PENTESTING REPORT

ON

https://roomeasy.net

Carried out at

CDAC Software Training and Development Centre Thiruvananthapuram

Under the Guidance of

Jayaram P Sir

Senior Cyber Security Specialist at CDAC

Thiruvananthapuram, Kerala, India

Submitted by

PUSHKAR ZEND

PRN:- 240860940032

GANESH SHELKE

PRN:- 240860940017

CONTENTS

EXECUTIVE SUMMARY

1.1 Summary:-

This penetration testing project was conducted on the website https://roomeasy.net, a platform that provides online room booking services. The goal of this assessment was to identify potential security vulnerabilities and risks within the website's infrastructure, applications, and services. The pentest aimed to evaluate the website's resilience against common attack vectors, such as SQL injection, Cross-Site Scripting (XSS), authentication flaws, and insecure configurations. A comprehensive analysis was performed to assess both the frontend and backend of the site, ensuring the integrity and confidentiality of user data and the overall system security.

1.2 Objective

The primary objective of this penetration testing project is to identify security weaknesses and provide actionable recommendations for improving the overall security posture of https://roomeasy.net. This includes:

- Evaluating the website for vulnerabilities like SQL injections, XSS, CSRF, and others.
- Assessing the strength of authentication mechanisms and access control policies.
- Conducting a thorough review of the web server configurations, SSL/TLS security, and other key components.
- Simulating real-world cyber-attacks to determine the potential impact on the system's confidentiality, integrity, and availability.
-

1.3 Scope

The scope of this pentest includes:

- Testing all publicly accessible parts of *roomeasy.net*, including the booking pages, user registration and login sections, and other web application features.
- Identifying security gaps in user authentication, data submission, and input validation processes.
- Reviewing the website's server-side security measures, such as encryption protocols and server configuration files.
- Conducting vulnerability assessments on external facing services like APIs and web servers.

- Providing recommendations on patching identified vulnerabilities and improving the overall security of the website.

Exclusions:

- Any testing that involves physical security or internal infrastructure that is not accessible remotely.
- Tests outside the scope of web application security.

METHODOLOGY

1.Web tools:-

1) **Website Informer**:- https://website.informer.com
2) **DNS DUMPSTER :- https://dnsdumpster.com/**

2.NMAP

3. Nikto Scan Report

4. Burp Suite

VULNERABILITIES

Sl.NO	NAME OF THE VULNERABILITY	RECOMMENDATION	RISK LEVEL
1.	Sensitive Information in Cleartext	Encrypt all sensitive data in transit using secure protocols like HTTPS and enforce strong encryption algorithms to prevent exposure in cleartext.	**HIGH**
2.	Insufficient Brute Force Protection	Implement account lockout mechanisms and CAPTCHA challenges after a set number of failed login attempts to mitigate brute force attacks.	**HIGH**
3.	File Upload	Files should be thoroughly scanned and validated before being made available to other users. If validation fails then, the file should be discarded.	**HIGH**
4.	Shared Credentials	Implement strict access control policies, enforce unique login credentials for each user, and regularly audit accounts to prevent shared credential vulnerabilities.	**HIGH**
5.	Clickjacking	Implement the X-Frame-Options header with the value DENY or SAMEORIGIN, or use the Content-Security-Policy (CSP) frame-ancestors directive to prevent clickjacking attacks.	**MEDIUM**
6.	X-Content-Type-Options	Set the X-Content-Type-Options header to nosniff to prevent browsers from interpreting files as a different MIME type	**MEDIUM**
7.	Weak Input Validation	Implement strict server-side validation and sanitization for all user inputs to ensure only expected and safe data is processed.	**MEDIUM**
8.	Exposed or Improperly managed API key	Ensure API keys are securely stored, never exposed in client-side code, and rotate them regularly to minimize the risk of unauthorized access.	**HIGH**
9.	Insecure Local Storage	Avoid storing sensitive data in local storage and use secure, encrypted storage mechanisms instead.	**MEDIUM**

TECHNICAL REPORT

1.Web tools

1. https://website.informer.com

   Using this tool, get to know that, The website is hosted by **Vercel.inc**, with the following Information.

Network and WHOIS Information Report

Network Information:

- **Hosting Company:** Vercel, Inc
- **IP Address:** 76.76.21.21
- **DNS Servers:**
  - ns1.dns-parking.com
  - ns2.dns-parking.com

IP Address Range:

- **NetRange:** 76.76.21.0 - 76.76.21.255
- **CIDR:** 76.76.21.0/24
- **NetName:** VERCEL-01
- **NetHandle:** NET-76-76-21-0-1
- **Parent Network:** NET76 (NET-76-0-0-0-0)
- **NetType:** Direct Allocation

Ownership and WHOIS Details:

- **Domain Name:** roomeasy.net
- **Registry Domain ID:** 2911705564_DOMAIN_NET-VRSN
- **Registrar:** Hostinger Operations, UAB
- **Registrar URL:** Hostinger
- **Created Date:** 2024-08-27
- **Expiration Date:** 2025-08-27

- **Updated Date:** 2024-10-27
- **Registrar WHOIS Server:** whois.hostinger.com
- **Owner (Privacy Protect, LLC):** PrivacyProtect.org
- **Owner Contact Email:** contact@privacyprotect.org

Proof of Concept

Network View All

ADDRESSING DETAILS

Hosting Company
Vercel, Inc

IPs
76.76.21.21

DNS
ns1.dns-parking.com
ns2.dns-parking.com

IP DETAILS

NetRange
76.76.21.0 - 76.76.21.255

CIDR
76.76.21.0/24

NetName
VERCEL-01

NetHandle
NET-76-76-21-0-1

Parent
NET76 (NET-76-0-0-0-0)

NetType
Direct Allocation

OriginAS

Organization
Vercel, Inc (ZEITI)

Whois View All

OWNERSHIP

Created
2024-08-27

Expires
2025-08-27

Owner
Domain Admin (Privacy Protect, LLC (Privacy Protect.org))

Registar
HOSTINGER operations, UAB

Owner Emails
contact@privacyprotect.org

WHOIS INFORMATION

Domain Name
ROOMEASY.NET

Registry Domain ID
2911705564_DOMAIN_NET-VRSN

Registrar WHOIS Server
whois.hostinger.com

Registrar URL
https://www.hostinger.com

Updated Date
2024-10-27T02:17:40Z

Creation Date
2024-08-27T18:37:18Z

Registrar Registration Expiration Date
2025-08-27T18:37:18Z

Registrar
Hostinger Operations, UAB

2. **DNS DUMPSTER :-** https://dnsdumpster.com/
   DNSDumpster.com is a FREE domain research tool that can discover hosts related to a domain. Finding visible hosts from the attackers perspective is an important part of the security assessment process.

2.NMAP Scan :-

Here we performed the NMAP scan to find the open ports and services.
We found only two ports are open and they are not vulnerable.

```
┌──(root㉿kaliPractice)-[/home/pushkarzend]
└─# nmap -sS -sV -p- 76.76.21.21
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-18 22:31 IST
Nmap scan report for 76.76.21.21
Host is up (0.014s latency).
Not shown: 65533 filtered tcp ports (no-response)
PORT    STATE SERVICE   VERSION
80/tcp  open  http      Vercel
443/tcp open  ssl/https Vercel
2 services unrecognized despite returning data. If you know the service/version, p
t.cgi?new-service :
```

3.Nikto Scan Report:-

Target Information:

- **Target IP**: 76.76.21.21
- **Target Hostname**: 76.76.21.21
- **Target Port**: 80
- **Scan Start Time**: 2024-11-18 22:48:27 (GMT+5.5)
- **Scan End Time**: 2024-11-18 22:51:27 (GMT+5.5)
- **Total Requests Made**: 8047

- **Errors Reported**: 1 error(s)
- **Issues Found**: 10 item(s)

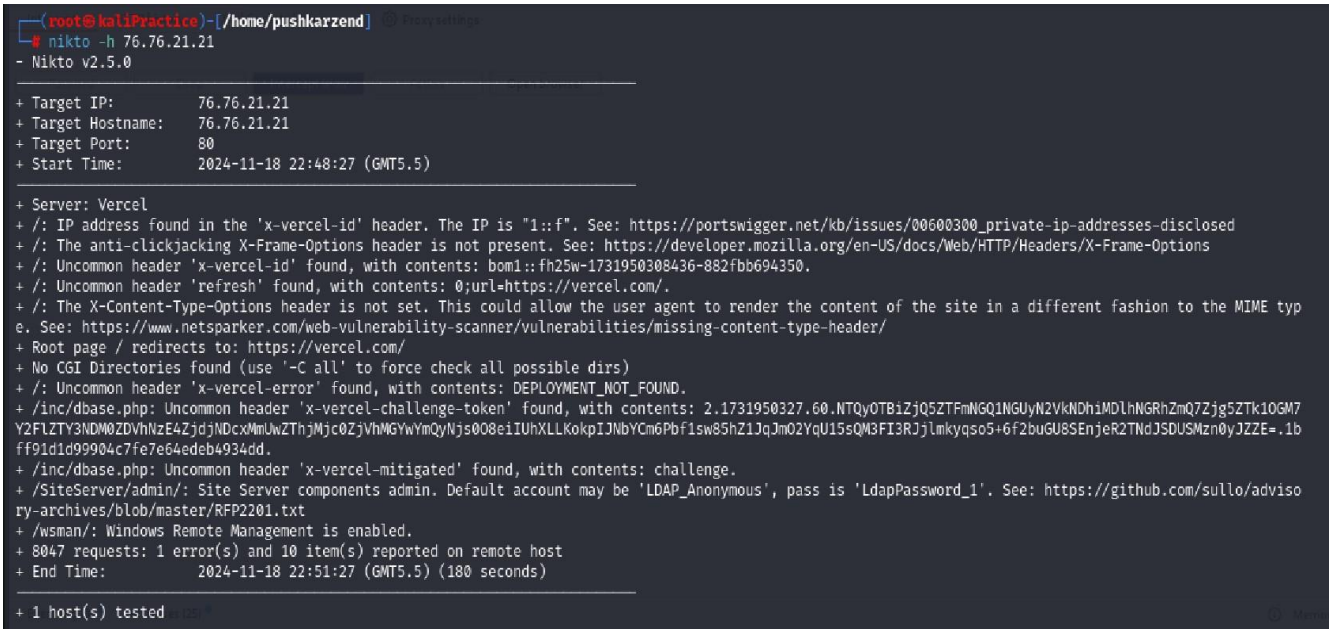
Vulnerabilities and Findings:

1. **Server Identification:**
   - The server was identified as **Vercel**.
2. **IP Disclosure:**
   - The target disclosed an internal **IP address** in the 'x-vercel-id' header (IP: `1::f`). This could lead to potential information leakage, revealing private IPs that may not be publicly intended.
   - Reference: Private IP Address Disclosure
3. **Clickjacking Protection Missing:**
   - The **X-Frame-Options** header is **not present**, leaving the application vulnerable to clickjacking attacks.
   - Reference: X-Frame-Options Header
4. **Uncommon Headers Found:**
   - Several uncommon headers were found:
     - `x-vercel-id`: `bom1::fh25w-1731950308436-882fbb694350`
     - `refresh`: `0;url=https://vercel.com/`
     - `x-vercel-error`: `DEPLOYMENT_NOT_FOUND`
     - `x-vercel-challenge-token`: `2.1731950327.60.NTQyOTBiZjQ5ZTFmNGQ1NGUyN2VkNDhiMDlhNGRhZmQ7Zjg5ZTk1OGM7Y2FlZTY3NDM0ZDVhNzE4ZjdjNDcxMmUwZThjMjc0ZjVhMGYwYmQyNjs0O8eiIUhXLLKokpIJNbYCm6Pbf1sw85hZ1JqJmO2YqU15sQM3FI3RJjlmkyqso5+6f2buGU8SEnjeR2TNdJSDUSMzn0yJZZE=.1bff91d1d99904c7fe7e64edeb4934dd`
     - `x-vercel-mitigated`: `challenge`

5. **Content-Type Header Missing:**
    - The **X-Content-Type-Options** header is **not set**, which could allow the user agent to render the content of the site in a different fashion than intended, potentially leading to cross-site scripting or other malicious content rendering.
    - Reference: Missing Content-Type Header
6. **Redirection:**
    - The root page (/) redirects to: https://vercel.com/.
7. **Potential Vulnerabilities in Components:**
    - **Admin Access**: The `/SiteServer/admin/` directory was found, which may contain a default account with the username **'LDAP_Anonymous'** and password **'LdapPassword_1'**. This could potentially allow attackers to gain unauthorized access if the credentials are not updated.
        - Reference: RFP2201 Advisory
    - **Windows Remote Management**: The `/wsman/` endpoint indicates that **Windows Remote Management** is enabled, which could be a security risk if not properly configured or secured.

Proof of Concept

To use this tool we, used the command
#nikto -h 76.76.21.21



RECOMMENDATION

- **Mitigate Information Disclosure**: Ensure that internal IP addresses are not disclosed in the response headers.
- **Implement X-Frame-Options**: Add the **X-Frame-Options** header to prevent clickjacking attacks.
- **Fix Content-Type Header Issue**: Implement the **X-Content-Type-Options** header to ensure content is rendered correctly and to prevent malicious content execution.
- **Review and Secure Admin Access**: Change default passwords for admin accounts and restrict access to sensitive directories like /SiteServer/admin/.
- **Disable Unnecessary Services**: If **Windows Remote Management** is not required, it should be disabled.

VULNERABILITY DETAILS

1)Sensitive Information In Cleartext

IMPACT	**HIGH**
AFFECTED URL	https://roomeasy.net
CWE	CWE-312: For storing sensitive data like credentials in plain text without encryption. CWE-319: For transmitting sensitive information over unencrypted communication channels.
OWASP	A02:2021:- Cryptographic Failures
CVSS SCORE:	7.5

Vulnerability Description

The **Plain Text Credentials Vulnerability** occurs when sensitive information such as login credentials (username and password) is transmitted without encryption. In this case, login credentials are visible in the intercepted HTTP request using Burp Suite, indicating they are being transmitted in plain text.

How It Works:

1. **Unencrypted Transmission:** The application transmits sensitive data over an insecure channel (e.g., HTTP instead of HTTPS).
2. **Interceptable Data:** An attacker monitoring the network can intercept the communication and access the credentials in plain text.
3. **Immediate Exploitation:** Once credentials are captured, an attacker can use them to log in and gain unauthorized access to the system.

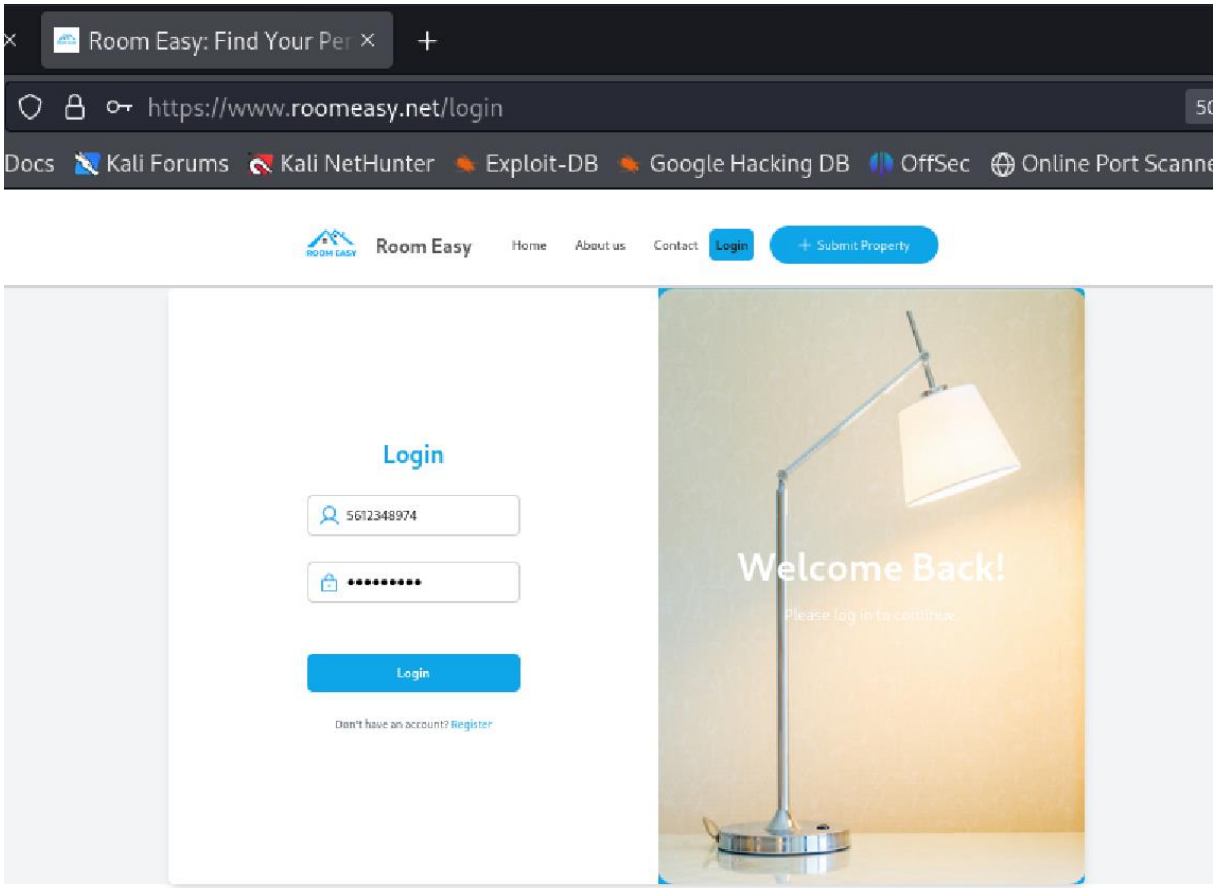
Impact:

- Unauthorized access to user accounts or sensitive data.
- Potential lateral movement within the application or system.
- Increased risk of credential reuse attacks if users reuse passwords across systems

Proof of Concept

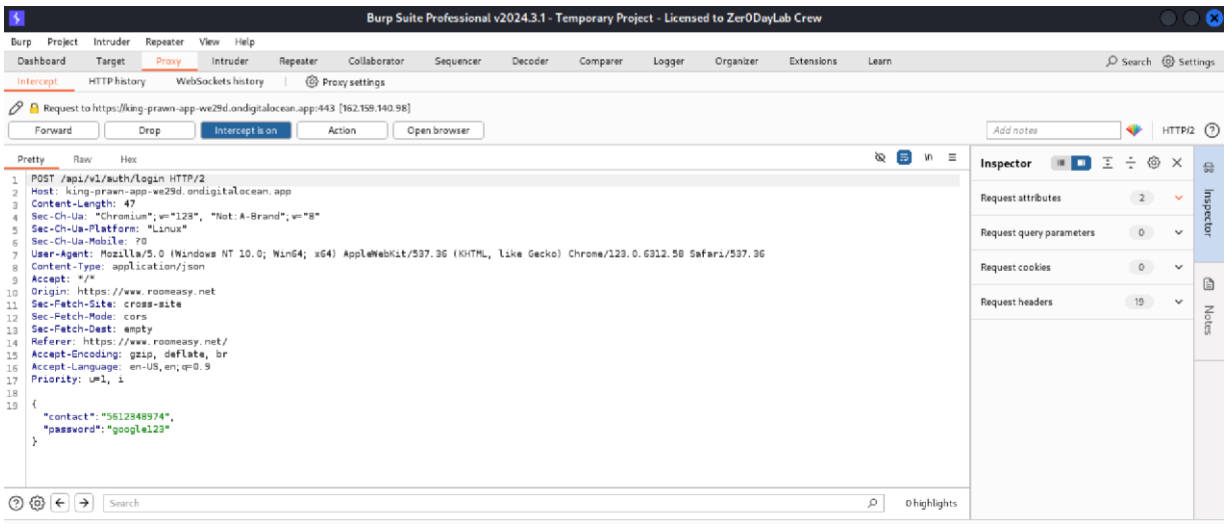
Steps Followed:-

First go to the login and type any username and password



Capture this request in the burp suite by setting proxy to the browser and turning on the intercept in burp suite.

