MONETHIC

BJustCoin Admin Dashboard

Penetration Testing Report

Prepared for:
BJustCoin s.r.c.

Version:
Final, 28.02.2025

Table of Contents

About Monethic 2
Disclaimer 3
Scoping Details 3
Timeframe 3
Approach and initial access level 4
Executive Summary 4
Methodology 4
Customized and hybrid testing approach 4
Risk rating 5
Risk categorization 6
Key findings 6
Security Model & Threat Modeling 7
Assets 7
Actors 7
Assumptions 7
STRIDE Methodology in Penetration Testing 7
Vulnerabilities summary 10
Technical summary 11
1. Any user can log into another user's account using any password 11
2. Lack of access control on administrative resources 14
3. The user registration process is insecure and may lead to multiple issues 19
4. Missing brute-force attacks protection in login form 28
5. Application leaks sensitive technical information to server logs 31
6. Lack of Web Application Firewall (WAF) mechanism usage 32
7. Lack of validation of special characters 34
8. Missing Subresource Integrity hashes 38
9. Missing security headers 39
10. Logout button does not work properly 41
11. Create application for ICO tokens for another user's email address 43
12. Software version disclosure 45
13. Directory listing is enabled 47
14. Vulnerable dependency in use 48

About Monethic

Monethic is a young and thriving cybersecurity company with extensive experience in various fields, including Smart Contracts, Blockchain protocols (layer 0/1/2), wallets and off-chain components audits, as well as traditional security research, starting from penetration testing services, ending at Red Team campaigns. Our team of cybersecurity experts includes experienced blockchain auditors, penetration testers, and security researchers with a deep understanding of the security risks and challenges in the rapidly evolving IT landscape. We work with a wide range of clients, including fintechs, blockchain startups, decentralized finance (DeFi) platforms, and established enterprises, to provide comprehensive security assessments that help mitigate the risks of cyberattacks, data breaches, and financial losses.

At **Monethic**, we take a collaborative approach to security assessments, working closely with our clients to understand their specific needs and tailor our assessments accordingly. Our goal is to provide actionable recommendations and insights that help our clients make informed decisions about their security posture, while minimizing the risk of security incidents and financial losses.

The testers team consisted of:

- **Jakub Heba** - CEO and Co-Founder of Monethic - with over 8 years of experience in the cybersecurity industry. Holder of OSCP and OSCE certifications, have assigned multiple CVEs, among others:
    - **CVE-2019-10070** - Apache Atlas, Stored Cross Site Scripting
    - **CVE-2020-6856** - JOC Cockpit, Jobscheduler, XML External Entity
    - **CVE-2020-6854** - JOC Cockpit, Jobscheduler, Multiple Stored Cross Site Scripting
    - **CVE-2020-6855** - JOC Cockpit, Jobscheduler, Denial of Service
    - **CVE-2021-3584** - Foreman, Authenticated Remote Code Execution via Sendmail configuration

- **Łukasz Mikuła** - CTO and Co-Founder of Monethic - with over 8 years of experience in the cybersecurity industry. Holder of eWPT, eWPTX, OSCP and OSCE certifications, with assigned CVEs such as:
    - **CVE-2020-5907** – F5 TMOS Shell privilege escalation vulnerability,
    - **CVE-2018-6498, CVE-2018-6499, CVE-2018-6498, CVE-2018-6499** - Microfocus - AutoPass License Server Remote Code Execution,
    - **CVE-2017-10068, CVE-2018-2651, CVE-2018-2652, CVE-2018-2653, CVE-2018-2695** - BI Publisher, PeopleSoft Enterprise PeopleTools XSS, XXE, SSRF, XSLT execution,
    - **CVE-2017-1181, CVE-2017-1183, CVE-2017-1182** IBM TEP Server - SQL Injection, Authorization Bypass, OS Command Injection

Disclaimer

This report reflects a rigorous security assessment conducted on the specified product, utilizing industry-leading methodologies. While the service was carried out with the utmost care and proficiency, it is essential to recognize that no security verification can guarantee 100% immunity from vulnerabilities or risks.

Security is a dynamic and ever-evolving field. Even with substantial expertise, it is impossible to predict or uncover all future vulnerabilities. Regular and varied security assessments should be performed throughout the code development lifecycle, and engaging different auditors is advisable to obtain a more robust security posture.

This assessment is limited to the defined scope and does not encompass parts of the system or third-party components not explicitly included. It does not provide legal assurance of compliance with regulations or standards, and the client remains responsible for implementing recommendations and continuous security practices.

Scoping Details

The purpose of the assessment was to conduct Penetration Tests against the **BJust Coin Admin Dashboard** and white-box review of **frontend** repository codebase.

Github repository:

- https://github.com/BJustCoin/frontend
    - Commit hash: 8352463a3d4685a22e789d2b6e7258a1c9c58c3b

Tests were performed on the production environment:

- https://dashboard.bjustcoin.com

The scope of work included a broadly understood analysis of the solution in terms of security. The work against deployed web applications was carried out in a black box methodology, however the user registration was open, which enabled the consultants to create their own user accounts. The only functionality of the platform available during the tests was sending information to the backend about being interested in BJustCoin token purchase. Moreover the consultants received the code repository containing the full frontend application code in order to perform an additional white box testing.

Timeframe

On December 4th 2024, an agreement was signed to conduct a penetration test of the Admin Dashboard web application, and BJustCoin Smart Contracts. The work began on December 5th.

On December 13th 2024, a Draft report from the security assessment was shared with the Client.

Between December 13th 2024, and February 28th 2025, eight rounds of retests were performed.

On February 28th 2025, a Final report from the security assessment was shared with the Client.

Approach and initial access level

The test has been conducted using the **Whitebox** approach. This means that testers were given the access to the **frontend** repository containing web application codebase. Most of the work was done manually, using automatic tools only to speed up individual tasks or to identify minor issues.

Executive Summary

Methodology

Our testing methodology is based on the OWASP Testing Guide, which provides a framework for identifying vulnerabilities in web applications. The OWASP guide is comprehensive and ensures that all potential security issues are considered. Additionally, during our assessment we take into account the OWASP Top Ten document, which is a standard awareness repository for developers and web application security as a whole. This source is updated regularly and provides a list of the ten most common and critical web application security risks.

In parallel, our test criteria were aligned with NIST (National Institute of Standards and Technology) Special Publication 800-53. This facilitated a dual-layered approach to security analysis: the OWASP framework provided granular, technique-specific insights, while NIST offered a broader view centered on risk management and cybersecurity hygiene. The connection of both frameworks ensured that the solution under scrutiny not only met tactical security benchmarks but also aligned with well-regarded risk management strategies.

Customized and hybrid testing approach

Our penetration testing approach employs a blend of manual and automated methodologies for a rigorous evaluation of the application's security posture.

Automated tools are employed to efficiently scan the application for known vulnerabilities and “low-hanging fruits”. This approach ensures that well-documented weaknesses are quickly identified, offering a baseline security assessment that is both fast and broad in scope.

In contrast, our manual approach involves specialized security experts actively exploring the application to identify complex and nuanced vulnerabilities that automated tools may overlook. This hands-on, research-oriented method allows for a more sophisticated and thorough assessment, capable of uncovering vulnerabilities in unique or bespoke elements of the application. By integrating these two methodologies, we achieve a comprehensive evaluation of the application's security posture. The automated testing provides a foundational layer of security assurance, while the manual approach digs deeper into the application's specific context, ultimately leading to the identification of a broader range of

vulnerabilities. Any non-standard techniques or hybrid approaches employed will be explicitly detailed in the report for complete stakeholder transparency.

Risk rating

Once vulnerabilities are identified, we rate their risk and severity using the Common Vulnerability Scoring System (CVSS) 3.1. CVSS provides a standardized method for rating vulnerabilities based on various metrics, such as the ease of exploitation and the potential impact.

The Common Vulnerability Scoring System (CVSS) provides an industry-standard way to define the characteristics and impacts of security vulnerabilities. It gives a numerical score reflecting the severity of a vulnerability based on various metrics.

However, while CVSS provides a technical severity score, it may not always capture the business risk associated with a vulnerability. That is why, after obtaining the CVSS score, we further categorize the vulnerabilities using the OWASP Risk Rating Methodology. This methodology considers factors like the likelihood of exploitation and the business impact to provide a more contextual risk rating.

Step 1: Obtain the CVSS Score
Start by evaluating the vulnerability using the CVSS 3.1 metrics. This will provide a base score that represents the severity of the vulnerability from a technical perspective.

Step 2: Determine the Likelihood and Impact using OWASP Methodology
The OWASP Risk Rating Methodology breaks down risk into two main components: Likelihood and Impact.

Likelihood: This is the probability that a vulnerability will be exploited. Factors influencing likelihood include:

- Threat Agent Factors: Skill level, motive, opportunity, and size of the threat agent.
- Vulnerability Factors: Ease of discovery, ease of exploitation, awareness, and intrusion detection.

Impact: This represents the potential damage that could result from an exploitation. Factors influencing impact include:

- Technical Impact Factors: Loss of confidentiality, loss of integrity, loss of availability, and loss of accountability.
- Business Impact Factors: Financial damage, reputation damage, non-compliance, and privacy violation.

Step 3: Calculate the Risk Score
Once the likelihood and impact have been determined, they can be combined to calculate the overall risk score. The formula is:

Risk Score = Likelihood x Impact

Risk categorization

Based on the calculated risk score, the vulnerability can be categorized into one of the following risk levels:

Critical	Represents an imminent threat to the application, requiring immediate remediation.
High	Indicates a significant threat, suggesting prompt remediation.
Medium	Denotes a moderate threat level, which should be addressed in due course.
Low	Implies a minor threat, which can be scheduled for remediation in the future.
Informational	Represents informational findings that don't pose a direct threat but are worth noting.

Key findings

The purpose of this penetration test was to identify vulnerabilities in the **BJustCoin Application** and assess the potential impacts on BJustCoin business operations and reputation. Our team conducted a comprehensive assessment and identified several vulnerabilities, some of which could have minor implications if exploited.

1. Lack of access control on administrative resources - endpoints verify authorization by `is_signed_in(&session)`, but it doesn't check if the user is administrator and allows any authenticated user to access the administrative pages.
2. The user registration process is insecure and may lead to multiple issues e.g. login with "1:1" credentials.
3. Another major security issue is the lack of password and email policies. It allows the creation of one-character credentials that are very vulnerable to brute forcing.
4. Missing brute-force protection - the application login page process is prone to mass password guessing attempts which means that an attacker might try to guess a user's password in order to gain unauthorized access to their accounts.
5. Leaking of the sensitive information to the server logs - login attempts with credentials are leaked to the server logs by login handler.
6. Lack of the Web Application Firewall makes the application vulnerable to various types of external attacks.

Overall, the state of the security of the assessed solution could be described as **requiring improvement**.

Security Model & Threat Modeling

The security model is used during the threat modeling process to identify all potential assets and actors that may participate in business processes, as well as to verify assumptions with the final version of the solution.

Threat modeling is a systematic process employed to identify, prioritize, and address potential security threats in a given system or environment. It serves as a foundational element in the design and development phase, ensuring that security considerations are integrated from the outset.

The significance of threat modeling lies in its proactive approach, enabling organizations to anticipate and mitigate vulnerabilities before they can be exploited, thereby safeguarding assets and upholding the integrity of operations.

Assets

1. Client instance (Dashboard) - login form on the BJust Coin web application.
2. Client instance (Profile) - allows you to manage account personal details and link the wallet.
3. One main subdomain being in scope: dashboard.bjustcoin.com

Actors

1. Standard users - visitors of the website hosting BJust Coin script, which manages the cookie usage and messages presented to the users related to cookies itself.
2. Tested instance users - users having an access to the BJust Coin platform and could edit all available data, scripts and others, leading to some unspecified consequences.
3. Web crawlers - automated tools that can invoke JavaScript code hosted on the site through web crawling or other actions.

Assumptions

1. All data that is not explicitly declared public should be treated as private and cannot be accessed by any actor outside the permission matrix within the instance.

STRIDE Methodology in Penetration Testing

The STRIDE methodology serves as a comprehensive framework for threat modeling, which is increasingly adapted for penetration testing scenarios. Originating from Microsoft, STRIDE stands for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privileges. This methodology allows penetration testers to systematically analyze potential vulnerabilities and risk vectors in software systems, including decentralized architectures like blockchains and smart contracts.

