



CredShields

Web Application Audit

Mar 4th, 2023 • CONFIDENTIAL

Description

This document details the process and result of the Web Application audit performed by CredShields Technologies PTE. LTD. on behalf of Juno between Dec 12th, 2022, and Jan 12th, 2023. And a retest was performed on Feb 23rd, 2023.

Author

Shashank (Co-founder, CredShields)

shashank@CredShields.com

Reviewers

Aditya Dixit (Research Team Lead)

aditya@CredShields.com

Prepared for

Juno

Table of Contents

| | |
|---|-----------|
| 1. Executive Summary | 3 |
| State of Security | 4 |
| 2. Methodology | 5 |
| 2.1 Preparation phase | 5 |
| 2.1.1 Scope | 6 |
| 2.1.2 Documentation | 6 |
| 2.1.3 Audit Goals | 6 |
| 2.2 Retesting phase | 7 |
| 2.3 Vulnerability classification and severity | 7 |
| 2.4 CredShields staff | 10 |
| 3. Findings | 11 |
| 3.1 Findings Overview | 11 |
| 3.1.1 Vulnerability Summary | 11 |
| 3.1.2 Findings Summary | 12 |
| 4. Remediation Status | 16 |
| 5. Bug Reports | 17 |
| Bug ID#1 [Fixed] | 17 |
| TLS 1.2 Supports Weak Cipher Suites | 17 |
| Bug ID#2 [Fixed] | 19 |
| Unrestricted Google Maps API | 19 |
| Bug ID#3 [Fixed] | 21 |
| Descriptive Stack Trace | 21 |
| Bug ID#4 [Fixed] | 24 |
| Content Spoofing/Text Injection | 24 |
| Bug ID#5 [Fixed] | 26 |
| Email Verification Bypass | 26 |
| Bug ID#6 [Fixed] | 28 |
| MFA Bypass | 28 |
| Bug ID#7 [Fixed] | 30 |
| JWT HS256 Algorithm Usage | 30 |
| Bug ID#8 [Won't Fix] | 32 |
| Misconfigured Content Security Policy | 32 |
| Bug ID#9 [Won't Fix] | 35 |

| | |
|---|-----------|
| Concurrent Sessions Allowed | 35 |
| Bug ID#10 [Fixed] | 37 |
| Missing Secure and HttpOnly attributes in Cookies | 37 |
| Bug ID#11 [Won't Fix] | 39 |
| Card PIN Update missing additional Security Validations | 39 |
| Bug ID#12 [Fixed] | 41 |
| Missing Access Control in Fetching Card Details | 41 |
| Bug ID#13 [Fixed] | 44 |
| Outdated Nginx and Version Disclosure | 44 |
| Bug ID#14 [Pending Fix] | 46 |
| Malicious File Upload | 46 |
| Bug ID#15 [Fixed] | 48 |
| Weak JWT Signing Key on Dev | 48 |
| Bug ID#16 [Won't Fix] | 50 |
| Weak Card PIN | 50 |
| Bug ID#17 [Fixed] | 52 |
| Missing Access Control in Fetching Pay Allocations | 52 |
| 6. Disclosure | 54 |

1. Executive Summary

Juno engaged CredShields to perform a Web Application audit from Dec 12th, 2022, to Jan 12th, 2023. During this timeframe, Seventeen (17) vulnerabilities were identified. **A retest was performed on Feb 23rd, 2023, and all the bugs have been addressed or acknowledged as won't fix.**

During the audit, Four (4) vulnerabilities were found with a severity rating of either High or Critical. These vulnerabilities represent the greatest immediate risk to "Juno" and should be prioritized for remediation, and fortunately, none were found.

The table below shows the in-scope assets and a breakdown of findings by severity per asset. Section 2.3 contains more information on how severity is calculated.

| Assets in Scope | Critical | High | Medium | Low | Info | Σ |
|-----------------|----------|----------|----------|----------|----------|-----------|
| Web Application | 0 | 4 | 5 | 7 | 1 | 17 |
| | 0 | 4 | 5 | 7 | 1 | 17 |

Table: Vulnerabilities Per Asset in Scope

The CredShields team conducted the security audit to focus on identifying vulnerabilities in Web Application's scope during the testing window while abiding by the policies set forth by Juno's team.

State of Security

To maintain a robust security posture, it is essential to continuously review and improve upon current security processes. Utilizing CredShields' continuous audit feature allows both Juno's internal security and development teams to not only identify specific vulnerabilities but also gain a deeper understanding of the current security threat landscape.

To ensure that vulnerabilities are not introduced when new features are added, or code is refactored, we recommend conducting regular security assessments. Additionally, by analyzing the root cause of resolved vulnerabilities, the internal teams at Juno can implement both manual and automated procedures to eliminate entire classes of vulnerabilities in the future. By taking a proactive approach, Juno can future-proof its security posture and protect its assets.

2. Methodology

Juno engaged CredShields to perform a Juno Web Application audit. The following sections cover how the engagement was put together and executed.

2.1 Preparation phase

The CredShields team intercepted all the APIs originating from the web application to understand the application's features and functionalities. They meticulously examined all functions and created a mind map to systematically identify potential security vulnerabilities, prioritizing those that were more critical and business-sensitive for the refactored code. To confirm their findings, the team also worked on the staging version of the Web application and performed verifications and validations during the audit phase.

A testing window from Dec 12th, 2022, to Jan 12th, 2023, was agreed upon during the preparation phase.

2.1.1 Scope

During the preparation phase, the following scope for the engagement was agreed-upon:

| IN SCOPE ASSETS |
|--|
| www.juno.finance api.juno.finance |

Table: List of Files in Scope

2.1.2 Documentation

Documentation was not required as the code was self-sufficient for understanding the project.

2.1.3 Audit Goals

CredShields uses both in-house tools and manual methods for comprehensive Web Application audits. The majority of the audit is done by manually reviewing the Web Application features and API calls, following [OWASP Web application security standards](#), and an extended industry standard self-developed checklist. The team places emphasis on understanding core concepts, preparing test cases, and evaluating business logic for potential vulnerabilities.

2.2 Retesting phase

Juno is actively partnering with CredShields to validate the remediations implemented towards the discovered vulnerabilities.

2.3 Vulnerability classification and severity

CredShields follows OWASP's Risk Rating Methodology to determine the risk associated with discovered vulnerabilities. This approach considers two factors - Likelihood and Impact - which are evaluated with three possible values - **Low**, **Medium**, and **High**, based on factors such as Threat agents, Vulnerability factors, Technical and Business Impacts. The overall severity of the risk is calculated by combining the likelihood and impact estimates.

| Overall Risk Severity | | | | |
|-----------------------|------------|--------|--------|----------|
| Impact | HIGH | Medium | High | Critical |
| | MEDIUM | Low | Medium | High |
| | LOW | Note | Low | Medium |
| | | LOW | MEDIUM | HIGH |
| | Likelihood | | | |

Overall, the categories can be defined as described below -

1. Informational

We prioritize technical excellence and pay attention to detail in our coding practices. Our guidelines, standards, and best practices help ensure software stability and reliability. Informational vulnerabilities are opportunities for improvement and do

not pose a direct risk to the organization. Code maintainers should use their own judgment on whether to address them.

2. Low

Low-risk vulnerabilities are those that either have a small impact or can't be exploited repeatedly or those the client considers insignificant based on their specific business circumstances.

3. Medium

Medium-severity vulnerabilities are those caused by weak or flawed logic in the code and can lead to exfiltration or modification of private user information. These vulnerabilities can harm the client's reputation under certain conditions and should be fixed within a specified timeframe.

4. High

High-severity vulnerabilities pose a significant risk to the Web Application and the organization. They can result in the loss of funds for some users, may or may not require specific conditions, and are more complex to exploit. These vulnerabilities can harm the client's reputation and should be fixed immediately.

5. Critical

Critical issues are directly exploitable bugs or security vulnerabilities that do not require specific conditions. They often result in the loss of funds for users or put sensitive user information at risk of compromise or modification. The client's

reputation and financial stability will be severely impacted if these issues are not addressed immediately.

2.4 CredShields staff

The following individual at CredShields managed this engagement and produced this report:

- **Shashank, Co-founder CredShields**
 - shashank@CredShields.com

Please feel free to contact this individual with any questions or concerns you have around the engagement or this document.

