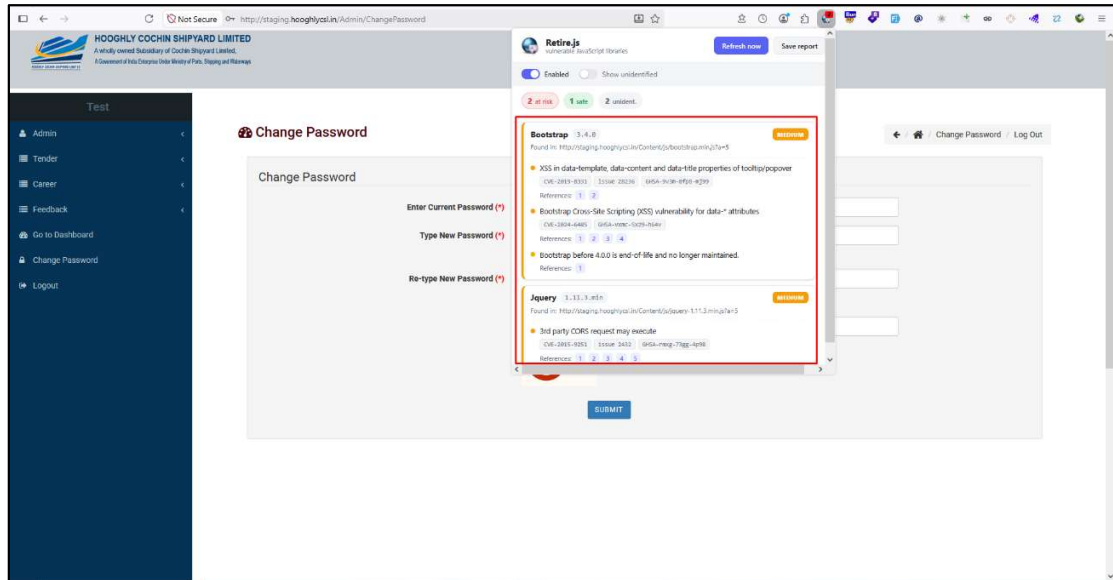
<https://cwe.mitre.org/data/definitions/937.html>

viii. New or Repeat observation

New

ix. References to evidence / Proof of Concept

Step I: Observe that application uses outdated and vulnerable version of Bootstrap, jquery and ASP.NET as shown as in snapshot below:



Note: Kindly, patch this vulnerability throughout the application.

Case XV

i. Affected Asset i.e. IP/URL/Application etc.

<http://staging.hooghlycs1.in/en/key-personnel>

ii. Observation/ Vulnerability title

Email Harvesting

iii. Detailed observation / Vulnerable point

An attacker can send SPAM mail to the application by using Email Harvester. The email harvesters are programs that scour the internet looking for email addresses on any website they come across.

iv. CVE/CWE

CWE-200

v. Severity

Low

vi. Recommendation

The application should properly customize the email addresses while posting on the website as: 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.

vii. Reference

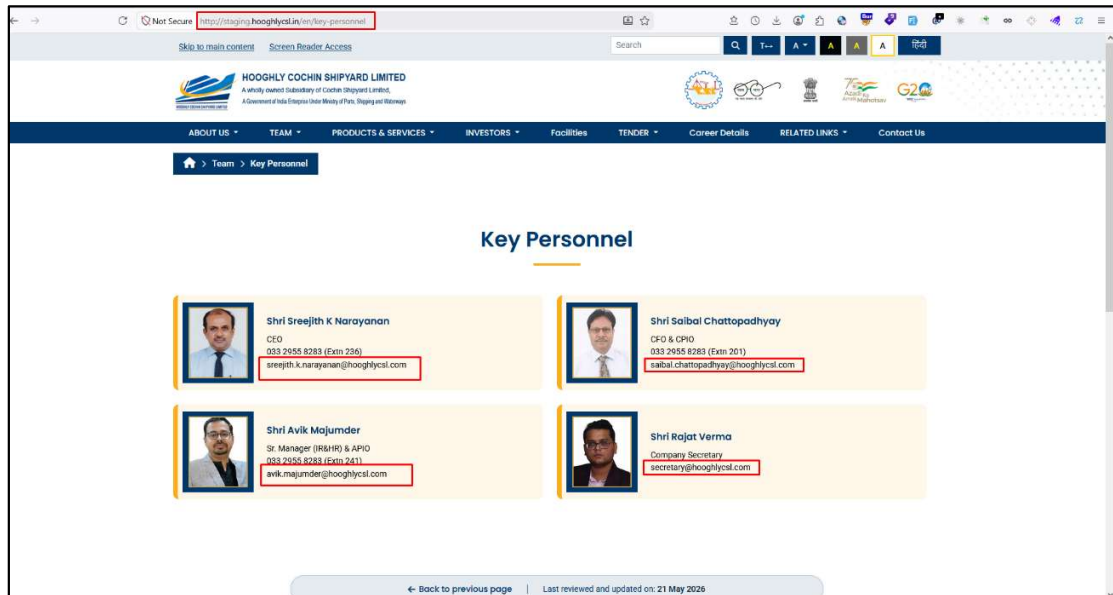
<https://cwe.mitre.org/data/definitions/200.html>