Components:

- **Spoofing (S)**: Addresses vulnerabilities related to identity spoofing and impersonation. Penetration tests should validate authentication mechanisms to mitigate this risk.
- **Tampering (T)**: Involves the unauthorized alteration of code or data. This is assessed by examining data integrity checks and input validation schemes.
- **Repudiation (R)**: Centers around the inability to deny the occurrence of an event. This involves testing auditing and logging mechanisms to ensure accountability.
- **Information Disclosure (I)**: Concerns the accidental or intentional leaking of sensitive information. Penetration tests focus on data encryption and access controls.
- **Denial of Service (DoS)**: Examines the system's resilience against resource exhaustion. This involves stress testing and resource allocation evaluations.
- **Elevation of Privileges (EoP)**: Addresses the unauthorized escalation of access rights. This involves testing access controls and role-based permissions.

Methodology:

- **Pre-Engagement**: Clearly define the scope and objectives of the penetration test, identifying assets that fall under each STRIDE category.
- **Threat Enumeration**: Use the STRIDE components as a framework for enumerating possible threat vectors. This phase correlates with system decomposition in traditional STRIDE analysis.
- **Vulnerability Mapping**: Identify vulnerabilities corresponding to each STRIDE category. Utilize both automated and manual testing techniques for a comprehensive analysis.
- **Risk Scoring**: Assign severity scores to each identified vulnerability based on industry-standard metrics such as CVSS (Common Vulnerability Scoring System).
- **Exploitation**: Execute controlled attacks to validate the identified vulnerabilities and assess real-world impact.
- **Mitigation and Remediation**: Propose countermeasures for each threat vector. Prioritize them based on their risk scores and potential impact.
- **Report Compilation**: Generate a detailed report outlining the vulnerabilities identified, the data obtained during the exploitation phase, and recommended mitigations.

Component	Threat
Spoofing	• The user can pretend to be another user in order to perform attacks against the identities of others. • Cookies can be leaked or stolen by Cross Site Scripting attacks, leading to active session takeover and performing actions on behalf of a valid user, if not configured properly. • Account takeover may lead to the loss of all assets. • Communication from the BJust Coin platform can be leveraged and forged to perform social engineering attacks.
Tampering	• Possible hardcoded sensitive information could allow an attacker to perform some serious actions on behalf of the user's account, such as withdrawing funds.

	• Possibility of injecting malicious payload at any point controlled by the client may lead to induce the attack
Repudiation	• Lack of sufficient logging and monitoring may allow users to deny actions they took within the application, as there is no robust mechanism to associate a user with performed actions conclusively. • Threat actors may overwrite or block the application from logging their actions.
Information Disclosure	• The users can try to access data of other instances e.g. by referencing accounts or IDs of other users when signing up. • The application may leak sensitive technical data such as application components code. • Private, sensitive information belonging solely to certain users might be readable by others. • Potential user enumeration can lead to obtaining a list of registered users and open a way for brute force attacks. • Passwords brute force attacks can potentially be executed if there is no security mechanism in place.
Denial of Service	• Database and storage of the application might be filled with dummy data, leading to inability to take more from the user, making the platform unusable in a short period of time. • Vulnerable JS libraries or backend frameworks might lead to Denial of Service scenarios. • Potential lack of firewall may lead to Denial of Service.
Elevation of privileges	• Administrator authentication could be bypassed by brute forcing possible passwords. • Possible accessing administrative pages without being administrator of the web application. • Access to the resources that belong to another user and should be restricted from being accessed by others.

Vulnerabilities summary

No.	Severity	Name	Status
1	Critical	Any user can log into another user's account, knowing only their email address	**Resolved**
2	Critical	Lack of access control on administrative resources	**Resolved**
3	High	The user registration process is insecure and may lead to multiple issues	**Resolved**
4	Medium	Missing brute-force attacks protection in login form	**Resolved**
5	Medium	Application leaks sensitive technical information to server logs	**Resolved**
6	Medium	Lack of Web Application Firewall (WAF) mechanism usage	**Resolved**
7	Low	Lack of validation of the special characters	**Resolved**
8	Low	Missing Subresource Integrity hashes	**Acknowledged**
9	Low	Missing security headers	**Acknowledged**
10	Low	Logout button does not work properly	**Acknowledged**
11	Low	Create application for ICO tokens for another user’s email address	**Resolved**
12	Informative	Software version disclosure	**Acknowledged**
13	Informative	Directory listing is enabled	**Acknowledged**
14	Informative	Vulnerable dependency in use	**Acknowledged**

Technical summary

1. Any user can log into another user's account using any password

Severity

CRITICAL

CVSS Score: 10.0

`CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:H/A:H`

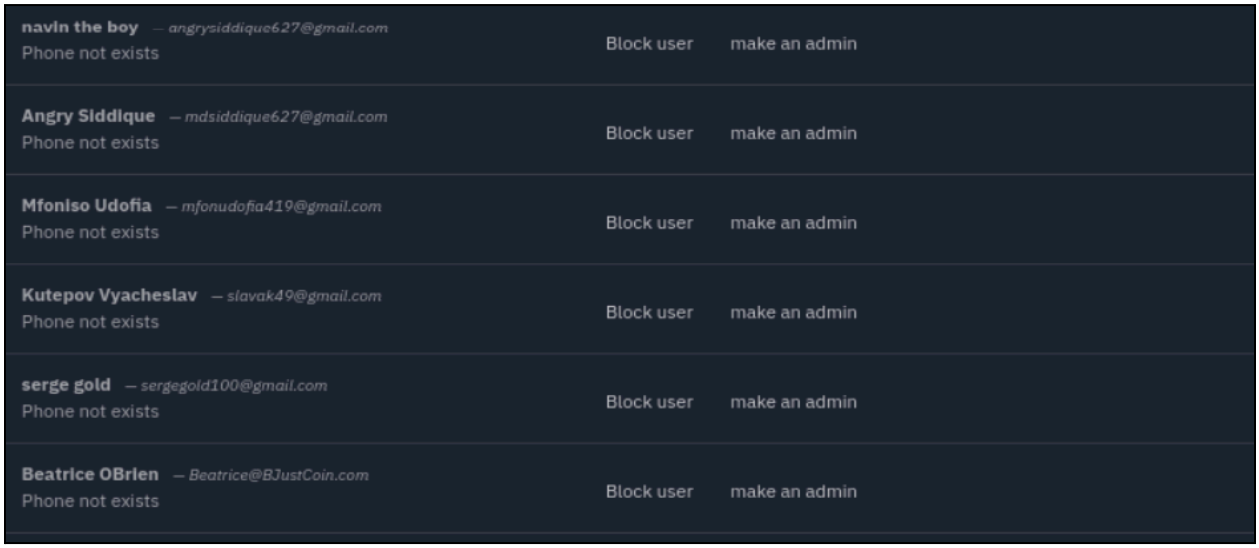
Description

Incorrectly implemented authentication leads to a situation where a user who only knows the login, in this case the email address of the registered user, can log in to his account using any password.

For example, an email address taken from the `/users/` administrative path:

- email address: `mdsiddique627@gmail.com`
- password: `Password12`

A snippet of list of users:



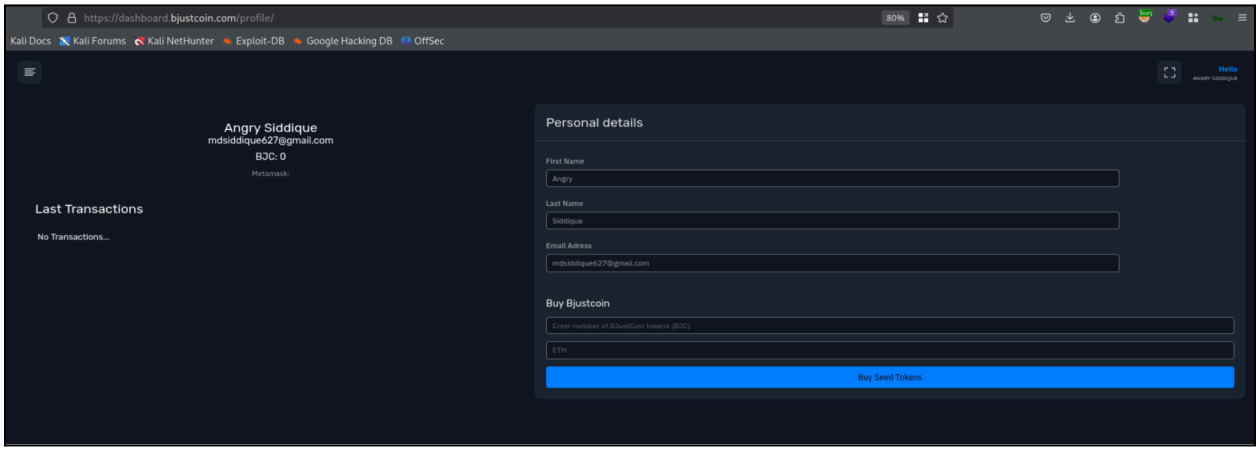
Request sent to the server:

```
POST /login/ HTTP/1.1

Host: dashboard.bjustcoin.com
Cookie: _gcl_au=1.1.1321471271.1737219315
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 59
Origin: https://dashboard.bjustcoin.com
Referer: https://dashboard.bjustcoin.com/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0
Te: trailers
Connection: keep-alive

{
"email":"mdsiddique627@gmail.com",
"password":"Password12"
}
```

Response:

```
HTTP/1.1 200 OK

Server: nginx/1.18.0 (Ubuntu)
Date: Sun, 19 Jan 2025 16:19:03 GMT
Content-Type: application/json
Content-Length: 16
Connection: keep-alive
set-cookie:
id=x2e5HDhwAezNywDv08+pzSXrFWL5xWvnpj368Zr0gyUmPome9YFfWcXnWxjCJHzZSe1yRzzY1f+Fnnqj
FscYvrdANXL6UyDAfR0%2FXPXMkE6I5WhYiAgE4Uy7SEqkL%2FI4Y4ROef1Y3vvpWreUGv+ktMqAw0t5l0V
MosMxseAL15c%2FN0nV89fyR+mlo%2Fsmz4SOZCJVi4GA2z2+HfNwZ3lBDDz89pOIrrdJi11UiG2tuSe8i5
1nFXQYLO2gkJJ+pVOhWPmomSVCYAnkUSUOuzJo13KF4GxKpofKOC8CuP5ewXxblhz3Gh4IgMYo3a%2F2Cu8
MuQnz3LF7sTGsRaru6G+RWpMrS93AWfkJ47h5u65X3V+gcegqqYJD1kG08h9XfRnl2Xcwu6IXjqR%2FVryI
h0UzJUPSMHY9JwILLaORqW5CxBfqEPOlByAKtqTz8%2Ffqp1U+Xr6I7+iUrhtBvk4GLP4NggI%3D;
HttpOnly; SameSite=Lax; Path=/

{"status":"ok!"}
```

Dashboard of the user `mdsiddique627@gmail.com`:



Remediation

A key requirement of authentication is that it is reliable and correctly matches the provided password with the password set by the user. We recommend implementation of proper authentication mechanisms.

Retest (26.02.2025)

The account authentication mechanism has been updated and implemented correctly, resulting in a failure to authenticate a user who enters an incorrect password.

For example, an email address taken from the list of users in the snippet above:

- email address: `slavak49@gmail.com`
- password: `Verysecurepassword12`

Request sent to the server:

```
POST /login/ HTTP/1.1
Host: dashboard.bjustcoin.com
Cookie: _gcl_au=1.1.1321471271.1737219315
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 67
Origin: https://dashboard.bjustcoin.com
Referer: https://dashboard.bjustcoin.com/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0
Te: trailers
```

```
Connection: keep-alive

{
"email":"slavak49@gmail.com",
"password":"Verysecurepassword12"
}
```

Server response:

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Wed, 26 Feb 2025 15:49:18 GMT
Content-Type: application/json
Content-Length: 56
Connection: keep-alive

{
"status":"The email address or password is incorrect."
}
```

Status: Resolved

---

2. Lack of access control on administrative resources

Severity

CRITICAL

CVSS Score: 9.0

`CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:H/A:H`

Description

The application's administrative interface at `src/views/admin_pages.rs` lacks authorization controls. While endpoints require authentication through `is_signed_in(&session)`, they don't verify administrative privileges. This allows any authenticated user to access sensitive administrative functionalities such as invoices, admin pages, or administrative settings.

Any authenticated user can bypass authorization by directly accessing admin URLs. The application exposes over 20 administrative endpoints. This could lead to unauthorized access to sensitive data and administrative functions.

Below is the sample code that controls access to one of administrative features in `admin_pages.rs`:

```
pub async fn admin_wallets_page(session: Session) ->
actix_web::Result<HttpResponse> {
    if !is_signed_in(&session) {
        return HttpResponse::Unauthorized().finish();
    }

    let user = get_current_user(&session).map_err(|_|
HttpResponse::Forbidden().finish())?;
    if !user.is_admin {  // Add proper role check
        return HttpResponse::Forbidden().finish();
    }

    // ... proceed with admin functionality
}
```

The actual access control is enforced in `src/utils/auth.rs` (lines 34-45):

