**Web Application - Papermill**

**Audit Report**

|  |  |
| --- | --- |
| **Report Release Date** | 02.05.2024 |
| **Type of Audit** | Web Application |
| **Type of Audit Report** | Final audit report |
| **Period** | 30-04-2024 (10:00 AM) to 02-05-2024 (4:00 PM). |

Document Control

|  |  |
| --- | --- |
| **Document Preparation** | |
| **Document Title** | Web Application – Papermill |
| **Document ID** |  |
| **Document Version** |  |
| **Prepared by** | Amil khan |
| **Reviewed by** |  |
| **Approved by** |  |
| **Released by** |  |
| **Release date** | 02.05.2024 |

|  |  |  |
| --- | --- | --- |
| **Document Change History** | | |
| **Version** | **Date** | **Remarks / Reason of change** |
| N/A | N/A | N/A |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Document Distribution List** | | | |
| **Name** | **Organization** | **Designation** | **Email Id** |
| Rohit Kumar |  |  |  |
|  |  |  |  |

## Contents

[Introduction 4](#_bookmark0)

[Engagement Scope 5](#_bookmark1)

[Details of the Auditing team 6](#_bookmark2)

[Audit Activities and Timelines 7](#_bookmark3)

[Audit Methodology and Criteria / Standard referred for audit 8](#_bookmark4)

[Tools/ Software used 9](#_bookmark5)

[Executive Summary 10](#_bookmark6)

[Detailed Observations 11](#_bookmark7)

[Appendices 12](#_bookmark8)

# Introduction

* This audit aims to assess and verify vulnerabilities, pinpointing exact weak points.
* Objectives include identifying risks, improving security, and ensuring compliance. Assumptions, exemptions, and limitations are documented.
* Through this process, the audit seeks to fortify defenses, mitigate potential threats, and uphold standards of data protection and system integrity.

# Engagement Scope

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S.**  **No** | **Asset Description** | **Criticality of Asset** | **Internal IP**  **Address** | **URL** | **Public IP**  **Address** | **Location** | **Hash Value (in case of applications)** | **Version (in**  **case of applications)** |
| 1 | Web application which is a premier source for high quality paper products. | Web Application -Papermill | 192.168.125.56 | http://192.168.125.56:8080 | 192.168.125.56 | N/A | N/A |  |

Date up to which the list has been updated: 02.05.2024

# Details of the Auditing team

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S. No | Name | Designation | Email Id | Professional Qualifications/ Certifications | Whether the  resource has been listed in the Snapshot information published on CERT-In’s  website(Yes/No) |
| 1 | Amil khan |  |  |  |  |
| 2 |  |  |  |  |  |

# Audit Activities and Timelines

# Audit Methodology and Criteria / Standard referred for audit

Methodology

A penetration test simulates covert and hostile network activities to identify specific exploitable vulnerabilities and to expose potential entryways to vital or sensitive data that, if discovered and misused by a malicious individual, could pose increased risk and liability to the organization, its executives, and shareholders.



* 1. **Information Gathering:**

The application details were comprehensively understood based on the information provided by the Application team. All pertinent documentation and additional details relevant to the application were considered within the assessment's scope.

* 1. **Planning and analysis:**

Scope was defined based on the nature, timing, and extent of the evaluation which was to be conducted in consultation with the team. The nature and type of the tools to be used were determined by theTeam and shared with the Team. The machine(s) which was to be used for conducting the tests was prepared accordingly. Testing was conducted during normal business hours post the consultation with the Application Testing Team.

* 1. **Attack & Penetration testing:**

The testing procedures were conducted with the presumption that the tester functioned as an external attacker. Specific tools were employed for this purpose. Proof of concept (POC) documentation was compiled for each vulnerability exploited during the assessment, intended for further reporting. Recommendations were also formulated to address these vulnerabilities systematically.

* 1. **Analysis & Reports:**

The assessment was conducted by prioritizing vulnerabilities according to their severity ratings. Subsequently, a comprehensive report was generated by articulating the analyses conducted, accompanied by recommendations for remediation.

* 1. **Severity Rating**

This rating depicts the severity of the impact on the organization. Depending on the criticality of the device, it could represent business impact, financial impact, or damage to a customer, partner, or reputation.

**HIGH**

It is imperative that effort is undertaken immediately to mitigate the vulnerabilities in this category.

**MEDIUM**

Medium threats need to be reviewed and discussed after which their treatment should be designed as a part of the security plan.

**LOW**

Low threats need to be reviewed and discussed.

* 1. **Result Collation and Report Preparation:**

Results are then collated in tabular form for easy reference. Vulnerability will be accompanied by a short description and screenshots.

# Tools/ Software used

|  |  |  |  |
| --- | --- | --- | --- |
| S. No | Name of Tool/Software used | Version of the tool  /Software used | Open Source/Licensed |
| 1 | Burp suite Community Edition | Latest | Open Source |

# Executive Summary

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S. No | Affected Asset i.e.  IP/URL/Application etc | Observation/ Vulnerability title | CVE/CWE | Control Objective # | Control Name # | Audit Requirement # | Severity | Recommendation |
| 1 | 192.168.125.56:8080/v1/client/profile | Reflected Cross Site Scripting | N/A | N/A | N/A | N/A | High | Yes |
| 2 | 192.168.125.56:8080/register | Admin account bypass via Duplicate Account Creation | N/A | N/A | N/A | N/A | High | Yes |
| 3 | 192.168.125.56:8080/v1/adm/add-item | Unrestricted File Upload | N/A | N/A | N/A | N/A | Medium | Yes |
| 4 | 192.168.125.56:8080/login | Weak Authentication | N/A | N/A | N/A | N/A | Medium | Yes |
| 5 | 192.168.125.56:8080/v1/adm/master | Admin Page Disclose | N/A | N/A | N/A | N/A | Medium | Yes |
| 6 | 192.168.125.56:8080/v1/adm/past-orders | HTML Injection | N/A | N/A | N/A | N/A | Medium | Yes |
| 7 | 192.168.125.56:8080/v1/client/profile | IFrame Injections | N/A | N/A | N/A | N/A | Medium | Yes |
| 8 | 192.168.125.56:8080 | Clickjacking | N/A | N/A | N/A | N/A | Medium | Yes |
| 9 | 192.168.125.56:8080 | Unencrypted Communication | N/A | N/A | N/A | N/A | Medium | Yes |
| 10 | 192.168.125.56:8080/login | Concurrent | N/A | N/A | N/A | N/A | Medium | Yes |
| 11 | 192.168.125.56:8080/v1/client/profile | Improper Input Validation | N/A | N/A | N/A | N/A | Medium | Yes |
| 12 | 192.168.125.56:8080/login | No Captcha Implemented | N/A | N/A | N/A | N/A | Medium | Yes |
| 13 | 192.168.125.56:8080/v1/api/orders/customer | Sensitive information discloser | N/A | N/A | N/A | N/A | Medium | Yes |
| 14 | 192.168.125.56:8080 | Numerous port open | N/A | N/A | N/A | N/A | Medium | Yes |
| 15 | 192.168.125.56:8080 | Cacheable http Response | N/A | N/A | N/A | N/A | Medium | Yes |
| 16 | 192.168.125.56:8080 | Improper error handling | N/A | N/A | N/A | N/A | Medium | Yes |
| 17 | 192.168.125.56:8080 | Insecure Content-Type | N/A | N/A | N/A | N/A | Medium | Yes |
| 18 | 192.168.125.56:8080 | Missing CSP Security Header | N/A | N/A | N/A | N/A | Medium | Yes |
| 19 | 192.168.125.56:8080 | Missing Same site Attribute | N/A | N/A | N/A | N/A | Medium | Yes |
| 20 | 192.168.125.56:8080 | Cookies Without Http Only Flag set | N/A | N/A | N/A | N/A | Medium | Yes |
| 21 | 192.168.125.56:8080 | Internal IP  Disclosure | N/A | N/A | N/A | N/A | Low | Yes |

# Detailed Observations

1. **Reflected XSS**

**Description**

Reflected Cross-Site Scripting (XSS) is a type of web application vulnerability that allows an attacker to inject malicious code into a website and execute it in the victim's browser.