We can see the username and password are in plaintext format. So attacker can easily intercept request and get to know about this sensitive information



RECOMMENDATION

- **Use Encryption for Data Transmission**:
  - Enforce HTTPS (TLS) to encrypt all communication between the client and server, preventing data interception during transmission.
- **Secure Storage of Credentials**:
  - Store sensitive data like passwords using strong hashing algorithms (e.g., bcrypt, Argon2) instead of storing them in plain text.
- **Implement Secure APIs**:
  - Ensure APIs do not expose sensitive information in plain text within responses or logs.
- **Mask Sensitive Data**:
  - Avoid exposing sensitive information like passwords, tokens, or session data in request/response payloads or logs.
- **Use Secure Configuration for Cookies**:
  - Mark cookies containing sensitive data as `Secure`, `HttpOnly`, and `SameSite` to prevent unauthorized access.
- **Enable Transport Layer Security (TLS) Everywhere**:
  - Ensure all internal and external communication is encrypted with strong TLS protocols.

2)Insufficient Brute Force Protection

IMPACT	**HIGH**
AFFECTED URL	https://roomeasy.net
CWE	CWE-307: Improper Restriction of Excessive Authentication Attempts
OWASP	A07:2021 – Broken Authentication
CVSS SCORE:	7.5

Vulnerability Description

Insufficient Brute Force Protection occurs when a web application fails to implement safeguards that limit or prevent multiple unsuccessful login attempts within a short period. Without mechanisms such as account lockout, CAPTCHA, or rate limiting, an attacker can exploit this flaw by using automated tools to perform brute force attacks. These attacks involve guessing a large number of username and password combinations to gain unauthorized access to a user account.

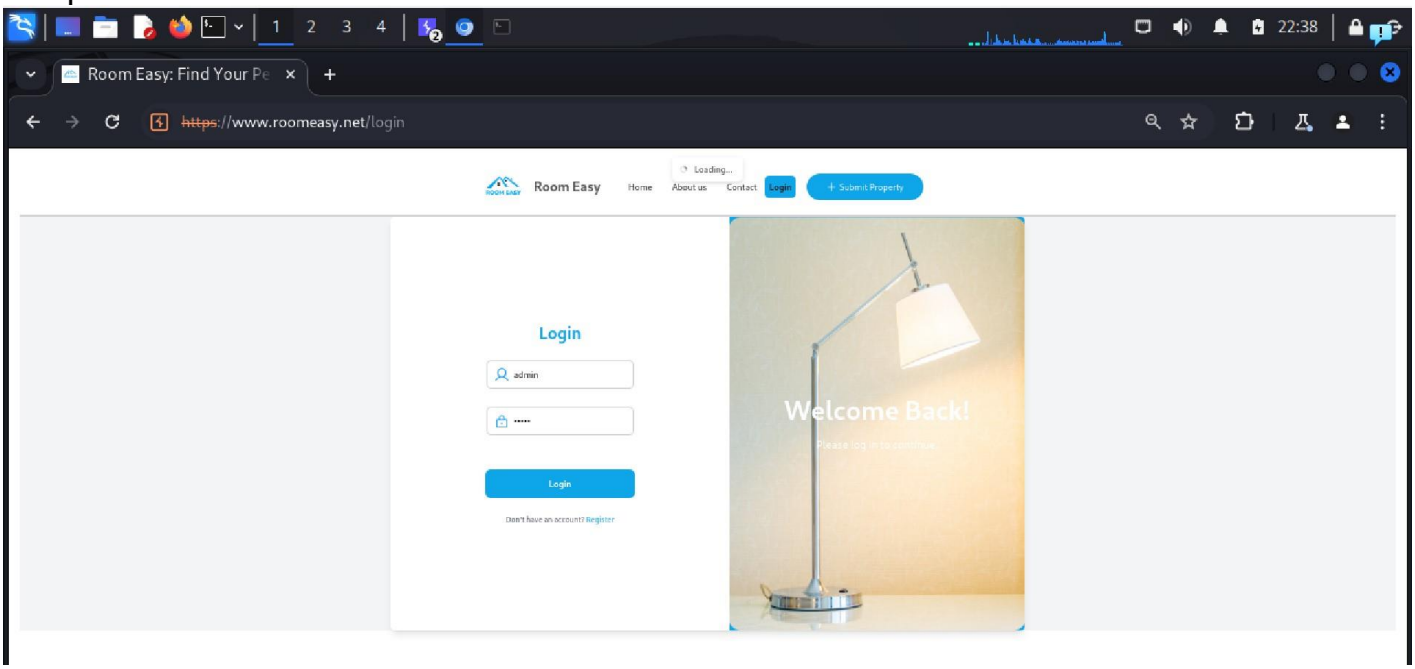
How It Works

- **Unrestricted Login Attempts**: The application does not limit or block login attempts after multiple failed tries.
- **Automated Tools**: Attackers can use automated tools (e.g., Burp Suite, Hydra, or custom scripts) to attempt numerous username and password combinations in a short time.
- **No Account Lockout or CAPTCHA**: There are no protections like account lockout after a certain number of failed attempts or CAPTCHA to verify the user is human.
- **Brute Force Attack**: Attackers can systematically try different combinations without any hindrance, increasing the chances of guessing valid credentials.
- **Higher Likelihood of Success**: With each failed attempt, the chances of success increase as the attacker tries more combinations, eventually gaining unauthorized access.

Proof of Concept

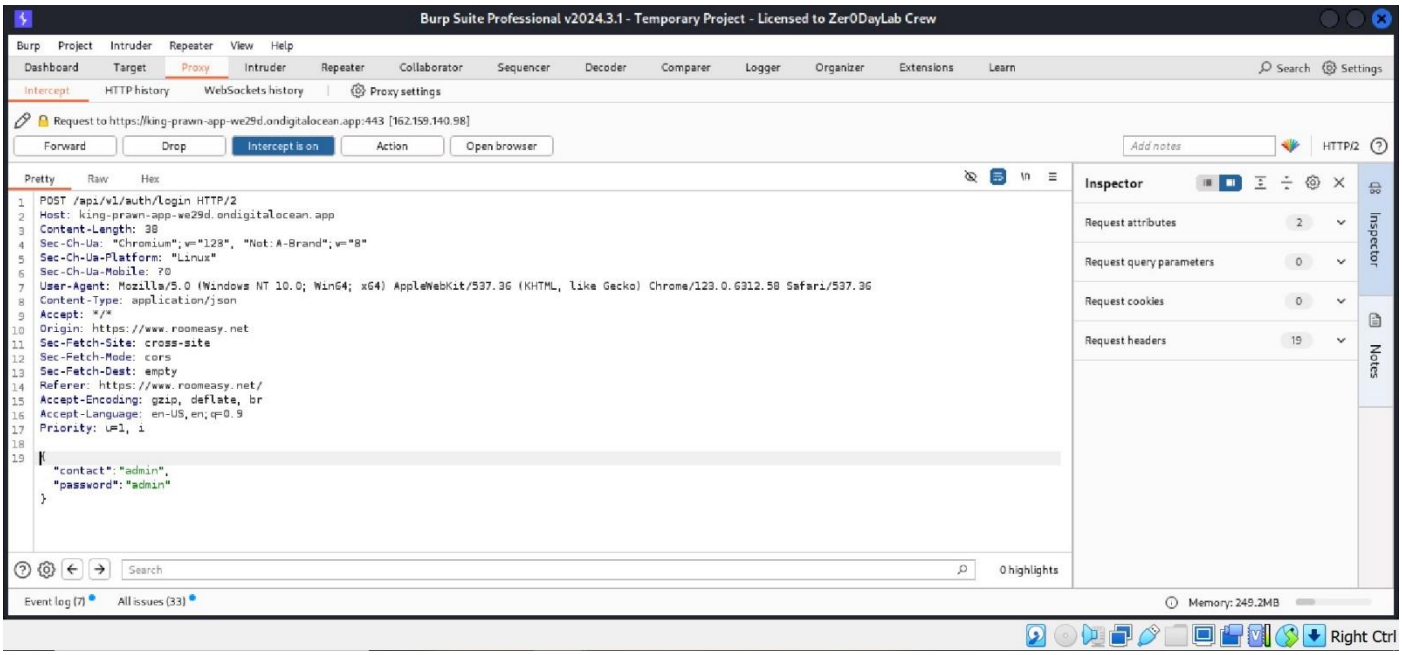
To check this vulnerability, we first created one account on the given website and try to brute force this website using burp suite intruder tool with mentioning our username and password in the list.

First, We go to the login page of roomeasy.net application and type any random username and password there

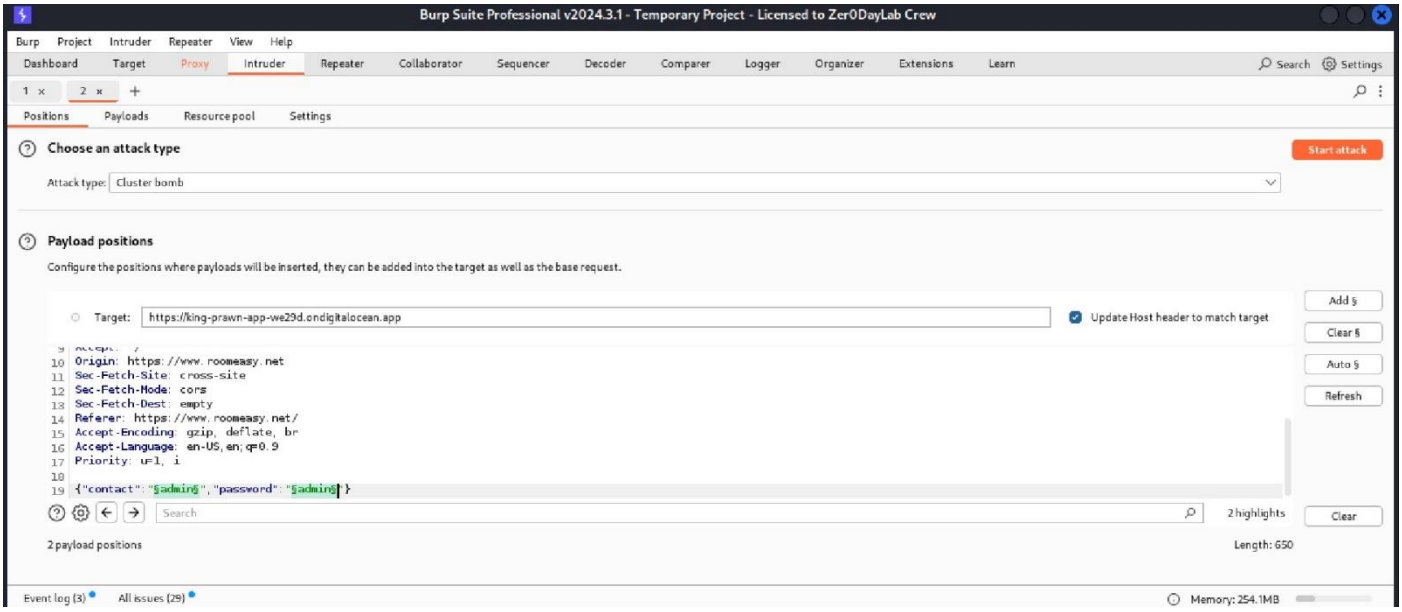


Before pressing the login, in proxy tab of burp suite, turned on the intercept to capture this request and then pressed login.