```
pub fn get_current_user(session: &Session) -> Result<AuthResp, AuthError> {
    let msg = "Error";
    session
        .get::<String>("user")
        .map_err(|_| AuthError::AuthenticationError(String::from(msg)))
        .unwrap()
        .map_or(
          Err(AuthError::AuthenticationError(String::from(msg))),
          |user| serde_json::from_str(&user).or_else(|_|
Err(AuthError::AuthenticationError(String::from(msg))))
        )
}
```

As per the above listing, the application checks if the user is logged in, but not if the user has proper privileges, or in any other way, if the user is the admin.

Remediation

We recommend implementation of proper authorization checks such as:

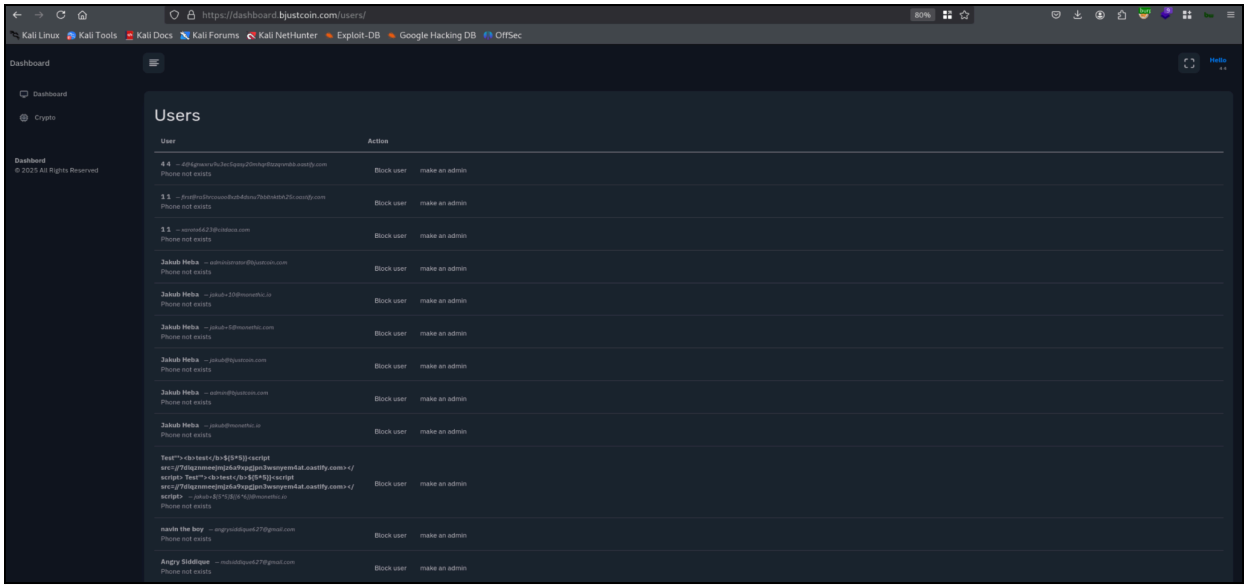
```
// Add role check to existing authentication
if is_signed_in(&session) && user.role == UserRole::Admin {
    // Process admin request
} else {
    // Return 403 Forbidden
    return HttpResponse::Forbidden().finish();
}
```

Consider implementing role-based access control (RBAC) middleware to handle authorization consistently across all admin routes.

Retest (19.01.2025)

Access control to administrative resources is implemented, but very poorly and inadequately. A user who signs in with the correct verification code has access to the administration panels. They can also perform actions for other users, e.g. `make an admin` functionality, which should only be available to the administrator.

The user `xaroto6623@citdaca.com` has been created with a valid verification code and has access to the administration panels, e.g. via the `/users/` path, which displays all users registered on the platform.



An authenticated user can click on the `make an admin` button and a query will be made to the server, which will return a result of 200 OK, but in the body of the response there will be an ERR, which creates further potential attacks, as the server responded with a 200 HTTP status code.

Example of the request made to the server:

```
POST /create_admin/ HTTP/1.1

Host: dashboard.bjustcoin.com
Cookie: [REDACTED]
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 10
Origin: https://dashboard.bjustcoin.com
```

```
Referer: https://dashboard.bjustcoin.com/users/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0
Te: trailers
Connection: keep-alive

{"id":101}
```

Example of the response:

```
HTTP/1.1 200 OK

Server: nginx/1.18.0 (Ubuntu)
Date: Sun, 19 Jan 2025 15:06:44 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 3
Connection: keep-alive

err
```

Retest (26.02.2025)

Previously implemented access control has been updated and improved. Correctly created users now don't have access to administrative resources such as `/users/` or `/admins/`.
When an authenticated user attempts to access the restricted resource - the server responds with status 200, but actually redirects to the dashboard page with no access to the administrative resource.

```
GET /users/ HTTP/1.1
Host: dashboard.bjustcoin.com
Cookie: [REDACTED]
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/p
ng,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Wed, 26 Feb 2025 16:07:24 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
Content-Length: 12769

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">
    <link rel="icon" href="/assets/images/favicon.ico">
    <title>Dashboard</title>

[. . .]
```

Status: **Resolved**

---

3. The user registration process is insecure and may lead to multiple issues

Severity

HIGH

CVSS Score: 8.8

`CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H`

Description

The application at `https://dashboard.bjustcoin.com/` implements an insecure user registration process that lacks basic validation and verification controls. Users can create accounts using any credentials, including arbitrary email addresses, without any verification. The registration process completes successfully with minimal input such as "1:1" as credentials.

- The absence of proper registration controls creates several security and operational risks.
- Accounts can be created with unverified or incorrect email addresses, preventing legitimate password recovery.
- Malicious actors can create accounts using others' email addresses, potentially leading to identity confusion and unauthorized access.
- The system is vulnerable to automated account creation, which could overwhelm system resources and create database pollution.
- It should be also noted that the user was not prevented from signing up with a single character instead of an email address.

Example of the request with the following login credentials:

- **email: 1**
- **password: 1**

```
POST /login/ HTTP/1.1
Host: dashboard.bjustcoin.com
Cookie: _gcl_au=1.1.1141853453.1733396594;
_ga_3YNSFDY25J=GS1.1.1733398060.1.1.1733399633.0.0.0;
SL_C_23361dd035530_SID={"0563212d2a846816aba8acd0191234c3acbfa214":{"sessionId":"-C
KJASKGUPGlLFrlk5L2n","visitorId":"XeZViQT2vdtG4Id-O5L6R"}}
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 28
Origin: https://dashboard.bjustcoin.com
Referer: https://dashboard.bjustcoin.com/
```

```
Te: trailers
Connection: keep-alive


{"email":"1","password":"1"}
```

Response from the server:

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Thu, 12 Dec 2024 20:25:44 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
set-cookie:
id=HySMXIvcT3tO8uVu5ByAjaA00psysUzXUF8yRsiCmmW9oAaNhrlAYR9OIRbw8MwZbKCKu5yzI3pfMhEu
afPBlbfubYW6U+hJc7WuTVPllZcVjr0LPoFV0hAvj6RXpOp0QrojwFdKukjRvSjfYTwY9qGwUV09iRq33tO
VKQjLSoiY%2Fkmhuli8UDQwq0o2rV0FvDjaX3Le; HttpOnly; SameSite=Lax; Path=/
Content-Length: 18803

[REDACTED]
```

Remediation

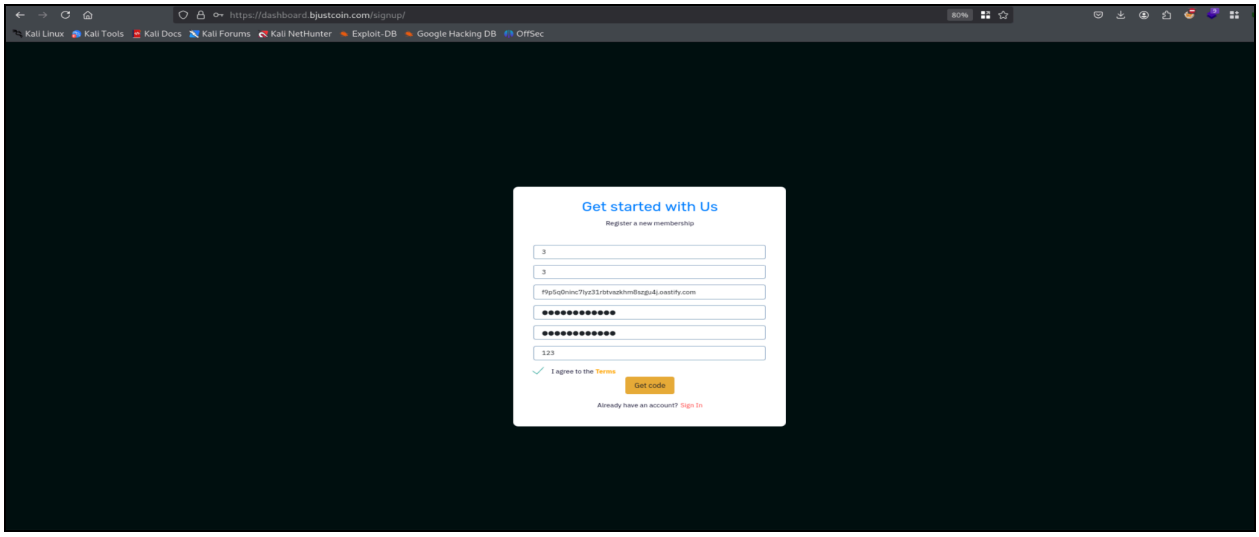
We recommend implementation of secure registration process that includes:

- Email verification through confirmation links or codes,
- Strong password requirements and validation (minimum 12 characters),
- Rate limiting on registration attempts,
- CAPTCHA or similar anti-automation measures,
- Account recovery mechanism through verified email addresses.

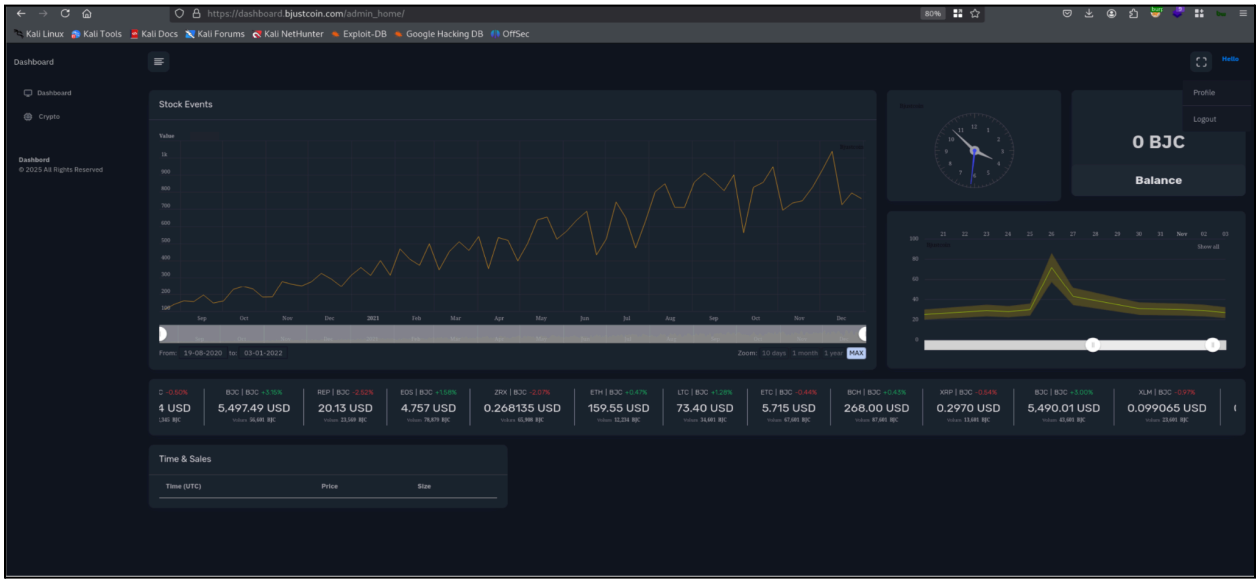
Additionally, we recommend adapting email policy to prevent users from creating single character or short character account logins. The email policy should contain at least the '**@**' character and the domain of the mail server.