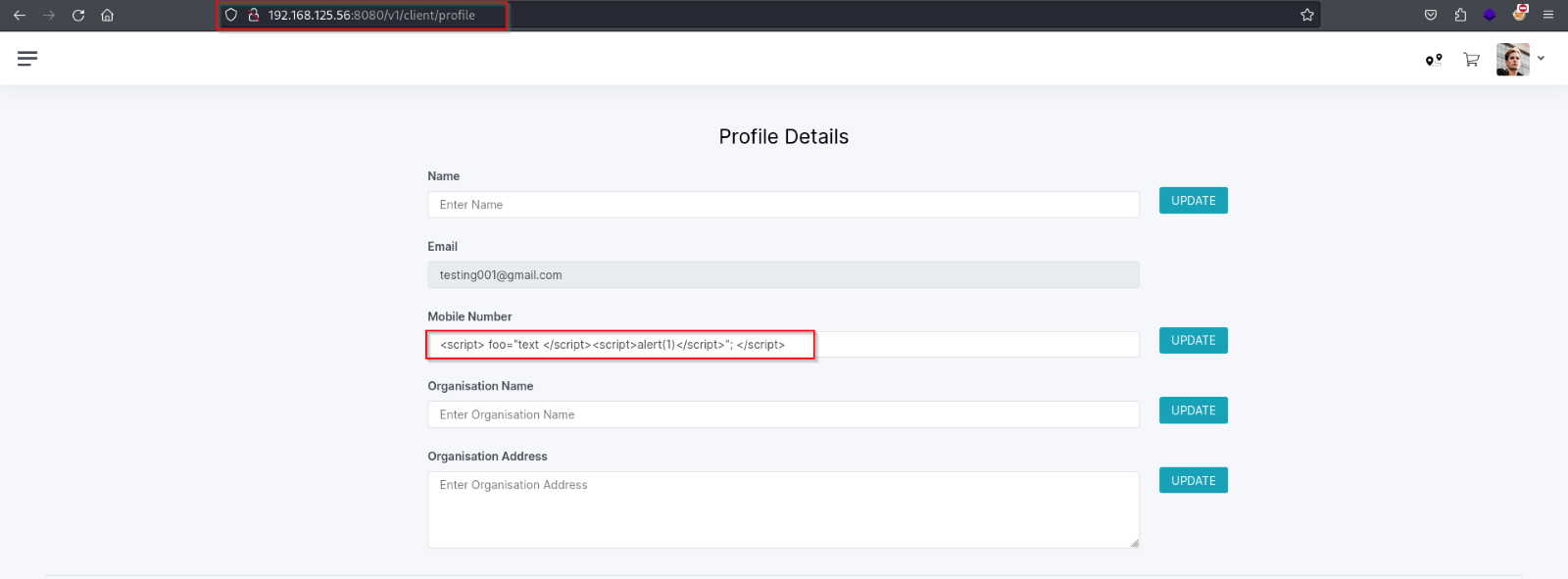
**Severity**

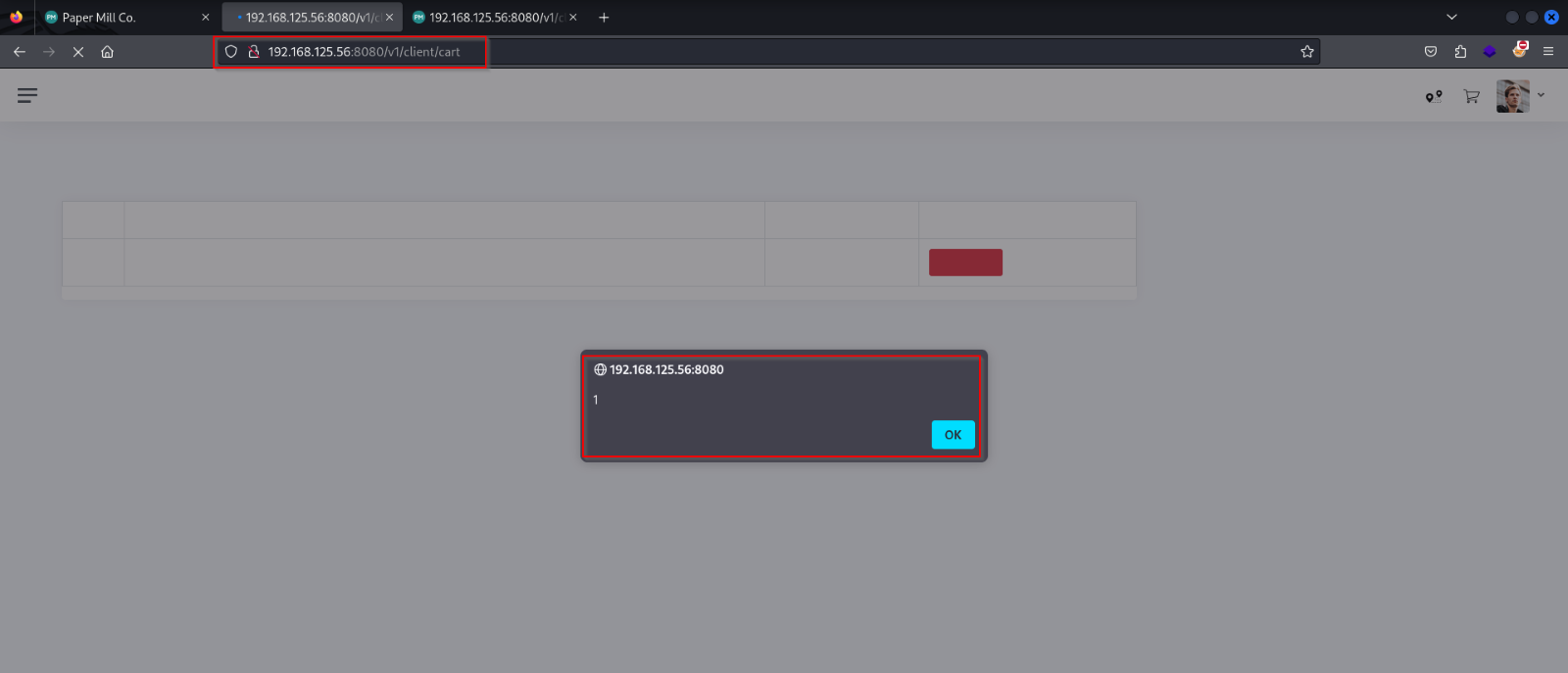
High

**Impact**

Reflected XSS attacks occur when an attacker is able to inject malicious code into a web application, which is then reflected back to the victim's browser in the form of an error message or other response. When the victim clicks on a link or visits a page that contains the reflected code, the code is executed in their browser, giving the attacker access to the victim's cookies, session tokens, or other sensitive information.

**Proof of Concept**





**Remediation**

To prevent Reflected XSS vulnerabilities, web developers should ensure that user input is properly validated

and sanitized before it is included in a response. This can include encoding user input or filtering out any

characters that could be used to execute malicious code.

1. **Admin account bypass via Duplicate Account Creation**

**Description**

The Duplicate Account Creation vulnerability allows an attacker to bypass the authentication mechanism of the admin account by exploiting a flaw that permits the creation of multiple accounts with identical credentials. This vulnerability arises from inadequate validation or enforcement of uniqueness constraints during the account creation process.

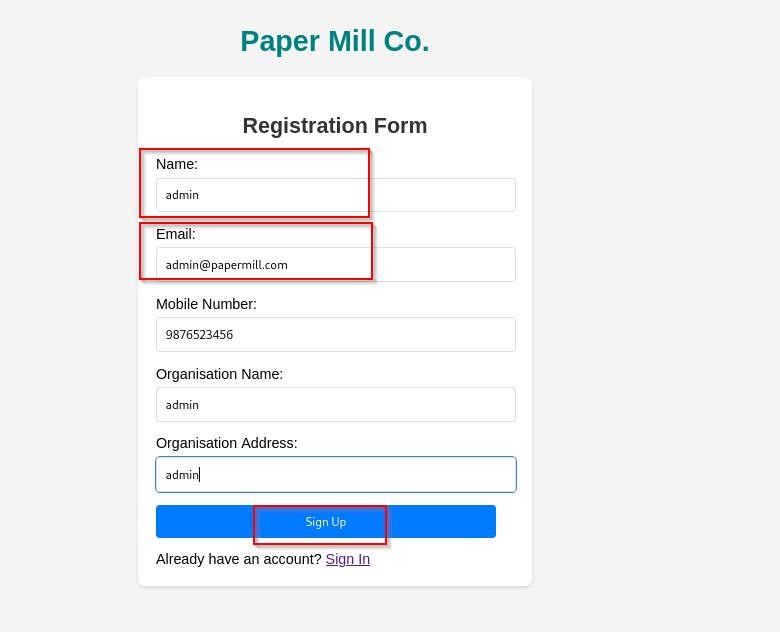
**Impact:**

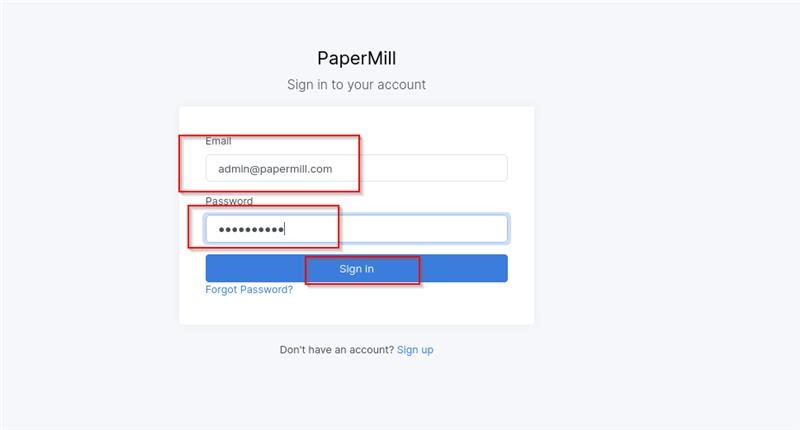
Unauthorized Access: Exploiting this vulnerability enables unauthorized users to gain administrative privileges, potentially leading to the compromise of sensitive data, manipulation of system configurations, or disruption of services.

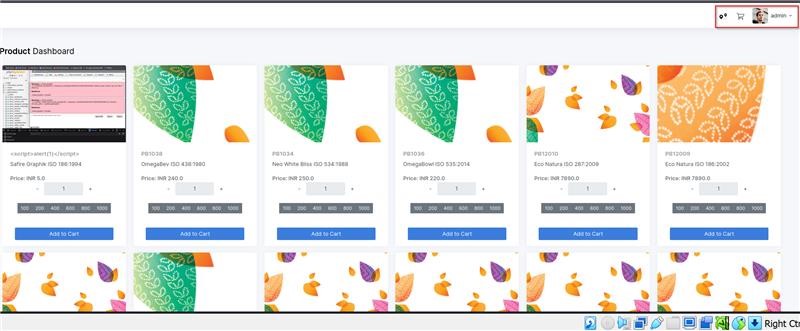
Data Breach: An attacker with admin privileges can access and manipulate confidential information stored within the system, including user data, financial records, and proprietary data.

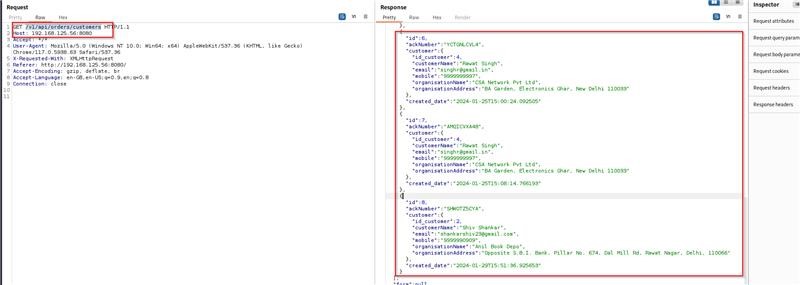
Reputation Damage: Successful exploitation of this vulnerability can tarnish the organization's reputation, leading to loss of trust among users, partners, and stakeholders.

**Proof of Concept**









**Remediation:**

Develop and deploy a patch that addresses the root cause of the Duplicate Account Creation vulnerability. Implement strict validation checks to prevent the creation of accounts with identical credentials.

Enforce rate limiting mechanisms to restrict the number of account creation attempts from a single IP address or device within a specific

Implement multi-factor authentication (MFA) for admin accounts to add an extra layer of security, making it more difficult for attackers to compromise accounts even if credentials are obtained

1. **Unrestricted file upload**

**Description**

Unrestricted file upload is a security vulnerability that occurs when a web application allows users to upload files without properly validating or sanitizing the input.

