EXTERNAL PENETRATION TESTING REPORT

FOR

(CLIENT NAME)

CLIENT LOGO

BY

AUDITED BY

DATE: XXXX

CONFIDENTIAL

This document is a confidential document of XXXXXX part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording or otherwise, without the written permission of Threatsys. This document includes confidential information related to Threatsys and should not be distributed to any persons other than those mentioned in the distribution list herein.

Document Control

Item	Description
Document Title:	XXXXXX WAPT Report 1.0 – xxxx
Version No:	1.0
Status:	Final
Type:	PDF
Publish Date:	xx-xx-xxxx
Revision Date:	xx-xx-xxxx

Author(s)		
Name	***Functional Section, Department***	***Signature/Date***
Approved by		
Name	***Functional Section, Department***	***Signature/Date***
Deepak Kumar Nath	Chief Technology Officer	xx-xx-xxxx

Document Amendment Record

Version	Date	Prepared By	Description
1.0	xx-xx-xxxx		Initial Report created for XXXX

Table of content

1. Executive Summary

1.1 Introduction

This report presents the results of the white box penetration testing for XXXXXX web Application. The purpose of this assessment was to test external web app to identify technical and functional vulnerabilities, discover whether a malicious user may leverage these flaws to compromise the security of the XXXXXX Organization and provide Preventions for risk mitigation that may arise on successful exploitation of these vulnerabilities.

The assessment was done within a controlled environment. The assessment was started on xx xxxx xxxx and ended on xx xxxx xxxx.

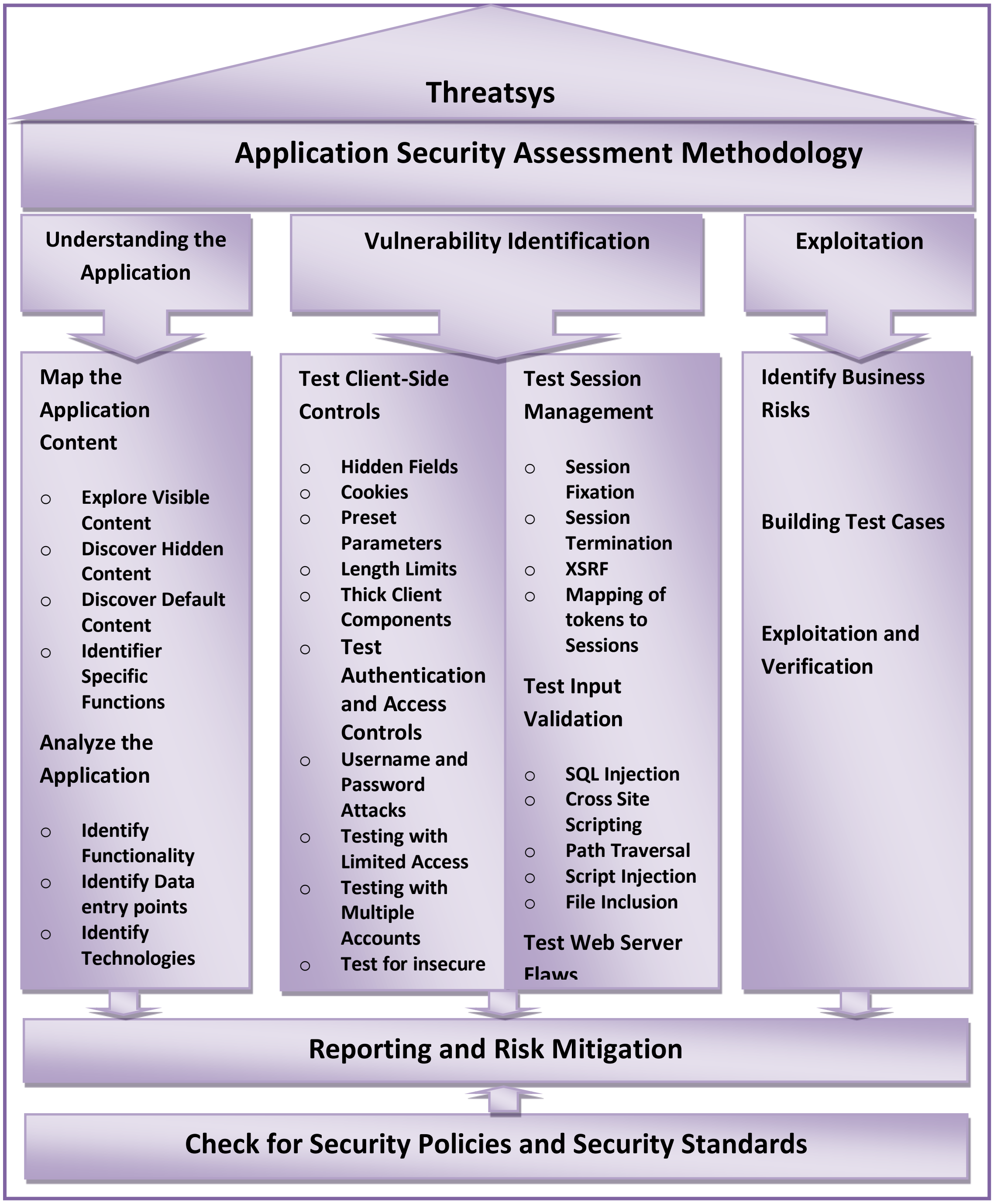
External Penetration testing assessment was conducted as a 'Black-box' & 'White-box' exercise. This was done to simulate as closely as possible from the viewpoint of an internal attacker.

The subsequent sections of this document provide statistics of the vulnerabilities identified. The detailed technical findings section constitutes identified vulnerabilities with Preventions to mitigate security risks associated with the servers.