Retest (19.01.2025)

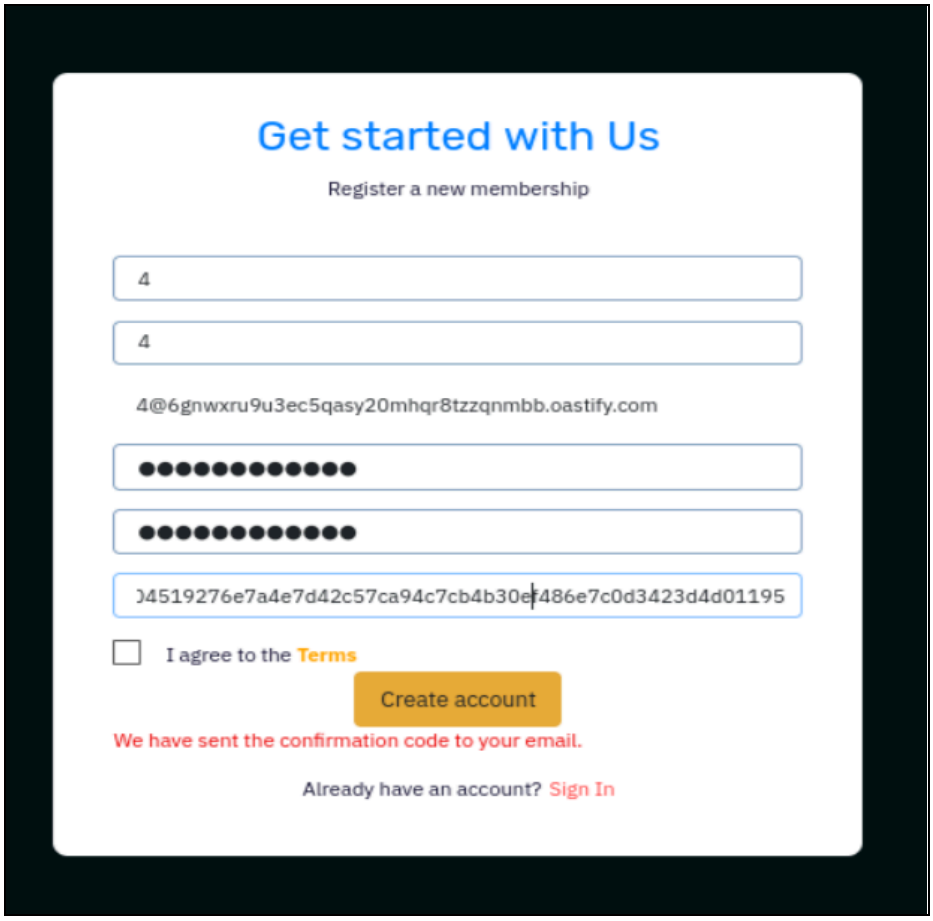
The process of creating a user is still dangerous because if the user enters the wrong code, he or she will still be able to access and browse the platform, e.g. access the `/admin_home/` path.
No account is created, but the user can check the paths and map the application without creating an account. They will not be able to perform any actions on this account, e.g. access the /users/ path - which in itself should be reserved for the administrator, but is available to any user who has entered the correct verification code.
The example of the user who passed `123` verification code in the registration process:



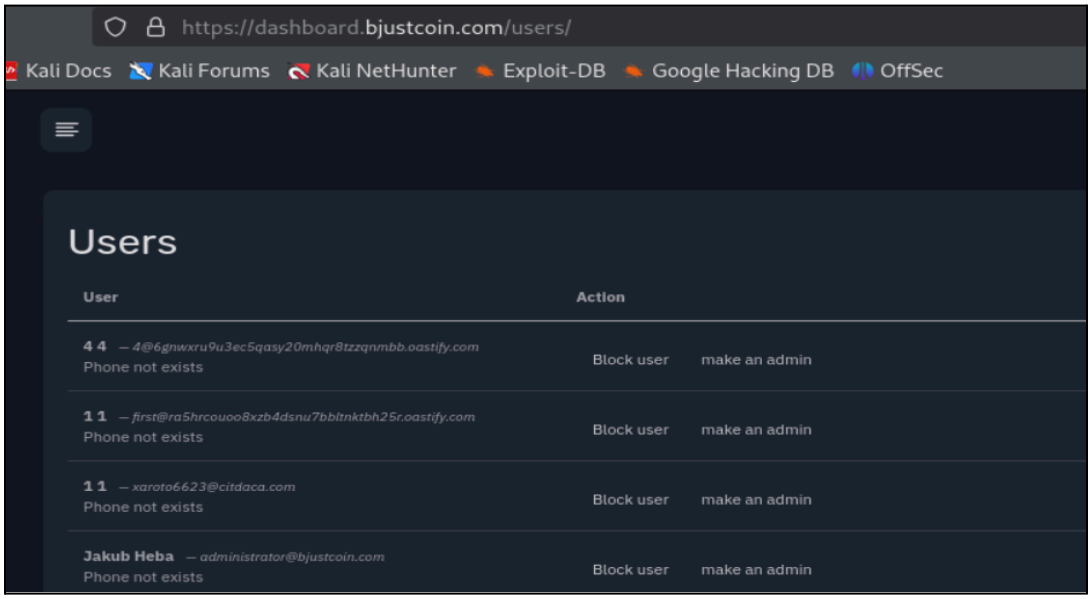
The example of access to the `/admin_home/` path with user created with incorrect certification code:



It is also possible to create a user without checking the terms and conditions.



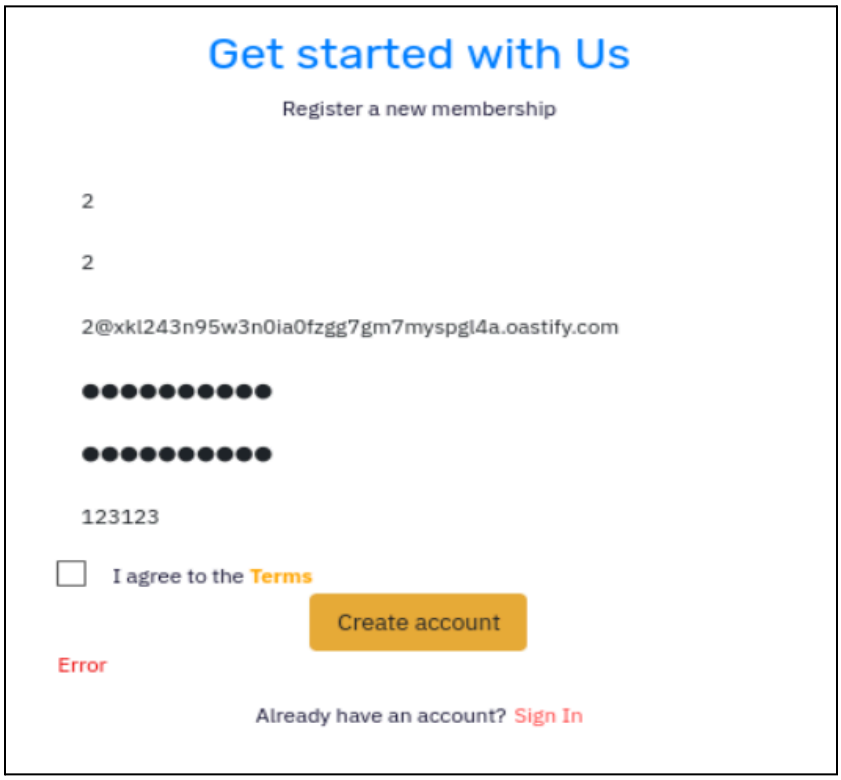
The user `4@6gnwxru9u3ec5qasy20mhqr8tzzqnmbb.oastify.com` was successfully created:



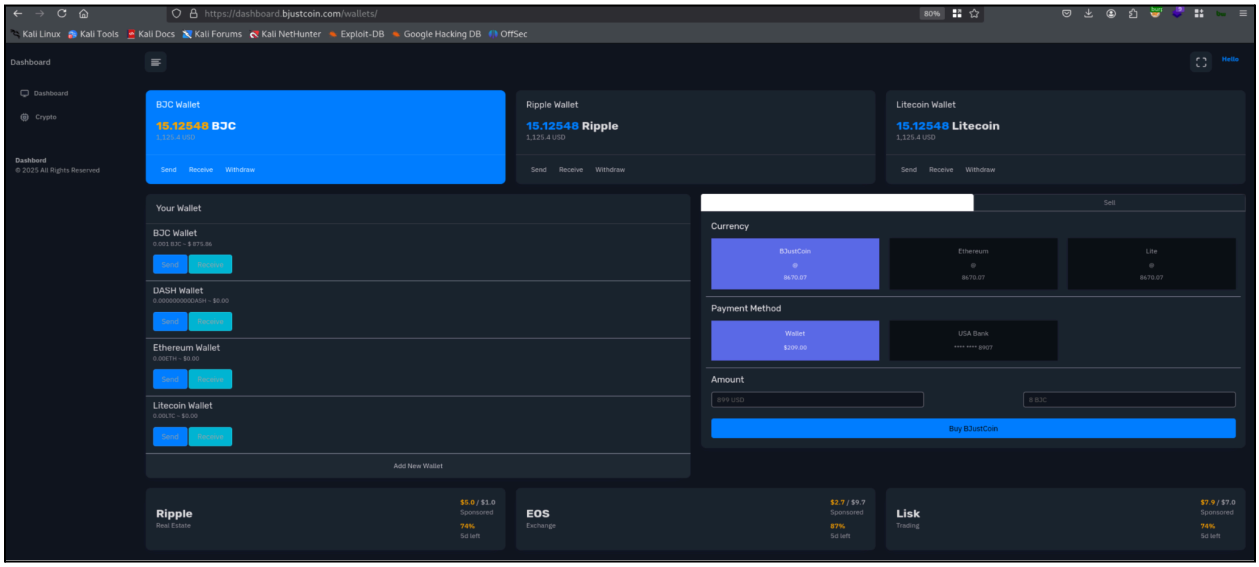
Retest (26.02.2025)

The process of creating a user is still dangerous because if the user enters the wrong code, they will still be able to access and browse the platform, e.g. access the `/admin_home/` path.

The example of the user who passed `123123` verification code in the registration process:



The message from the server indicates that there is an error but the user can check the paths and map the application without creating an account e.g. `/wallets/` or `/setting/`.



The application also does not check if an account already exists for the email, and the system sends the activation code to the email address used.

First, the user creates an account on the email address: `retest@y653q49arxpomjw1102hth888zeq2pqe.oastify.com` and get the activation code.

```
Our confirmation code - <strong>4804115a6203ed4f010aab7c95af55474ceb26cdbc5d76661b6db9351a427277</strong><img
```

Request to the server with obtained code to create new account:

```
POST /signup/ HTTP/1.1
Host: dashboard.bjustcoin.com
Cookie: _gcl_au=1.1.1321471271.1737219315
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 205
Origin: https://dashboard.bjustcoin.com
Referer: https://dashboard.bjustcoin.com/signup/
Te: trailers
Connection: keep-alive

{
"first_name":"Retest",
"last_name":"Retest",
"email":"retest@y653q49arxpomjw1102hth888zeq2pqe.oastify.com",
"password":"Password12",
"token":"4804115a6203ed4f010aab7c95af55474ceb26cdbc5d76661b6db9351a427277"
}
```

Response from the server:

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Wed, 26 Feb 2025 20:46:29 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
set-cookie:
id=cKL5+7aO+0xKfgmJL+Wg99KeJ5alm+xfI8KNhlh8eWfymIDM6ybZpsddZJRTfg86d+uHIgcoeg3Y0FgI
OGMUnrFRfi7Wxvc47ONcs+IWs6hK6ayNJ5hSPbCoymmN8Iw9kY5Qpc+IPe3GM3TMtu7RpU4CyGY3DUieOdC
GR3bd1uatiSHAT%2FCRJaByn8EtyqFCV%2FS32gdSL7LE2mfxCe2VNm3hfUCz11%2FcsNtGYS25FsnIiOFH
q7ak27Jg5P%2FDtUTFzQXhIEDCkZGF5LEGMb1ZDn%2FbVCZfsDqFNyQcO9QOufHmW2a%2FeIRvKZNR1+3RA
trw6+EK+y0Ptz8XPGkn2iBwQaSToesTwLiFJrjMWwQfIYr89ENC4bft4if0mmHlTu7fkY5eZjZsWHlZiulP
GeVbXrKK9u1e5ATDsoJUbdyiUePkLnMynglphijBdCAFP1LjaAi1JJs7XZWgj93Tb6uHQB9AkvghZwWPhQ%
```

```
3D%3D; HttpOnly; SameSite=Lax; Path=/
Content-Length: 12761

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">
    <link rel="icon" href="/assets/images/favicon.ico">
    <title>Dashboard

[. . .]
```

Secondly, the user tries to create a new account on the same email address: `retest@y653q49arxpomjw1102hth888zeq2pqe.oastify.com` and get another activation code.