3. Findings

This chapter contains the results of the security assessment. Findings are sorted by their severity and grouped by the asset and CWE classification. Each asset section will include a summary. The table in the executive summary contains the total number of identified security vulnerabilities per asset per risk indication.

3.1 Findings Overview

3.1.1 Vulnerability Summary

During the security assessment, Seventeen (17) security vulnerabilities were identified in the asset.

| VULNERABILITY TITLE | SEVERITY | CWE Vulnerability Type |
|-------------------------------------|----------|--|
| TLS 1.2 Supports Weak Cipher Suites | Medium | Inadequate Encryption Strength - CWE-326 |
| Unrestricted Google Maps API | Low | Configuration - CWE-16 |
| Descriptive Stack Trace | Low | Generation of Error Message Containing Sensitive Information - CWE-209 |
| Content Spoofing/Text Injection | Low | Content Spoofing - CWE-451 |
| Email Verification Bypass | High | Improper Access Control - CWE-284 |

| | | |
|---|-------------|--|
| MFA Bypass | High | Improper Authorization - CWE-285 |
| JWT HS256 Algorithm Usage | Low | Configuration - CWE-16 |
| Misconfigured Content Security Policy | Low | Configuration - CWE-16 |
| Concurrent Sessions Allowed | Informative | Manage User Sessions - CWE-1018 |
| Missing Secure and HttpOnly attributes in Cookies | Medium | Configuration - CWE-16 |
| Card PIN Update missing additional Security Validations | Medium | Configuration - CWE-16 |
| Missing Access Control in Fetching Card Details | High | Improper Access Control - CWE-284 |
| Outdated Nginx and Version Disclosure | Low | Using Components with know vulnerabilities |
| Malicious File Upload | Low | File Upload |
| Weak JWT Signing Key on Dev | High | Configuration - CWE-16 |
| Weak Card PIN | Medium | Configuration - CWE-16 |
| Missing Access Control in Fetching Pay Allocations | Medium | Improper Access Control - CWE-284 |

Table: Findings in Web Application

4. Remediation Status

Juno is actively partnering with CredShields from this engagement to validate the discovered vulnerabilities' remediations. **A retest was performed on Feb 23rd, 2023, and all the issues have been addressed or acknowledged as won't fix.**

Also, the table shows the remediation status of each finding.

| VULNERABILITY TITLE | SEVERITY | REMEDIATION STATUS |
|---|-------------|-----------------------|
| TLS 1.2 Supports Weak Cipher Suites | Medium | Fixed [23/02/2023] |
| Unrestricted Google Maps API | Low | Fixed [23/02/2023] |
| Descriptive Stack Trace | Low | Fixed [23/02/2023] |
| Content Spoofing/Text Injection | Low | Fixed [23/02/2023] |
| Email Verification Bypass | High | Fixed [23/02/2023] |
| MFA Bypass | High | Fixed [23/02/2023] |
| JWT HS256 Algorithm Usage | Low | Won't Fix |
| Misconfigured Content Security Policy | Low | Won't Fix |
| Concurrent Sessions Allowed | Informative | Won't Fix |
| Missing Secure and HttpOnly attributes in Cookies | Medium | Fixed [23/02/2023] |

| | | |
|---|--------|-------------------------------------|
| Card PIN Update missing additional Security Validations | Medium | Won't Fix |
| Missing Access Control in Fetching Card Details | High | Fixed [23/02/2023] |
| Outdated Nginx and Version Disclosure | Low | Fixed [23/02/2023] |
| Malicious File Upload | Low | Fixed [23/02/2023] |
| Weak JWT Signing Key on Dev | High | Fixed [23/02/2023] |
| Weak Card PIN | Medium | Won't Fix |
| Missing Access Control in Fetching Pay Allocations | Medium | Fixed [23/02/2023] |

Table: Summary of findings and status of remediation

5. Bug Reports

Bug ID#1 [Fixed]

TLS 1.2 Supports Weak Cipher Suites

Vulnerability Type

Inadequate Encryption Strength - [CWE-326](#)

Severity

Medium

Description

The software stores or transmits sensitive data using an encryption scheme that is theoretically sound but is not strong enough for the level of protection required.

A weak encryption scheme can be subjected to brute force attacks that have a reasonable chance of succeeding using current attack methods and resources.




TLS 1.2 implementations by the server **juno.finance** were found to be supporting weak CBC and other cipher suites. These cipher suites offer additional security over Electronic Codebook (ECB) mode but have the potential to leak information if used improperly.

Vulnerable URL

- [juno.finance](#)

PoC

1. Scan the server using SSLabs at <https://www.ssllabs.com/ssltest/analyze.html?d=juno.finance&s=35.160.188.228&hideResults=on&latest>

| Configuration | | | |
|---|---------------------------------------|------|---|
|  | | | |
| Protocols | | | |
| TLS 1.3 | | | No |
| TLS 1.2 | | | Yes* |
| TLS 1.1 | | | No |
| TLS 1.0 | | | No |
| SSL 3 | | | No |
| SSL 2 | | | No |
| (*) Experimental: Server negotiated using No-SNI | | | |
|  | | | |
| Cipher Suites | | | |
| # TLS 1.2 (suites in server-preferred order) | | |  |
| TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f) | ECDH secp256r1 (eq. 3072 bits RSA) FS | | 128 |
| TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027) | ECDH secp256r1 (eq. 3072 bits RSA) FS | WEAK | 128 |
| TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030) | ECDH secp256r1 (eq. 3072 bits RSA) FS | | 256 |
| TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028) | ECDH secp256r1 (eq. 3072 bits RSA) FS | WEAK | 256 |
| TLS_RSA_WITH_AES_128_GCM_SHA256 (0x9c) | | WEAK | 128 |
| TLS_RSA_WITH_AES_128_CBC_SHA256 (0x3c) | | WEAK | 128 |
| TLS_RSA_WITH_AES_256_GCM_SHA384 (0x9d) | | WEAK | 256 |
| TLS_RSA_WITH_AES_256_CBC_SHA256 (0x3d) | | WEAK | 256 |

Impacts

An attacker may be able to exploit this issue to conduct man-in-the-middle attacks and decrypt and tamper with the communications between the affected service and clients.

Remediation

As a best practice, consider supporting only cipher suites that are known to be secure. Disable any cipher suites that use encryption with less than 128-bit key lengths or utilize RC4 algorithms. Enabled TLS cipher suites must be ranked as MEDIUM strength by the current version of OpenSSL at a minimum, however, HIGH is ideal. Ensure that the cipher suites are ordered from strongest to weakest.

Here's a reference - <https://www.acunetix.com/blog/articles/tls-ssl-cipher-hardening/>

Retest

Support for weak cipher suits has been removed.

Bug ID#2 [Fixed]

Unrestricted Google Maps API

Vulnerability Type

Configuration - [CWE-16](#)

Severity

Low

Description

Google Maps API is a paid service. These API keys require security configurations for blocking unauthorized usage by malicious attackers, which is not enabled by default. Developers need to configure these security controls when implementing the API key. While these API keys are designed as public API keys and do not have any impact in terms of customer data confidentiality/integrity, these security configurations still need to be implemented for blocking unauthorized usage, otherwise, attackers can abuse these keys in order to bill the Company for the expenses of using Google Maps API services.

Vulnerable Key

- AlzaSyAfC3TtVQmRZQUbiakxatIEwsQOixyOSgY

PoC

1. Go through the sign-up flow and search for an address. Observe the request to Maps API sent in the background with the unrestricted API key.

| Results | Cost Table/Reference to Exploit: |
|----------------------------|----------------------------------|
| - Find Place From Text | \$17 per 1000 elements |
| - Autocomplete | \$2.83 per 1000 requests |
| - Autocomplete Per Session | \$17 per 1000 requests |
| - Place Details | \$17 per 1000 requests |
| - Nearby Search-Places | \$32 per 1000 requests |
| - Text Search-Places | \$32 per 1000 requests |
| - Places Photo | \$7 per 1000 requests |

Impacts

If this API key is not properly configured, an attacker could consume the company's monthly quota or can over-bill with unauthorized usage of this service and do financial damage to the company, if the company does not have any limitation settings on API budgets.