Our opinion provided in this report is valid for the period during which the assessment was carried out and is based on the information provided for the assessment. Projection of any conclusions based on our findings for future periods and applications / operating versions is subject to the risk that the validity of such conclusions may be altered because of changes made to the web applications or. Furthermore, the findings in this report reflect the conditions found during the assessment, and do not necessarily reflect current conditions.

2. METHODOLOGY

Threatsys used a combination of the Open Web Application Security Project (OWASP) testing guide and ISECOM's Open-Source Security Testing Methodology Manual (OSSTMM) for conducting penetration test of the server and applications. The testing was done to simulate as closely as possible the viewpoint of completely external attacker and Application user.


Threatsys
Application Security Assessment Methodology
Understanding the Application
Vulnerability Identification
Exploitation
Map the Application Content
○ Explore Visible Content
○ Discover Hidden Content
○ Discover Default Content
○ Identifier Specific Functions
Analyze the Application
○ Identify Functionality
○ Identify Data entry points
○ Identify Technologies
Test Client-Side Controls
○ Hidden Fields
○ Cookies
○ Preset Parameters
○ Length Limits
○ Thick Client Components
○ Test Authentication and Access Controls
○ Username and Password Attacks
○ Testing with Limited Access
○ Testing with Multiple Accounts
○ Test for insecure
Test Session Management
○ Session Fixation
○ Session Termination
○ XSRF
○ Mapping of tokens to Sessions
Test Input Validation
○ SQL Injection
○ Cross Site Scripting
○ Path Traversal
○ Script Injection
○ File Inclusion
Test Web Server Flaws
Identify Business Risks
Building Test Cases
Exploitation and Verification
Reporting and Risk Mitigation
Check for Security Policies and Security Standards

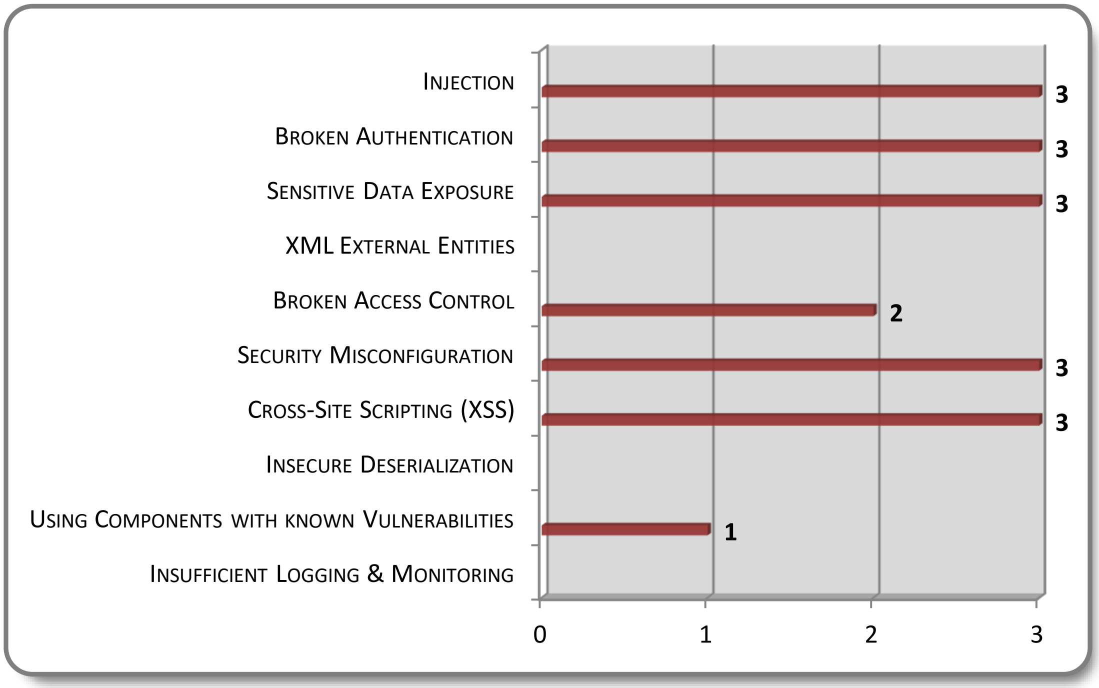
2.1 Scope of Work

The scope of the assessment was to perform a White-box web application PT

WEB application URL
http://example.com

2.2 Benchmarking with OWASP Top 10 for Application PT

The below chart depicts the number of discovered vulnerabilities mapped to OWASP 2017 testing category.



OWASP has rated the Top Ten Vulnerabilities found in Web applications worldwide. OWASP Top Ten Vulnerabilities details can be found at below link.