```
Our confirmation code - <strong>d0d309ca779b6a4810e0a21c9af380c793617fc12006fbaf6eed3d0e06f761e3</strong><img
```

Request to the server with new code:

```
POST /signup/ HTTP/1.1
Host: dashboard.bjustcoin.com
Cookie: _gcl_au=1.1.1321471271.1737219315
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 214
Origin: https://dashboard.bjustcoin.com
Referer: https://dashboard.bjustcoin.com/signup/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0
Te: trailers
Connection: keep-alive

{
"first_name":"NextRestest",
"last_name":"NextRetest",
"email":"retest@y653q49arxpomjw1102hth888zeq2pqe.oastify.com",
```

```
"password":"Password12",
"token":"d0d309ca779b6a4810e0a21c9af380c793617fc12006fbaf6eed3d0e06f761e3"
}
```

Response from the server:

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Wed, 26 Feb 2025 21:06:42 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 3
Connection: keep-alive

err
```

The response from the server shows that the new account has not been created, but the new activation code has been sent to the user without checking if the email address is in use.

Retest (27.02.2025)

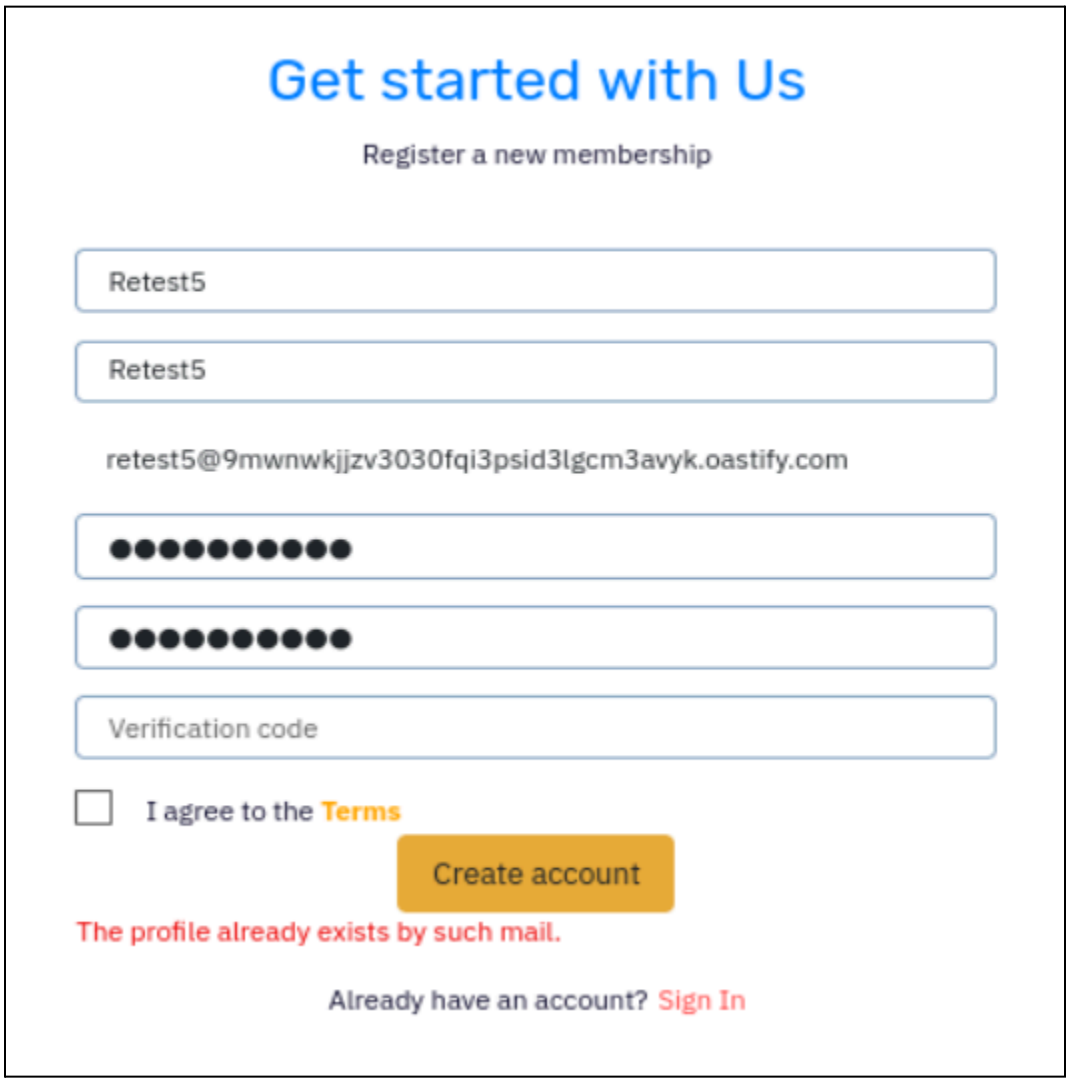
The user creation process has been improved and now redirects to the login page if an incorrect verification code is entered as shown in the request/response to the server.

```
POST /signup/ HTTP/1.1
Host: dashboard.bjustcoin.com
Cookie: _gcl_au=1.1.1321471271.1737219315
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 156
Origin: https://dashboard.bjustcoin.com
Referer: https://dashboard.bjustcoin.com/signup/
Priority: u=0
Te: trailers
Connection: keep-alive

{
"first_name":"Retest",
"last_name":"Retest",
"email":"retest321@9mwnwkjjzv3030fqi3psid3lgcm3avyk.oastify.com",
"password":"Password12",
"token":"123132132212"
}
```

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Thu, 27 Feb 2025 18:08:00 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 0
Connection: keep-alive
```

The server responds with status 200 OK, but with no content with the header `Content-Length: 0`.

It also checks the user's email address and if the email is used for another account, the verification code will not be sent to that account.



Status: **Resolved**

4. Missing brute-force attacks protection in login form

Severity

MEDIUM

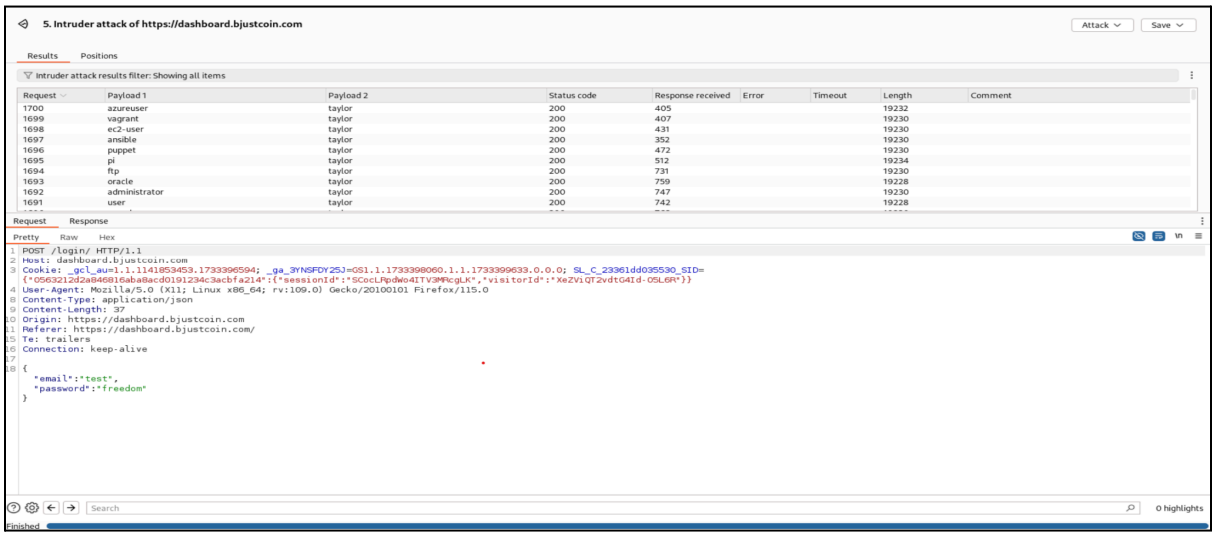
CVSS Score: 5.4

`CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:L/I:L/A:N`

Description

During the penetration test, it was observed that the target application does not implement adequate protections against brute-force attacks. Specifically, an attacker could send a large number of incorrect login attempts without the target account being locked or rate-limited. This vulnerability allows malicious actors to attempt multiple password guesses, potentially leading to unauthorized access if a weak password is used for the account.

An attacker has to intercept the login request in Burp Suite and then sends it to Burp Intruder. Next he has to place a password or email between **§§** signs as a place of injection and start the attack. View from the BurpSuite Intruder tool, showing that 1700 requests were done during a short amount of time.



Remediation

We recommend implementation of an account lockout mechanism where, after a certain number of consecutive failed login or “change password” attempts (e.g., five attempts), the account is temporarily locked for a specific period (e.g., 30 minutes). This will deter attackers from trying multiple passwords in a short period. In order to prevent DoS attack vectors, the lock could be communicated to the user's email, and clicking a specific link would clear the lockout.
Additionally, a CAPTCHA mechanism can be implemented on the login feature to limit the amount of possible login attempts.

Retest (19.01.2025)

The account lockout mechanism is not implemented and the login form is still vulnerable to brute force attacks. We were able to make over 500 requests using the same email address in a short period of time.

6. Intruder attack of https://dashboard.bjustcoin.com

Attack | Save

Results | Positions

Intruder attack results filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
508	sony	200	4223			724	
507	smilebig!	200	4136			714	
506	smha13	200	4228			722	
505	smart1985	200	4237			718	
504	sleepless	200	4188			714	
503	silver520932	200	4202			722	
502	shaunamaloney	200	4290			716	
501	shahrooz	200	4278			712	
500	sexyback	200	4111			722	
499	sergioteamo	200	4245			712	
498	sept12	200	4267			716	

Request | Response

Pretty | Raw | Hex

```
POST /login/ HTTP/1.1
Host: dashboard.bjustcoin.com
Cookie: _gcl_au=1.1.1321471271.1737219315
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 58
Origin: https://dashboard.bjustcoin.com
Referer: https://dashboard.bjustcoin.com/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0
Te: trailers
Connection: keep-alive

{
  "email":"mdsiddique627@gmail.com",
  "password":"smart1985"
}
```

Retest (26.02.2025)

The account lockout mechanism is not implemented and the login form is still vulnerable to brute force attacks. We were able to make over 500 requests using the same email address in a short period of time.

Attack | Save

4. Intruder attack of https://dashboard.bjustcoin.com

Attack | Save

Results | Positions

Capture filter: Capturing all items | Apply capture filter

View filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
507	marina	200	4124			219	
506	147852	200	4209			219	
505	stupid	200	4224			219	
504	claire	200	4180			219	
503	linkinpark	200	4152			219	
502	marlon	200	4188			219	
501	turtle	200	4164			219	
500	xavier	200	4211			219	
499	spiders	200	4235			219	

Payloads | Resource pool | Settings

Request | Response

Pretty | Raw | Hex

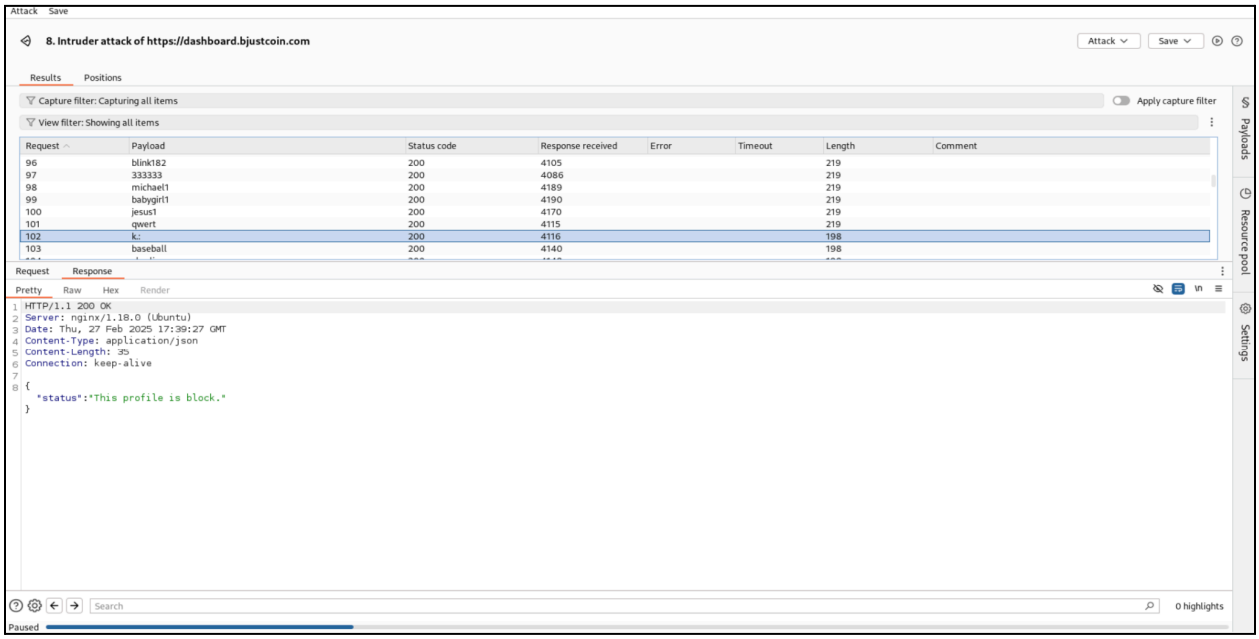
```
POST /login/ HTTP/1.1
Host: dashboard.bjustcoin.com
Cookie: _gcl_au=1.1.1321471271.1737219315
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 82
Origin: https://dashboard.bjustcoin.com
Referer: https://dashboard.bjustcoin.com/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0
Te: trailers
Connection: keep-alive

{
  "email":"test4@yeg3y4hazxxouj4190ah1hg8gzmqanyc.oastify.com",
  "password":"xavier"
}
```

Search | 0 highlights

Paused

Retest (27.02.2025)

The account lockout mechanism is implemented and after 101 invalid login attempts the application will lock the profile. We recommend reducing the number of invalid login attempts before suspending the account.



Status: Resolved

5. Application leaks sensitive technical information to server logs

Severity

MEDIUM

CVSS Score: 6.3

`CVSS:3.1/AV:L/AC:H/PR:H/UI:R/S:U/C:H/I:H/A:H`

Description

The application logs sensitive authentication credentials during login attempts. In the `login` handler function located at `src/views/auth_pages.rs` (lines 59-73), user passwords are printed to server logs in plaintext.