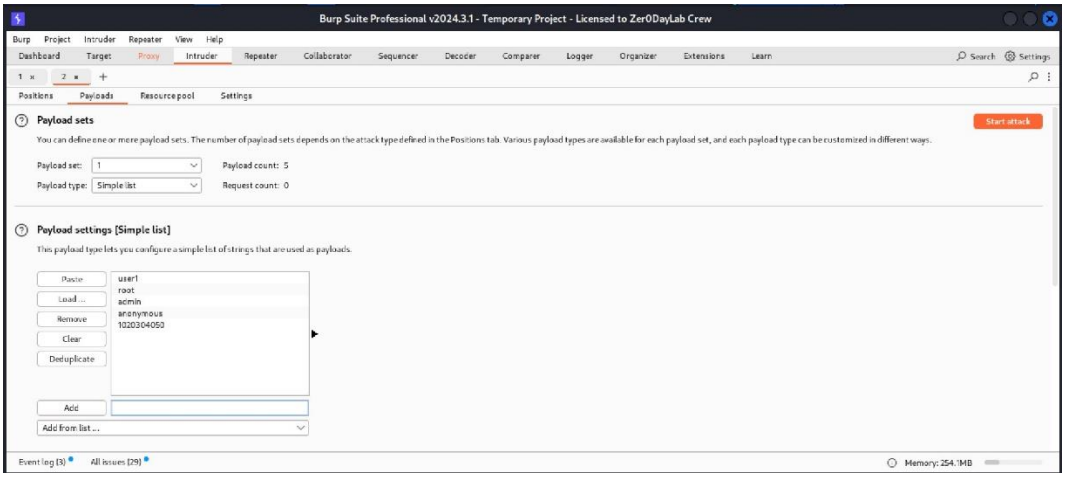
We can see the captured request in the burp suite with plaintext credentials.

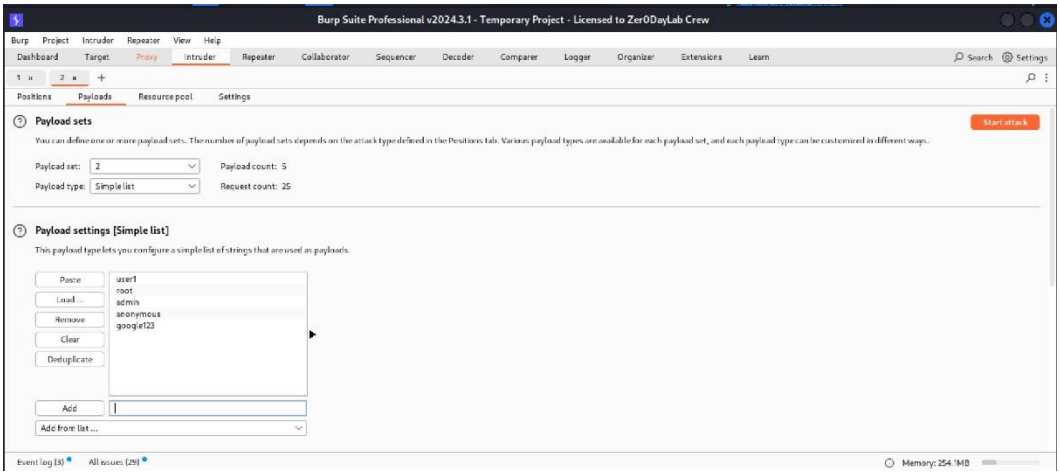


To perform Brute force attack, sent this request to the intruder and highlighted the username $admin$ and password $admin$. Selected the Attack type as Cluster Bomb Attack.



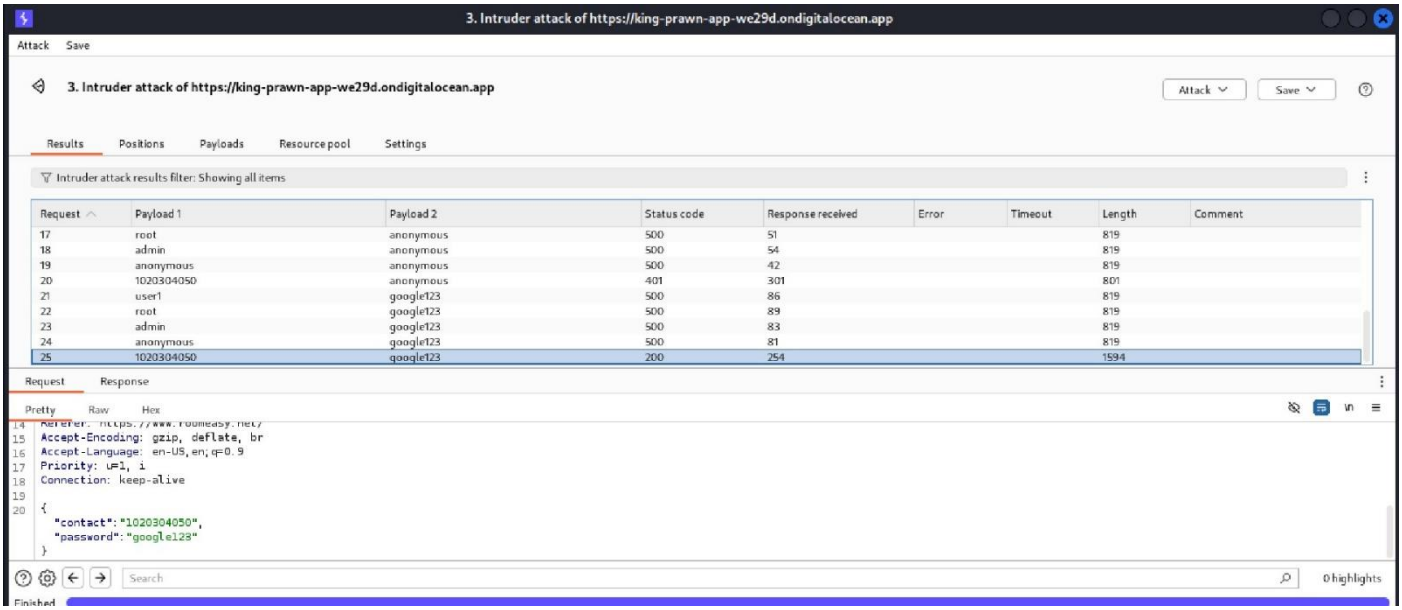
Added 2 payload with random usernames and passwords including one correct credentials that is we made on this web application.





After adding this payloads started the attack and it checked almost 25 combinations and as we have added one correct credentials we get it by looking at the length.

As it has attempted 25 combinations we can conclude that this web application is vulnerable to Brute force and vulnerability is knows as "**Insufficient Brute Force Protection**"



RECOMMENDATION

- **Implement Account Lockout Mechanism**: Introduce an account lockout or throttling mechanism after a predefined number of failed login attempts (e.g., 5 attempts), which prevents further login attempts for a specified period (e.g., 15 minutes).
- **Enable CAPTCHA**: Implement CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) after a certain number of failed login attempts to prevent automated tools from continuously attempting to brute force login credentials.
- **Use Multi-Factor Authentication (MFA)**: Enforce the use of MFA to add an extra layer of security, even if an attacker successfully guesses the username and password.
- **Monitor Login Attempts**: Implement logging and alerting mechanisms to monitor unusual login patterns or a high number of failed login attempts, enabling timely detection of brute force attacks.
- **Limit Login Attempts Per IP**: Restrict the number of login attempts from the same IP address within a specific time frame to mitigate attacks coming from a single source.

3)File Upload

IMPACT	**HIGH**
AFFECTED URL	https://roomeasy.net
CWE	CWE-434: Unrestricted Upload of File with Dangerous Type
OWASP	A06:2021 – Vulnerable and Outdated Components
CVSS SCORE:	9.8

Vulnerability Description

The **Unrestricted File Upload** vulnerability occurs when a web application allows users to upload files without validating the file type or content. This can lead to attackers uploading malicious files (e.g., PHP scripts) disguised as harmless file types.

If the malicious file is executed, it can result in **Remote Code Execution (RCE)**, data breaches, defacement, or privilege escalation.

How It Works:

The application allows file uploads without proper validation of file types or content. An attacker can upload a malicious file, such as a PHP backdoor disguised as an image. If the file is executed by the server, the attacker gains unauthorized access, potentially leading to remote code execution or further exploitation.