viii. New or Repeat observation

New

ix. References to evidence / Proof of Concept

Step I: Email addresses provided in the application are not in the proper format, as shown in the snapshot below:



Note: Kindly, patch this vulnerability throughout the application.

Case XVI

i. Affected Asset i.e. IP/URL/Application etc.

<http://staging.hooghlycs1.in/en/>

ii. Observation/ Vulnerability title

Server Banner Disclosure

iii. Detailed observation / Vulnerable point

Applications can unintentionally leak information about their configuration, internal workings, or violate privacy through a variety of application problems. Attackers use this weakness to steal sensitive data or conduct more serious attacks.

iv. CVE/CWE

CWE-200

v. Severity

Low

vi. Recommendation

Hide Server version details in the response.

vii. Reference

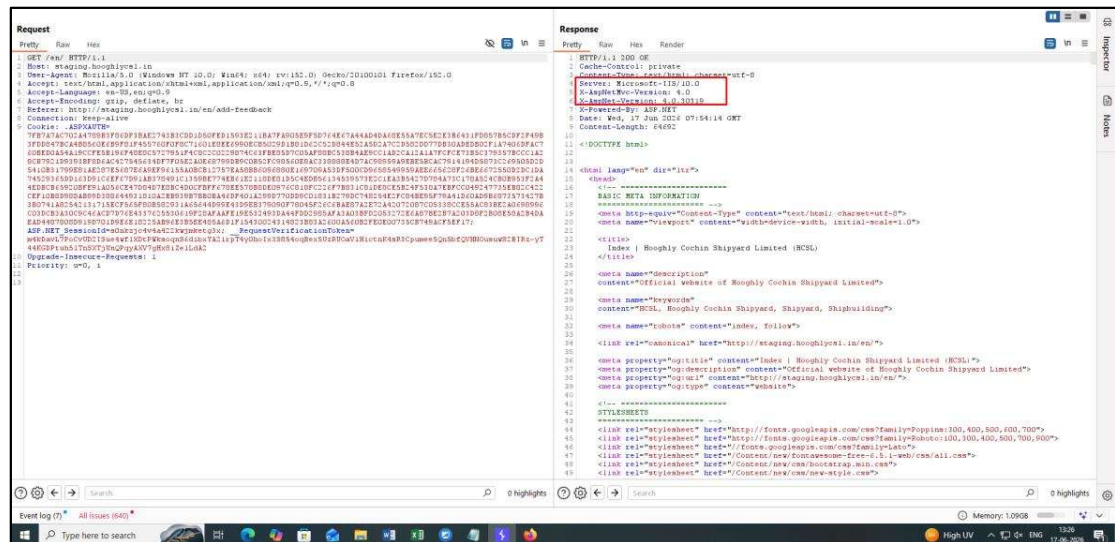
<https://cwe.mitre.org/data/definitions/200.html>

viii. New or Repeat observation

New

ix. References to evidence / Proof of Concept

Step I: Observe that server banner and ASP.NET version is disclosed as shown in snapshot below:



Note: Kindly, patch this vulnerability throughout the application.

Case XVII

i. Affected Asset i.e. IP/URL/Application etc.

http://staging.hooghlycs1.in/

http://staging.hooghlycs1.in/Admin/Login

ii. Observation/ Vulnerability title

SSL/TLS Not Implemented

iii. Detailed observation / Vulnerable point

Applications that do not implement SSL/TLS transmit data over unencrypted channels (HTTP), making them highly vulnerable to interception, manipulation, and unauthorized access. As communication occurs in plaintext, attackers can easily perform man-in-the-middle (MITM) attacks, steal sensitive information, or compromise user sessions. This lack of encryption significantly weakens the application's security posture and can lead to severedata breaches or loss of integrity.

iv. CVE/CWE

CWE-311

v. Severity

Low

vi. Recommendation

Obtain and install a valid SSL/TLS certificate from a trusted Certificate Authority (CA).