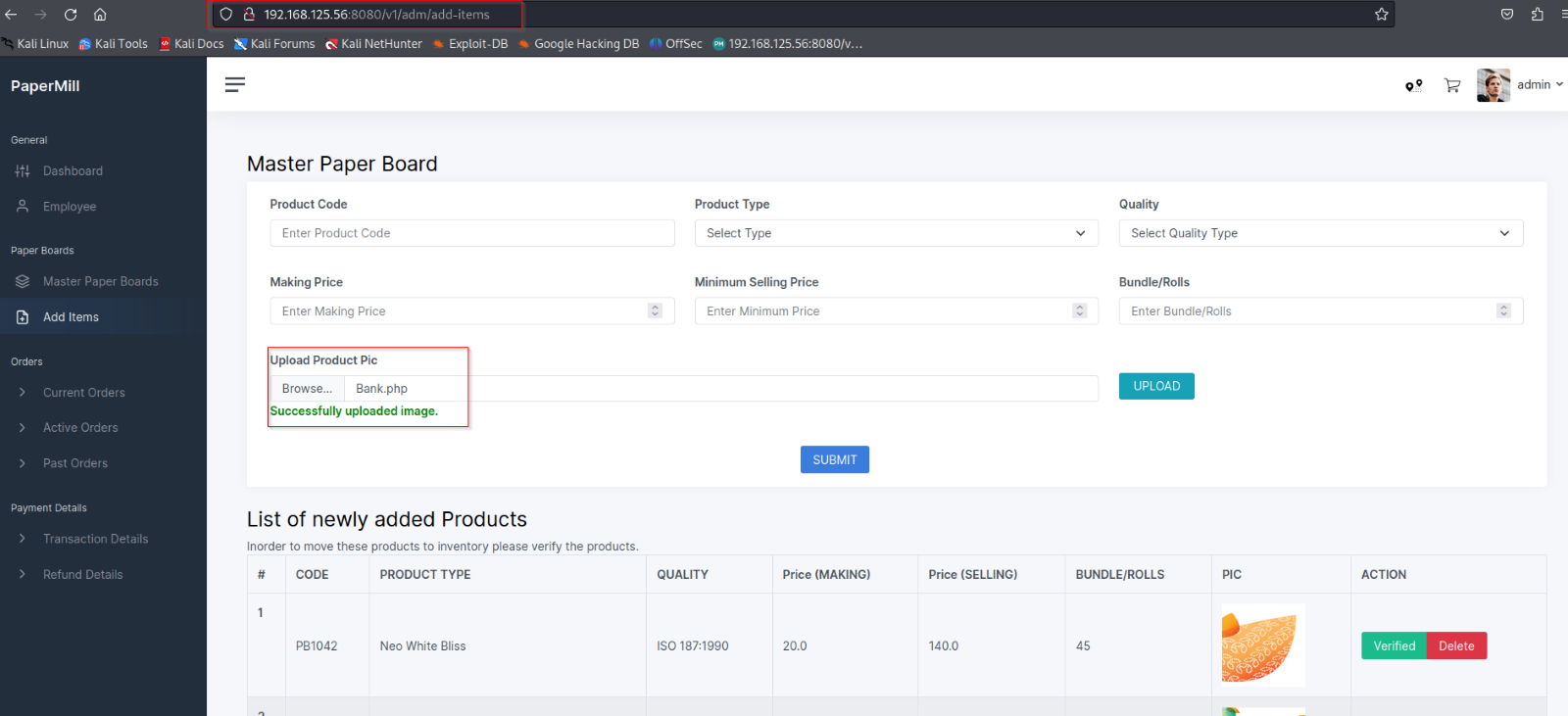
**Severity**

High

**Impact**

Malicious file upload can allow attackers to upload and execute malicious files on the server or on other users' systems. For example, an attacker may be able to upload a file containing malicious code, such as a virus or a backdoor, and then execute that code on the server or on other users' systems.

**Proof of Concept**



**Remediation**

Malicious file upload can allow attackers to upload and execute malicious files on the server or on other users' systems. For example, an attacker may be able to upload a file containing malicious code, such as a virus or a backdoor, and then execute that code on the server or on other users' systems.

1. **Brute Force Attack (weak Authentication)**

**Description**

A brute force vulnerability in the web application allows malicious actors to repeatedly and systematically guess usernames and/or passwords until they gain unauthorized access to an account or system. This vulnerability arises when there are insufficient security measures to limit login attempts, making it possible for attackers to automate the process of trying various combinations until they succeed.

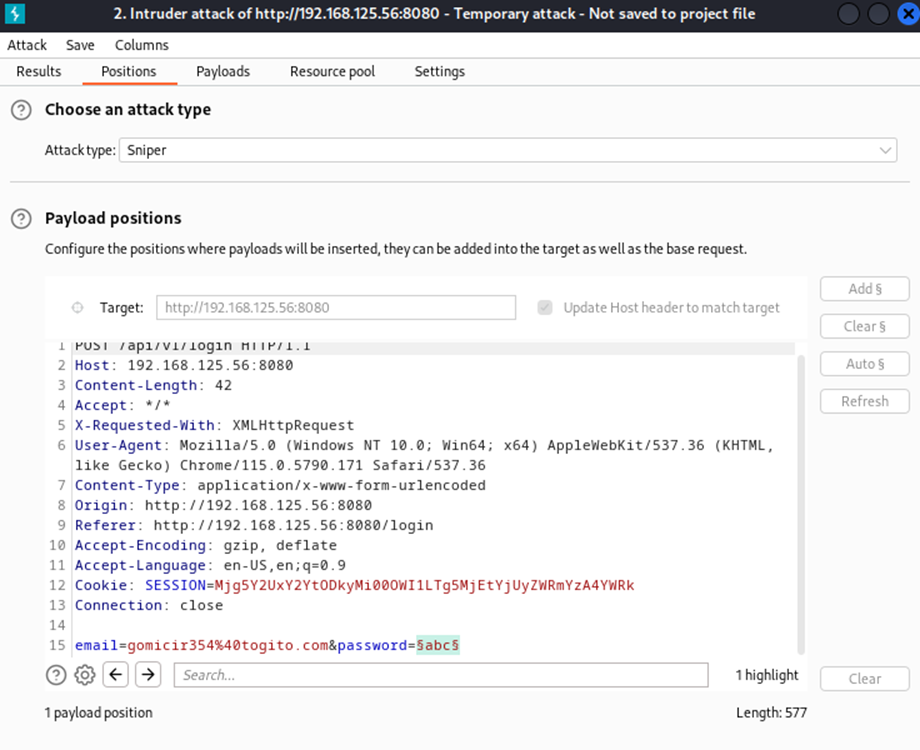
**Severity**

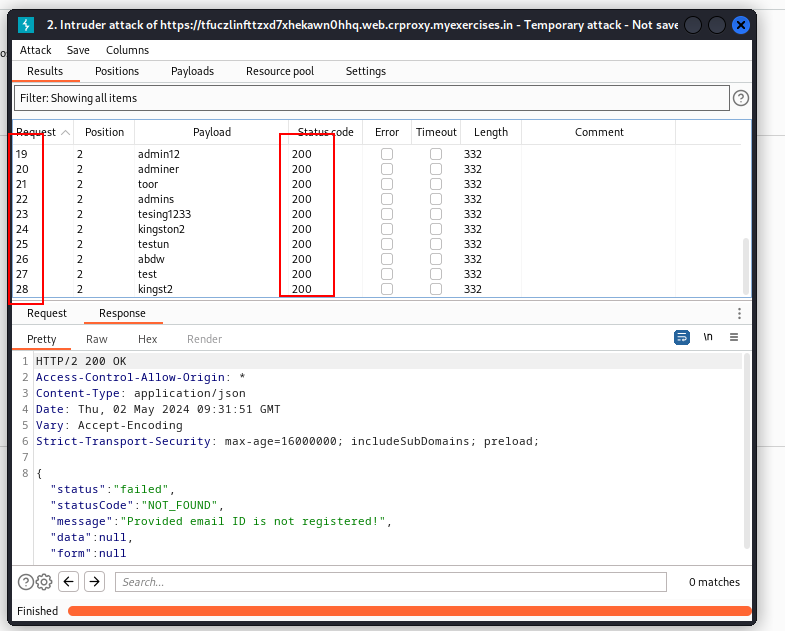
High

**Impact**

The impact of this vulnerability can be severe, potentially leading to unauthorized access to user accounts, sensitive data, or the web application itself. It could result in data breaches, compromised user privacy, or even complete system takeover, depending on the level of access the compromised account provides. This could lead to financial losses, reputational damage, and legal consequences.

**Proof of Concept**





**Remediation**

Implement account lockout policies that temporarily or permanently lock an account after a certain number of failed login attempts, making it difficult for attackers to keep guessing.

Implement account lockout policies that temporarily or permanently lock an account after a certain number of failed login attempts, making it difficult for attackers to keep guessing.

1. **Admin page bypass via URL change**

**Description**

This allows an attacker to login to the admin panel with a user of his choice, e.g the root user with highest privileges or even a non-existing user. An attacker needs to have network access to the admin interface.