Remediation

Google suggests adding proper restrictions to the API keys such as -

- HTTP referrers: restricts usage to one or more URLs and is intended for keys that are used in websites and web apps. This type of restriction allows you to set restrictions to a specific domain, page, or set of pages on your website.
- IP addresses: restricts usage to one or more IP addresses, and are intended for securing keys used in server-side requests, such as calls from web servers and cron jobs.
- Android and iOS app restriction: restricts usage to calls from an Android app with a specified package name.

References: https://developers.google.com/maps/api-key-best-practices#restrict_apikey

Retest

The key has been updated and restricted except staticmap API.

Bug ID#3 [Fixed]

Descriptive Stack Trace

Vulnerability Type

Generation of Error Message Containing Sensitive Information - [CWE-209](#)

Severity

Low

Description

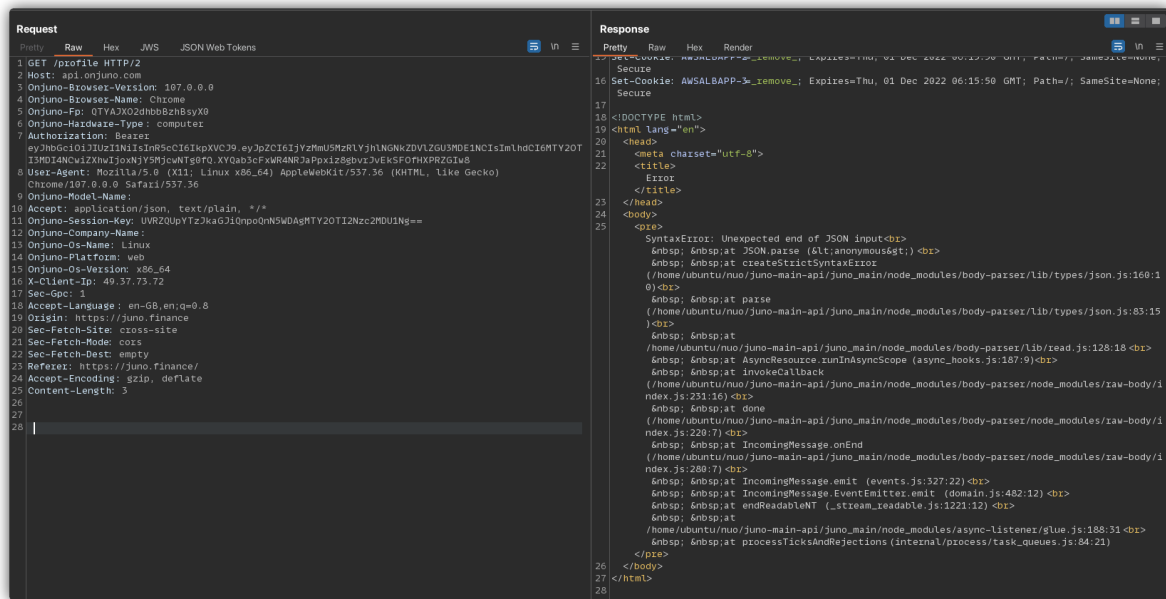
The application generates an error message that includes sensitive information about its environment, users, or associated data. The sensitive information could be valuable information on its own (such as a password), or it could be useful for launching further attacks. If an attack fails, an attacker may use the error information provided by the server to launch another more focused attack.

Vulnerable Key

- <https://api.onjuno.com/profile>

PoC

1. Log in to the application and send a malformed JSON request as shown below. The descriptive error will be displayed in the response due to improper error handling.



```
GET /profile HTTP/2
Host: api.onjuno.com
Onjuno-Browser-Version: 107.0.0.0
Onjuno-Browser-Name: Chrome
Onjuno-Fp: QTYAJXO2dhhbBzhBsyX0
Onjuno-Hardware-Type: computer
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYzMmU5MzRlYjhlNGNkZDVlZGU3MDE1NCIsIm1hdCI6MTY2OTI3MDI4NCwiZmxhZSI6bnVjY5MjcwNTg0fQ.XYQab3cFwR4NRJaPpxiz8gbvrJvEksFOfHXPRZGIw8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/107.0.0.0 Safari/537.36
Onjuno-Model-Name:
Accept: application/json, text/plain, */*
Onjuno-Session-Key: UVRZQUpYTzJkaGJiQnpOQnN5WDAGMTY2OTI2Nzc2MDU1Ng==
Onjuno-Company-Name:
Onjuno-OS-Name: Linux
Onjuno-Platform: web
Onjuno-OS-Version: x86_64
X-Client-IP: 49.37.73.72
Sec-Gpc: 1
Accept-Language: en-GB,en;q=0.8
Origin: https://juno.finance
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://juno.finance/
Accept-Encoding: gzip, deflate
Content-Length: 3
```

Impacts

This exposes sensitive information such as software used, function names, and the stack used in the application. This does not represent a vulnerability in itself but the knowledge obtained may be abused in carrying out other attacks.

Improper handling of errors can introduce a variety of security problems for a website. The most common problem is when detailed internal error messages such as stack traces, database dumps, and error codes are displayed to the user.

These messages reveal implementation details that should never be revealed. Such details can provide hackers with important clues on potential flaws in the site and such messages are also disturbing to normal users.

Remediation

The errors should be generalized so that instead of excessive information, the API only returns a 403 Forbidden error.

A specific policy for how to handle errors should be documented, including the types of errors to be handled and for each, what information is going to be reported back to the user, and what information is going to be logged. All developers need to understand the policy and ensure that their code follows it.

In the implementation, ensure that the site is built to gracefully handle all possible errors. When errors occur, the site should respond with a specifically designed result that is helpful to the user without revealing unnecessary internal details. Certain classes of errors should be logged to help detect implementation flaws in the site and/or hacking attempts.

Retest

This is fixed. Error handling is done properly without showing a descriptive stack trace.

Bug ID#4 [Fixed]

Content Spoofing/Text Injection

Vulnerability Type

Content Spoofing - [CWE-451](#)

Severity

Low

Description

Content spoofing, also referred to as content injection, "arbitrary text injection" or virtual defacement, is an attack targeting a user made possible by an injection vulnerability in a web application. When an application does not properly handle user-supplied data, an attacker can supply content to a web application, typically via a parameter value, that is reflected back to the user. This presents the user with a modified page under the context of the trusted domain.

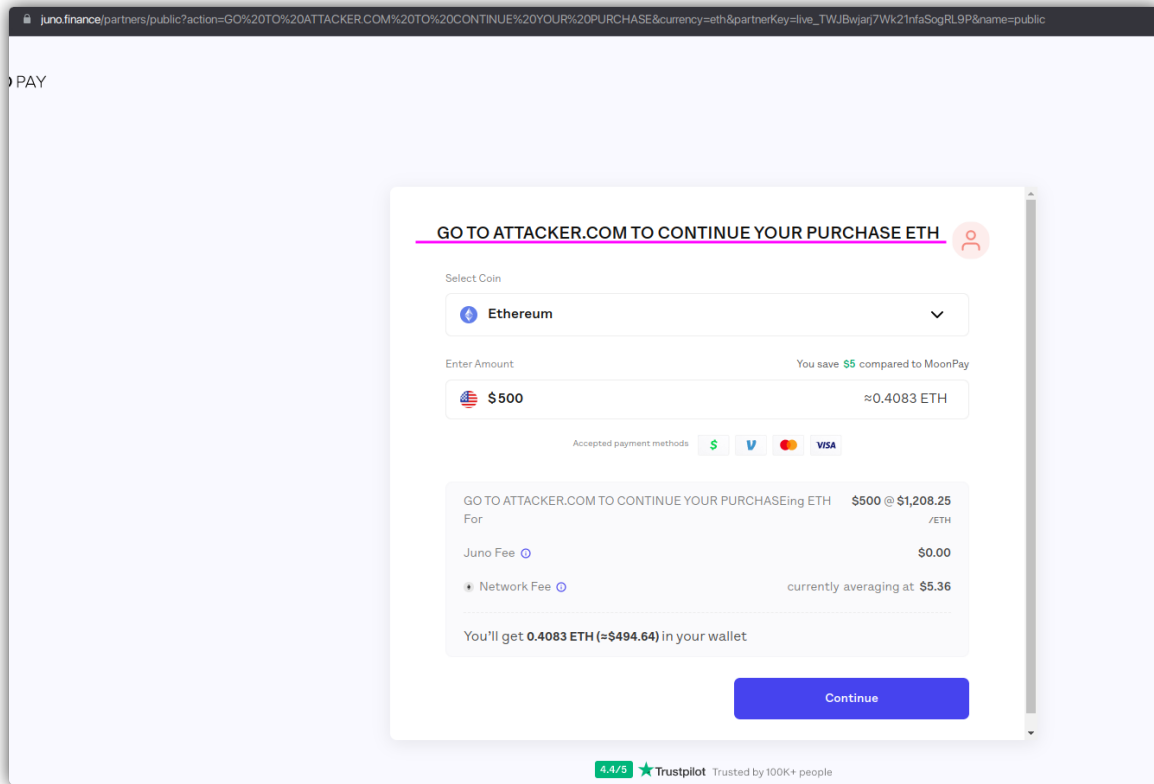
This attack is typically used as, or in conjunction with, social engineering because the attack is exploiting a code-based vulnerability and a user's trust.