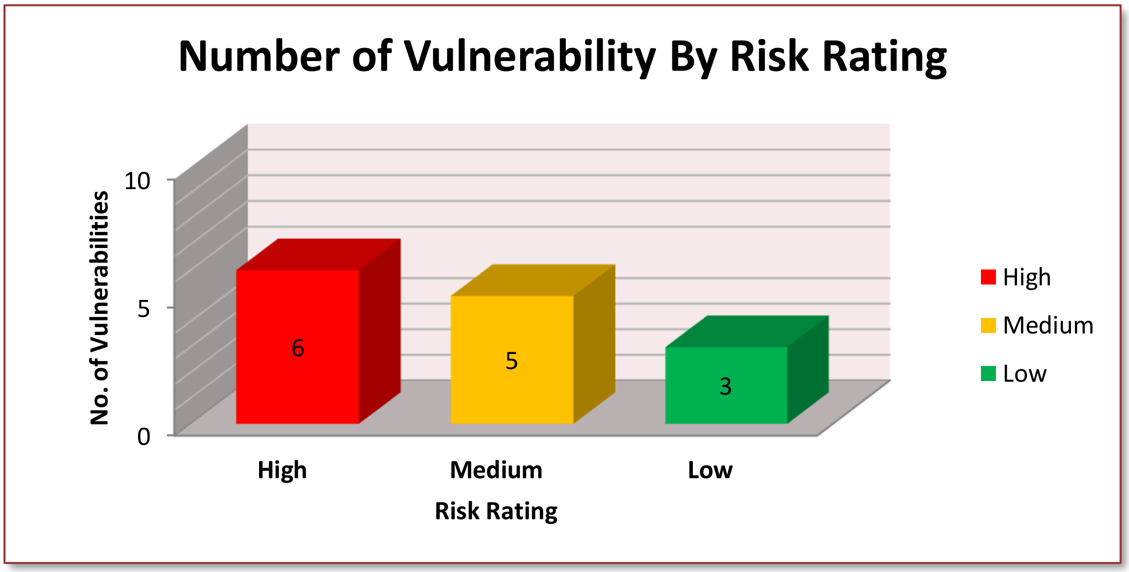
https://www.owasp.org/index.php/Top_10-2017_Top_10

The table below shows how Vulnerabilities compares with respect to the OWASP 2017 Top 10 list.

#	VULNERABILITIES	STATUS
1	Injection	Un-Protected
2	Broken Authentication	Un-Protected
3	Sensitive Data Exposure	Un-Protected
4	XML External Entities	Protected
5	Broken Access Control	Un-Protected
6	Security Misconfiguration	Un-Protected
7	Cross-Site Scripting	Un-Protected
8	Insecure Deserialization	Protected
9	Using Components with known Vulnerabilities	Un-Protected
10	Insufficient Logging & Monitoring	Protected

3. Summary of Findings

3. 1 Total Number of Findings



3.2 Findings Summary

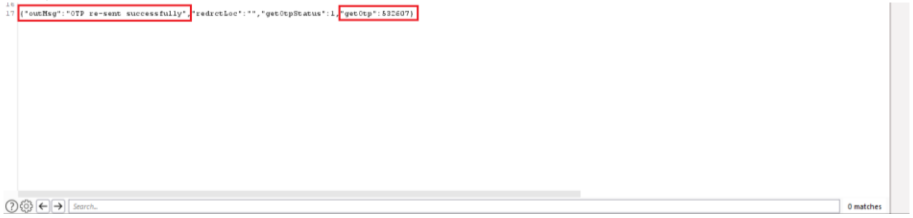
Device Type	IP Address	Risk Rating				
		Critical	High	Medium	Low	Total
WEB Application	http://example.com	1	5	4	3	13
Total		1	5	4	3	13

Note: It was not possible to add all the endpoints on the report. As the below listed bugs are present in all the similar endpoints. All the below mentioned fix should be applied with all the similar endpoints. Each parameter should be protected as mentioned below.

3.3 Findings Vulnerability Summary

S. No	Vulnerability	Severity
1.	ACCOUNT TAKEOVER	CRITICAL
2.	CROSS SITE SCRIPTING (Stored)	HIGH
3.	HTML INJECTION	HIGH
4.	CROSS SITE REQUEST FURGERY	HIGH
5.	INFORMATION DISCLOSURE	HIGH
6.	BROKEN AUTHENDICATION	HIGH
7.	IMPROPER SESSION MANAGEMENT	MEDIUM
8.	BROKEN ACCESS CONTROL	MEDIUM
9.	OPEN REDIRECTION	MEDIUM
10.	NORATE LIMITING	MEDIUM
11.	PASSWORD IN CLEAR TEXT	LOW
12.	LOW VERSION JQUARY	LOW
13.	LOW VERSION BOOTSTRAP	LOW