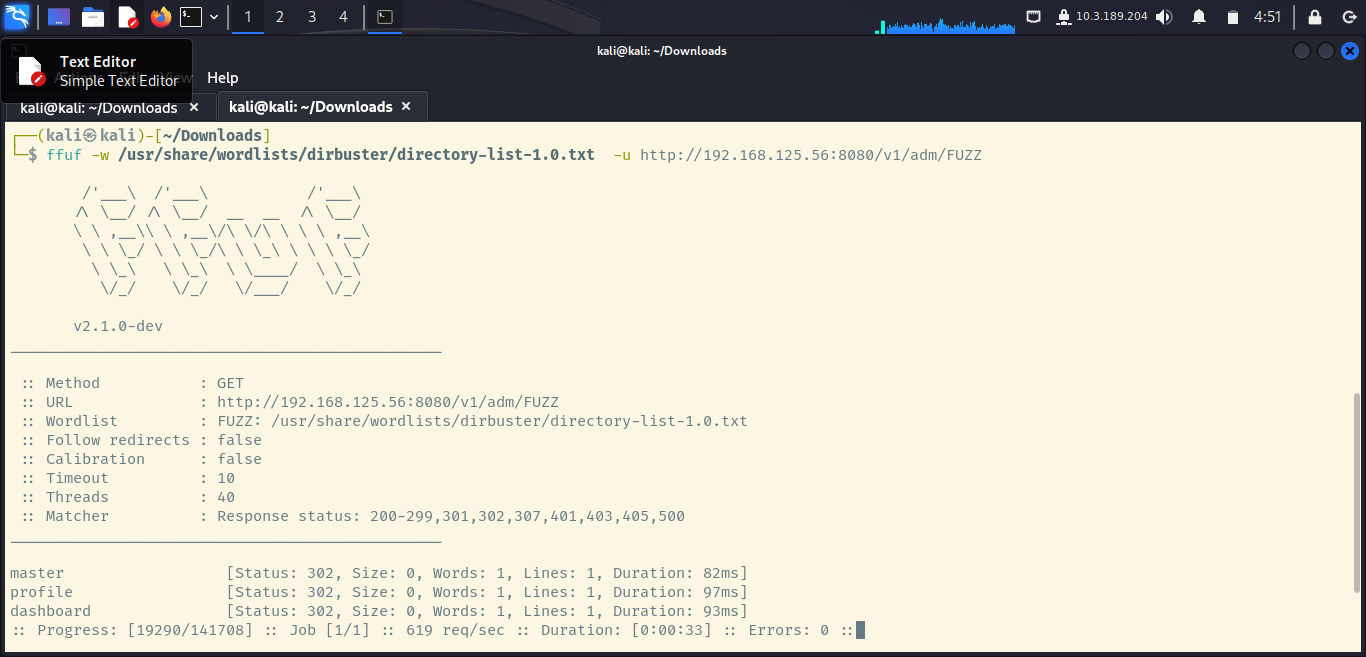
**Severity**

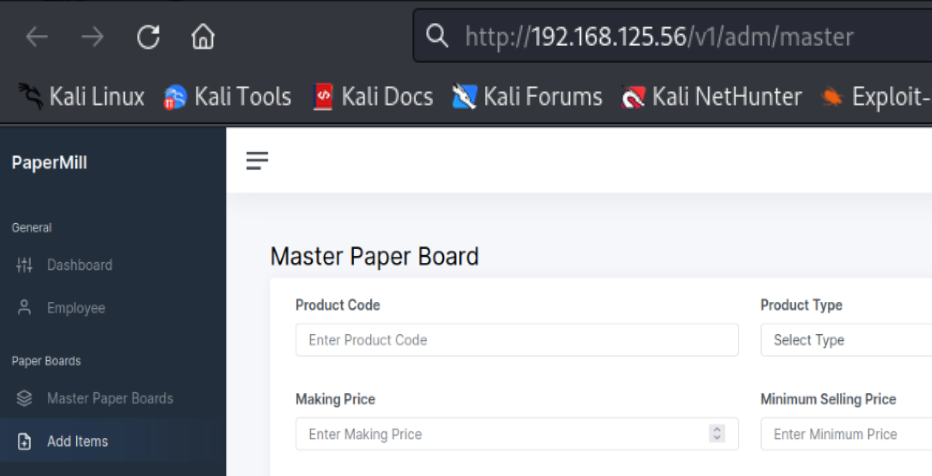
High

**Business Impact**

The impact of authentication vulnerabilities can be very severe. Once an attacker has either bypassed authentication or has brute-forced their way into another user's account, they have access to all the data and functionality that the compromised account has.

**Proof of Concept**





**Remediation**

Authentication bypass is a vulnerable point where criminals gain access to the application and get users' sensitive information. Authentication bypass vulnerability allows hackers to perform malicious activities by bypassing the authentication mechanism of the devices.

1. **HTML Injection**

**Description**

This is a type of web application security vulnerability that allows an attacker to inject malicious HTML code into a web application, which is then executed by unsuspecting users.

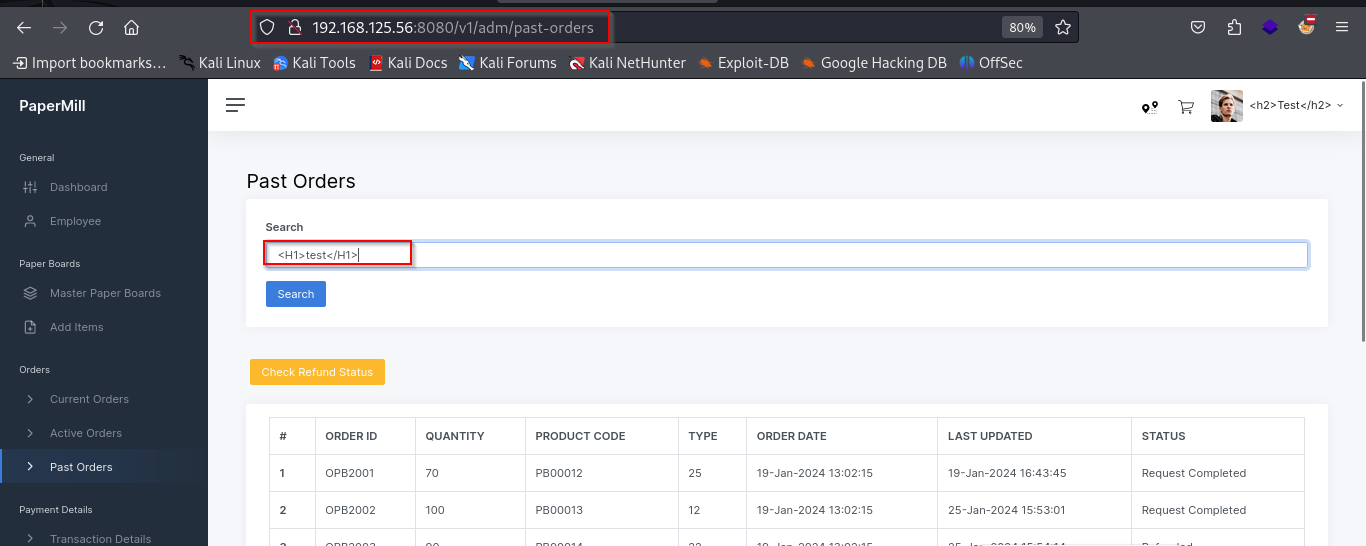
**Severity**

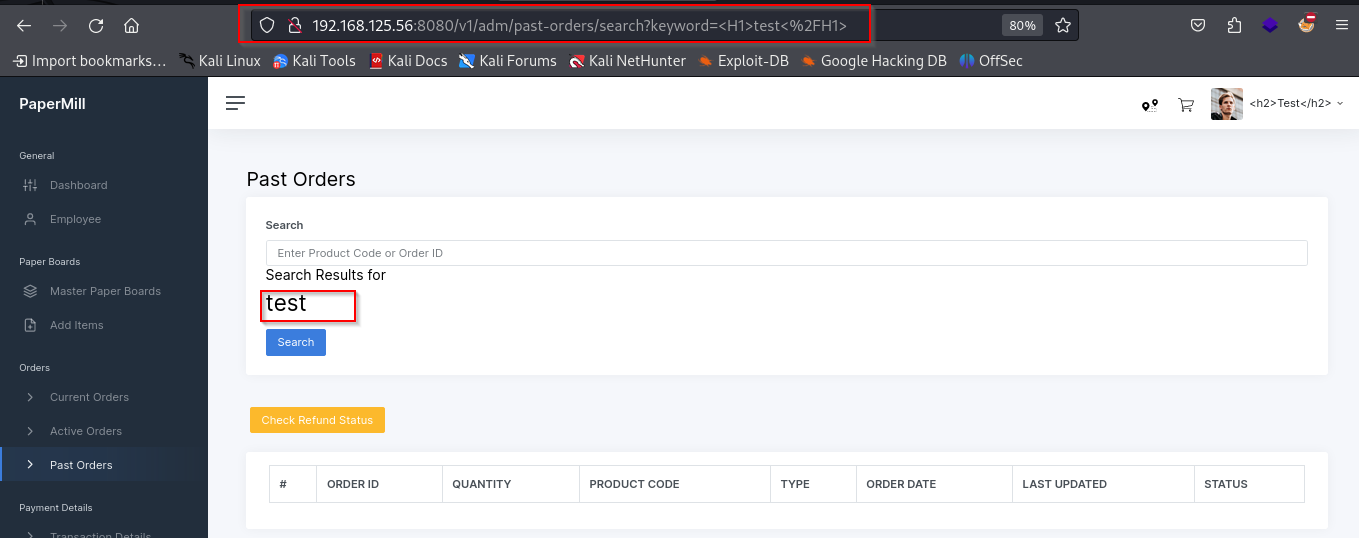
Medium

**Impact**

In an HTML Injection attack, an attacker injects malicious HTML code into a web application's input fields, such as a search box or login form. When other users access the web application and enter search queries or login credentials, the injected code is executed, potentially allowing the attacker to steal sensitive information or to carry out malicious actions on behalf of the user.

**Proof of Concept**





**Remediation**

To remediate HTML Injection, web developers can implement the following measures:

1. Use input validation and output encoding: Web developers should validate and sanitize all user input, such as form data or input fields, and encode output to prevent the execution of any injected HTML code.

2. Use secure development practices: Web developers should follow secure coding practices, such as using a web application framework or secure coding guidelines.

3. Use a Content Security Policy (CSP): Implementing CSP can help prevent HTML Injection attacks by allowing web developers to define a whitelist of trusted sources for scripts and other resources allowed to execute on their web application.

1. **Iframe injection**

**Description**

Iframe injection is a type of web application security vulnerability that allows an attacker to inject an iframe element into a web page, which can then be used to execute malicious code or steal sensitive information.