Vulnerable Endpoint

- https://juno.finance/partners/public?action=GO%20TO%20ATTACKER.COM%20TO%20CONTINUE%20YOUR%20PURCHASE¤cy=eth&partnerKey=live_TWJBwj7Wk21nfaSogRL9P&name=public

PoC

1. Go to the endpoint mentioned above and observe the text rendered from the "action" parameter.



Impacts

Text injection vulnerabilities can be combined with social engineering to execute phishing attacks and manipulate end-users since the data is being rendered on a trusted application/domain.

Remediation

As a best practice, handle dynamic text input and use a whitelist of proper inputs. Validate user input before processing it on the server or client side.

Retest

The page has been removed.

Bug ID#5 [Fixed]

Email Verification Bypass

Vulnerability Type

Improper Access Control - [CWE-284](#)

Severity

High

Description

The application discloses OTPs that are supposed to be sent via email during signup in the JSON response body. This could allow users to bypass the email verification process and sign-up with any email.

Vulnerable Endpoint

- <https://api.onjuno.com/get-started>
- https://juno.finance/partners/public/enter-email?action=buy¤cy=eth&partnerKey=live_TWJBwjarj7Wk21nfaSogRL9P&name=public
- <https://api.onjuno.com/security/public-email-init>

PoC

1. Create an account by going to the partner's URL above.
2. Enter the required details and use any email address.
3. Once the application sends OTP, it can be seen in the response as shown below:

| Request | | Response | |
|---|-----|--|-----|
| Pretty | Raw | Pretty | Raw |
| <pre> 1 POST /security/public-email-init HTTP/2 2 Host: api.onjuno.com 3 Content-Length: 58 4 Onjuno-Browser-Version: 107.0.0.0 5 Onjuno-Browser-Name: Chrome 6 Onjuno-Fp: QTYAJXO2dhhbB2h8syX0 7 Onjuno-Hardware-Type: computer 8 Onjuno-Partner-Id: 62d9d33a1b7fe2c96b142e3 9 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36 10 Content-Type: application/json 11 Onjuno-Model-Name: 12 Accept: application/json, text/plain, */* 13 Onjuno-Session-Key: UVRZ0UpYT2KaGJ0np0QnNSWDagMTY2OTI2Nzc2MDU1Ng== 14 Onjuno-Company-Name: 15 Onjuno-OS-Name: Linux 16 Onjuno-Platform: web 17 Onjuno-OS-Version: x86_64 18 X-Client-Id: 49.37.73.72 19 Sec-Gpc: 1 20 Accept-Language: en-GB,en;q=0.8 21 Origin: https://juno.finance 22 Sec-Fetch-Site: cross-site 23 Sec-Fetch-Mode: cors 24 Sec-Fetch-Dest: empty 25 Referer: https://juno.finance/ 26 Accept-Encoding: gzip, deflate 27 28 { "email": "zbrains9@gmail.com", "type": "LOGIN_EMAIL_VERIFY" } </pre> | | <pre> 1 HTTP/2 200 OK 2 Date: Thu, 24 Nov 2022 05:54:59 GMT 3 Content-Type: application/json; charset=utf-8 4 Content-Length: 224 5 Server: nginx/1.10.3 (Ubuntu) 6 X-Dns-Prefetch-Control: off 7 X-Frame-Options: SAMEORIGIN 8 Strict-Transport-Security: max-age=3552000; includeSubDomains 9 X-Download-Options: noopen 10 X-Content-Type-Options: nosniff 11 X-Xss-Protection 1: mode=block 12 Access-Control-Allow-Origin: https://juno.finance 13 Vary: Origin, Accept-Encoding 14 Access-Control-Expose-Headers: date 15 X-RateLimit-Limit: 100 16 X-RateLimit-Remaining: 98 17 X-RateLimit-Reset: 1669355599 18 Etag: W/"e0-DaYLV0N1KJZkio9xwQmFDHLLto" 19 Set-Cookie: AWSALBAPP-Q_remove_; Expires=Thu, 01 Dec 2022 05:54:59 GMT; Path=/; SameSite=None; Secure 20 Set-Cookie: AWSALBAPP-Q_remove_; Expires=Thu, 01 Dec 2022 05:54:59 GMT; Path=/; SameSite=None; Secure 21 Set-Cookie: AWSALBAPP-Q_remove_; Expires=Thu, 01 Dec 2022 05:54:59 GMT; Path=/; SameSite=None; Secure 22 Set-Cookie: AWSALBAPP-Q_remove_; Expires=Thu, 01 Dec 2022 05:54:59 GMT; Path=/; SameSite=None; Secure 23 24 { "status": "success", "message": "Otp sent", "data": { "rediskey": "user:632e934eb6e4cd05ede70194:LOGIN_EMAIL_VERIFY:zbrains9@gmail.com", "redisData": { "retries": 1, "otp": "758673" } }, "request_id": "87e83840-cbbc-11ed-81b3-9b8469a73227" } </pre> | |

Impacts

This information disclosure allows users to sign up with any email address on the platform by bypassing the email verification process.

Remediation

Do not disclose the OTP in the response body or anywhere except the email address.

Retest

This has been fixed. The response body is not disclosing the OTP anymore.

Bug ID#6 [Fixed]

MFA Bypass

Vulnerability Type

Improper Authorization - [CWE-285](#)

Severity

High

Description

Multi-Factor Authentication (MFA) often known as Two-Factor Authentication (2FA) is an added layer of protection added to an application in order to enhance the overall security of the user's account.

The application enforces multifactor authentication after logging in. This is not enforced properly on all the API endpoints. It can be bypassed by force browsing or by capturing the authentication token and using it directly on the APIs.

Vulnerable Endpoint

- https://api.onjuno.com/*

PoC

1. Log in to the application and note that it asks the user to enter the OTP sent to the mobile number.
2. Ignore the OTP and browse the API endpoints as shown below. There's no MFA validation on the APIs.

```
Request
Pretty Raw Hex
1 GET /profile HTTP/2
2 Host: api.onjuno.com
3 Sec-Ch-Ua: "Google Chrome";v="107", "Chromium";v="107",
  "Not=A?Brand";v="24"
4 Onjuno-Browser-Name: Chrome
5 Onjuno-Fp: R042OnzASXvOGPB0ddyk
6 Onjuno-Hardware-Type: computer
7 Authorization: Bearer
  eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYzNjRkMTVhNTc1YmRhZm
  VmYzZGQwY3ZSIzInlhdCI6MTY2OTM1NjIwNjYzXhwIjoxNjY5MzU2NTA2fQ.nei-Hi
  9R7dohqXLRcnYVnH0Rw0_jttEP40C_gdCP84
8 Onjuno-Company-Name:
9 X-Client-IP: 171.76.83.33
10 Sec-Ch-Ua-Platform: "macOS"
11 Onjuno-Browser-Version: 107.0.0.0
12 Sec-Ch-Ua-Mobile: ?0
13 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0
  Safari/537.36
14 Onjuno-Model-Name:
15 Accept: application/json, text/plain, */*
16 Onjuno-Session-Key:
  UjA0Mk9uekFTWHZPR1BCT2RkeWsgMTY2OTM1NTIwOTM2MA==
17 Onjuno-OS-Name: Mac OS
18 Onjuno-Platform: web
19 Onjuno-OS-Version: 10.15.7
20 Origin: https://juno.finance
21 Sec-Fetch-Site: cross-site
22 Sec-Fetch-Mode: cors
23 Sec-Fetch-Dest: empty
24 Referer: https://juno.finance/
25 Accept-Encoding: gzip, deflate
26 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
27 If-None-Match: W/"1414-U4UM0z2UY9HsadsadKs3qNW/048gJc4x8"
28
29

Response
Pretty Raw Hex Render
23 {
  "status": "success",
  "message": "Profile fetched successfully",
  "data": {
    "edit_merchants": {
      "count": 0,
      "first_edited_on": null,
      "last_edited_on": null,
      "is_edit": true,
      "select_max_limit": 10,
      "select_min_limit": 1
    },
    "email_verified": {
      "first_date": "2022-11-04T09:01:56.766Z",
      "last_date": "2022-11-24T10:47:38.645Z"
    },
    "permission": {
      "transaction": {
        "extra": {
          "settlement_delay": null,
          "process_on": null,
          "speed": null
        },
        "credit": true,
        "debit": true
      },
      "flag": null,
      "is_crm": 2,
      "edit_merchant": true,
      "vpn_allowed": false,
      "added_on": null,
      "data": {
        "activation": {
          "account_activated": 1,
          "card_issue": 1,
          "card_pin": 2,
          "select_brand": 2,
          "first_deposit": 3
        }
      }
    }
  }
}
```