Configure the server to enforce HTTPS for all communication.

Redirect all HTTP traffic to HTTPS to prevent unencrypted access.

Implement HTTP Strict Transport Security (HSTS) headers to protect against downgrade attacks.

Regularly monitor, update, and renew SSL/TLS certificates before expiration.

Disable insecure protocols (e.g., SSLv2, SSLv3, TLS 1.0/1.1) and use strong cipher suites.

vii. Reference

<https://cwe.mitre.org/data/definitions/311.html>

viii. New or Repeat observation

New

ix. References to evidence / Proof of Concept

Implement SSL/TLS to secure data transmission and prevent interception or tampering of sensitive information.

Case XVIII

i. Affected Asset i.e. IP/URL/Application etc.

<http://staging.hooghlycs1.in/Admin/Login>

ii. Observation/ Vulnerability title

Security Logging and Monitoring

iii. Detailed observation / Vulnerable point

The application does not maintain any record of the application usage in the form of a report or audit trail. Any malicious activity cannot be monitored or traced back. In case of any misuse or attack, it may be difficult to trace and locate the origin.

iv. CVE/CWE

CWE-778

v. Severity

Low

vi. Recommendation

Information to be logged includes the following: IP address of the originating Source, Date, Time, Username (No Password), session details, Referrer, Process id, URL, User Agent, Countries if any in addition to other details to be logged in the website.

Logging of Authentication Process which includes number of successful and failed login attempts.

To create audit logs, use auto numbering so that every logged entry has an un-editable log number. Then if one audit entry is deleted a gap in the numbering sequence will appear.

Report of the website logs to be generated weekly by the administrator to keep track of the website activities.

vii. Reference

<https://cwe.mitre.org/data/definitions/778.html>

viii. New or Repeat observation

New

ix. References to evidence / Proof of Concept

There were no logs found in the application.

Ex: Failure and successful login attempts can be something like below:

Username	Time Stamp	IP Address	Login/Logout
Admin	02/02/2020;2.3 0.18	192.168.1.177	Login Failed
User	03/02/2020;2.3 0.18	192.168.1.177	Login Successful
Testing	04/02/2020;2.4 0.18	192.168.1.177	Logout

It is at the discretion of the website owner to determine what application specific information is to be logged.

Note: If logs are maintained on the server side, kindly share the relevant screenshots for verification.

<Confidential>

Observation

Case I: Add Career Not working

HOOGHLY COCHIN SHIPYARD LIMITED
A wholly owned Subsidiary of Cochin Shipyard Limited,
A Government of India Enterprise Under Ministry of Ports, Shipping and Waterways

Test

Admin <
Tender <
Career <
Feedback <
Go to Dashboard
Change Password
Logout

Add Career

Change Password / Log Out

Add Career Details (*) Fields are Mandatory

Advertisement Number (*) test

Career Type (*) Opportunities-Walk-in

Career Subject/ Description (In English) (*) test

Career Subject/ Description (In Hindi-UNICODE) (*) test

Career Last Date (*) 19-06-2026

Career Expiry Date 30-06-2026

Upload Career (In English) Browse... No file selected

Upload Career (In Hindi-UNICODE) Browse... No file selected

Redirect Link

Is Active? (*) YES

Is Important ? ☐

Position Position

SUBMIT RESET VIEW LIST

Case II: Add Tender not working

HOOGHLY COCHIN SHIPYARD LIMITED
A wholly owned Subsidiary of Cochin Shipyard Limited,
A Government of India Enterprise Under Ministry of Ports, Shipping and Waterways

Test

Admin <
Tender <
Career <
Feedback <
Go to Dashboard
Change Password
Logout

Add Tender

Change Password / Log Out

Add Tender (*) Fields are Mandatory

Tender No (*) test

Tender Subject/ Description (In English) (*) test

Tender Subject/ Description (In Hindi-UNICODE) (*) test

Tender Start Date 01-06-2026

Tender Last Date (*) 30-06-2026

Tender Opening Date (*) 01-06-2026

Tender Expiry Date (*) 30-06-2026

Upload Tender (In English) Browse... test.pdf

Upload Tender (In Hindi-UNICODE) Browse... test.pdf

Redirect Link

Is Active? (*) YES

Is Important ? ☐

Position Position

SUBMIT RESET VIEW LIST

Case III: Feedback not working

HOOGHLY COCHIN SHIPYARD LIMITED
A wholly owned Subsidiary of Cochin Shipyard Limited,
A Government of India Enterprise Under Ministry of Ports, Shipping and Waterways

ABOUT US TEAM PRODUCTS & SERVICES INVESTORS Facilities TENDER Career Details RELATED LINKS Contact Us

HCSL Feedback Portal

We are ready to lead you, send us your query
Fields marked with * are mandatory

Title *
Mrs.