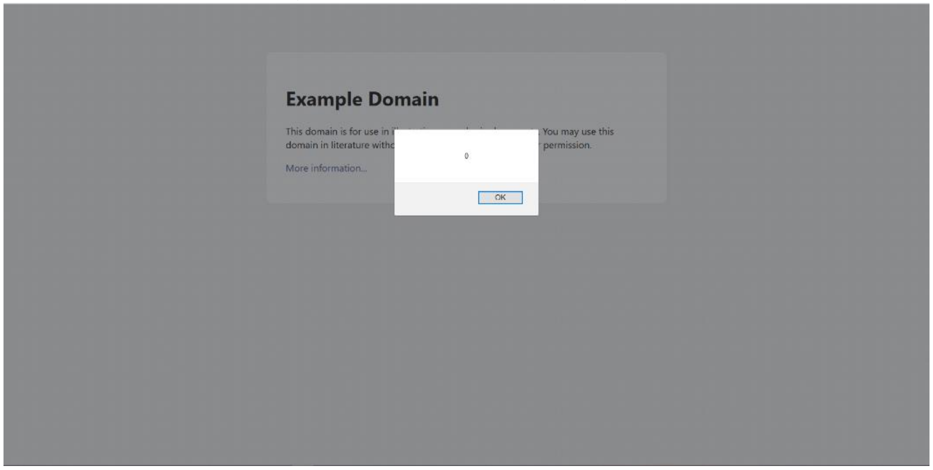
4. Detailed Findings

Reference	**FND - 01**	***Finding Ownership***	
Raised By	Red Team	***Department***	Threatsys Red Team
Date Raised	xx-xx-xxxx	***Assigned To***	
Severity	**CRITICAL**	***Remediation Plan Expected***	
ISD Representative		***Remediation Expected***	
Server's Affected	http://example.com/forgotPassword		
Vulnerability Ref	https://cheatsheetseries.owasp.org/cheatsheets/Multifactor_Authentication_Cheat_Sheet.html		
Title	**ACCOUNT TAKEOVER – OTP LEAKAGE IN RESPONSE**		
Description	Our team detected that the URL mansion above has a functionality for forget password and Reset password through OTP, and that OTP is being Disclosed in response which makes it vulnerable to Bypass OTP and takeover Admin account.		
Prevention	OTP should not be Disclosed in Response and also use hashing or encryption Technique to Encode the OTP so if attack anyhow intercept the communication, it becomes hard to understand the real valid OTP.		
Evidence	Below screenshot shows that Valid OTP is being disclosed in response:  `{"outMsg":"OTP re-sent successfully","redrctLoc":"","getOtpStatus":1,"getOtp":532607}` 		

Impact	It is observed that Attacker having malicious intension can able to takeover Admin account By Resetting Password Through This Vulnerability and fully compromise the security of web application.

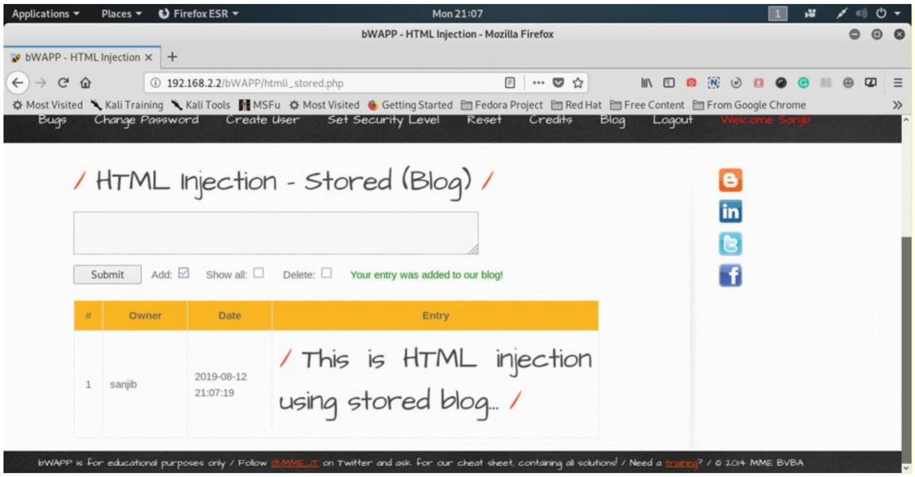
5. Detailed Findings

Reference	**FND - 02**	***Finding Ownership***	
Raised By	Red Team	***Department***	
Date Raised	xx-xx-xxxx	***Assigned To***	
Severity	**HIGH**	***Remediation Plan Expected***	
ISD Representative		***Remediation Expected***	
Server's Affected	http://example.com/console/entities/edit/3		
Vulnerability Ref	https://owasp.org/www-community/attacks/xss/		
Title	**CROSS SITE SCRIPTING (Stored)**		
Description	Our team observed that the above URL has a functionality for adding a value in a input filed which is storing the user input in an unsafe way which led to execute the JavaScript code. The code is storing in the server and this makes it a stored xss.		
Recommendation	It is recommended to sanitize all user input with the help of input validation and text filtering methodologies in URLs, textboxes and other input fields to avoid cross-site scripting.		

Impact	Attacker can steal user cookies and hence the session.
Evidence	Below Screenshot is the proof for xss, see the pop-up below.  Example Domain This domain is for use in i . You may use this domain in literature witho permission. More information... 0

6. Detailed Findings

Reference	**FND - 03**	***Finding Ownership***	
Raised By	Red Team	***Department***	
Date Raised	xx-xx-xxxx	***Assigned To***	
Severity	**HIGH**	***Remediation Plan Expected***	
ISD Representative		***Remediation Expected***	

Server's Affected	http://example.com
Vulnerability Ref	https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/11-Client-side_Testing/03-Testing_for_HTML_Injection
Title	**HTML INJECTION**
Description	Our team observed that the above URL has an input filed which is vulnerable to HTML Injection. due to misconfiguration, we were able to inject a HTML Code Because it is not sanitizing the user input properly which results in the HTML payloads being executed.
Recommendation	It is recommended to sanitize all user input with the help of input validation and text filtering methodologies in URLs, textboxes and other input fields to avoid cross-site scripting.
Evidence	Below screenshot shows that HTML payload has been successfully Executed.  Applications Places Firefox ESR Mon 21:07 bWAPP - HTML Injection - Mozilla Firefox bWAPP - HTML Injection 192.168.2.2/bWAPP/htmli_stored.php Most Visited Kali Training Kali Tools MSFu Most Visited Getting Started Fedora Project Red Hat Free Content From Google Chrome Bugs Change Password Create User Set Security Level Reset Credits Blog Logout Welcome Sanjib Submit Add: ☑ Show all: □ Delete: □ / HTML Injection - Stored (Blog) / Your entry was added to our blog! # | Owner | Date | Entry 1 | sanjib | 2019-08-12 21:07:19 | / This is HTML injection using stored blog... / bWAPP is for educational purposes only / Follow @MME_IT on Twitter and ask for our cheat sheet, containing all solutions! / Need a training? / © 2014 MME BVBA
Impact	It can cause phishing, spoofing etc.