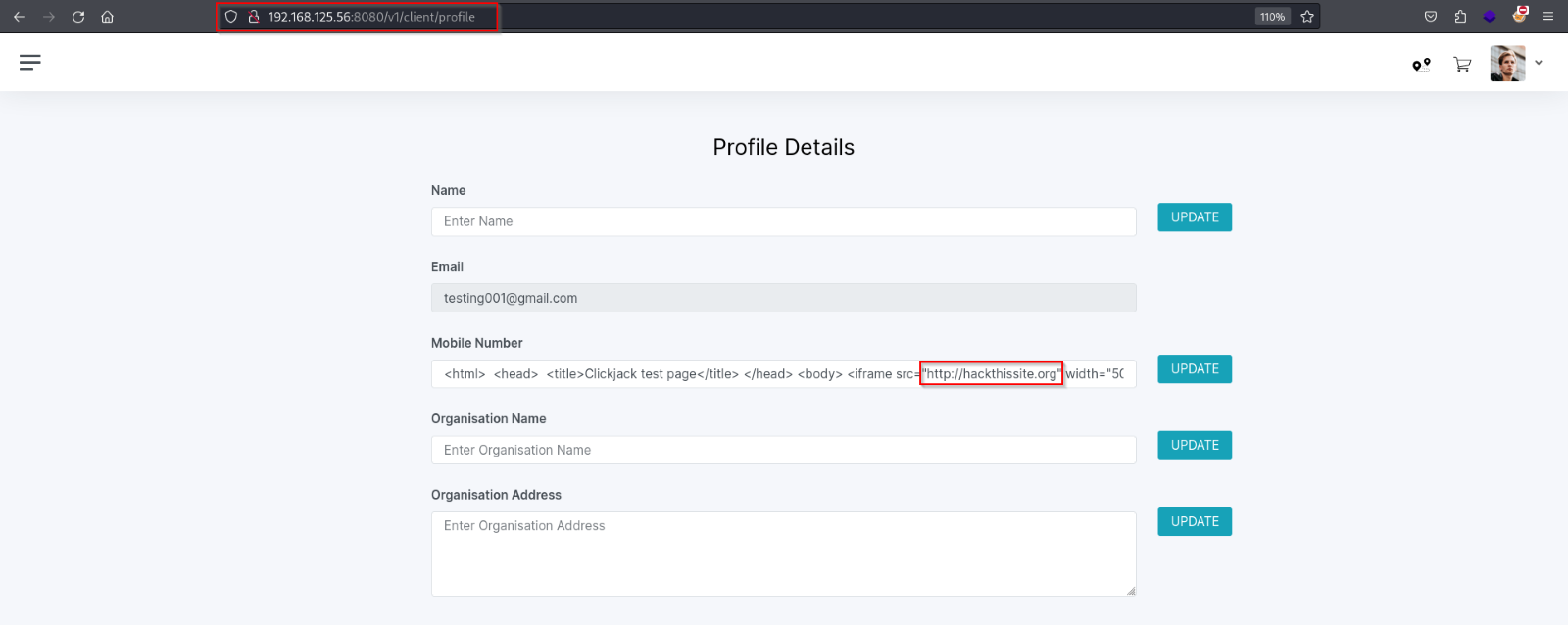
**Severity**

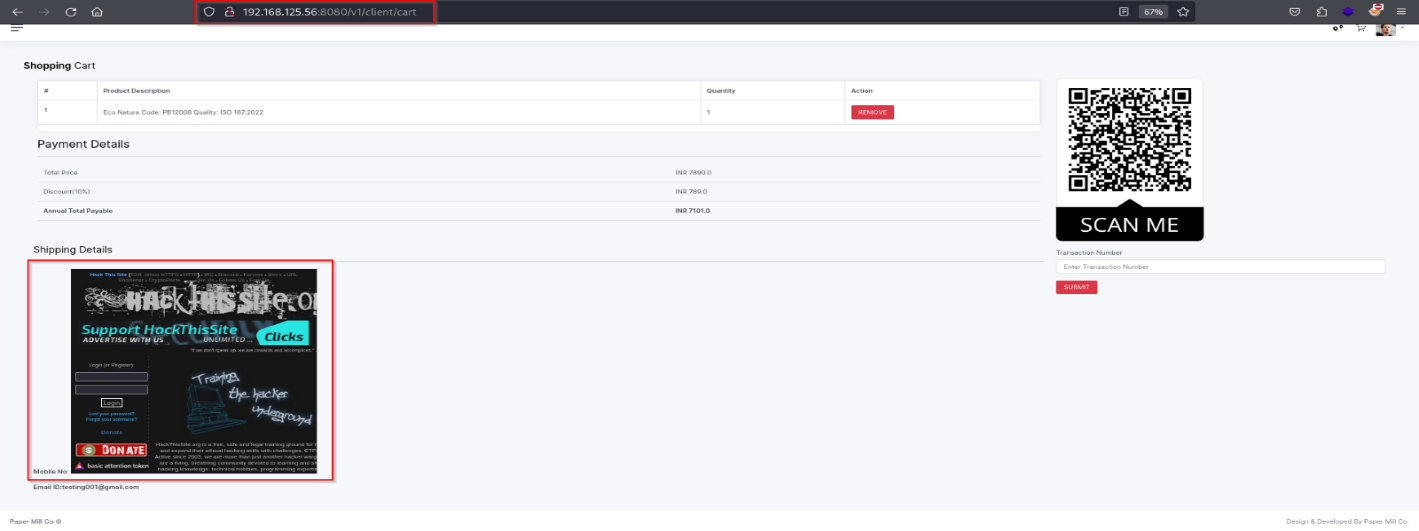
Medium

**Business Impact**

In an Iframe Injection attack, an attacker can inject an iframe element into a web page, often by exploiting vulnerabilities in the web application's code or by injecting code into user input fields. The iframe element can then be used to load content from a different website, often controlled by the attacker, and execute malicious code or steal sensitive information from the user.

**Proof of Concept**





Should use X-Frame-Opt which will automatically protect your website and it will definitely help in detecting and preventing against such attacks.

**Remediation**

To remediate Iframe injection, web developers can implement the following measures:

1. Code Analysis: Conduct a thorough review of your website's codebase to identify the source of the iframe injection.

2.Code Remediation: Remove or modify the injected iframe code from affected web pages to eliminate the vulnerability.

3.Security Hardening: Implement security measures such as input validation, output encoding, and access controls to prevent future iframe injections and enhance overall website security.

1. **Clickjacking**

**Description**

Clickjacking (User Interface redress attack, UI redress attack, UI redressing) is a malicious technique of tricking a Web user into clicking on something different from what the user perceives they are clicking on, thus potentially revealing confidential information or taking control of their computer while clicking on seemingly innocuous web pages.

The server didn't return an X-Frame-Options header which means that this website could be at risk of a clickjacking attack. The X-Frame-Options HTTP response header can be used to indicate whether a browser should be allowed to render a page inside a frame or iframe. Sites can use this to avoid clickjacking attacks, by ensuring that their content is not embedded into other sites.

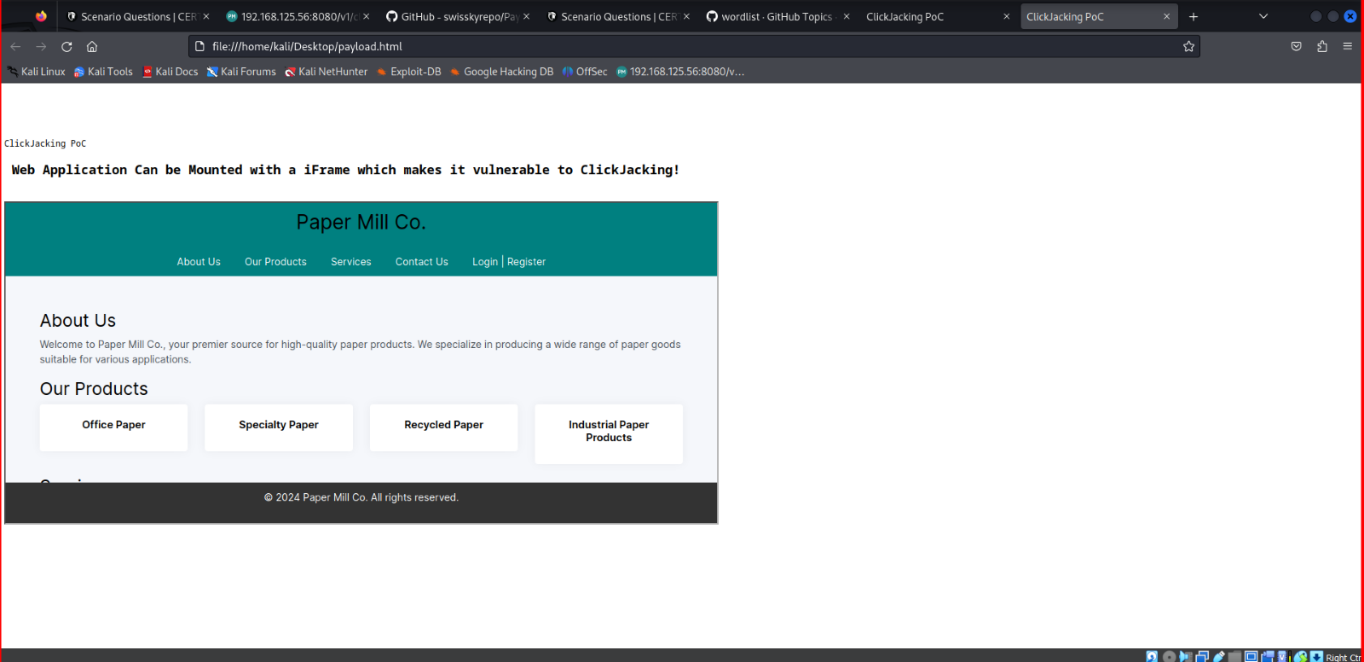
**Severity**

Medium

**Impact**

Clickjacking is when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on a framed 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 other another page, most likely owned by another application, domain, or both.

**Proof of Concept**

z

**Remediation**

Sending the proper X-Frame-Options in HTTP response headers that instruct the browser to not allow framing from other domains.  
X-Frame-Options: DENY It completely denies being loaded in frame/iframe.  
X-Frame-Options: SAMEORIGIN It allows only if the site which wants to load has a same origin.  
X-Frame-Options: ALLOW-FROM URL It grants a specific URL to load itself in an iframe. However please pay attention to that, not all browsers support this.  
Employing defensive code in the UI to ensure that the current frame is the most top-level window.Certainly! Here's a template you could use for reporting the unencrypted communication vulnerability:

1. **Unencrypted Communication**

**Description**

**Description:**

The unencrypted communication vulnerability refers to the absence of encryption protocols in data transmissions between endpoints. This means that sensitive information, such as login credentials, personal data, or financial details, is transmitted in plaintext, making it susceptible to interception by malicious actors. This vulnerability poses a significant security risk as it exposes sensitive data to eavesdropping and potential exploitation.

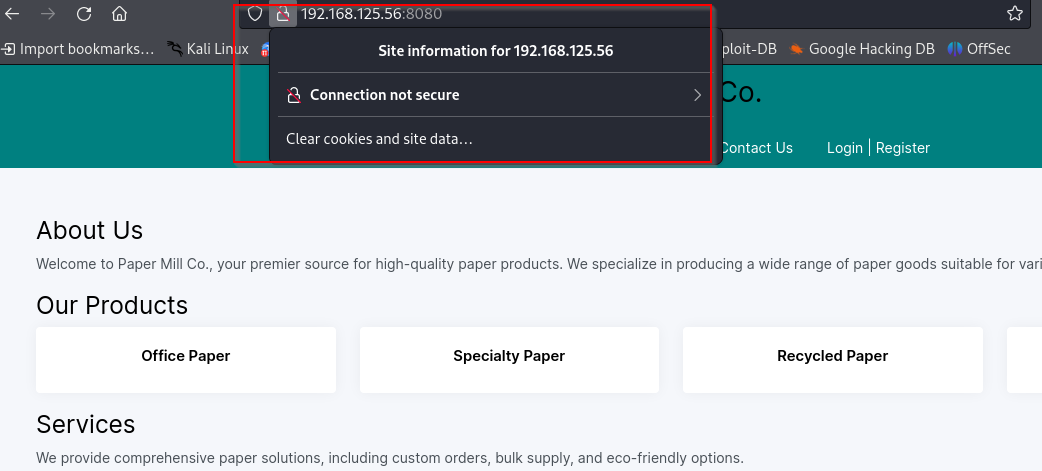
**Impact:**

Without encryption, sensitive data transmitted over the network can be intercepted and accessed by unauthorized parties.