Impact:

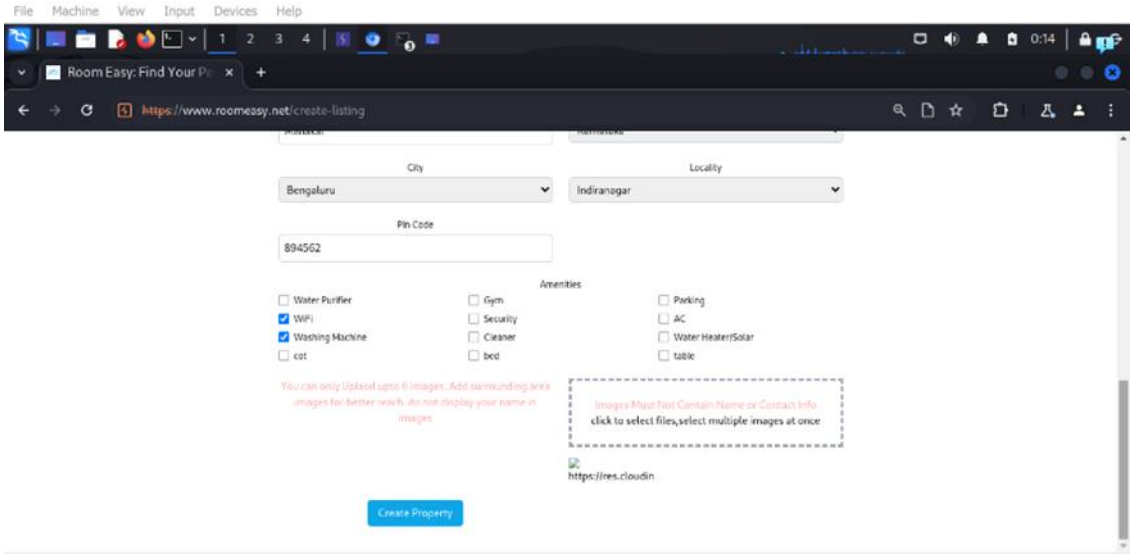
- **Remote Code Execution (RCE)**
- **Data Breach** or modification
- **Defacement** or disruption
- **Privilege Escalation**
- **Reputation Damage**

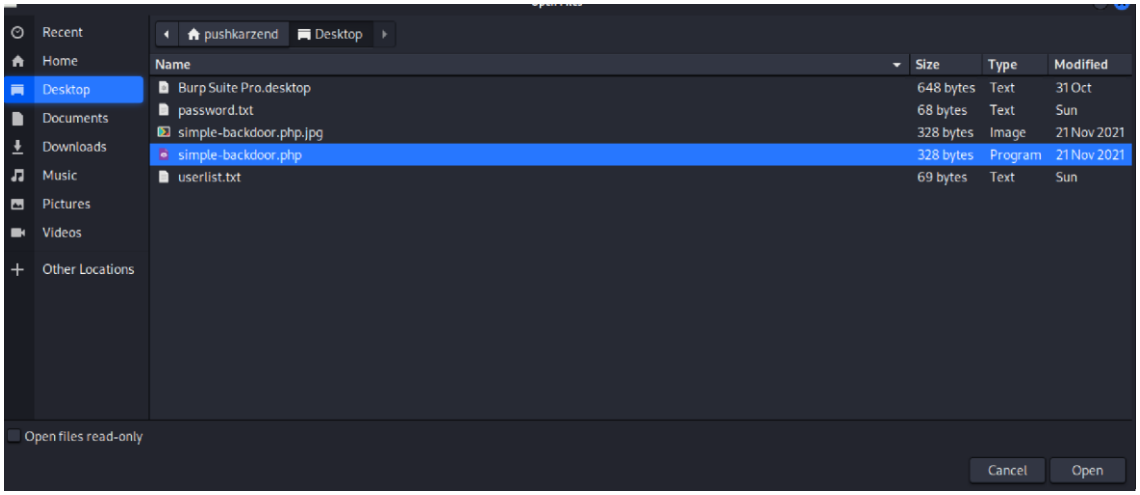
This vulnerability is critical and requires immediate remediation to prevent exploitation.

Proof of Concept

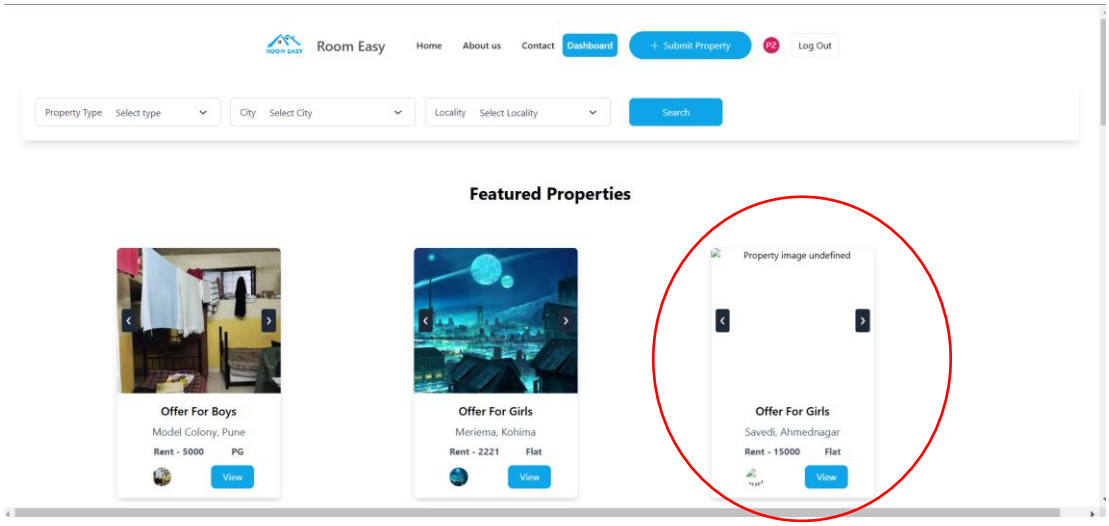
We created one dummy user account and submit one dummy property here.
We tried different image extensions like .jpg, .png, etc it is uploading all types of files.

Then tried to upload simple-backdoor.php file and it is also get successfully uploaded.





After submitting the property, it is showing our property at the home page.



RECOMMENDATION

1. Restrict File Types: Only allow specific, trusted file types (e.g., images in formats like `.jpg`, `.png`) and reject others.

2. Validate File Content: Check the file's MIME type and verify its content matches the expected file type (e.g., inspect image headers for images).

3. Rename Files: Rename uploaded files to random or sanitized names to prevent execution of malicious code.

4. Store Files Securely: Store uploaded files in a non-executable directory with proper access controls.

5. Limit File Size: Restrict file size to prevent large payloads from being uploaded.

6. Use File Scanners: Implement antivirus and malware scanners on uploaded files to detect potential threats.

7. Disable File Execution: Ensure that the directory where files are uploaded does not allow execution of scripts.

4)Shared Credentials Vulnerability / Weak Session Management

IMPACT	**Medium to high**
AFFECTED URL	https://roomeasy.net
CWE	CWE-288 - Authentication Bypass Using an Alternate Path or Method CWE-284 – Improper Access Control
OWASP	A5:2021 - Security Misconfiguration
CVSS SCORE:	6.5

Vulnerability Description

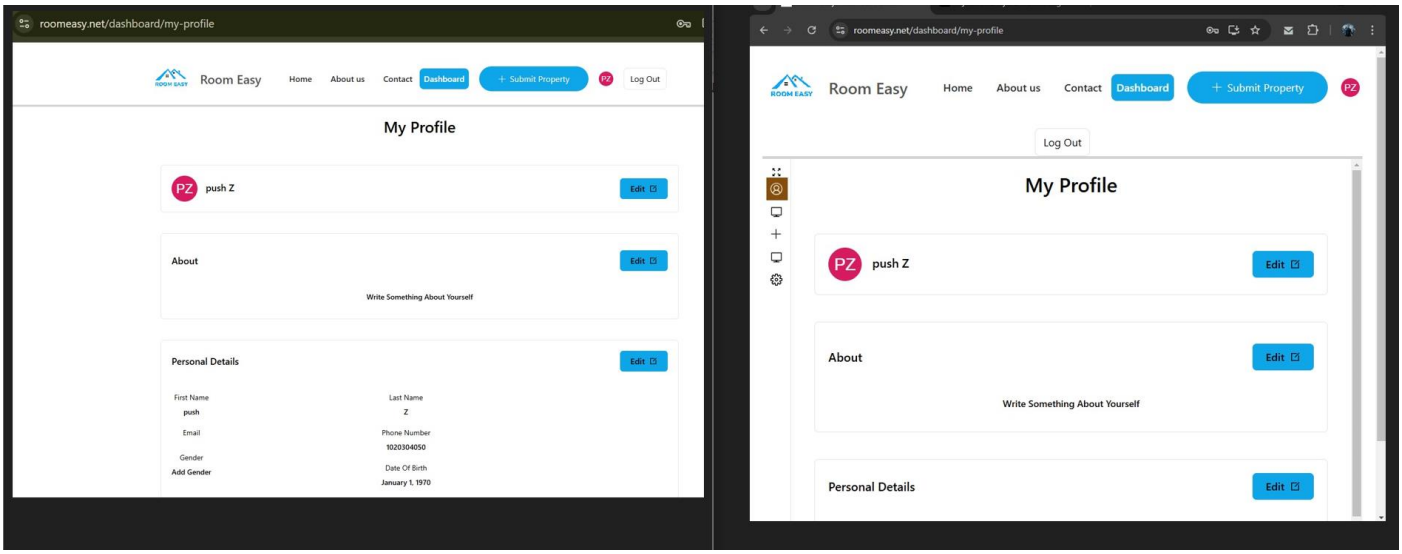
Weak Session Management or **Shared Credentials**, this occurs when a user can log in with the same credentials (username and password) from multiple locations, even if the user is not on the same network. This suggests that the website is not properly managing or isolating user sessions, allowing multiple users to access the same account without proper session handling or validation.

How it Works:

- The user logs in using a set of credentials (username and password).
- Since the application doesn't implement proper session management, the session remains active across multiple devices or networks.
- Both you and your friend can use the same credentials to log into the application, which indicates that the session isn’t being properly isolated or terminated after login.
- This can lead to unauthorized access, data manipulation, or session hijacking by malicious actors.

Proof of Concept

Here though the One session is logged in on one device, by using the same credentials another session can be logged in for the same user who is in another network.



So it means that it allows multiple sessions of same User on different devices

RECOMMENDATION

- **Implement Unique Session Management:**
  - Ensure each user session is unique and tied to a single active login.
  - Terminate existing sessions when a new session is initiated for the same user credentials.
- **Use Session Tokens:**
  - Employ secure, randomly generated session tokens for each login.
  - Validate tokens for every user request to ensure session integrity.
- **Enforce Device and IP Restrictions:**
  - Restrict multiple simultaneous logins from different devices or IP addresses.
  - Alert users of any suspicious activity when multiple logins are detected.
- **Enable Two-Factor Authentication (2FA):**
  - Add an extra layer of security to the login process to prevent unauthorized access.
- **Session Expiration:**
  - Set a timeout for idle sessions and terminate inactive sessions automatically.
  - Provide users with a visible logout option and ensure it properly destroys the session.

5)Clickjacking (UI redress attack)

IMPACT	**Medium**
AFFECTED URL	https://roomeasy.net
CWE	**CWE-1021: Improper Restriction of Rendered UI Layers or Frames**
OWASP	A04:2021 – Insecure Design
CVSS SCORE:	5.0

Vulnerability Description

Clickjacking occurs when an attacker overlays an invisible iframe over a legitimate webpage. Users believe they are interacting with visible elements, but their clicks are actually directed to the hidden content in the iframe. This can lead to unintended actions, such as submitting data, changing account settings, or authorizing transactions without the user's knowledge or consent.