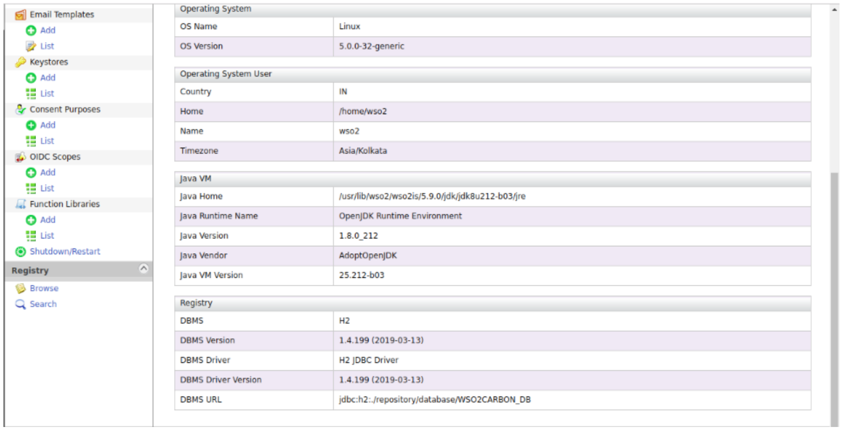
7. Detailed Findings

Reference	**FND - 04**	***Finding Ownership***	
Raised By	Red Team	***Department***	Threatsys Red Team
Date Raised	xx-xx-xxxx	***Assigned To***	
Severity	**HIGH**	***Remediation Plan Expected***	
ISD Representative		***Remediation Expected***	
Server's Affected	http://example.com/profile		
Vulnerability Ref	https://owasp.org/www-community/attacks/csrf		
Title	**CROSS SITE REQUEST FORGERY**		
Description	Our team detected that FOR Changing any data in the above URL is vulnerable to CSRF attacks as the requests being sent to server doesn't contain anti-CSRF tokens and server is not verifying that the request is forged or not Because of that a User manage to create a link for changing any data which content the action, can be send to admin to change their profile data and after that its lead to account takeover through reset password using email.		
Prevention	It is recommended to implement anti-CSRF token in each and every request which have state changing functionality or Request verification token should be expired after being used for a single time & The token should be random and unique for each user.		
Evidence	**Below POC of CSRF:** **Step 1:** **<body>** **<script>history.pushState('', '', '/')</script>** **<form action="http://example.com/profile" method="POST">**		

	<input type="hidden" name="name" value="EVIL CSRF" /> <input type="hidden" name="email id" value="AccountTakeOver@gmail.com" /> </form> </body> </html> **STEP 2:** **“Save this POC in a .html file and send it to the admin”** **STEP 3:** **Send .html file to Admin.** **“After Submitting Request, the data of admin profile will be changed.”**
Impact	It is observed that the admin can be tricked into clicking a link to change the data or take an action which they never intended to.

8. Detailed Findings

Reference	**FND-06**	*Finding Ownership*	
Raised By	RED TEAM	*Department*	Threatsys

<table>
<tr><td></td><td></td><td></td><td>Red Team</td></tr>
<tr><td><em>Date Raised</em></td><td>xx-xx-xxxx</td><td><em>Assigned To</em></td><td></td></tr>
<tr><td><em>Severity</em></td><td><strong>HIGH</strong></td><td><em>Remediation Expected Date</em></td><td></td></tr>
<tr><td><em>ISD Representative</em></td><td></td><td><em>Remediation Expected</em></td><td></td></tr>
<tr><td><em>Server's Affected</em></td><td colspan="3">http://example.com/</td></tr>
<tr><td><em>Title</em></td><td colspan="3"><strong>Information Disclosure</strong></td></tr>
<tr><td><em>Description</em></td><td colspan="3">Our team has detected that the application Disclose some very sensitive data like "DBMS name" "Host name" "DBMS PASSWORD" "Environment Detail" and many sensitive data which an attack can use for Further exploitation.</td></tr>
<tr><td><em>Prevention</em></td><td colspan="3">It is recommended to mask the Sensitive information about the Data base and environment details hidden from the Users.</td></tr>
<tr><td><em>Evidence</em></td><td colspan="3">Below screenshot shows that Information is Disclosed:


<table>
<tr><th colspan="2">Operating System</th></tr>
<tr><td>OS Name</td><td>Linux</td></tr>
<tr><td>OS Version</td><td>5.0.0-32-generic</td></tr>
<tr><th colspan="2">Operating System User</th></tr>
<tr><td>Country</td><td>IN</td></tr>
<tr><td>Home</td><td>/home/wso2</td></tr>
<tr><td>Name</td><td>wso2</td></tr>
<tr><td>Timezone</td><td>Asia/Kolkata</td></tr>
<tr><th colspan="2">Java VM</th></tr>
<tr><td>Java Home</td><td>/usr/lib/wso2/wso2is/5.9.0/jdk/jdk8u212-b03/jre</td></tr>
<tr><td>Java Runtime Name</td><td>OpenJDK Runtime Environment</td></tr>
<tr><td>Java Version</td><td>1.8.0_212</td></tr>
<tr><td>Java Vendor</td><td>AdoptOpenJDK</td></tr>
<tr><td>Java VM Version</td><td>25.212-b03</td></tr>
<tr><th colspan="2">Registry</th></tr>
<tr><td>DBMS</td><td>H2</td></tr>
<tr><td>DBMS Version</td><td>1.4.199 (2019-03-13)</td></tr>
<tr><td>DBMS Driver</td><td>H2 JDBC Driver</td></tr>
<tr><td>DBMS Driver Version</td><td>1.4.199 (2019-03-13)</td></tr>
<tr><td>DBMS URL</td><td>jdbc:h2:./repository/database/WSO2CARBON_DB</td></tr>
</table></td></tr>
<tr><td><em>Impact</em></td><td colspan="3">It is observed that if an attacker has access to this information, then use that information for further malicious exploitation.</td></tr>
</table>