Personal and confidential information of users or customers may be compromised, leading to privacy violations and potential legal repercussions.

Incidents of data breaches resulting from unencrypted communication can tarnish the organization's reputation and erode trust among stakeholders.

**Proof of Concept**



**Remediation:**

**I**mplement Transport Layer Security (TLS): Encrypt data transmissions using TLS protocols to secure communication channels between endpoints. Ensure that strong encryption algorithms and key exchange mechanisms are employed.

Enable HTTPS: Use HTTPS (HTTP over SSL/TLS) for web communications to encrypt data exchanged between clients and servers. Configure web servers to enforce HTTPS connections.

Encrypt Data at Rest: In addition to encrypting data in transit, consider implementing encryption mechanisms for data stored on servers or databases to provide end-to-end security.

1. **Concurrent Session**

**Description**

A vulnerability in concurrent sessions arises when a web application allows multiple sessions to be established for a single user account simultaneously, without proper restrictions or controls.

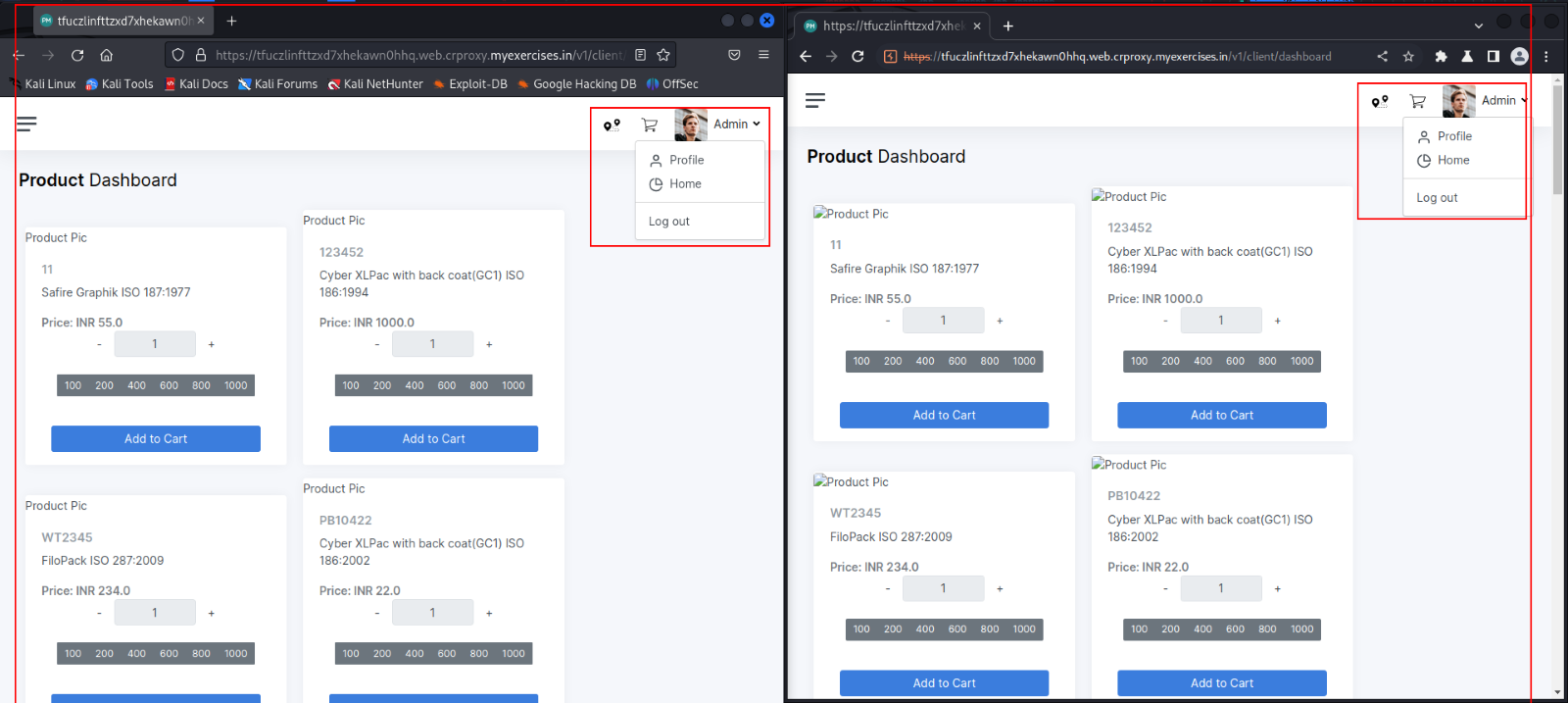
**Severity**

Medium

**Business Impact**

This can occur if an attacker uses stolen credentials to establish a new session while the legitimate user is still logged in.

**Proof of Concept**



**Remediation**

To mitigate these vulnerabilities, web developers can implement controls such as session tracking, session timeouts, and limiting the number of concurrent sessions per user account

1. **Improper Input validation**

his vulnerability occurs when a web application does not properly validate input from users or other sources, such as APIs or file uploads, before using that input to generate dynamic content or interact with back-end systems.

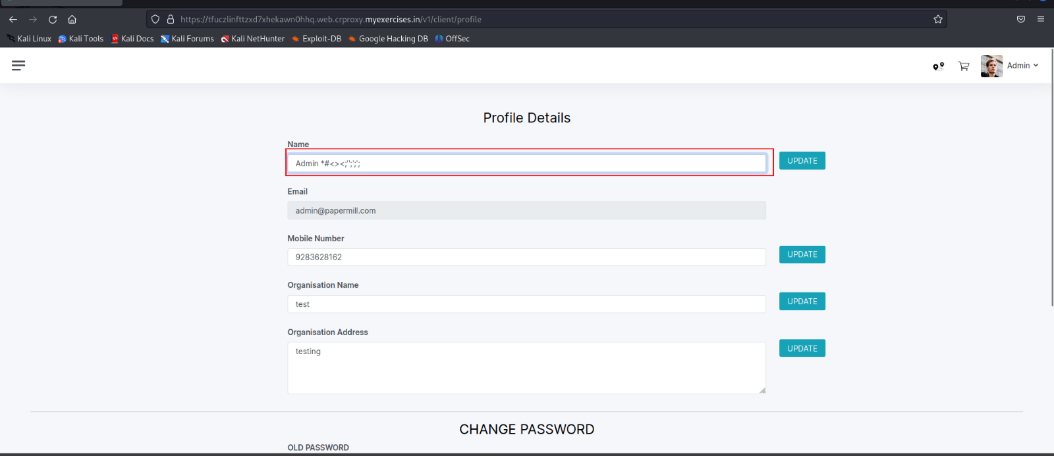
**Severity**

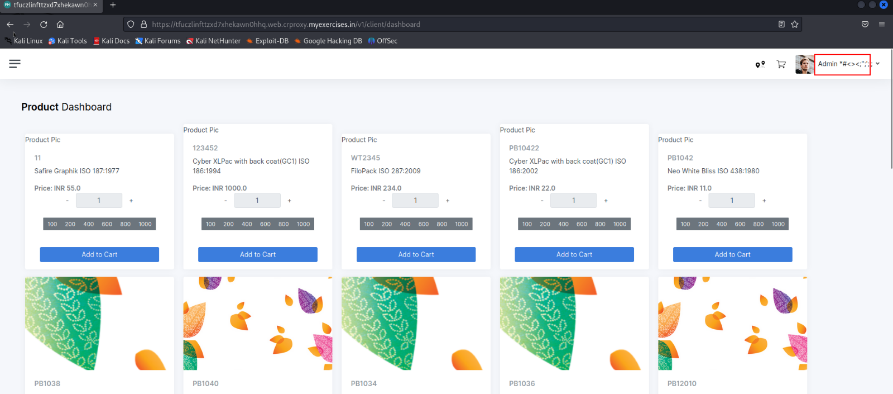
Low

**Impact**

This can allow attackers to inject malicious code or data, such as SQL injection, cross-site scripting (XSS), or buffer overflow attacks, into the web application and potentially compromise the security of the system.

**Proof of Concept**





**Remediation**

To mitigate this vulnerability, web developers must implement input validation and sanitization mechanisms that verify the accuracy, validity, and security of user input. This can be accomplished through regular expression patterns, white-listing or black-listing of input data, and validation checks for data type, length, and format.

1. **No Captcha Implementation**

**Description**

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is a security mechanism used to prevent automated scripts from submitting forms or performing other actions on a web application.

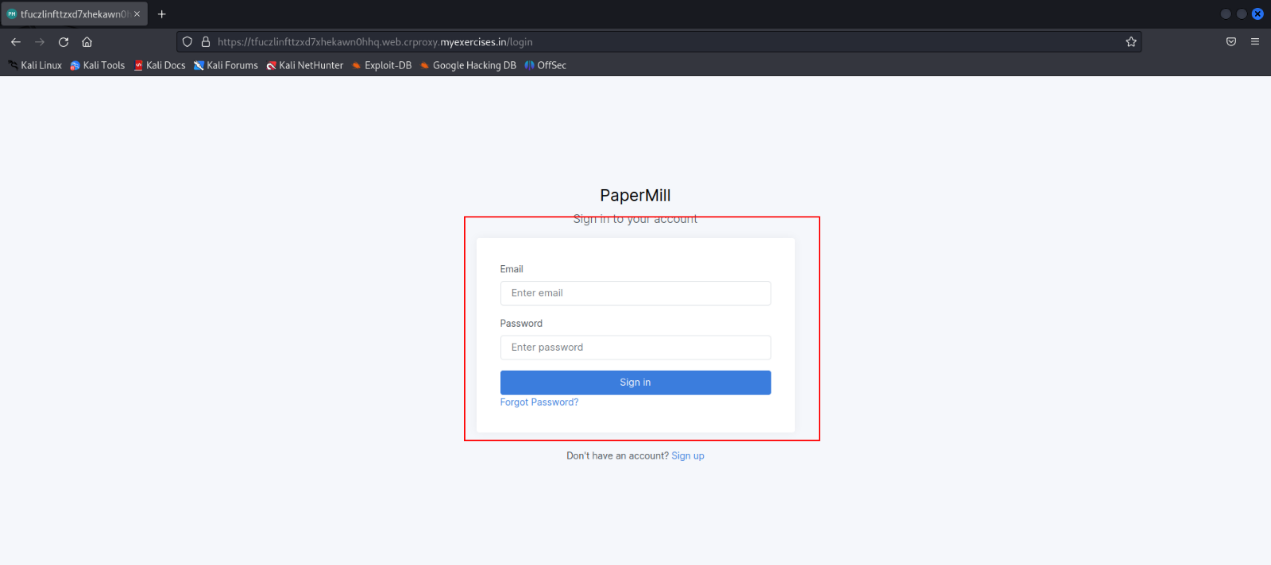
**Severity**

Medium

**Impact**

Without a CAPTCHA mechanism, attackers can use automated scripts or bots to submit forms or perform other actions on a web application, such as creating multiple accounts or spamming the system with unwanted content. This can lead to a range of security issues, such as account takeover, denial-of-service attacks, or data theft.