```
pub async fn login(session: Session, data: Json<LoginUser>) ->
actix_web::Result<HttpResponse> {
    if is_signed_in(&session) {
        return crate::views::not_found_page(session).await;
    }
    let l_data = LoginUser {
        email:    data.email.clone(),
        password: data.password.clone(),
    };
    println!("email: {:?}", data.email.clone());
    println!("password: {:?}", data.password.clone());
    let res = request_post::<LoginUser, AuthResp> (
        URL.to_owned() + &"/login/".to_string(),
        &l_data,
        false
    ).await;
```

While likely implemented for debugging purposes, this creates a security risk as server logs are often stored unencrypted and may be accessible to various personnel or attackers who gain system access.

Password exposure in logs can lead to unauthorized account access if logs are compromised. The risk is amplified by common password reuse across services. Additionally, leaked credentials in logs may violate data protection regulations and expose sensitive data during log shipping to monitoring or backup systems.

Remediation

We recommend removing all password logging from the application. For debugging purposes, log only authentication events and outcomes without credentials.

Retest (19.01.2025)

This issue has been fixed by changing the indicated lines of code as shown below in the `src/views/auth_pages.rs` (lines 103-117).

```
pub async fn login(session: Session, data: Json<LoginUser>) -> Json<Resp> {
    if is_signed_in(&session) {
        return Json(Resp {
            status: "error".to_string(),
        });
    }
    let l_data = LoginUser {
        email:    data.email.clone(),
        password: data.password.clone(),
    };
    let res = request_post::<LoginUser, AuthResp2> (
        URL.to_owned() + &"/login/".to_string(),
        &l_data,
        "".to_string()
    ).await;
```

Status: **Resolved**

---

6. Lack of Web Application Firewall (WAF) mechanism usage

Severity

MEDIUM

CVSS Score: 5.8

`CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:C/C:L/I:L/A:L`

Description

During security testing of `dashboard.bjustcoin.com`, which has a `67.220.95.91` IP address, it was discovered that there was no Web Application Firewall (WAF) protection in place. In the present threat landscape, automated tools and bots continuously scan websites every day, making WAF protection essential for basic security hygiene. Without this fundamental protection layer, the server is exposed to various automated attacks and reconnaissance attempts. It is worth noting that the domain listed above was **production** one, not staging, dev or testing environment.

This absence of WAF enabled comprehensive port scanning of the server, revealing open ports and service versions. Using basic scanning tools, it was possible to enumerate running services and their specific versions, which could aid attackers in identifying targeted attack vectors. Additionally, the lack of

WAF protection made the application vulnerable to special character injection due to insufficient input filtering.

Below list shows results from a nmap port scan of the application server:

```
PORT     STATE SERVICE    VERSION
22/tcp   open  ssh        OpenSSH 8.9p1 Ubuntu 3ubuntu0.10(Ubuntu Linux; protocol 2.0)
25/tcp   open  smtp?
53/tcp   open  domain     PowerDNS Authoritative Server 4.9.2
80/tcp   open  http       nginx 1.18.0 (Ubuntu)
110/tcp  open  pop3       Dovecot pop3d
111/tcp  open  rpcbind    2-4 (RPC #100000)
143/tcp  open  imap       Dovecot imapd
443/tcp  open  ssl/http   nginx 1.18.0 (Ubuntu)
465/tcp  open  ssl/smtp   Exim smtpd 4.98
587/tcp  open  smtp       Exim smtpd 4.98
993/tcp  open  imaps?
995/tcp  open  pop3s?
1720/tcp open  tcpwrapped
5060/tcp open  tcpwrapped
8010/tcp open  ssl/xmpp?
9999/tcp open  abyss?
```

Remediation

We recommend implementing Cloudflare or similar services for the domain to provide WAF protection and additional security features. Direct IP address access to the application should be disabled, with all traffic routed through Cloudflare. This configuration will help prevent port scanning attempts and protect against malicious input including special character injection attacks.

Retest (19.01.2025)

The basic nmap versions and scripts scans show that the 995 ports are filtered, which means that some kind of WAF is implemented.

```
Not shown: 995 filtered tcp ports (no-response)
PORT     STATE SERVICE    VERSION
22/tcp   open  tcpwrapped
|_ssh-hostkey: ERROR: Script execution failed (use -d to debug)
53/tcp   open  tcpwrapped
80/tcp   open  tcpwrapped
|_http-server-header: nginx/1.18.0 (Ubuntu)
443/tcp  open  tcpwrapped
| ssl-cert: Subject: commonName=ai.justlaw.network
| Subject Alternative Name: DNS:ai.justlaw.network
| Not valid before: 2024-12-08T06:17:53
```

```
|_Not valid after:  2025-03-08T06:17:52
|_ssl-date: TLS randomness does not represent time
|_http-server-header: nginx/1.18.0 (Ubuntu)
8080/tcp open  tcpwrapped
```

Status: **Resolved**

---

7. Lack of validation of special characters

Severity

LOW

CVSS Score: 3.1

`CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N`

Description

The web application has no mechanism to validate or filter the characters and strings entered. It is possible to enter any string, e.g. `<script>alert(document.domain)</script>` or `" OR 1 = 1 -- -"`. Such an environment is highly vulnerable to all kinds of client-side and server-side attacks. Without character filtering, the attacker can type any string, which can either execute the code or the intended attack, depending on the attack.

Example of the request/response in the login form:

```
POST /login/ HTTP/1.1
Host: dashboard.bjustcoin.com
Cookie: _gcl_au=1.1.1141853453.1733396594;
_ga_3YNSFDY25J=GS1.1.1733398060.1.1.1733399633.0.0.0;
SL_C_23361dd035530_SID={"0563212d2a846816aba8acd0191234c3acbfa214":{"sessionId":"-C
KJASKGUPGlLFrlk5L2n","visitorId":"XeZViQT2vdtG4Id-O5L6R"}};
id=VIx%2FK9t5SCpYPo9cwrqhJ6RL4bNrCLnWqe3TdiK2EGdssq7qP6QNQoSCvoB7R43r1nd261D7friU3W
W6nDxNPJJzl56XyJermYs8zC5CtMo+dxxnQ9QbGKMCXJS1NZqsCHPJCiBfjvCWflHoxy78OV4ix4Rl3usFl
5onngbMiH7Lw77LCPohgChWk6csWzxCBqYe72gu
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 73
Origin: https://dashboard.bjustcoin.com
```

```
Referer: https://dashboard.bjustcoin.com/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers
Connection: keep-alive

{"email":"<script>alert(document.domain)</script>",
"password":"' or 1=1"}
```

Server response:

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Thu, 12 Dec 2024 21:00:00 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
Content-Length: 7499

[REDACTED]
```

Example of the request/response in the signup form with the following payloads:

- first_name - Os Command Injection,
  - `;system('cat%20/etc/passwd')`
- last_name - Cross Site Scripting,
  - `<script\\x20type=\"text/javascript\">javascript:alert(1);</script>`
- email - SQL Injection.
  - `" OR 1 = 1 -- -"`

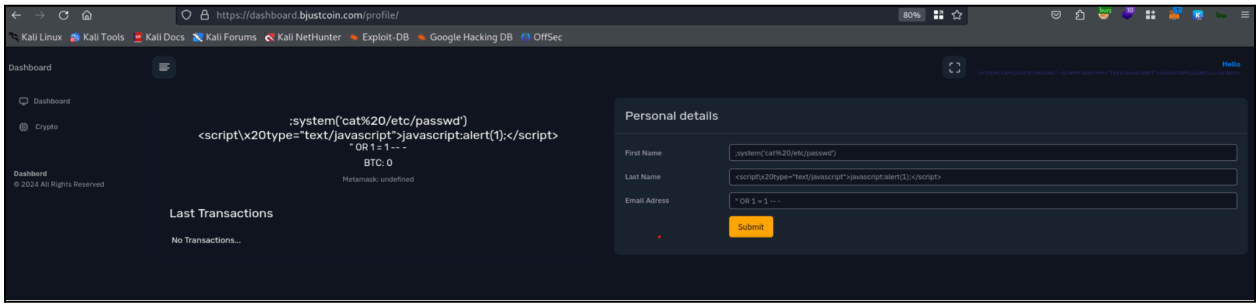
```
POST /signup/ HTTP/1.1
Host: dashboard.bjustcoin.com
Cookie: _gcl_au=1.1.1141853453.1733396594;
_ga_3YNSFDY25J=GS1.1.1733398060.1.1.1733399633.0.0.0;
SL_C_23361dd035530_SID={"0563212d2a846816aba8acd0191234c3acbfa214":{"sessionId":"-C
KJASKGUPGlLFrlk5L2n","visitorId":"XeZViQT2vdtG4Id-O5L6R"}}
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 168
Origin: https://dashboard.bjustcoin.com
Referer: https://dashboard.bjustcoin.com/signup/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
```

```
Sec-Fetch-Site: same-origin
Te: trailers
Connection: keep-alive


{"first_name":";system('cat%20/etc/passwd')",
"last_name":"<script\\x20type=\"text/javascript\">javascript:alert(1);</script>",
"email":"\" OR 1 = 1 -- -",
"password":"2"}
```

Server response:

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Thu, 12 Dec 2024 20:57:01 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
Content-Length: 7499