How It Works: The attacker embeds the target website inside an invisible iframe and places it over their malicious interface. When the user clicks on a visible button or link, they are unknowingly interacting with the hidden iframe, triggering actions on the underlying site, such as clicking a “submit” button or confirming a transaction.

Impact:

- Users are deceived into performing unintended actions.
- Potential for data theft, account compromise, and unauthorized transactions.

Proof of Concept

- The malicious page loads `www.roomeasy.net` in an iframe.
- The `iframe` is semi-transparent (via `opacity: 0.1`), so users see the attacker's content and interact with the embedded target page.
- The attacker can mislead users into clicking on elements of `www.roomeasy.net` (e.g., a "Login" button), potentially exposing sensitive data or executing unintended actions.

Request

```
GET / HTTP/1.1
Host: www.roomeasy.net
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.58 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Sec-Ch-Ua: "Chromium";v="123", "Not:A-Brand";v="8"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=0, i
Connection: close
```

Response

```
HTTP/2 200 OK
Server: Vercel
X-Vercel-Id: 22u7iFIULn9celQiLRIelKb4FKk3QQFu
Accept-Ranges: bytes
Access-Control-Allow-Origin: *
Age: 2366969
Cache-Control: public, max-age=0, must-revalidate
Content-Disposition: inline
Content-Type: text/html; charset=utf-8
Date: Mon, 18 Nov 2024 18:25:36 GMT
Etag: "c5f0cd0a7cbc6811f1590fe2543f75d5"
Last-Modified: Tue, 22 Oct 2024 08:56:07 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000
X-Vercel-Cache: HIT
X-Vercel-Id: bom1::bjwnz-1731954336949-394277e19ec5
Content-Length: 644

<!doctype html><html lang="en"><head><meta charset="utf-8"/><link rel="icon" href="/favicon.ico"/><meta name="viewport" content="width=device-
width,initial-scale=1"/><meta name="theme-color" content="
```

In the above reponse as X-Frame-Options are missing. Without this header, the page can be embedded in an frame on a malicious website.

Also Content-Security-Policy is also missing, this could specify frame-ancestors to control iframe embedding.

Access-Control-Allow-Origin indicates permissive cross-origin resource sharing (CORS), which may exacerabate the risk in some scenarios

RECOMMENDATION

1. Review the settings, management and handling of privileges.
2. Explicitly manage trust zones in the software.
3. Before making changes at the server & sending response to client, verify the role and privilege of the user to prevent unauthorized access to data.
4. Enforce strict validation at client side and server side.
5. To effectively prevent framing attacks, the application should return a response header with the name **X-Frame-Options** and the value **DENY** to prevent framing altogether, or the value **SAMEORIGIN** to allow framing only by pages on the same origin as the response itself. Note that the SAMEORIGIN header can be partially bypassed if the application itself can be made to frame untrusted websites.

6) X-Content-Type-Options (MIME-type sniffing)

IMPACT	**Medium**
AFFECTED URL	https://roomeasy.net
CWE	CWE-204: Confusion of Functionality and Intent (Due to lack of MIME type declaration) CWE-116: Improper Encoding or Escaping of Output
OWASP	OWASP A5: Security Misconfiguration:
CVSS SCORE:	6.1

Vulnerability Description

The **X-Content-Type-Options** header is a security measure that prevents browsers from performing **MIME-type sniffing**. MIME-type sniffing occurs when a browser attempts to determine the content type of a resource based on its content, instead of strictly adhering to the `Content-Type` header set by the server. If this header is not set correctly, browsers may misinterpret the content and potentially execute it in an unintended manner, such as running JavaScript from a file intended to be something else, like an image or text file. This misinterpretation can lead to security vulnerabilities, such as **cross-site scripting (XSS)** attacks, where malicious code is executed in the user's browser.

How It Works:

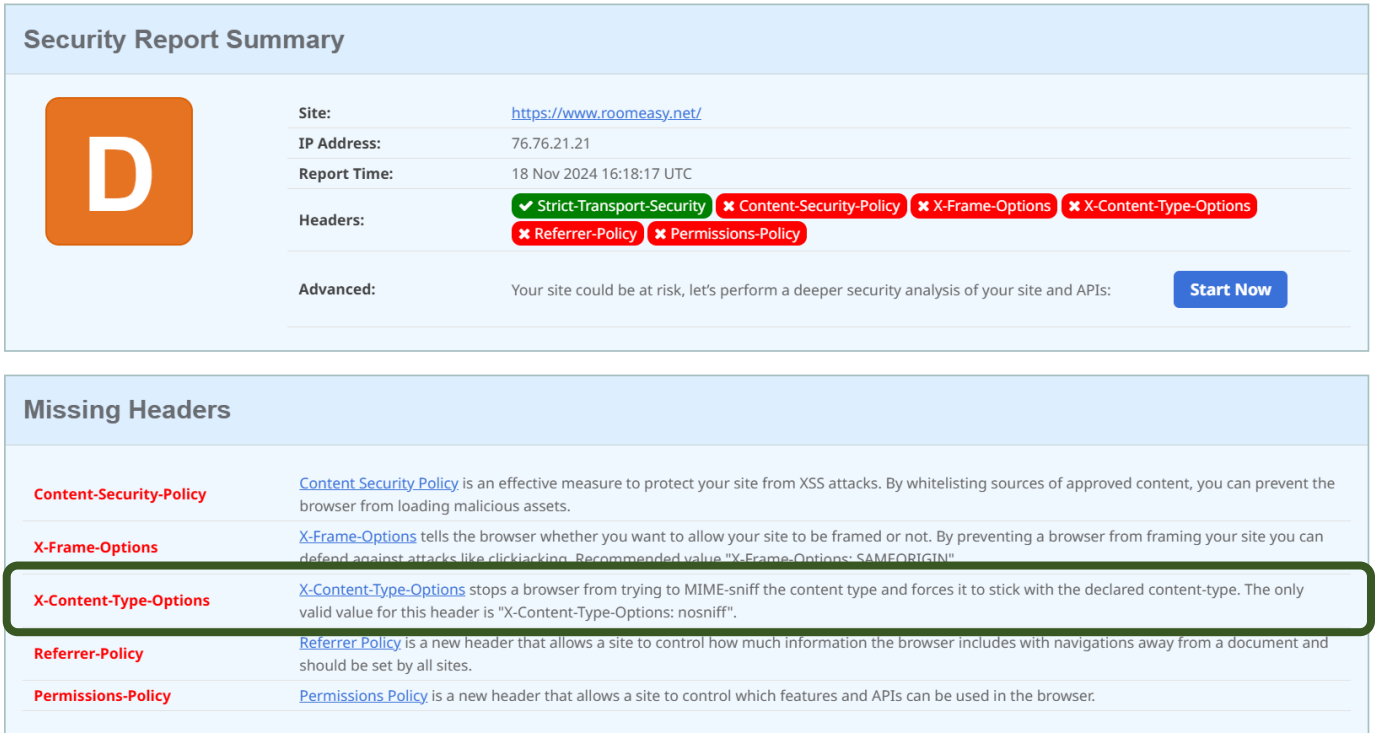
When a browser requests a resource, it relies on the `Content-Type` header to determine how to handle that content (e.g., as text, an image, or a script). If the server provides an incorrect or missing `Content-Type` header, or if the header is ambiguous, browsers might attempt to guess the content type based on the actual content of the resource. This is called **MIME-type sniffing**. For example, a `.js` file could be served with the `Content-Type: text/plain`, causing the browser to treat it as plain text instead of JavaScript. Without the **X-Content-Type-Options** header set to `nosniff`, the browser may misinterpret the content and execute it as JavaScript, potentially leading to security vulnerabilities such as **XSS**.

Risk and Impact:

- **Cross-Site Scripting (XSS)**: An attacker could exploit this vulnerability by injecting malicious JavaScript into files that are improperly interpreted by the browser. The attacker could then execute arbitrary scripts in the context of the user's session, stealing sensitive data, hijacking user accounts, or defacing websites.
- **Content Misinterpretation**: Files intended to be non-executable, like images or text files, might be executed as malicious scripts by the browser, leading to further vulnerabilities or unexpected behavior.

Proof of Concept

We scanned the URL of this web application on https://securityheaders.com/ this website and we get various security headers are missing.



RECOMMENDATION

To mitigate the risk of MIME-type sniffing, ensure that the **X-Content-Type-Options** header is set to `nosniff` in the HTTP response.

Key Actions:

1. **Set the Header**: Add `X-Content-Type-Options: nosniff` to your server's response headers.
2. **Verify Configuration**: Ensure the header is included for all responses across the web application.
   - **Apache**: `Header set X-Content-Type-Options "nosniff"`
   - **Nginx**: `add_header X-Content-Type-Options "nosniff";`
3. **Check Content-Type**: Ensure that the `Content-Type` header is correctly configured for all resources.
4. **Test Regularly**: Use security scanners to check for missing or misconfigured headers.

Setting this header prevents browsers from misinterpreting content and helps protect against **XSS** and **drive-by downloads**.

7)Weak Input Validation

IMPACT	**Medium**
AFFECTED URL	https://roomeasy.net
CWE	CWE-521: Weak Password Requirements CWE-20: Improper Input Validation
OWASP	A07:2021 – Identification and Authentication Failures
CVSS SCORE:	5.3

Vulnerability Description

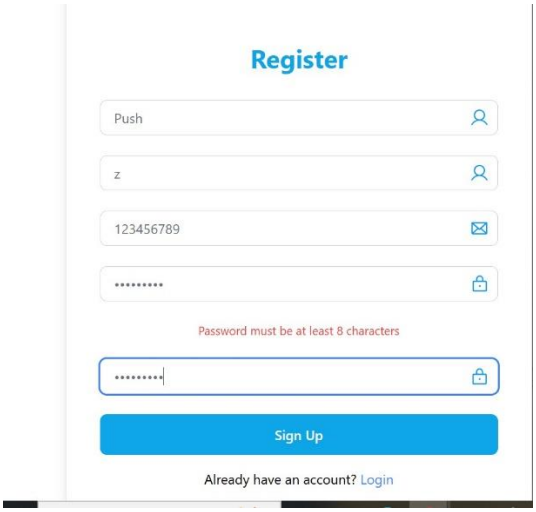
The application allows registration with invalid or weak inputs, such as non-verified mobile numbers and simple passwords without complexity requirements. This indicates Weak Input Validation and a Weak Password Policy, making the system vulnerable to brute force attacks, fake account creation, and unauthorized access. These flaws compromise authentication security and may lead to account misuse or exploitation.