**Proof of Concept**



**Remediation**

To mitigate this vulnerability, web developers should implement a CAPTCHA mechanism to verify that the user is a human and not a bot. There are several types of CAPTCHA mechanisms available, such as image-based CAPTCHAs, audio-based CAPTCHAs, or math-based CAPTCHAs

**13. Sensitive Information disclosure**

**Description**

The identified vulnerability is a Sensitive Information Disclosure issue, which occurs when a web application unintentionally exposes sensitive data to unauthorized users. This disclosure can happen due to various reasons, such as inadequate access controls, improper error handling, or misconfigured security settings within the application.

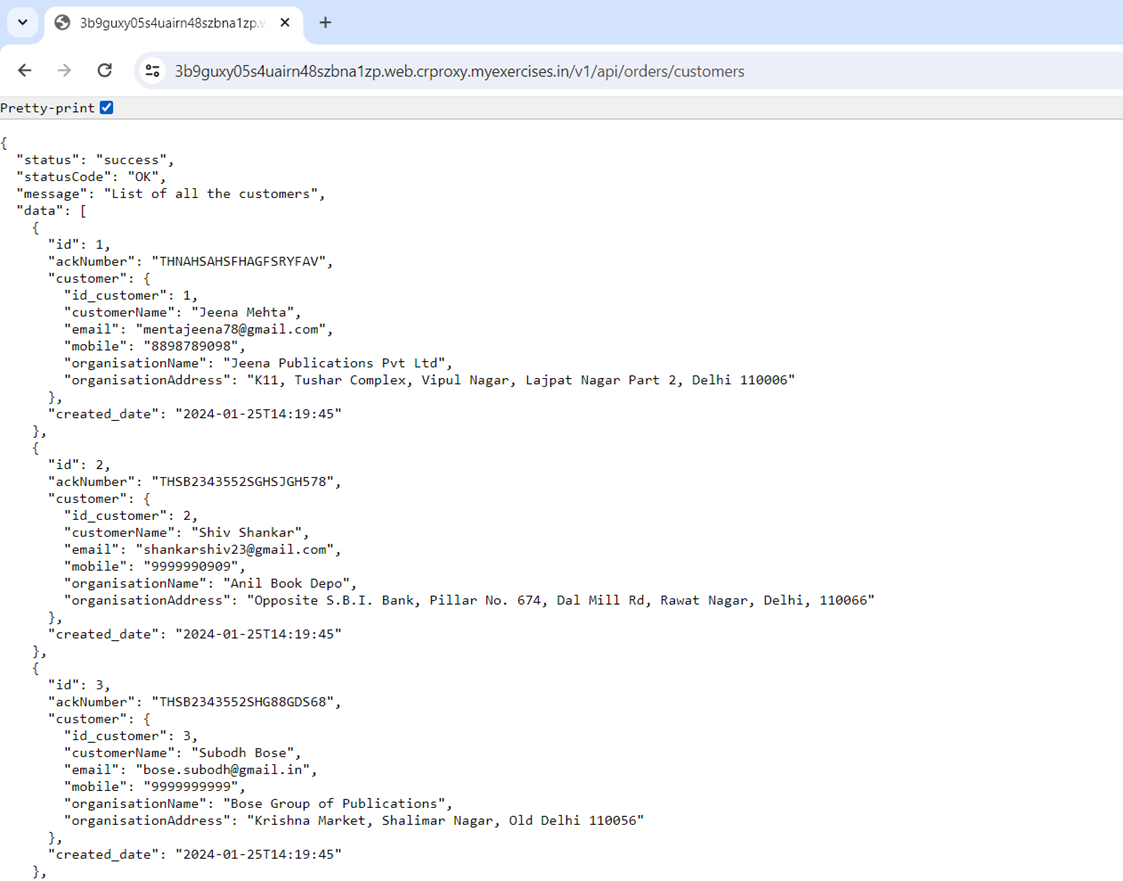
**Severity**

Medium

**Impact**

Unauthorized users can gain access to sensitive information, including personal data, credentials, or confidential documents.

**Proof of Concept**



**Remediation**

Encrypt sensitive data both at rest and in transit using strong encryption algorithms and protocols.

implement strict input validation and sanitize user inputs to prevent data leakage through injection attacks.

Use security headers like Content Security Policy (CSP) and X-Content-Type-Options to control content rendering and prevent information leakage.

**14. Numerous Ports Open**

**Description**

Open ports are used by applications and services and, as any piece of code, they may have vulnerabilities or bugs. The more applications and services run using open ports for Internet communication, the higher the risk of one of them having a vulnerability that can be exploited.

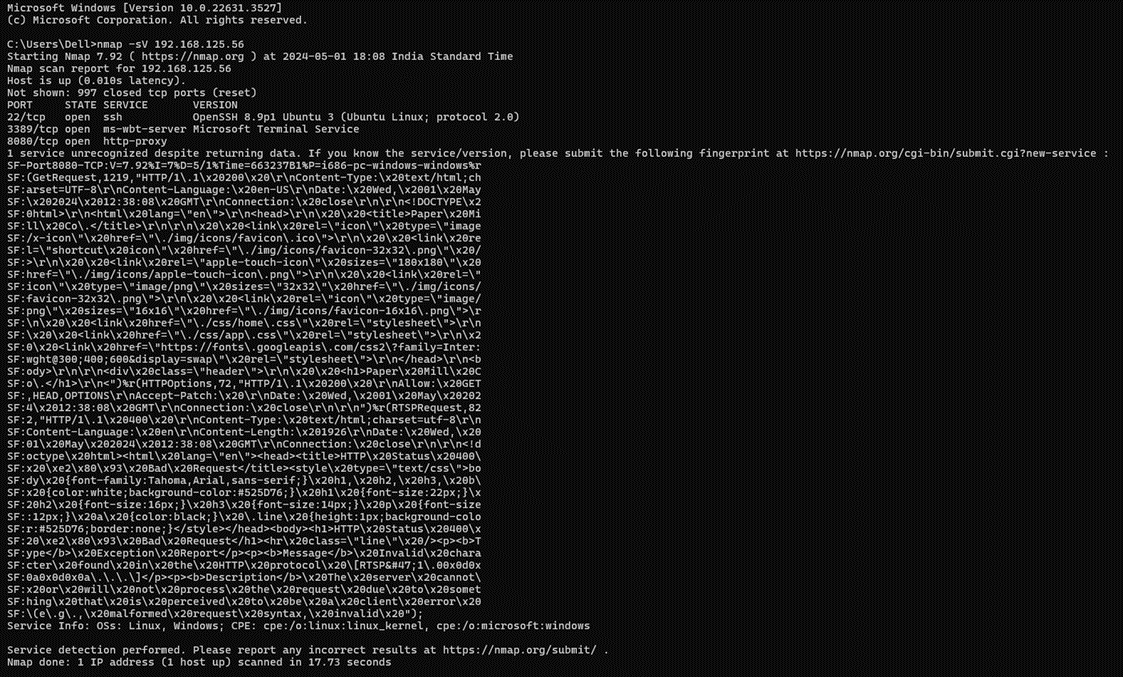
**Severity**

Medium

**Impact**

An attacker can leverage this information when trying an attack.

**Proof of Concept**



**Remediation**

Ports should be filtered, and banners should be removed/generalized

**15. Cacheable HTTP Response**

**Description**

In the context of web application security, cacheable HTTP response refers to the caching of HTTP

Responses by web browsers or intermediary devices such as proxy servers, to improve web application performance. However, this can also lead to security vulnerabilities if sensitive information is cached and then later retrieved by unauthorized users.

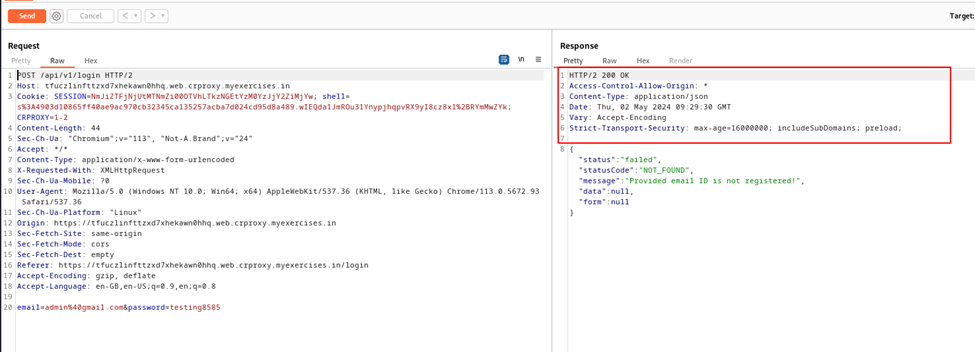
**Severity**

Medium

**Impact**

This risk refers to the possibility of sensitive information, such as user authentication credentials, being cached in a way that could be later retrieved by an attacker.

**Proof of Concept**



**Remediation**

To mitigate this risk, web developers can use HTTP headers to control caching behavior and ensure

that sensitive information is not cached or is cached only for a limited time. For example, the Cache-Control header can be used to specify that a response should not be cached, or that it should be cached only for a limited time. Additionally, web developers can use secure session management practices to prevent sensitive data from being stored in the cache in the first place.

Set cache-control header value to “ no-cache, no-store, must revalidate

**16. Improper Error Handling**

**Description**

The improper error handling vulnerability occurs when a program fails to handle errors or exceptions properly during its execution. This could manifest in various ways, such as not providing adequate error messages to users, leaking sensitive information in error messages, or failing to log errors for further analysis.