[REDACTED]
```

The injected strings are also reflected in the personal details and rendered by the frontend mechanisms in the `/profile/` directory.

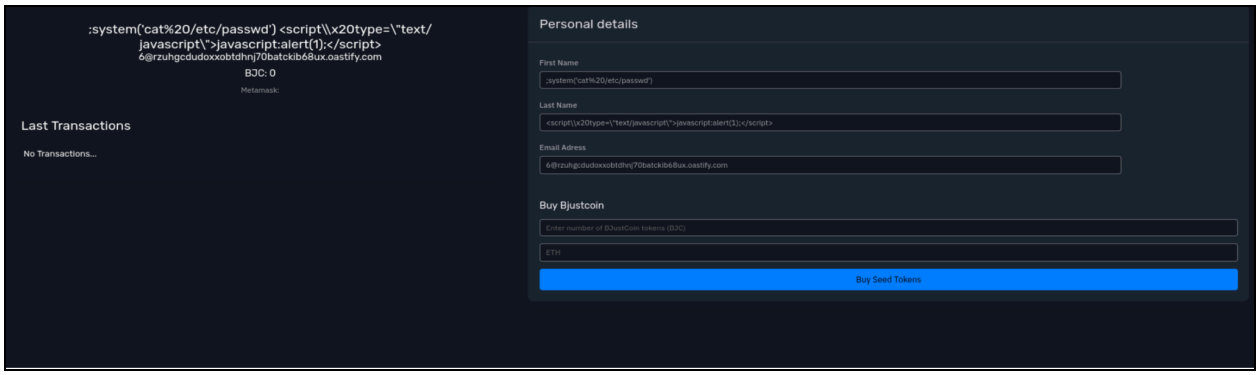


Remediation

We recommend applying a mechanism to validate and filter dangerous characters and strings such as HTML special characters. For filtering it is recommended to use regular expressions which filter out characters that could induce an attack as well as use HTML encoding for any potentially unsafe characters, so they are not stored or reused by the application in its canonical form.

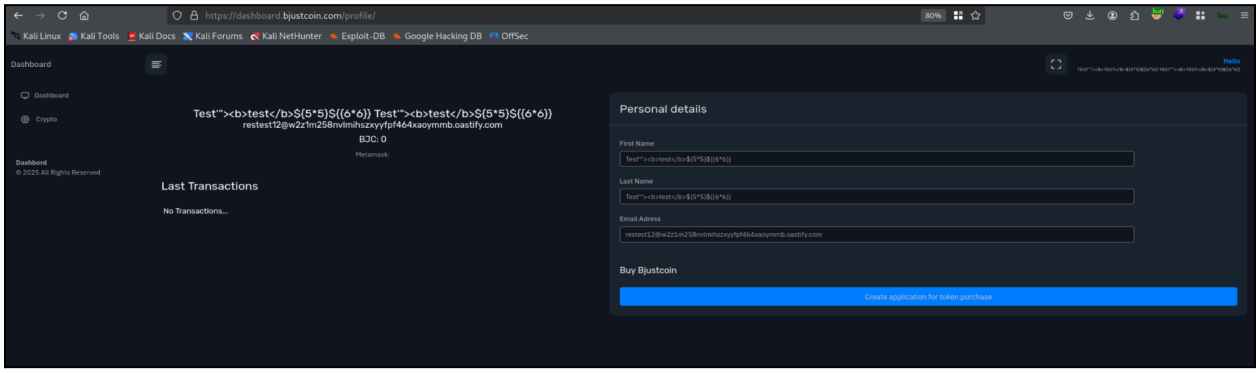
Retest (19.01.2025)

There is no improved validation of special characters, the user can still sign in using any special character, which can lead to an insecure environment.



Retest (26.02.2025)

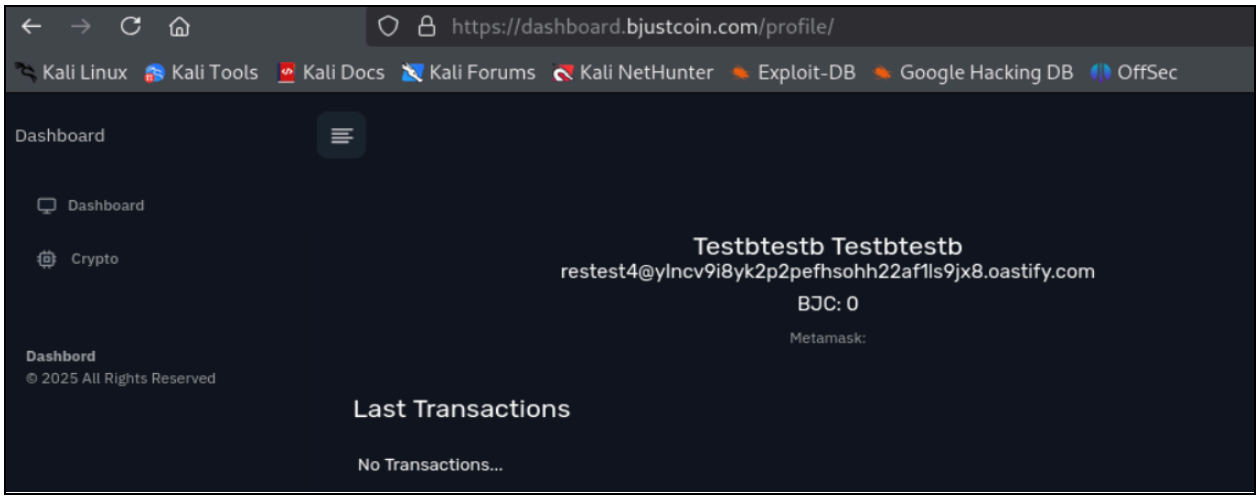
There is no improved validation of special characters, the user can still sign in using any special character, which can lead to an insecure environment.



Retest (27.02.2025)

Special character validation is now implemented and successfully validates special characters from user input. During profile creation, the user can enter any character, but when the user clicks on the `create account` button, the user's input is validated.

The view after successful account creation with special character validation.



Status: **Resolved**

---

8. Missing Subresource Integrity hashes

Severity

LOW

CVSS Score: 3.1

`CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N`

Description

The application at dashboard.bjustcoin.com loads external JavaScript resources without Subresource Integrity (SRI) hashes. Several third-party scripts, including Web3 and Coinponent libraries, are loaded from CDNs without integrity verification:

`/profile/` - Web3 library

```
<script src="https://cdn.jsdelivr.net/npm/web3@latest/dist/web3.min.js"></script>
```

`/admin_home2/` - Coinponent library

```
<script
src="https://cdn.jsdelivr.net/gh/coinponent/coinponent@1.2.6/dist/coinponent.js"></
script>
```

A compromised CDN or modified library could lead to execution of unauthorized JavaScript in the application context, potentially exposing sensitive user data or enabling malicious actions.

Remediation

We recommend that for current scripts, integrity attributes should be added to prevent unauthorized modifications. When implementing external resources in the future, use specific versions instead of "latest" tags, calculate appropriate SRI hashes, and include both integrity and crossorigin attributes. The development team should document and follow this process consistently for all external resources.

References: https://developer.mozilla.org/en-US/docs/Web/Security/Subresource_Integrity

Status: Acknowledged

---

9. Missing security headers

Severity

LOW

CVSS Score: 3.1

`CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N`

Description

During the assessment of the web application at https://dashboard.bjustcoin.com/, several security headers were found to be missing from HTTP responses:

- **HTTP Strict Transport Security (HSTS)** header enforces secure HTTPS connections, preventing protocol downgrade attacks and cookie hijacking. Recommended value: "max-age=31536000; includeSubDomains".
- **Content Security Policy (CSP)** protects against XSS attacks by specifying trusted content sources for scripts, styles, and other resources.
- **X-Content-Type-Options** prevents MIME-sniffing attacks by forcing browsers to honor the declared content-type. Required value: "nosniff".
- **Referrer Policy** controls information disclosure in navigations away from the site, enhancing user privacy.

The absence of these headers increases exposure to XSS attacks, protocol downgrades, MIME-type confusion, and unintended information disclosure through referrer headers.

Example of the request/response pair showing missing security headers:

```
POST /login/ HTTP/1.1
Host: dashboard.bjustcoin.com
Cookie: _gcl_au=1.1.1141853453.1733396594;
_ga_3YNSFDY25J=GS1.1.1733398060.1.1.1733399633.0.0.0;
```

```
SL_C_23361dd035530_SID={"0563212d2a846816aba8acd0191234c3acbfa214":{"sessionId":"-C
KJASKGUPGlLFrlk5L2n","visitorId":"XeZViQT2vdtG4Id-O5L6R"}};
id=VIx%2FK9t5SCpYPo9cwrqhJ6RL4bNrCLnWqe3TdiK2EGdssq7qP6QNQoSCvoB7R43r1nd261D7friU3W
W6nDxNPJJzl56XyJermYs8zC5CtMo+dxxnQ9QbGKMCXJS1NZqsCHPJCiBfjvCWflHoxy78OV4ix4Rl3usFl
5onngbMiH7Lw77LCPohgChWk6csWzxCBqYe72gu
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 73
Origin: https://dashboard.bjustcoin.com
Referer: https://dashboard.bjustcoin.com/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers
Connection: keep-alive

{"email":"<script>alert(document.domain)</script>",
"password":"' or 1=1"}
```

Server response:

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Thu, 12 Dec 2024 21:00:00 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
Content-Length: 7499
[REDACTED]
```

Remediation

We recommend implementation of the missing security headers in the web server configuration or application code.

Status: Acknowledged

---

10. Logout button does not work properly

Severity

LOW

CVSS Score: 3.1

`CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N`

Description

The logout button doesn't work as it should. The intended function of the logout button is to log the user out of the current session, remove the cookie from storage and make the cookie unusable. In this application, the logout button only removes cookies from storage, but the cookie is still valid and can be used again.

Example of the request sent to the server after clicking logout button:

```
GET /logout/ HTTP/1.1
Host: dashboard.bjustcoin.com
Cookie: _gcl_au=1.1.1321471271.1737219315;
id=cKL5+7aO+0xKfgmJL+Wg99KeJ5alm+xfI8KNhlh8eWfymIDM6ybZpsddZJRTfg86d+uHIgcoeg3Y0FgI
OGMUnrFRfi7Wxvc47ONcs+IWs6hK6ayNJ5hSPbCoymmN8Iw9kY5Qpc+IPe3GM3TMtu7RpU4CyGY3DUieOdC
GR3bd1uatiSHAT%2FCRJaByn8EtyqFCV%2FS32gdSL7LE2mfxCe2VNm3hfUCz11%2FcsNtGYS25FsnIiOFH
q7ak27Jg5P%2FDtUTFzQXhIEDCkZGF5LEGMb1ZDn%2FbVCZfsDqFNyQcO9QOufHmW2a%2FeIRvKZNR1+3RA
trw6+EK+y0Ptz8XPGkn2iBwQaSToesTwLiFJrjMWwQfIYr89ENC4bft4if0mmHlTu7fkY5eZjZsWHlZiulP
GeVbXrKK9u1e5ATDsoJUbdyiUePkLnMynglphijBdCAFP1LjaAi1JJs7XZWgj93Tb6uHQB9AkvghZwWPhQ%
3D%3D
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://dashboard.bjustcoin.com/profile/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0
Te: trailers
Connection: keep-alive
```

After sending such a request, the server should invalidate the cookie and the user should not be able to use that cookie to establish the session, but in this application the server doesn't do that.

Example of the request to another endpoint e.g. `/profile`:

```
GET /profile/ HTTP/1.1
Host: dashboard.bjustcoin.com
```

```
Cookie: _gcl_au=1.1.1321471271.1737219315;
id=cKL5+7aO+0xKfgmJL+Wg99KeJ5alm+xfI8KNhlh8eWfymIDM6ybZpsddZJRTfg86d+uHIgcoeg3Y0FgI
OGMUnrFRfi7Wxvc47ONcs+IWs6hK6ayNJ5hSPbCoymmN8Iw9kY5Qpc+IPe3GM3TMtu7RpU4CyGY3DUieOdC
GR3bd1uatiSHAT%2FCRJaByn8EtyqFCV%2FS32gdSL7LE2mfxCe2VNm3hfUCz11%2FcsNtGYS25FsnIiOFH
q7ak27Jg5P%2FDtUTFzQXhIEDCkZGF5LEGMb1ZDn%2FbVCZfsDqFNyQcO9QOufHmW2a%2FeIRvKZNR1+3RA
trw6+EK+y0Ptz8XPGkn2iBwQaSToesTwLiFJrjMWwQfIYr89ENC4bft4if0mmHlTu7fkY5eZjZsWHlZiulP
GeVbXrKK9u1e5ATDsoJUbdyiUePkLnMynglphijBdCAFP1LjaAi1JJs7XZWgj93Tb6uHQB9AkvghZwWPhQ%
3D%3D
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://dashboard.bjustcoin.com/profile/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0
Te: trailers
Connection: keep-alive
```

Response from the server:

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Wed, 26 Feb 2025 20:46:57 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
Content-Length: 12761

[. . .]
<a href="#" class="waves-effect waves-light dropdown-toggle w-auto l-h-12
bg-transparent py-0 no-shadow" data-bs-toggle="dropdown" title="User">
<div class="d-flex pt-5">
<div class="text-end me-10">
<p class="pt-5 fs-14 mb-0 fw-700 text-primary">Hello</p>
<small class="get_user_info fs-10 mb-0 text-uppercase text-mute"
first-name="Retest"
last-name="Retest"
email="retest@y653q49arxpomjw1102hth888zeq2pqe.oastify.com"
id="134">
                                                Retest Retest
</small>
[. . .]
```

The above response indicates that the cookie is still valid and returns the account information, in this case: first name, last name, email and id.

Remediation

We recommend implementing a mechanism to invalidate cookies on the server side after the logout button has been used. After logging out and invalidating the cookie, a user attempting to use the same cookie again should receive a 403 response from the server and be forced to log in again.

Status: Acknowledged

---

11. Create application for ICO tokens for another user's email address

Severity

LOW

CVSS Score: 3.1

`CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N`

Description

The Create Application for ICO Tokens functionality has the ability to specify the user's email address. Any authenticated user can enter another user's email address, which could trigger the application for ICO tokens for another user.

The token purchase request form consists of a number of parameters that can be specified by the user, namely: first_name, middle_name, last_name, email, phone, mobile, address, tokens and is_agree.

The example of request/response to the server:

```
POST /create_suggest_item/ HTTP/1.1
Host: dashboard.bjustcoin.com
Cookie: _gcl_au=1.1.1321471271.1737219315;
id=a5X1HwVQtmIBP9kTz+Ah4udwHEQwt+bTTH56xwP9q9NG+BHzzcc9o7TgQGyUPIAnaUMHrbiF6FoY1itQ
9NmOJo51nrEbg+yTuM6jFzLST4tHzY941qPoaZDk5gz7F+LNDrZC4lhl0xF6qX19lhqWsuwa6rZbJ1BbyIe
1AxZ+H1sMNRZgDHdkYGZKi4SKPwlWPxcKWUqEt2MsvgRt19uKik967J%2FusxdAIGYx9lgPsMoA%2FE1d9N
GScr5S2ZDH32qjSTkVgLoQ1X0IKYPtdJhayJNypHHO9GGOYRY3mEY0VjRGDl3EOZAQ5+fAoodQI3cqW61hb
VlzZMWVFiVrXC6PDQKAcpHeiNAxzsE916ZqsgKiZWRfE5oVMtzX2zRjapJGgz8JPDpx5c9%2F4JpgVmgJrI
0hyFlKC7IR2p5qz+ZSLPCielsNHGDP8m2Io2zMs7d%2FwjM9QSrcwCPdcIdNN2d9gfVGdXWBy8Dapw%3D%3
D
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
```

```
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 212
Origin: https://dashboard.bjustcoin.com
Referer: https://dashboard.bjustcoin.com/create_suggest_item/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0
Te: trailers
Connection: keep-alive

{"first_name":"Retest",
"middle_name":"",
"last_name":"Retest",
"email":"Another_User_Email_Address@example.com",
"phone":"321321321",
"mobile":"123123123",
"address":"0X000000000000000",
"tokens":"150",
"is_agree":"on"}
```

Response:

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Wed, 26 Feb 2025 21:48:32 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 3
Connection: keep-alive

err
```

The response shows that there is an error processing this request.