How It Works:

The system accepts registration with invalid mobile numbers (e.g., incorrect length or format) and weak passwords (e.g., only lowercase letters and numbers). There is no validation or enforcement of strong password policies or verification of the mobile number, allowing users to register with insecure credentials and potentially gain unauthorized access.

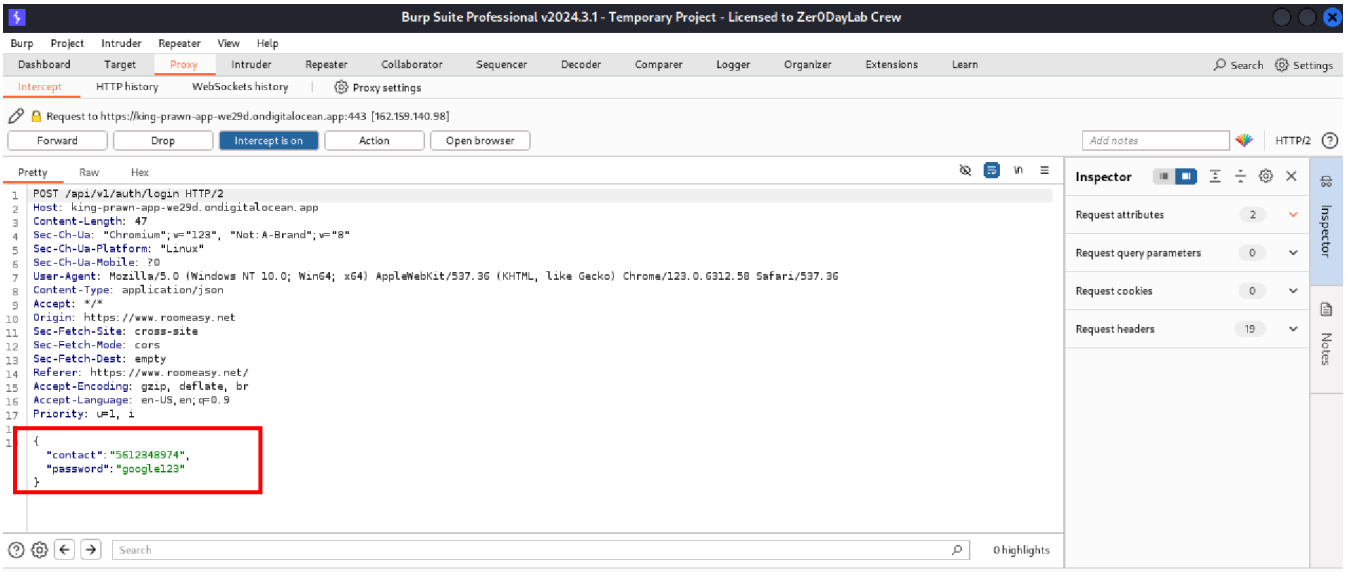
Proof of Concept

While registering a new user to this particular web application, we write a wrong mobile number, and very basic simple password containing only characters and numbers.

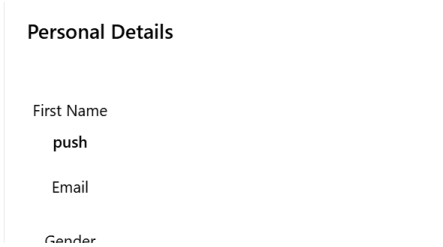


It doesn’t validated the content we typed there and registered the user successfully.

We can see the plaintext credentials in burp suite,



After Signing up, we successfully logged in with the dummy credentials we have made.



RECOMMENDATION

Recommendations for Weak Input Validation Vulnerability:

1. **Enforce Strong Password Policies**:
    - Require passwords to be at least 12 characters long, with a mix of uppercase, lowercase, numbers, and special characters.
2. **Implement Mobile Number Validation**:
    - Use OTP (One-Time Password) or email verification to validate mobile numbers during registration.
3. **Server-Side Input Validation**:
    - Ensure proper validation of user inputs for both mobile numbers and passwords to meet predefined formats and complexity requirements.
4. **Error Handling and Messaging**:
    - Provide clear error messages for invalid inputs without exposing system internals.
5. **Rate Limiting and Brute Force Protection**:
    - Implement rate limiting or CAPTCHA to prevent brute force attacks during login or registration attempts.
6. **Regular Security Audits**:
    - Conduct regular audits and tests of input validation mechanisms to identify and address potential weaknesses.

8)Exposed or Improperly Managed API key

IMPACT	**HIGH**
AFFECTED URL	https://roomeasy.net
CWE	CWE-522: Insufficiently Protected Credentials CWE-598: Information exposure through query strings in GET Request CWE-200: Exposure of Sensitive Information to an Unauthorized actor
OWASP	A05:2021 – Security Misconfiguration (If API keys are exposed due to misconfigured permissions of headers.
CVSS SCORE:	5.3

Vulnerability Description

API Key Mismanagement in the context of the application occurs due to the exposure of an API key in client-side requests (e.g., in the `GET` or `POST` request headers or URLs). This allows anyone with access to the intercepted traffic to extract the API key and misuse it. In this case, the API key was found in the `GET` and `POST` requests sent to the backend API (`/api/v1/image/assets` and `/api/v1/auth/login`), which exposes the system to unauthorized access or malicious activities.

How It Works

1. **Exposure of API Key**: The API key is visible in network requests (e.g., in the `apiKey` parameter). This makes it accessible to anyone inspecting the traffic using tools like Burp Suite or browser developer tools
2. **Lack of Restrictions**: The API key does not have adequate usage restrictions, such as IP whitelisting or domain-specific enforcement, allowing it to be exploited by unauthorized users.
3. **Unauthorized API Access**: An attacker can capture the exposed key and use it to make API requests on behalf of legitimate users, potentially gaining access to sensitive data or performing malicious actions.
4. **Compromise of Integrity and Confidentiality**: If the API key allows critical actions, it could lead to data breaches, manipulation of data, or unauthorized transactions within the application.

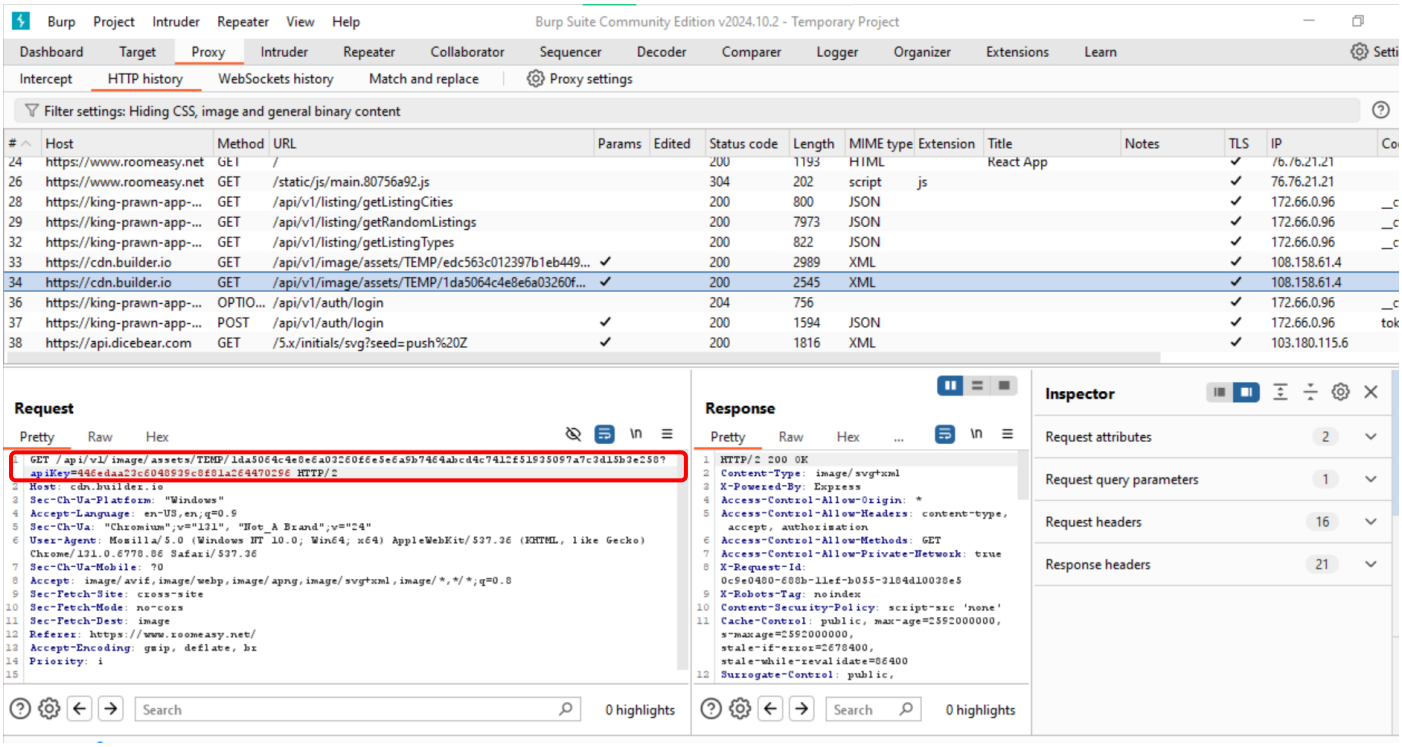
Proof of Concept

1. Intercepting the Traffic in burp suite.

Intercepted and analysed the login request of the dummy account we created using burp suite.