**Severity**

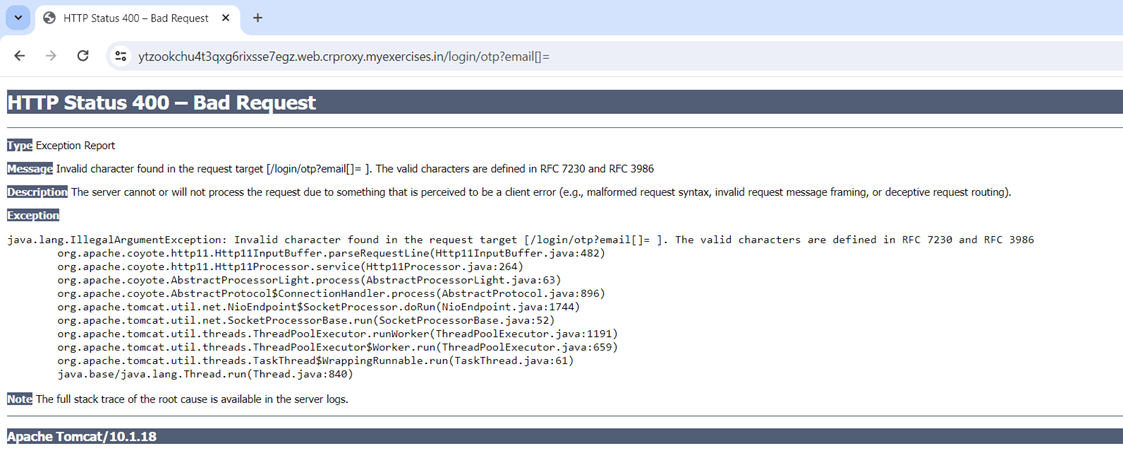
Medium

**Impact**

The impact of improper error handling can be significant. Without proper error handling mechanisms in place, attackers may exploit these vulnerabilities to gather sensitive information about the system or its users, leading to potential data breaches or unauthorized access. Additionally, the lack of error logging could impede the detection and mitigation of security incidents, prolonging the exposure of vulnerabilities.

**Proof of Concept**





**Remediation**

Custom Error Pages: Develop custom error pages that provide concise and informative messages to users without divulging sensitive system details. These pages should guide users on how to proceed in case of errors and avoid exposing internal system workings. Input Validation: Strengthen input validation mechanisms to prevent malformed or malicious inputs from triggering unexpected errors. Validate all user-supplied data before processing it to reduce the risk of exploitation

**17. Insecure Content-type**

**Description**

When a web server receives a request, it typically examines the Content-Type header to determine the type of data being sent, such as text, images, or JSON. If this header is manipulated or forged by an attacker, it can trick the server into processing the request in unexpected ways. For example, an attacker could send a request with a Content-Type of "application/json" but include malicious HTML or JavaScript content, leading to Cross-Site Scripting (XSS) attacks.

**Severity**

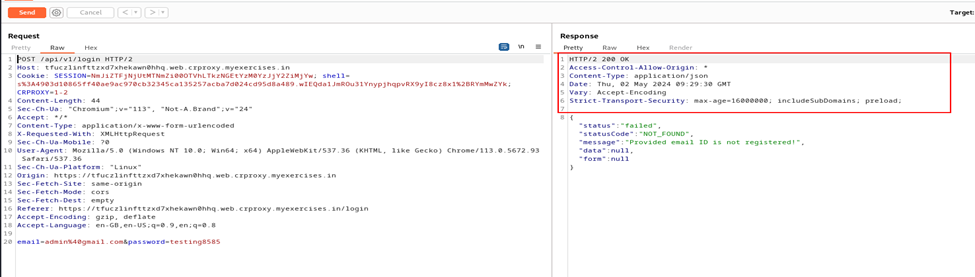
Medium

**Impact**

Insecure Content-Type headers in web applications can lead to severe security risks. They pave the way for attacks like Cross-Site Scripting (XSS), enabling malicious scripts to infiltrate web pages and compromise user data. Additionally, attackers can exploit these vulnerabilities for content spoofing, tricking systems into treating harmful content as benign. Moreover, insecure handling of Content-Type headers can facilitate request smuggling, circumventing security measures and escalating the impact of

attacks. Mitigation strategies such as stringent validation and regular security audits are crucial for safeguarding against these threats and bolstering overall application security.

**Proof of concept**



**Remediation**

To remediate insecure Content-Type headers, begin by implementing thorough validation and sanitization protocols for incoming data. Employ Content Security Policy (CSP) to restrict content sources and bolster defense mechanisms. Utilize trusted libraries and frameworks equipped with built-in security features. Conduct regular security audits and penetration testing to identify and address vulnerabilities promptly. Additionally, prioritize developer education on secure coding practices to foster a culture of vigilance. Collaborate with security experts to establish comprehensive remediation plans tailored to specific vulnerabilities. Finally, ensure swift and effective patching of any identified vulnerabilities to minimize exposure to potential threats.

**18. Missing CSP Header**

**Description**

The Content Security Policy (CSP) header is a crucial security mechanism that allows web developers to mitigate various types of attacks, such as Cross-Site Scripting (XSS) and data injection attacks. It works by defining a whitelist of trusted sources for content loading and execution, including scripts, stylesheets, and other resources. When a web application fails to include a CSP header or includes an incomplete or ineffective one, it becomes vulnerable to a range of security threats.

**Severity**

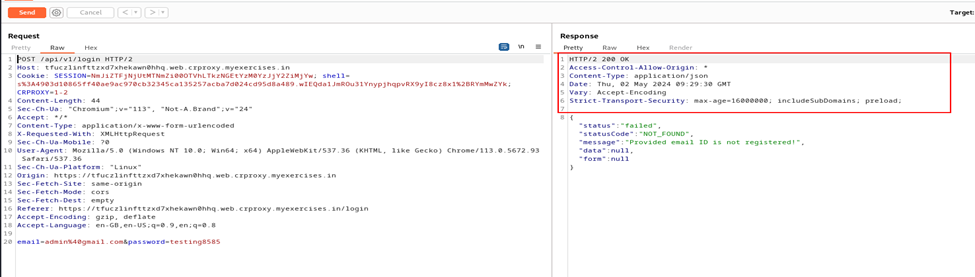
Low

**Impact**

The absence or inadequacy of a Content Security Policy (CSP) header in a web application poses significant

security risks. Without CSP, the application is vulnerable to Cross-Site Scripting (XSS), enabling attackers to execute malicious scripts and compromise user data. Additionally, it opens the door to data injection attacks, allowing for the manipulation of content to steal information. Furthermore, without CSP protection, clickjacking attacks can exploit user interactions for unintended actions. Implementing a robust CSP header is crucial for fortifying security and safeguarding against these threats.

**Proof of Concept**



**Remediation**

To remediate the absence or inadequacy of a Content Security Policy (CSP) header, implement a well-configured CSP header with strict policies to restrict content loading. Minimize the use of 'unsafe-inline' and 'unsafe-eval' keywords and enable CSP violation reporting for monitoring. Thoroughly test CSP policies across browsers, stay updated on threats, and educate staff on CSP importance. These steps are crucial for enhancing security and mitigating risks associated with CSP header deficiencies.

**19. Missing same-site in cookies**

**Description**

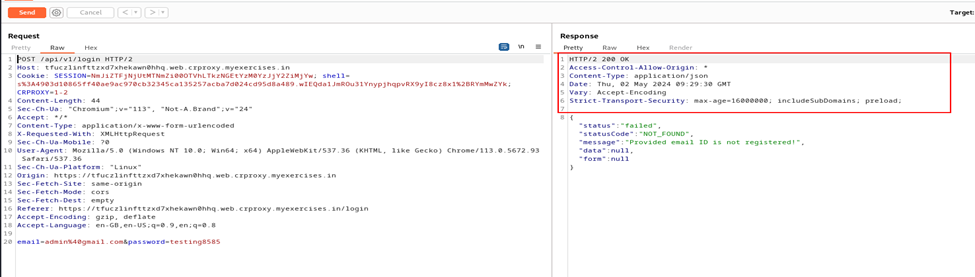
The "SameSite" attribute in cookies is a critical security feature that specifies whether cookies should be restricted to same-site or cross-site requests. It helps mitigate Cross-Site Request Forgery (CSRF) and Cross-Site Script Inclusion (XSSI) attacks by controlling the behavior of cookies across different origins. However, if cookies are issued without the "SameSite" attribute or with an improper configuration, it can expose web applications to various security risks.

**Severity**

**Impact**

Cross-Site Request Forgery (CSRF) occurs when a malicious website tricks a user's browser into making an unintended request to a different website where the user is authenticated. If the request involves a state-changing action (like transferring money or changing account settings), it can have serious consequences.

**Proof of Concept**



**Remediation**

To remediate the absence of the SameSite attribute in cookies, first, implement SameSite attribute for cookies, choosing between "Strict," "Lax," or "None" based on needs. Upgrade dependencies to support SameSite attribute configuration, then thoroughly test for functionality and compatibility. Establish continuous monitoring for CSRF attempts and unexpected cookie behavior. Educate the development team on CSRF protection and SameSite attribute usage. Ensure all cookies are configured with SameSite attribute to mitigate CSRF risk. Following these steps rigorously enhances web application security against CSRF attacks.

Thorough testing should follow to ensure that enabling the HttpOnly flag does not interfere with the functionality of the web application. Regression testing can help verify that all features continue to operate as expected post-implementation.

**21. Internal IP Disclosure**

**Description**

Internal IP disclosure occurs when a web application inadvertently reveals internal IP addresses to external users. These internal IPs are meant for internal network communication and should not be exposed to the public internet. This vulnerability can occur due to various factors such as misconfigured servers, improper error handling, or insufficient input validation.

**Severity**

Low

**Impact**

The exposure of internal IP addresses can provide attackers with valuable information about the organization's network infrastructure, facilitating reconnaissance for potential cyber attacks. Armed with internal IP addresses, attackers can map out network topology, identify vulnerable systems, and launch targeted attacks, including lateral movement within the network. This information disclosure increases the organization's susceptibility to various threats, including infiltration, data exfiltration, and service disruption.

**Proof of Concept**



**Remediation**

Remediating internal IP disclosure involves several key steps to mitigate risks and enhance security. Firstly, thoroughly review and update server configurations to ensure internal IP addresses are not inadvertently exposed in HTTP responses or error messages. Implement robust input validation and output encoding mechanisms to sanitize user inputs and prevent injection attacks. Improve error handling to provide generic error messages without divulging internal details. Consider implementing network segmentation to restrict access to internal resources and prevent direct exposure of internal IP addresses. Conduct regular security assessments, including penetration testing, to identify and remediate vulnerabilities proactively. Lastly, educate employees and developers on the risks associated with internal IP disclosure and promote best practices for secure configuration and development. By diligently following these steps, organizations can strengthen their defenses against potential threats and safeguard sensitive information.

# Appendices

<This section should include evidences and additional information that supports the main body of the report, such as detailed tables, figures, or descriptions of audit tools and techniques used and glossary of terms for clarity.>