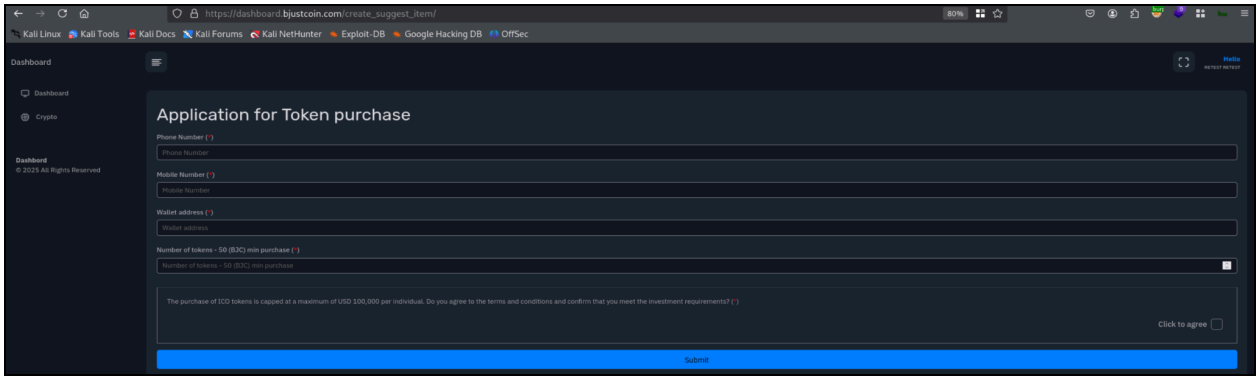
Remediation

This type of functionality should be assigned to the current account, and it should not be possible for the user to provide key information such as an email address.

We recommend that the email address as well as other key parameters should be set here permanently, without the user being able to change them. We suggest leaving as few fields as possible for user input.

Retest (27.02.2025)

This issue has been fixed by removal of a field where user can input the email address of another user.



Status: Resolved

---

12. Software version disclosure

Severity

INFORMATIONAL

CVSS Score: N/A

Description

During testing, it was found that the server responses return headers related to the web server used in the backend, including its exact name and version. While this does not pose a risk to the platform in itself, such information can help a potential attacker to conduct other, more sophisticated attacks, and also makes it possible for issues that will be detected in that particular version in the future, in this case **nginx**, to make the platform and server subject to attacks from the public network.

Example of a request that returns the exact **nginx** version in response:

```
GET / HTTP/1.1
Host: dashboard.bjustcoin.com
Cookie: _gcl_au=1.1.1141853453.1733396594;
_ga_3YNSFDY25J=GS1.1.1733398060.1.1.1733399633.0.0.0;
SL_C_23361dd035530_SID={"0563212d2a846816aba8acd0191234c3acbfa214":{"sessionId":"-C
KJASKGUPGlLFrlk5L2n","visitorId":"XeZViQT2vdtG4Id-O5L6R"}};
id=VIx%2FK9t5SCpYPo9cwrqhJ6RL4bNrCLnWqe3TdiK2EGdssq7qP6QNQoSCvoB7R43r1nd261D7friU3W
W6nDxNPJJzl56XyJermYs8zC5CtMo+dxxnQ9QbGKMCXJS1NZqsCHPJCiBfjvCWflHoxy78OV4ix4Rl3usFl
5onngbMiH7Lw77LCPohgChWk6csWzxCBqYe72gu
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0
.8
```

```
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Te: trailers
Connection: keep-alive
```

Server response:

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Thu, 12 Dec 2024 20:42:44 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
Content-Length: 3494

[REDACTED]
```

Remediation

We recommend that the "Server" header be disabled so that it is not returned with every server response. For Nginx, the configuration process is described, for example, in the following paper:

- https://www.tecmint.com/hide-nginx-server-version-in-linux/

Status: Acknowledged

---

13. Directory listing is enabled

Severity

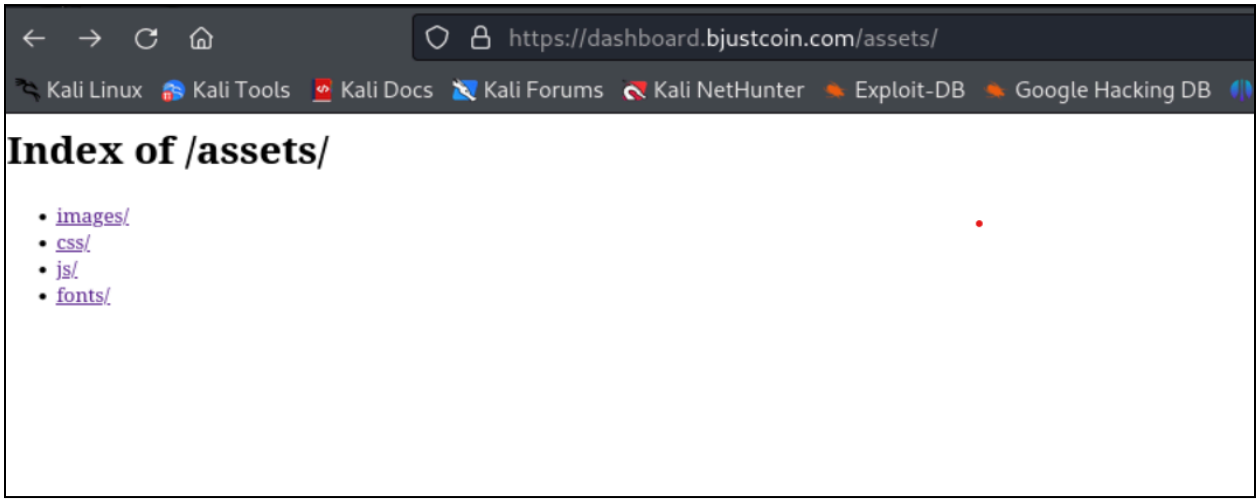
INFORMATIONAL

CVSS Score: N/A

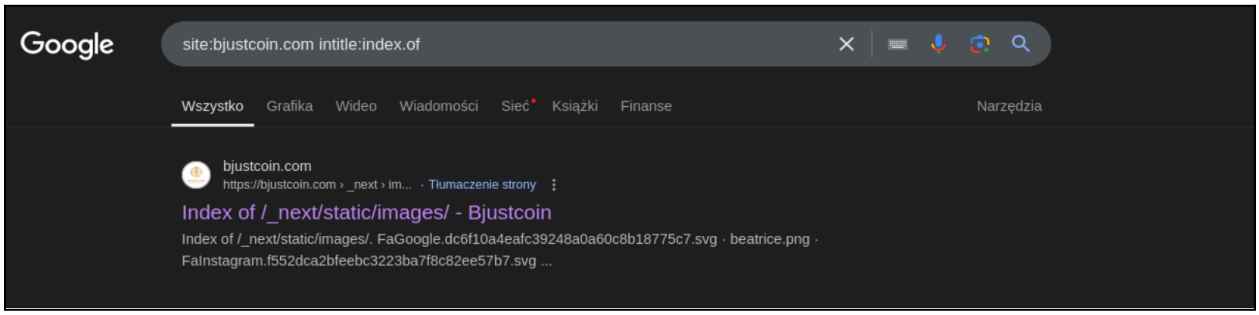
Description

During the penetration test, it was discovered that any user visiting the site could perform a directory listing on the specific paths. For example, the entire contents of the `/assets/` directory can be listed. A potential attacker can see all the files in this directory. This is one of the main directories that contains e.g. all the javascript files.

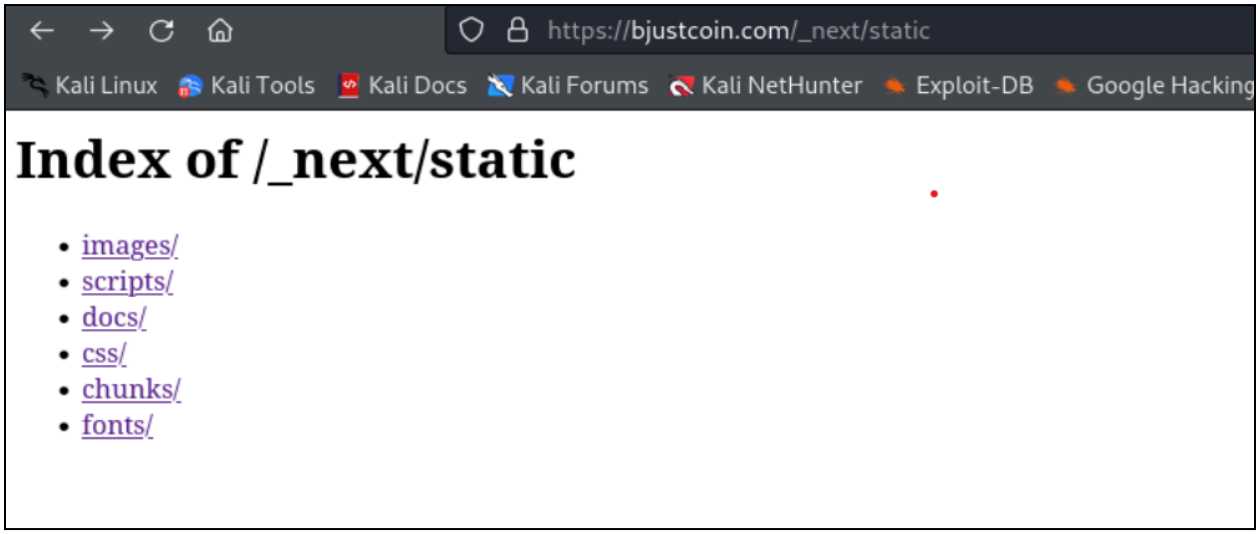
While not being a direct threat to the application, directory listing is considered insecure, as if anytime a sensitive file is uploaded to the server, it becomes instantly visible and potentially indexed by search engines. This increases the probability of an attack against the application.



In addition, any internet user could perform google dorks (`site:bjustcoin.com intitle:index.of`) to find directory listing on the main page of the BJust Coin.



The above URL shows an attacker the `/images/` directory but he can freely browse other catalogues, which contains more potentially interesting files.



Remediation

We recommend disabling the ability of directory listing on the web application server.

Status: Acknowledged

---

14. Vulnerable dependency in use

Severity

INFORMATIVE

CVSS Score: N/A

Description

A security audit of the frontend application repository and its dependencies using `cargo audit` tool revealed an outdated version of the Diesel ORM framework that contains a known vulnerability. The application uses Diesel version 1.4.8, which is affected by a binary protocol misinterpretation vulnerability due to truncating or overflowing casts.

Binary protocol misinterpretation vulnerabilities can potentially lead to data corruption or unexpected behavior in database operations. However, it was not proven that current implementation is exploitable.

`Cargo audit` output:

```
Crate:     diesel
```

```
Version:   1.4.8
Title:     Binary Protocol Misinterpretation caused by Truncating or Overflowing
Date:      2024-08-23
ID:        RUSTSEC-2024-0365
URL:       https://rustsec.org/advisories/RUSTSEC-2024-0365
Solution:  Upgrade to >=2.2.3
Dependency tree:
diesel 1.4.8
└── dach_front 0.1.0

error: 1 vulnerability found!
```

Remediation

We recommend updating the Diesel dependency to version 2.2.3 or newer.

Status: Acknowledged

END OF THE REPORT