Impacts

This vulnerability allows attackers to bypass the multi-factor authentication and the need to enter the OTP sent to the victim's mobile number. This could compromise the account in case the attacker gets a hold of the victim's password.

Remediation

Enforce MFA protection on all the API and application endpoints except the one that verifies the OTP.

Retest

This is fixed. All the sensitive endpoints are behind MFA that can not be bypassed.

Bug ID#7 [Fixed]

JWT HS256 Algorithm Usage

Vulnerability Type

Configuration - [CWE-16](#)

Severity

Low

Description

The application uses the HS256 (HMAC+SHA-256) algorithm which is a symmetric algorithm to create the JWT token. HS256 uses shared secrets to verify the authenticity and to generate a valid token. Since the shared secrets are used, it is possible for an attacker to perform a brute-force attack in order to get hold of the "shared secret" that may allow an attacker to forge a valid JWT Token.

Vulnerable Endpoint

- https://api.onjuno.com/*

PoC

1. Log in to the application and note the JWT being used in the requests to the API inside the Authorization header.

Impacts

A malicious actor with enough resources and time could obtain the "shared secret" by brute force, and then can generate a valid JWT token.

Remediation

While both HS256 and RS256 can be used to allow verification of the integrity of JWTs, the recommended algorithm at this time is RS256.

A signature must ensure authenticity, which means that the JWT content is the same as that generated by the sender. Both HS256 and RS256 algorithms ensure JWT authenticity.

RS256 also ensures non-repudiation, which means it guarantees that the sender has generated the signature. In addition, if the secret key is compromised, you can rotate signing keys without re-deploying your application with a new secret, as you would have to do with HS256.

References: <https://auth0.com/blog/rs256-vs-hs256-whats-the-difference/>

Retest

This is fixed. The algorithm has been changed to RS256.

Bug ID#8 [Won't Fix]

Misconfigured Content Security Policy

Vulnerability Type

Configuration - [CWE-16](#)

Severity

Low

Description

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross-Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft, to site defacement, to malware distribution.

The CSP is found to be misconfigured in the application and allows the unsafe-inline attribute in script-src along with whitelisted domains that are known to host JSONP endpoints and Angular libraries that render the CSP useless in case an attack like XSS is found.

Vulnerable Endpoint

- <https://juno.finance/>*

PoC

1. Copy the CSP shown in the response headers and evaluate it using Google's CSP Evaluator at <https://csp-evaluator.withgoogle.com/>
2. It can be seen that the script-src directive allows unsafe-inline and various whitelisted domains that are considered unsafe.

| | |
|---|---|
| ✓ default-src | |
| ❗ script-src | Host whitelists can frequently be bypassed. Consider using 'strict-dynamic' in combination with CSP nonces or hashes. |
| 🔍 'self' | 'self' can be problematic if you host JSONP, Angular or user uploaded files. |
| ❗ 'unsafe-inline' | 'unsafe-inline' allows the execution of unsafe in-page scripts and event handlers. |
| 🔍 'unsafe-eval' | 'unsafe-eval' allows the execution of code injected into DOM APIs such as eval(). |
| 🔍 https://cdn.juno.finance | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 https://cdn.juno.finance/ | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 https://juno-fi-cdn.s3.us-west-1.amazonaws.com/ | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 *.juno.finance | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 *.onjuno.com | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 https://juno.finance | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 https://onjuno.com | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 https://utt.impactcdn.com/A3616483-3c96-4bcc-8f84-a65bf71b01f61.js | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 https://unpkg.com/@lottiefiles/lottie-interactivity@latest/dist/lottie-interactivity.min.js | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 *.ubembed.com/ | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 https://www.tp88trk.com/scripts/sdk/everflow.js | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 https://widget.frill.co/v2/widget.js | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| ❗ https://www.googleadservices.com | www.googleadservices.com is known to host JSONP endpoints and Angular libraries which allow to bypass this CSP. |
| ❗ https://googleads.g.doubleclick.net | googleads.g.doubleclick.net is known to host JSONP endpoints which allow to bypass this CSP. |
| 🔍 https://tags.srv.stackadapt.com/events.js | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 https://tpnqmodn.net/ | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 https://sc-static.net/scevent.min.js | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 https://www.redditstatic.com/ads/pixel.js | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 *.atomicfi.com/transact.js | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 https://www.recaptcha.net/recaptcha/api.js | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 *.firebaseio.com/ | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 https://argyle-link.firebaseio.com/ | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 https://plugin.argyle.com/argyle.web.v3.js | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 https://accounts.google.com/gsi/client | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 https://appleid.cdn-apple.com | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 *.sardine.ai | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 *.clevertap.com | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| ❗ *.twitter.com | syndication.twitter.com is known to host JSONP endpoints which allow to bypass this CSP. |
| 🔍 http://static.ads-twitter.com/ | Allow only resources downloaded over HTTPS. |
| 🔍 https://browser.sentry-cdn.com/ | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 https://analytics.tiktok.com/ | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |
| 🔍 https://widget.trustpilot.com/bootstrap/v5/tp.widget.bootstrap | No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries. |

Impacts

Avoid using the unsafe-inline keyword in the CSP policy. This keyword annuls most of the security benefits that Content-Security-Policy provides. This leaves the application vulnerable to attacks like XSS and makes the presence of a Content Security Policy useless. The whitelisted domains are also known to host JSONP endpoints and Angular libraries that can be used to bypass the policy and exploit issues like XSS.

Remediation

- Do not use unsafe-inline.

- Remove the whitelisted domains or use a nonce.
- Use Google's CSP Evaluator to analyze the CSP directives and make changes accordingly.

Retest

This can not be fixed right now as it requires massive architecture changes.

Bug ID#9 [Won't Fix]

Concurrent Sessions Allowed

Vulnerability Type

Manage User Sessions - [CWE-1018](#)

Severity

Informative

Description

The application allows multiple concurrent sessions for the same user. In applications where the integrity of data is essential allowing concurrent sessions could compromise that integrity.

It is generally considered a good approach to implement some type of thread-awareness into web applications. There are multiple dangerous scenarios that can arise from multiple threads interacting with an object simultaneously. The results of this situation can vary largely, depending on the application language and framework, and environmental factors such as server load.

Vulnerable Endpoint

- https://api.onjuno.com/*

PoC

1. Log in to the application using a valid account from Browser 1.
2. Open a private session or another browser, where the session caches and cookies are not shared, and log into the same account from this Browser 2.
3. Observe that both sessions work simultaneously.

Impacts

This situation does not necessarily constitute a vulnerability, and may sometimes even be considered a helpful feature - this is something the administrators of the application need to consider when judging whether this allows for a security vulnerability vector.

The vector by itself would be reported as informational. If a race condition is found or another aspect of the site's operation can be affected through this multi-threading aspect, the vector would increase in criticality. In this instance, no such situation was detected.

Remediation

- As a best practice, consider disallowing multiple concurrent user sessions or logins.
- Sessions should ideally be hooked to the database ID
- User data should be associated with the active session

Retest

This won't be fixed. The application allows 3 devices at a time.