9. Detailed Findings

Reference	**FND-07**	*Finding Ownership*	

Raised By	Red Team	***Department***	Threatsys Red Team
Date Raised	xx-xx-xxxx	***Assigned To***	
Severity	**HIGH**	***Remediation Plan Expected***	
ISD Representative		***Remediation Expected***	
Server's Affected	http://example.com/changePassword		
Vulnerability Ref	https://owasp.org/www-project-top-ten/2017/A2_2017-Broken_Authentication		
Title	**BROKEN AUTHENDICATION**		
Description	Our team detected that the app need authentication to view the inside functionality like "change password" which requires login, but anyone can directly access the "Change password" Functionally without any authentication & it allows the malicious user to change the password of "admin" which makes it Vulnerable for Broken Authentication.		
Recommendation	It is recommended that the inside panel structure shouldn't be available with a bad access, it should show 302 permanently moved.		
Evidence	**Steps to Reproduce:** **Step 1: Open login page http://example.com in a tab.** **Step 2: Now open This url "http://example.com/changePassword" Now you can see You can able to change the password by giving "OLD password" "NEW password" in the input filed without being authenticate.**		
Impact	It is observed that an attacker can view or Do unauthorized content, Functionality or Action without authentication and compromise the security of web application.		

10. DETAILED FINDINGS

Reference	**FND-08**	***Finding Ownership***	
Raised By	RED TEAM	***Department***	Threatsys Red Team
Date Raised	xx-xx-xxxx	***Assigned To***	
Severity	**MEDIUM**	***Remediation Expected Date***	
ISD Representative		***Remediation Expected***	
Server's Affected	http://example.com		
Vulnerability Ref	https://affinity-it-security.com/what-is-a-session-management-vulnerability/		
Title	**IMPROPER SESSION MANAGEMENT**		
Description	Our team detected that, if we are logged in from multiple browsers and we change password from one browser, we are not logged out of other browsers. In other words, the session doesn't expire once a user is logged in even if the user changes his password.		
Impact	It is observed that if an attacker has the credentials or he/she is logged in to an account by the help of some social engineering attack, the genuine user can't secure his account by simply changing		

	his password as the session on the attacker's end won't expire granting him complete access to the victim's account.
Prevention	It is recommended to implement proper session management.
Evidence	1: Login into http://example.com using your credentials. 2: Open another browser and login using the same credentials. Now change the password in this browser. 3: Switch to the other browser where the password wasn't changed and click on any link there and it will open without asking you to login again. This confirms that all the active sessions are not expiring even after a password change.

11. Detailed Findings

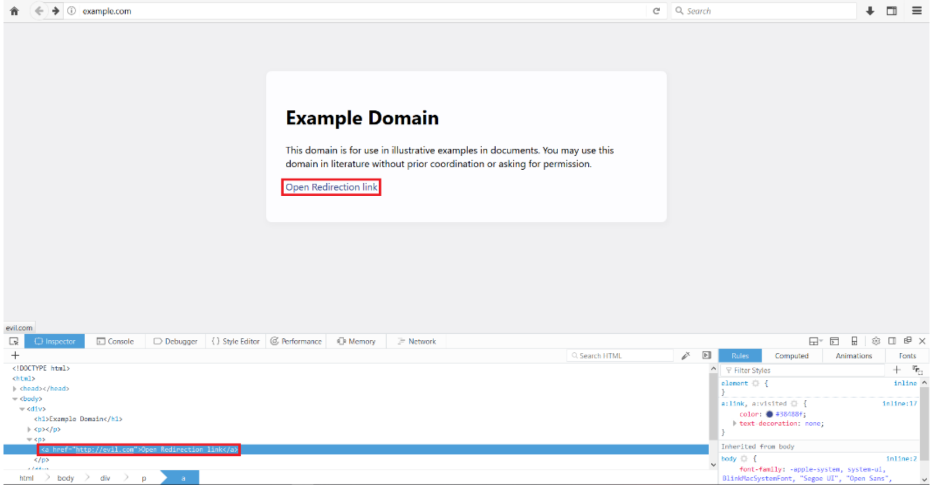
Reference	**FND-09**	***Finding Ownership***	
Raised By	Red Team	***Department***	Threatsys Red Team
Date Raised	xx-xx-xxxx	***Assigned To***	
Severity	**MEDIUM**	***Remediation Plan Expected***	
ISD Representative		***Remediation Expected***	
Server's Affected	http://example.com/		
Vulnerability Ref	https://owasp.org/www-project-top-ten/2017/A5_2017-Broken_Access_Control		

Title	**BROKEN ACCESS CONTROL**
Description	Our team detected that the above URL has a functionality to view the personal detail of a user which is in pdf format. We can able to view that PDF which contain “sensitive data of a user” without any authentication this issue led to broken access control. hence, we can able to view any user Sensitive Data files just by changing the endpoint value in URL without any authentication.
Recommendation	It is recommended to implement an access control, i.e., the user needs to be authorized before the server provides the requested information. It is often recommended to use something less obvious that is harder to enumerate as a reference or encrypt the file name in the URL. This is a good mitigation for multiple scenarios but it should not be considered as the only mitigation against such attacks.
Evidence	Through This URL http://example.com /application/1111/user1_Details.pdf We can able to view the pdf file a user which contain sensitive information.
Impact	It is observed that an attacker can view unauthorized details of a user which contain sensitive information.