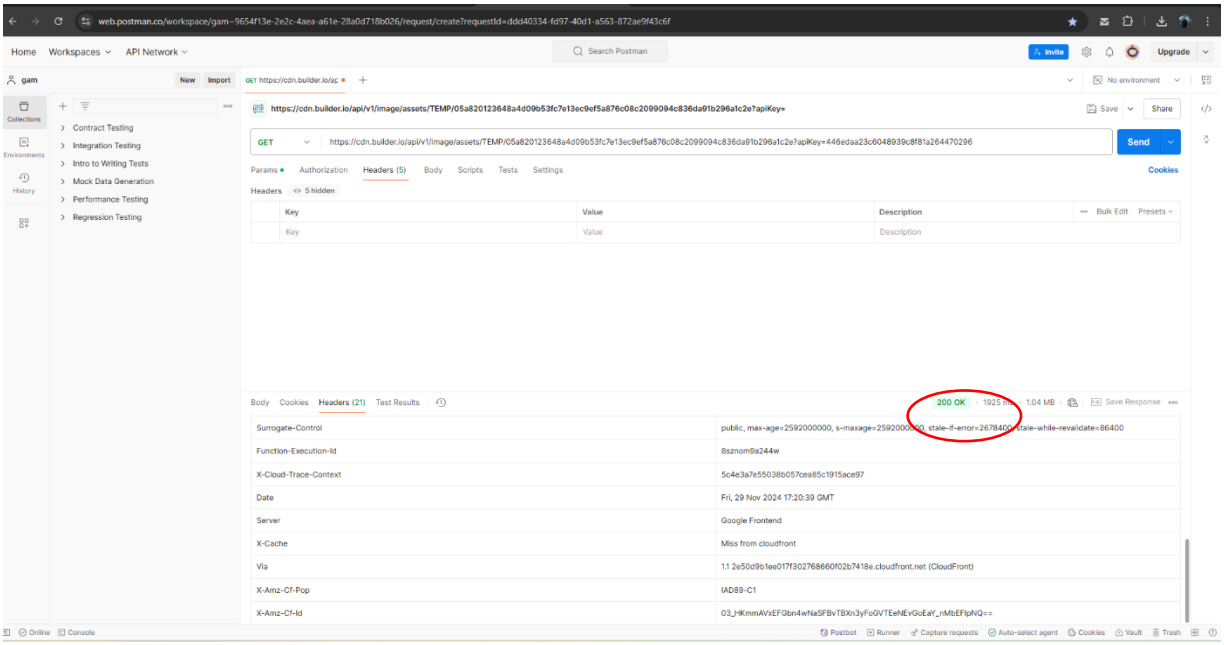
2. Captured the API key;

We identified the API key in and it is exposed in the URL of request



3.Testing the API key using POSTMAN to see it is working or not

To do this copied the URL from the request and checked it in the postman tool.



The request is successful, so it confirms that the API key works without any restrictions.

RECOMMENDATION

1. **Use Secure Storage**:

- Store API keys securely, using environment variables or secret management tools (e.g., AWS Secrets Manager, HashiCorp Vault).
- Avoid storing API keys in public code repositories or client-side code.

2. **Limit API Key Permissions**:

- Implement the principle of least privilege by limiting API key permissions to only the necessary resources and actions.
- Ensure API keys are scoped to specific roles or endpoints, and avoid granting excessive permissions.

3. **Implement IP Whitelisting**:

- Restrict API key usage to a predefined set of IP addresses or IP ranges. This ensures that only authorized systems can make requests using the key.

4. **Use Strong Authentication**:

- Employ more secure authentication mechanisms (e.g., OAuth, API tokens) to replace static API keys where appropriate.
- Implement multi-factor authentication (MFA) for critical API endpoints.

5. **Rotate and Expire API Keys**:

- Regularly rotate API keys to minimize the risk of exposure over time.
- Set expiration dates on API keys to ensure they are only valid for a limited time.
- Revoke API keys immediately if they are suspected to be compromised.

6. **Implement Rate Limiting and Throttling**:

- Use rate limiting to prevent brute-force attacks or abuse of API endpoints.
- Throttle the number of requests an API key can make in a given time frame.

9)Insecure Local Storage

IMPACT	**Medium (can escalate to high depending on the sensitivity of the data stored)**
AFFECTED URL	https://roomeasy.net
CWE	CWE-922 - Insecure Storage of Sensitive Information CWE-312: Cleartext Storage of Sensitive Information. CWE-200: Exposure of Sensitive Information to an Unauthorized Actor
OWASP	A01:2021 - Broken Access Control
CVSS SCORE:	6.4

Vulnerability Description

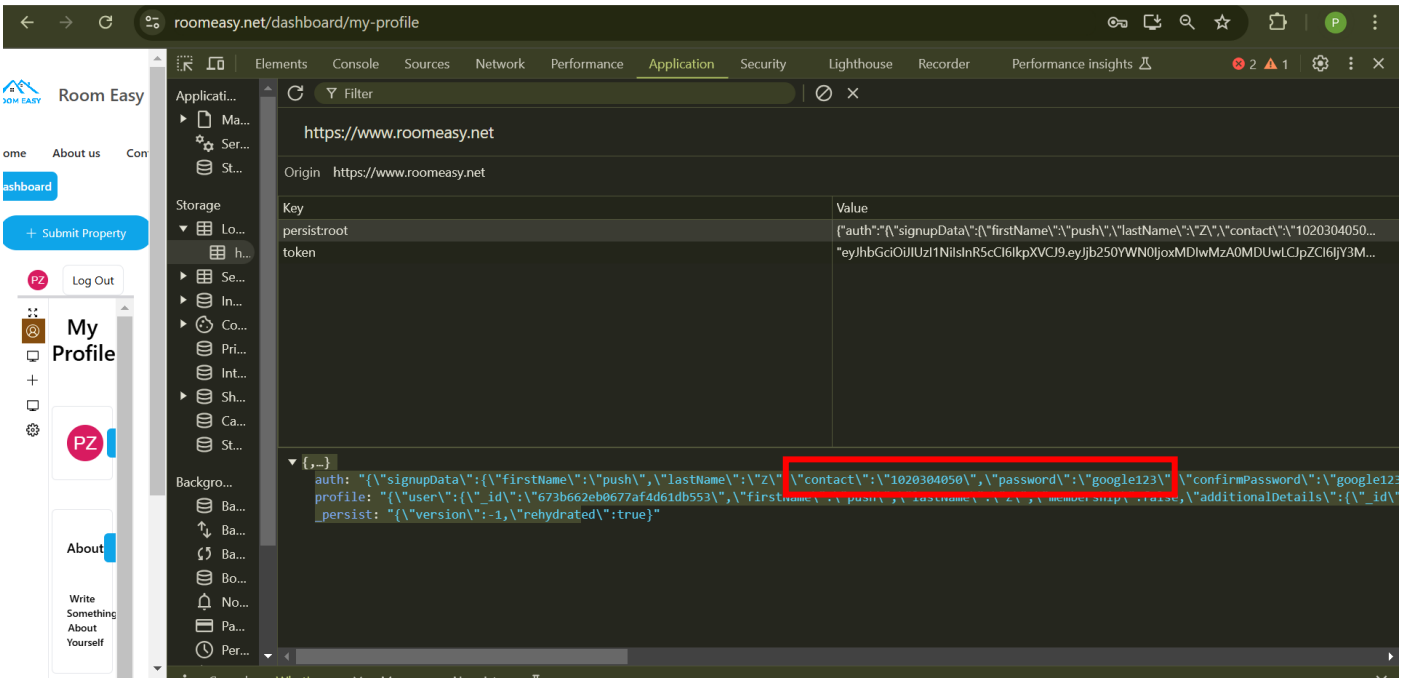
Insecure Local Storage refers to the unsafe practice of storing sensitive information, such as API keys, user credentials, or tokens, in the browser's local storage, session storage, or other client-side storage mechanisms. These storage areas lack sufficient security and can be easily accessed by attackers using browser-based attacks like Cross-Site Scripting (XSS), browser extensions, or physical device access.

How It Works:

- **Data Storage:** Sensitive data is stored in client-side storage mechanisms (e.g., localStorage, sessionStorage) without encryption or security measures.
- **Exploitation via XSS:** If the application is vulnerable to XSS, attackers can inject malicious scripts that access and exfiltrate data from the insecure storage.
- **Browser Access:** If an attacker gains unauthorized access to a user's browser (via malware or browser vulnerabilities), they can directly read the stored sensitive data.
- **Shared Environment Risks:** In shared or public devices, sensitive data stored in local storage remains accessible even after a user logs out or closes the browser.

Proof of Concept

After login with the dummy user that we created, after inspecting the page of our profile on the website, in the local storage the username and password are displayed in plaintext form leading to insecure local storage vulnerability.



RECOMMENDATION

- **Avoid Storing Sensitive Data Client-Side:**
  - Refrain from storing sensitive information like API keys, tokens, or user credentials in `localStorage`, `sessionStorage`, or cookies without proper security mechanisms.
- **Use Secure Storage Mechanisms:**
  - Store sensitive data on the server-side where possible. Use secure session management with short-lived tokens and server-side validation.
- **Implement Token Security:**
  - Use HTTP-only, secure, and SameSite cookies for storing session tokens. These cookies are inaccessible to JavaScript, mitigating the risk of XSS attacks.
- **Encrypt Data:**
  - If storing data client-side is unavoidable, ensure it is encrypted using strong encryption algorithms. However, remember that encryption keys should not be hardcoded or stored in the same environment.
- **Prevent XSS Attacks:**
  - Implement robust input validation and output encoding to prevent Cross-Site Scripting (XSS), which could enable attackers to access client-side storage.
- **Token Expiration and Revocation:**
  - Ensure tokens stored client-side are short-lived and support easy revocation if misuse is detected.
- **Regularly Clear Storage:**
  - Implement mechanisms to automatically clear client-side storage upon user logout or session expiration.
- **Monitor and Log Access:**
  - Log and monitor access patterns to detect and respond to unauthorized attempts to retrieve or misuse client-stored data.
- **Educate Developers:**
  - Train developers on secure coding practices, particularly on the risks associated with insecure client-side storage and how to mitigate them.

CONCLUSION

The network penetration testing conducted on **roomeasy.net** uncovered several critical security vulnerabilities that could compromise the integrity and confidentiality of the web application and its associated infrastructure. Notable findings include the ability to upload potentially malicious files due to insufficient file validation, weak session management allowing unauthorized access, and the exposure of sensitive login credentials during site interactions.

The shared hosting environment introduces additional risks, as vulnerabilities in neighboring websites could also impact **roomeasy.net**. Furthermore, a lack of robust security headers and improper handling of HTTP methods and cookies highlight the need for stronger security measures. These gaps, coupled with the potential for exploitation via weak password policies and unrestricted access, emphasize the critical need for immediate action.

To mitigate these risks, the organization should implement the recommended security measures, including enforcing strict file validation, strengthening session management, and ensuring secure header configurations. Regular security audits and continuous monitoring of the application will be essential to preempt potential attacks and maintain a secure environment. Addressing these vulnerabilities will not only safeguard the application but also enhance user trust and confidence in the platform.