Bug ID#10 [Fixed]

Missing Secure and HttpOnly attributes in Cookies

Vulnerability Type

Configuration - [CWE-16](#)

Severity

Medium

Description

The Secure flag prevents the browser from sending a cookie to a URL outside of an SSL connection, thereby protecting the confidentiality of the cookie.

The HttpOnly flag protects the cookie from being accessed via the DOM, and therefore being accessed by client-side scripts. Whilst it may be desirable to have some cookies accessible by client scripts, critical cookies such as the Session ID should be protected in order to prevent session hijacking.

Vulnerable Endpoint

- <https://juno.finance/>*

PoC

1. Log in to the application and open the browser's Storage console to check the missing HttpOnly and Secure attributes in Cookies.

| Name | Value | Domain | Path | Expires / Max-Age | Size | HttpOnly | Secure | SameSite | Last Accessed |
|----------------------|---------------------------------------|--------------|------|-------------------------------|------|----------|--------|----------|-------------------------------|
| .ce.s | v=50183009d52f831204a0d2116e7... | juno.finance | / | Wed, 29 Nov 2023 04:44:33 GMT | 53 | false | false | None | Tue, 29 Nov 2022 05:16:39 GMT |
| .fig | 86.1166969797263.1679934284 | juno.finance | / | Mon, 27 Feb 2023 05:16:42 GMT | 83 | false | false | None | Tue, 29 Nov 2022 05:16:42 GMT |
| .js_K0NRE1LV3 | Q51.1.1669697069.1.1.1669697069.0... | juno.finance | / | Thu, 28 Nov 2024 05:16:42 GMT | 51 | false | false | None | Tue, 29 Nov 2022 05:16:42 GMT |
| .jsgt_UA-149641098-1 | 1 | juno.finance | / | Tue, 29 Nov 2022 05:17:41 GMT | 20 | false | false | None | Tue, 29 Nov 2022 05:16:41 GMT |
| .js_ga | GA1.2.78497890.1669697069 | juno.finance | / | Thu, 28 Nov 2024 05:16:41 GMT | 29 | false | false | None | Tue, 29 Nov 2022 05:16:41 GMT |
| .js_gd | GA1.1.1167421640.1669697069 | juno.finance | / | Mon, 27 Feb 2023 04:44:28 GMT | 32 | false | false | None | Tue, 29 Nov 2022 05:16:39 GMT |
| .js_gd | GA1.2.176313661.1669697072 | juno.finance | / | Wed, 30 Nov 2022 05:16:41 GMT | 30 | false | false | None | Tue, 29 Nov 2022 05:16:41 GMT |
| .js_gd | 1669697069.367a80e-6564-463... | juno.finance | / | Mon, 27 Feb 2023 05:13:57 GMT | 59 | false | true | Strict | Tue, 29 Nov 2022 05:16:39 GMT |
| .js_gd | 1669697069 | juno.finance | / | Session | 12 | false | false | Lax | Tue, 29 Nov 2022 05:16:39 GMT |
| .js_gd | c9396ac-3a0d-4d5e-4b67-1a478b1... | juno.finance | / | Fri, 29 Dec 2023 21:02:47 GMT | 41 | false | false | Lax | Tue, 29 Nov 2022 05:16:39 GMT |
| .js_gd | 11166966020000 | juno.finance | / | Fri, 29 Dec 2023 21:02:47 GMT | 20 | false | false | Lax | Tue, 29 Nov 2022 05:16:39 GMT |
| .js_gd | LS5B56WwRfTE1GwQK7mqQQQL... | juno.finance | / | Wed, 29 Nov 2023 04:44:50 GMT | 98 | false | false | Lax | Tue, 29 Nov 2022 05:16:39 GMT |
| .js_gd | 1 | juno.finance | / | Session | 5 | false | false | None | Tue, 29 Nov 2022 05:16:39 GMT |
| .js_gd | ucB8p8rnp1JubBdQ0 | juno.finance | / | Session | 22 | false | false | None | Tue, 29 Nov 2022 05:16:39 GMT |
| .js_gd | gmgle | juno.finance | / | Fri, 31 Dec 9999 13:00:00 GMT | 20 | false | false | None | Tue, 29 Nov 2022 05:16:39 GMT |
| .js_gd | 95c39c41.31.42.41.4e-bd73-1f6d86... | juno.finance | / | Sat, 26 Aug 2023 05:47:29 GMT | 63 | false | false | Lax | Tue, 29 Nov 2022 05:16:39 GMT |
| .js_gd | 5dc2a07.5917.4108-a264-fdbd494c... | juno.finance | / | Sat, 26 Aug 2023 05:47:29 GMT | 56 | false | false | Lax | Tue, 29 Nov 2022 05:16:39 GMT |
| .js_gd | QThrsEzNHNEkRd5L3jKXBBz20M... | juno.finance | / | Tue, 06 Dec 2022 05:14:09 GMT | 187 | false | false | Lax | Tue, 29 Nov 2022 05:16:39 GMT |
| .js_gd | 166969883691%7CON7C166969883... | juno.finance | / | Session | 47 | false | false | None | Tue, 29 Nov 2022 05:16:39 GMT |
| .js_gd | juno.finance | juno.finance | / | Session | 18 | false | false | None | Tue, 29 Nov 2022 05:16:39 GMT |
| .js_gd | 687a035-ac95-3487-0442-429168f... | juno.finance | / | Mon, 18 Nov 2024 05:13:58 GMT | 57 | false | false | None | Tue, 29 Nov 2022 05:16:39 GMT |
| .js_gd | null | juno.finance | / | Session | 13 | false | false | None | Tue, 29 Nov 2022 05:13:58 GMT |
| .js_gd | edtya@credshields.com | juno.finance | / | Session | 31 | false | false | None | Tue, 29 Nov 2022 05:13:58 GMT |
| .js_gd | eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVC... | juno.finance | / | Tue, 29 Nov 2022 06:38:41 GMT | 187 | false | false | None | Tue, 29 Nov 2022 05:13:53 GMT |
| .js_gd | eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVC... | juno.finance | / | Session | 180 | false | false | None | Tue, 29 Nov 2022 05:13:53 GMT |
| .js_gd | 9N253AR0Dmduq5.3pE0516076d... | juno.finance | / | Wed, 29 Nov 2023 05:14:00 GMT | 94 | false | false | None | Tue, 29 Nov 2022 05:14:00 GMT |
| .js_gd | 9N253AR0Dmduq5.3pE0516076d... | juno.finance | / | Wed, 29 Nov 2023 05:14:00 GMT | 110 | false | false | None | Tue, 29 Nov 2022 05:14:00 GMT |
| .js_gd | 300 | juno.finance | / | Session | 14 | false | false | None | Tue, 29 Nov 2022 05:13:58 GMT |
| .js_gd | 046815f9e1541a5a58de5162876f... | juno.finance | / | Fri, 29 Nov 2023 05:13:57 GMT | 38 | false | false | None | Tue, 29 Nov 2022 05:16:39 GMT |
| .js_gd | %7B%22%22%3A%22%22%22%22%22... | juno.finance | / | Tue, 29 Nov 2022 05:36:18 GMT | 82 | false | false | None | Tue, 29 Nov 2022 05:16:18 GMT |

Impacts

This issue can be leveraged in order to capture session cookies and impersonate other users by forging new tokens.

Remediation

Set the Secure and HTTPOnly cookie flags for all the critical cookies.

Retest

This is fixed. HttpOnly and Secure attributes have been added to sensitive cookies.