12. Detailed Findings

Reference	**FND-10**	***Finding Ownership***	
Raised By	Red Team	***Department***	
Date Raised	xx-xx-xxxx	***Assigned To***	

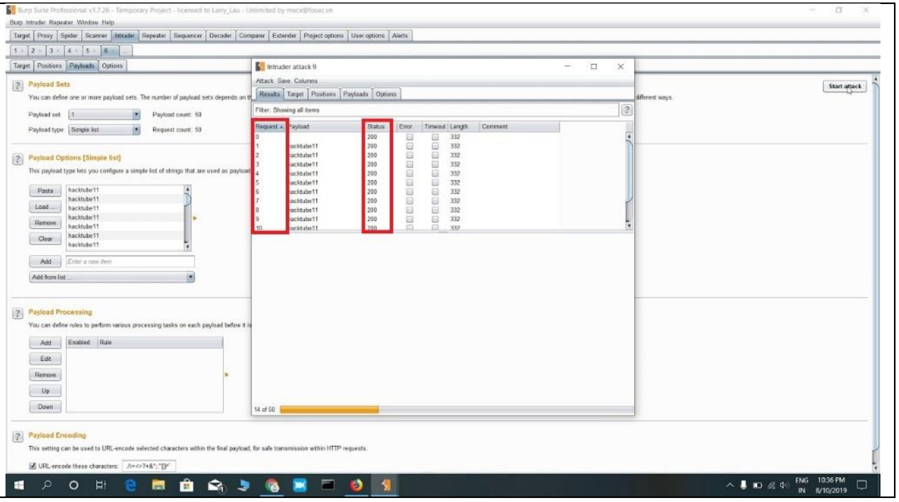
Severity	**MEDIUM**	***Remediation Plan Expected***	
ISD Representative		***Remediation Expected***	
Server's Affected	http://example.com/		
Vulnerability Ref	https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/11-Client_Side_Testing/04-Testing_for_Client_Side_URL_Redirect		
Title	**OPEN REDIRECTION**		
Description	Our team found that the URL above has an input filed which stores the user input data in a unsafe manner through which we can able to create a stored buttons which will redirect the user to a malicious website which is controlled by the attacker. The user's data will be theft in this way.		
Recommendation	It is recommended to sanitize all user input with the help of input validation and text filtering methodologies in URLs, textboxes and other input fields to avoid cross-site scripting.		

Evidence	The below screenshot show there is a button for redirection.
	
	Example Domain
	This domain is for use in illustrative examples in documents. You may use this domain in literature without prior coordination or asking for permission.
	Open Redirection link
	`<!DOCTYPE html>`
	`<html>`
	`<head></head>`
	`<body>`
	`<div>`
	`<h1>Example Domain</h1>`
	`<p></p>`
	`<p>`
	`<a href="http://evil.com">Open Redirection link</a>`
	`</p>`
	`element { } inline`
	`a:link, a:visited { inline:17`
	`color: #38488f;`
	`text decoration: none;`
	`}`
	`Inherited from body`
	`body { inline:2`
	`font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans",`		

Impact	This can cause phishing.		

13. DETAILED FINDINGS

Reference	**FND-11**	***Finding Ownership***	
Raised By	RED TEAM	***Department***	Threatsys Red Team
Date Raised	xx-xx-xxxx	***Assigned To***	
Severity	**MEDIUM**	***Remediation Expected Date***	
ISD Representative		***Remediation Expected***	
Server's Affected	http://example.com		
Vulnerability Ref	https://www.cloudflare.com/learning/bots/what-is-rate-limiting/		
Title	**No Rate Limiting**		
Description	Our team detected that the Login and OTP parameter field can be brute forced as there is no rate limiting functionality. This allowed our team to send numerous requests for the end points. Which will lead to bypass the Login functionality and hence it will lead to the account take over.		
Impact	An attacker can bypass the login panel, OTP panel and takeover the admin panel through this vulnerability		
Prevention	It is suggested to restrict the maximum number of consecutive failed tries by a user and give an error response, "429: Too Many Tries". Another safeguard would be to block login attempts for some time after few consecutive failures.		
Evidence	Below screenshot shows that we can able to send multiple request:		



Burp Suite Professional v1.7.26 - Temporary Project - licensed to Larry_Lau - Unlimited by mxcx@fosec.vn

Payload Sets

You can define one or more payload sets. The number of payload sets depends on th

Payload set: 1 Payload count: 50

Payload type: Simple list Request count: 50

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payload

hacktube11
hacktube11
hacktube11
hacktube11
hacktube11
hacktube11
hacktube11

Payload Processing

You can define rules to perform various processing tasks on each payload before it is

Payload Encoding

This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

☑ URL-encode these characters: ./\=<>?+&*;:"{}|^`

Intruder attack 9

Filter: Showing all items

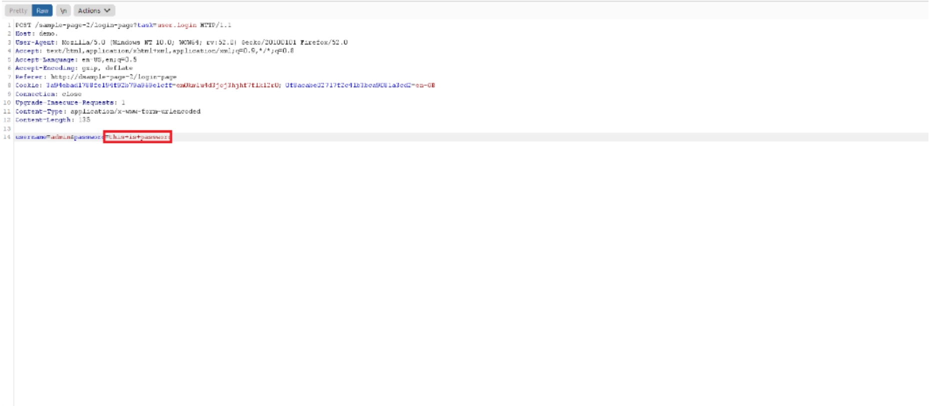
Request ▲	Payload	Status	Error	Timeout	Length	Comment
0		200	□	□	332	
1	acktube11	200	□	□	332	
2	acktube11	200	□	□	332	
3	acktube11	200	□	□	332	
4	acktube11	200	□	□	332	
5	acktube11	200	□	□	332	
6	acktube11	200	□	□	332	
7	acktube11	200	□	□	332	
8	acktube11	200	□	□	332	
9	acktube11	200	□	□	332	
10	acktube11	200	□	□	332	