Full Name *
Odysseus Shields

Email ID *
xasan@mailinator.com

Phone Number
+1 (644) 623-2164

Mobile No. *
7868768768

Address *
Numquam tempor guide

↑

HOOGHLY COCHIN SHIPYARD LIMITED
A wholly owned Subsidiary of Cochin Shipyard Limited,
A Government of India Enterprise Under Ministry of Ports, Shipping and Waterways

ABOUT US TEAM PRODUCTS & SERVICES INVESTORS Facilities TENDER Career Details RELATED LINKS Contact Us

Phone Number
+1 (644) 623-2164

Mobile No. *
7868768768

Address *
Numquam tempor guide

Feedback / Enquiry *
Molestias voluptatum

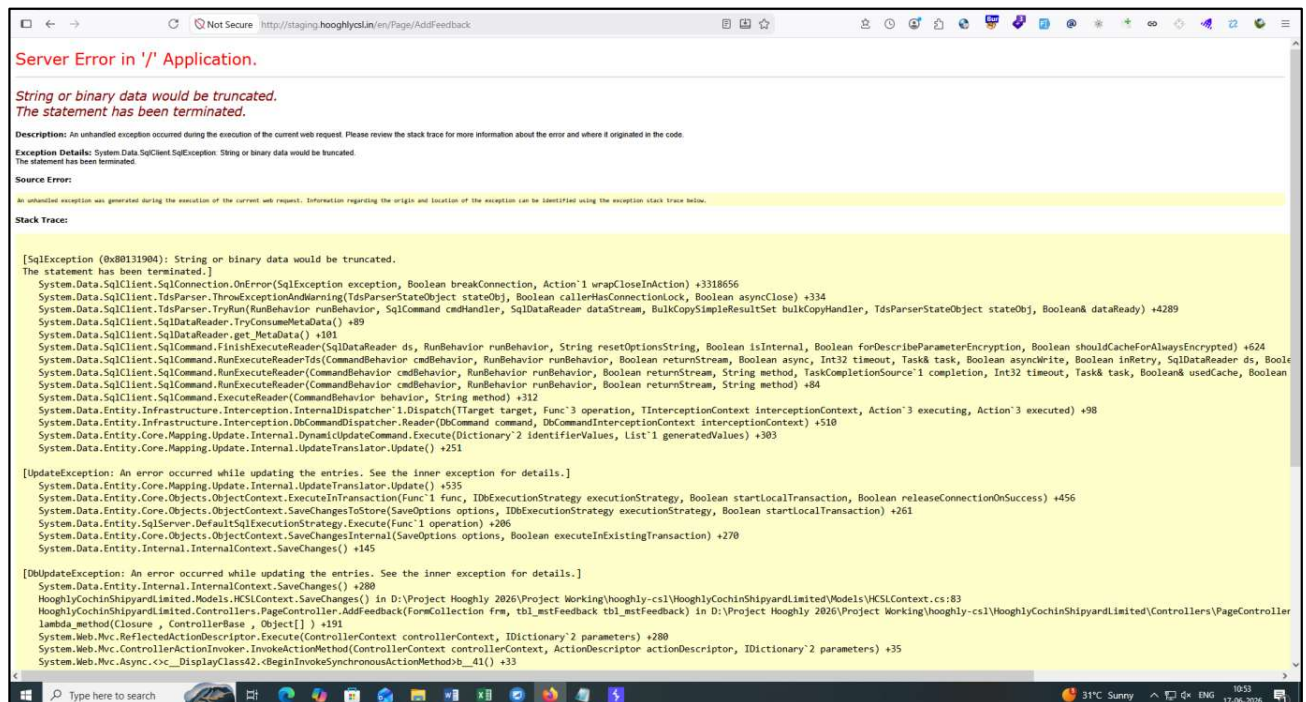
Security Check
4+8 = ?
12

Submit Reset

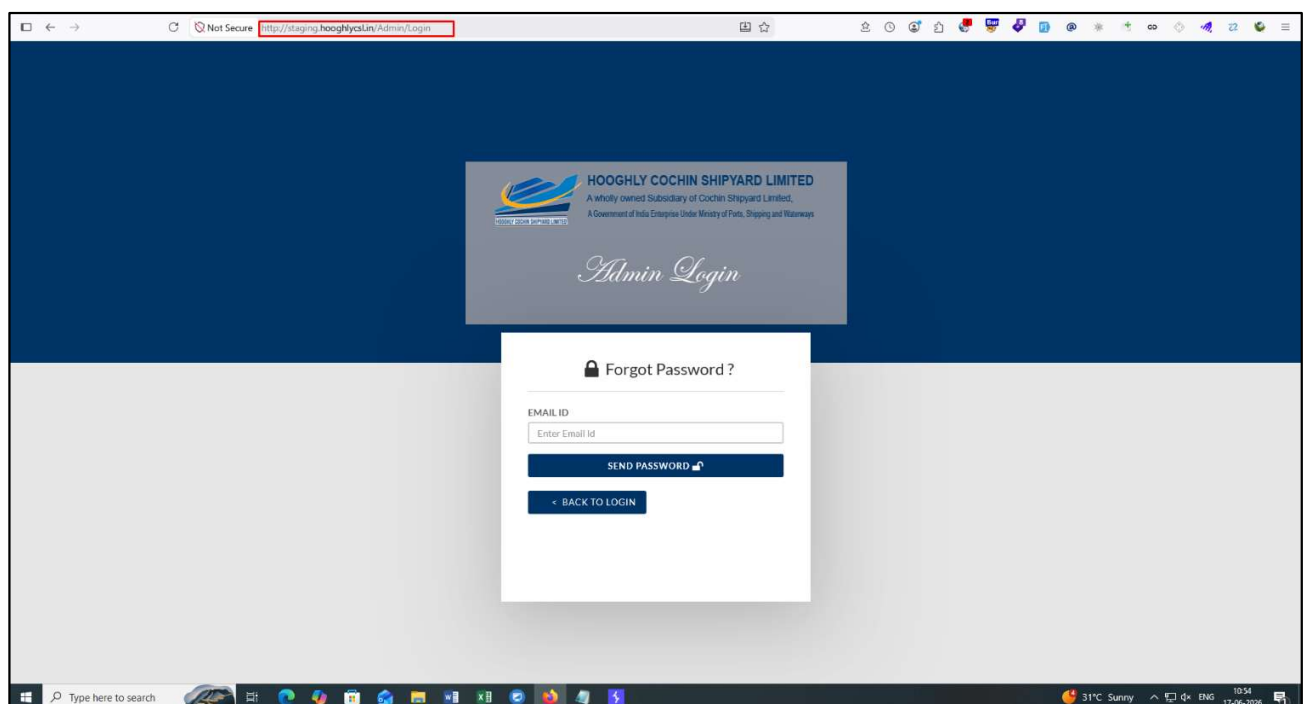
← Back to previous page | Last reviewed and updated on: 12 June 2026

↑

<Confidential>



Case IV: Forgot password not working



Appendix 'A'

Tools Description

Burp Suite Professional

Portswigger's Burp Suite Professional is an advanced set of tools for testing web security. Burp Suite offers the features for both manual and automated scans. Through Burp Suite, a user can intercept HTTP traffic, find hidden attack surface, assess strength of tokens, perform brute-forcing and fuzzing, construct CSRF exploits, modify HTTP messages, scan for common vulnerabilities including the OWASP Standard.

Acunetix Vulnerability Scanner

Acunetix is a web vulnerability scanner which is also a complete Web Application security testing solution that can be used both standalone and as part of complex environments. It offers built-in vulnerability assessment and vulnerability management, as well as many options for integration with market-leading software development tools.

DirSearch

DirSearch is a command-line tool written in Python for brute-forcing directories and files on web servers. It's widely used in security testing to discover hidden or restricted content on websites by making HTTP requests with various dictionary-based wordlists. DirSearch supports multi-threading, custom request headers, and multiple HTTP methods, making it a versatile tool for web reconnaissance.

SQLMAP

SQLMAP is an open-source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many niche features for the ultimate penetration tester and a broad range of switches lasting from database fingerprinting, over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out-of-band connections.