Bug ID#11 [Won't Fix]

Card PIN Update missing additional Security Validations

Vulnerability Type

Configuration - [CWE-16](#)

Severity

Medium

Description

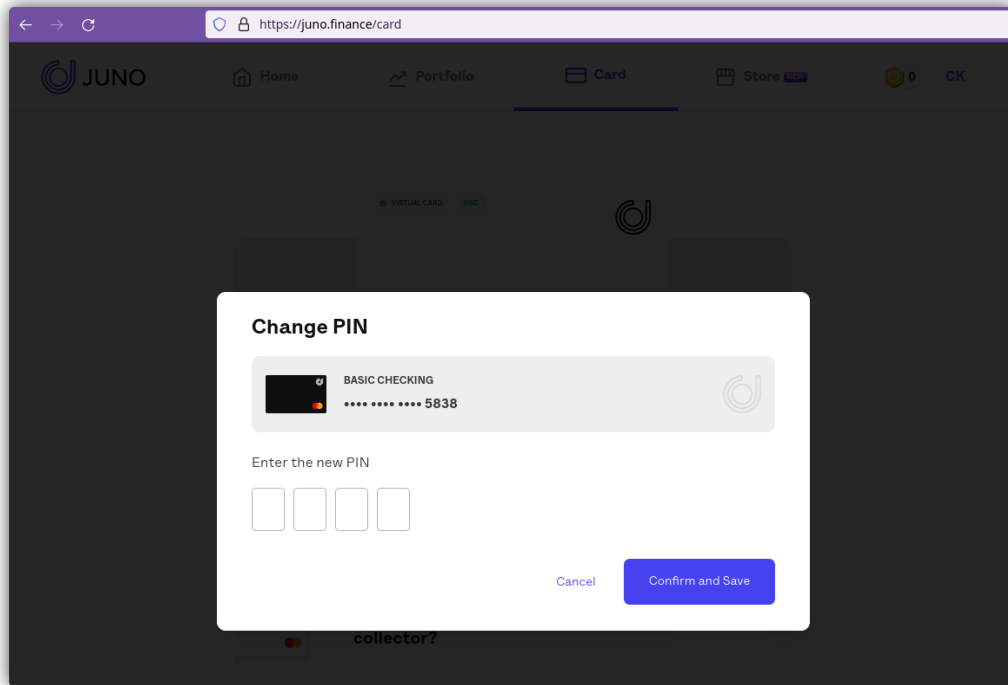
The application does not ask for any additional validations when updating card PIN details.

Vulnerable Endpoint

- <https://api.onjuno.com/cards/637f4b6cd2b472ccfcab1a8a/set-pin>

PoC

1. Log in to the application and head over to Cards.
2. Try to update the card pin and note that it does not ask for any additional validations before changing the PIN.



Impacts

This vulnerability might allow authenticated attackers to update card details without requiring any additional validations.

Remediation

Require users to have additional security when they update their card details. Two-step validation such as old pin number or OTP can be used.

Retest

This won't be fixed since 2FA is implemented during login and a card pin is not necessary for transactions in the USA.

Bug ID#12 [Fixed]

Missing Access Control in Fetching Card Details

Vulnerability Type

Improper Access Control - [CWE-284](#)

Severity

High

Description

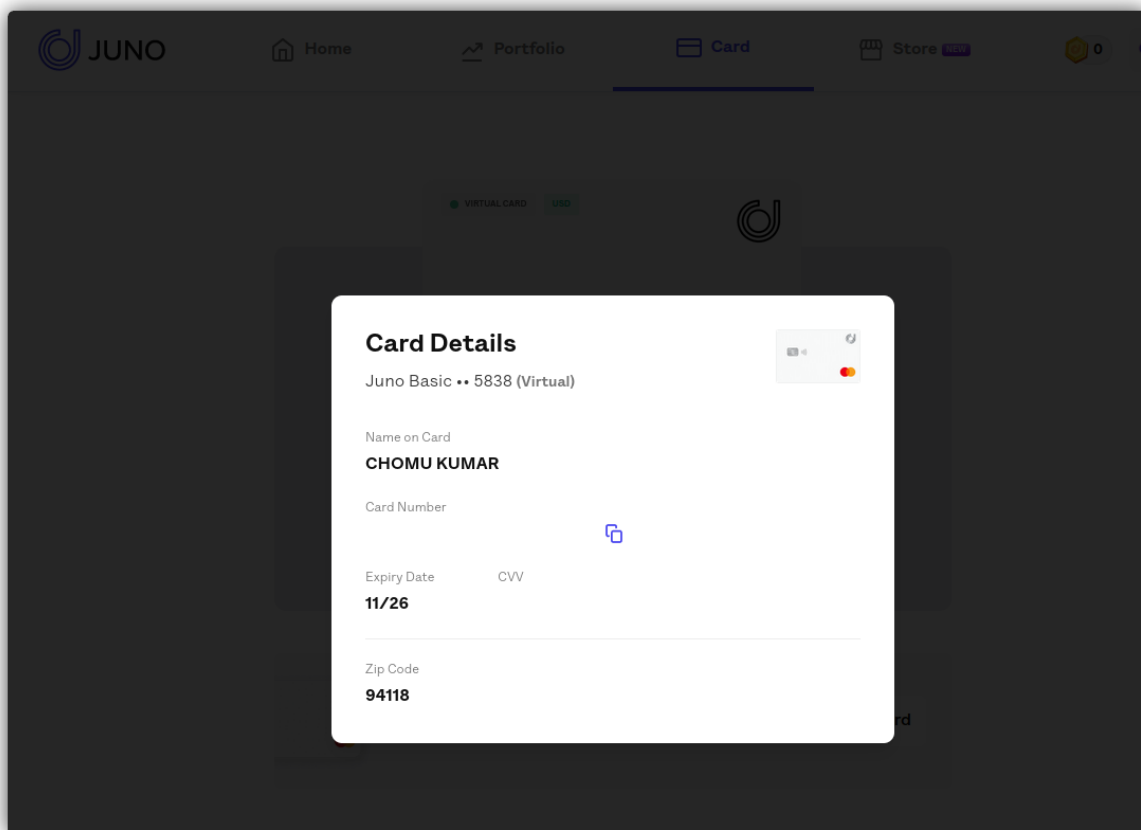
The application lacks access control validations on the endpoint to fetch card details allowing users to get card details for any user on the platform by sending requests to "/cards/<card_id>".

Vulnerable Endpoint

- <https://api.onjuno.com/cards/6364d691ba2f213fdb4b9376>

PoC

1. Log in to the application and head over to Cards.
2. Click on "View Details" to view the card details and change the value of the card ID with any card ID that is owned by another user.
3. The response will display the card details of the specified card ID.



| Request | Response |
|--|---|
| <pre> 1 GET /cards/6364d691ba2f213fdb4b9376 HTTP/2 2 Host: api.onjuno.com 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:107.0) Gecko/20100101 Firefox/107.0 4 Accept: application/json, text/plain, */* 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Referer: https://juno.finance/ 8 Onjuno-Platform: web 9 Onjuno-OS-Name: Linux 10 Onjuno-OS-Version: x86_64 11 Onjuno-Browser-Name: Firefox 12 Onjuno-Browser-Version: 107.0 13 Onjuno-Hardware-Type: computer 14 Onjuno-Session-Key: dwlj0HJicEVyc3AxSVXJaTh4RDagMTY20TY50DUx0TcxNw== 15 Onjuno-Fp: uic8rbpErspIiui8xD0 16 X-Client-Id: 49.37.73.72 17 Origin: https://juno.finance 18 Sec-Fetch-Dest: empty 19 Sec-Fetch-Mode: cors 20 Sec-Fetch-Site: cross-site 21 Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYzN2YzNjgyZDcyMTVjZDZj NGJkMjJmYyIsImh0dCI6MTY20TcwMTExOjYzN2YzNjgyZDcyMTVjZDZj Ni4RbXAvYzZlLDM6ZDk0SAa4quFymN58 22 Onjuno-Company-Name: 23 Onjuno-Model-Name: 24 Te: trailers 25 26 </pre> | <pre> { "updated_at": null, "id": "6364d691ba2f213fdb4b9376", "activation_date": "2022-11-29T05:44:42.848Z", "added_on": "2022-11-04T09:08:33.782Z", "astra_juno_card_deleted_at": null, "astra_juno_card_id": null, "card_limit": null, "expiry_date": "2026-11-30T00:00:00.000Z", "name": "Virtual Card", "number": "9885", "preferences": { "allow_cash_transactions": true, "allow_foreign_transactions": true, "daily_cash_limit": 1875, "daily_transaction_limit": 7500, "monthly_cash_limit": null, "monthly_transaction_limit": null }, "status": "ACTIVE", "sub_type": null, "type": "virtual", "vgs_cvc": "3d284f0e-c8de-452a-b15c-db973de9a606", "vgs_number": "9915104043210369885", "is_verified": true, "isProcessedData": true, "vgs_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYzN2YzNjgyZDcyMTVjZDZjNGJkMjJmYyIsImh0dCI6MTY20TcwMTExOjYzN2YzNjgyZDcyMTVjZDZjLCJpYXQiOiJlZDZjNGJkMjJmYyIsImh0dCI6MTY20TcwMTExOjYzN2YzNjgyZDcyMTVjZDZjJnChT5hIz-b6eudBU0eC8kz7QU", "vgs_token_expiry_time": 3600 } </pre> |

Impacts

This vulnerability allows authenticated users to get card details owned by any user account on the platform given that the card ID for the other user is known.