14 of 50

Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
1 2 3 4 5 6 ...
Target Positions Payloads Options
Paste Load ... Remove Clear Add
Enter a new item
Add from list ...
Add Edit Remove Up Down
Enabled Rule
Attack Save Columns
Results Target Positions Payloads Options
Start attack
different ways.
ENG IN 10:36 PM 8/10/2019

14. Detailed Findings

Reference	**FND-12**	***Finding Ownership***	
Raised By	RedTeam	***Department***	
Date Raised	27-01-2021	***Assigned To***	
Severity	**Low**	***Remediation Plan Expected***	
ISD Representative		***Remediation Expected***	
Server's Affected	http://example.com/changePassword		

Vulnerability Ref	https://owasp.org/www-community/vulnerabilities/Password_Plaintext_Storage
Title	**PASSWORD IN CLEAR TEXT**
Description	Our team has detected that the application is transmitting passwords in clear text over unencrypted connections which makes it vulnerable to interception.
Recommendation	It is recommended to encrypt all data in transit using TLS v5 to avoid any kind of data theft.
Evidence	The below screenshot shows the server is transmitting the password in plain text.  Pretty Raw \n Actions POST /sample-page-2/login-page?task=user.login HTTP/1.1 Host: demo. User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:52.0) Gecko/20100101 Firefox/52.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Referer: http://dsample-page-2/login-page Cookie: 3a94ebad1788fe194t92b79a969e1cft=emUkmlu4d3joj3hjht7tlk12r0; Ut8acabe02717f2c41b3bca9081a3cd2=en-GB Connection: close Upgrade-Insecure-Requests: 1 Content-Type: application/x-www-form-urlencoded Content-Length: 135 username=admin&password=this+is+password
Impact	It is observed that if an attacker is suitably positioned then he/she can eavesdrop on the victim’s network. This scenario is possible if a user communicates over an insecure connection, like public Wi-Fi.

15. Detailed Findings

Reference	**FND-13**	***Finding Ownership***	
Raised By	Red Team	***Department***	

Date Raised	xx-xx-xxxx	***Assigned To***	
Severity	**Low**	***Remediation Plan Expected***	
ISD Representative		***Remediation Expected***	
Server's Affected	http://example.com/public/js/jquery-3.3.1.min.js		
Vulnerability Ref	https://owasp.org/www-project-top-ten/2017/A9_2017-Using_Components_with_Known_Vulnerabilities		
Title	**OUTDATED JQUERY**		
Description	Our team detected that the server is using out dated version of jQuery which is version 3.3.1. It has some previously known vulnerabilities which can be exploited.		
Recommendation	It is recommended to update jQuery to the latest version 3.5.1		
Evidence	N/A		
Impact	This version has the vulnerability for jQuery prototype pollution.		

16. Detailed Findings

Reference	**FND-14**	***Finding Ownership***	
Raised By	Red Team	***Department***	Threatsys

			Red Team
Date Raised	xx-xx-xxxx	***Assigned To***	
Severity	**LOW**	***Remediation Plan Expected***	
ISD Representative		***Remediation Expected***	
Server's Affected	http://example.com/public/js/bootstrap.min.js		
Vulnerability Ref	https://owasp.org/www-project-top-ten/2017/A9_2017-Using_Components_with_Known_Vulnerabilities		
Title	**OUTDATED BOOTSTRAP**		
Description	Our team detected that the server is using out dated version of BOOTSTRAP which is 4.3.1 version. It has some previously known vulnerabilities which can be exploited.		
Recommendation	It is recommended to update BOOTSTRAP to the latest version 4.5.3		
Evidence	N/A		
Impact	Since this is an old version of the software, it may be vulnerable to attacks		

17. Conclusion

Threatsys has done a thorough Internal Penetration testing for XXXXXX web Application, Threatsys suggests XXXXXX Organization to implement the Preventions in this document with respect to affected applications and infrastructure. The following table suggests the timelines to implement the Preventions based on the risk rating of the vulnerabilities.

Risk Rating	Timelines
CRITICAL	Implement controls within 1 week
High	Implement controls within 2 to 3 weeks
Medium	Implement controls within 1 months
Low	Implement controls within 1 to 2 months

APPENDIX A – DEFINITIONS -VULNERABILITY RATING & EXPLOITATION

VULNERABILITY LEVELS	DESCRIPTION
High	If the exploitation of the vulnerability can result in complete takeover of the / destruction of the / disclosure of highly sensitive information. Exploit for this vulnerability is easily available
Medium	If the exploitation of the vulnerability can result in Partial control of the / Partial destruction of the / disclosure of semi sensitive information. Exploit for this vulnerability is possible but not available
Low	If the exploitation of the vulnerability can result in little or no impact on the / disclosure of less sensitive information. Exploit for this vulnerability is very difficult to obtain