Remediation

Implement access control validation on the endpoint to make sure a user is only allowed to access and modify details belonging to their own account.

Retest

This has been fixed. The validations are properly implemented and will throw an error if the user tries to access unauthorized card details.

Outdated Nginx and Version Disclosure

Using Components with known vulnerabilities

Low

An outdated version of Nginx (1.10.3) was found on the Juno Web application and the API which is affected by multiple publicly disclosed exploits and CVEs.

- <https://juno.finance>
- <https://api.juno.finance>

1. The version of Nginx can be checked in the response headers as shown below:

| Request | Response |
|---|--|
| <pre> Pretty Raw Hex GET /cards/6364d091ba2f13fdb4b9376 HTTP/2 Host: api.onjuno.com User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:107.0) Gecko/20100101 Firefox/107.0 Accept: application/json, text/plain, */* Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Referer: https://jjuno.finance/ Onjuno-Platform: web Onjuno-Os-Name: Linux Onjuno-Os-Version: x86_64 Onjuno-Browser-Name: Firefox Onjuno-Browser-Version: 107.0 Onjuno-Hardware-Type: computer Onjuno-Session-Key: dwljOHJicEYvc3AKSXVJaTh4RDAGMY2TYc2M04MJQ3MEJQ== Onjuno-Fp: uic8rbpErspIuIi8x00 X-Client-Ip: 49.37.73.72 Origin: https://jjuno.finance/ Sec-Fetch-Dest: empty Sec-Fetch-Mode: cors Sec-Fetch-Site: cross-site Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjZlNzY2NGMyTDZlZjNmIGwKImVjbmVzImhlbGMIGMTYT0tZC1MjQ4MWwiZXhpdjI1bnJmY2ZyZWgNZGVfLj04LWZlcnR5ZWgkHGFScFkpID0tQDh0IHA Onjuno-Company-Name: Onjuno-Model-Name: If-None-Match: W/"4b5-APXqJgwKeJR/lr0jLBAlmx/YGU" Te: trailers </pre> | <pre> Pretty Raw Hex Render 1 HTTP/2 401 Unauthorized 2 Date: Wed, 30 Nov 2022 05:11:44 GMT 3 Content-Type: application/json; charset=utf-8 4 Content-Length: 154 5 Server: nginx/1.10.3 (Ubuntu) 6 X-Ons-Prefetch-Control: off 7 X-Frame-Options: SAMEORIGIN 8 Strict-Transport-Security: max-age=15552000; includeSubDomains 9 X-Download-Options: noopen 10 X-Content-Type-Options: nosniff 11 X-Xss-Protection: 1; mode=block 12 Access-Control-Allow-Origin: https://jjuno.finance 13 Vary: Origin, Accept-Encoding 14 Access-Control-Expose-Headers: date 15 X-Ratelimit-Limit: 300 16 X-Ratelimit-Remaining: 299 17 X-Ratelimit-Reset: 1669795165 18 Etag: W/"9a-vinOfT4Wjt7ijtyScB-Gn5wg" 19 Set-Cookie: AWSALBAPP-0_remove_; Expires=Wed, 07 Dec 2022 05:11:44 GMT; Path=/ 20 Set-Cookie: AWSALBAPP-1_remove_; Expires=Wed, 07 Dec 2022 05:11:44 GMT; Path=/ 21 Set-Cookie: AWSALBAPP-2_remove_; Expires=Wed, 07 Dec 2022 05:11:44 GMT; Path=/ 22 Set-Cookie: AWSALBAPP-3_remove_; Expires=Wed, 07 Dec 2022 05:11:44 GMT; Path=/ 23 24 { "status": "failed", "message": "Restricted: User Authentication Required", "data": null, "error_code": 4008, "request_id": "7b61b320-706d-11ed-a070-ad04b95d5447" } </pre> |

Impacts

This version of Nginx is affected by multiple vulnerabilities and exploits such as Remote code executions, denial of service, Information disclosure, etc. A complete list can be found here - <https://www.cybersecurity-help.cz/vdb/nginx/nginx/1.10.3/>

Remediation

- Update the Nginx to its latest patched version.
- Do not disclose the software name and version anywhere in the response headers and the body.

Retest

Nginx version is not disclosed anymore in the response headers.

Bug ID#14 [Fixed]

Malicious File Upload

Vulnerability Type

File Upload

Severity

Low

Description

The application allows users to upload images of check deposits to its S3 bucket. There is a file type validation that can be bypassed by forcefully uploading files with content other than images. The backend only checks for mime type and extension but regardless of them, keeps the content type based on the content of the file.

Vulnerable Endpoint

- <https://dev-api.juno.finance/transactions/rdc>

PoC

1. Log in to your account and go to Check Deposits.
2. Fill in all the required fields and proceed to upload the images.
3. There's a file type validation that only allows uploading for image files. This can be bypassed by changing the file content and keeping the extension and mime type as images as shown below.
4. The file will be uploaded as SVG regardless of the mime and extension - https://juno-develop.s3.us-west-2.amazonaws.com/rdc/6380b7ef6dab000afcf6010/1669786722156_file1_front.jpg

| Request | Response |
|---|---|
| <pre> 33 Content-Disposition: form-data; name="from" 34 35 "" 36 -----8857984272589777392771870031 37 Content-Disposition: form-data; name="note" 38 39 "" 40 -----8857984272589777392771870031 41 Content-Disposition: form-data; name="file1"; filename="front.jpg" 42 Content-Type: image/jpeg 43 44 <?xml version="1.0" standalone="no"?> 45 <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" 46 "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd"> 47 <svg version="1.1" baseProfile="full" 48 xmlns="http://www.w3.org/2000/svg"> 49 <polygon id="triangle" points="0,0 0,50 50,0" fill="#009900" 50 stroke="#004400"/> 51 <script type="text/javascript"> 52 alert('XSS\n'+document.domain+'\n'+document.cookie); 53 </script> 54 </svg> 55 -----8857984272589777392771870031 56 Content-Disposition: form-data; name="file2"; filename="back.jpg" 57 Content-Type: image/jpeg 58 59 <?xml version="1.0" standalone="no"?> 60 <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" 61 "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd"> 62 <svg version="1.1" baseProfile="full" 63 xmlns="http://www.w3.org/2000/svg"> 64 <polygon id="triangle" points="0,0 0,50 50,0" fill="#009900" 65 stroke="#004400"/> 66 <script type="text/javascript"> 67 alert('XSS\n'+document.domain+'\n'+document.cookie); 68 </script> 69 </svg> 70 -----8857984272589777392771870031-- </pre> | <pre> { "lon":0 }, { "fees":{ "amount":0, "title":null, "note":null, "list":[] }, "referral_bonus":{ "user_id":null, "user_type":null }, "status":{ "code":0, "date":"2022-11-30T05:38:42.433Z", "note":null, "title":"INITIATED" }, "interest_bonus":{ "period":1, "from_date":null, "to_date":null }, "cheque":{ "cheque_id":"6386ec62185a9681d3242abb", "front_url": "https://juno-develop.s3.us-west-2.amazonaws.com/rdc/6380b7ef6dab0 00afcf6e010/1669786722156_file1_front.jpg", "back_url": "https://juno-develop.s3.us-west-2.amazonaws.com/rdc/6380b7ef6dab0 00afcf6e010/1669786722156_file2_back.jpg" }, "plaid_fraud":{ "bank_score":null, "consumer_score":null, "scores":null, "core_attributes":null }, "wise":{ "static":/ </pre> |

Impacts

This vulnerability allows users to bypass file upload and upload files with any type and store them on Juno's S3 bucket. These files can then be shared with users to exploit their systems.

Remediation

Do not allow users to upload files with any arbitrary content. It is recommended to validate the file content as well and discard any invalid files.

Retest

Bug ID#15 [Fixed]

Weak JWT Signing Key on Dev

Vulnerability Type

Configuration - [CWE-16](#)

Severity

High

Description

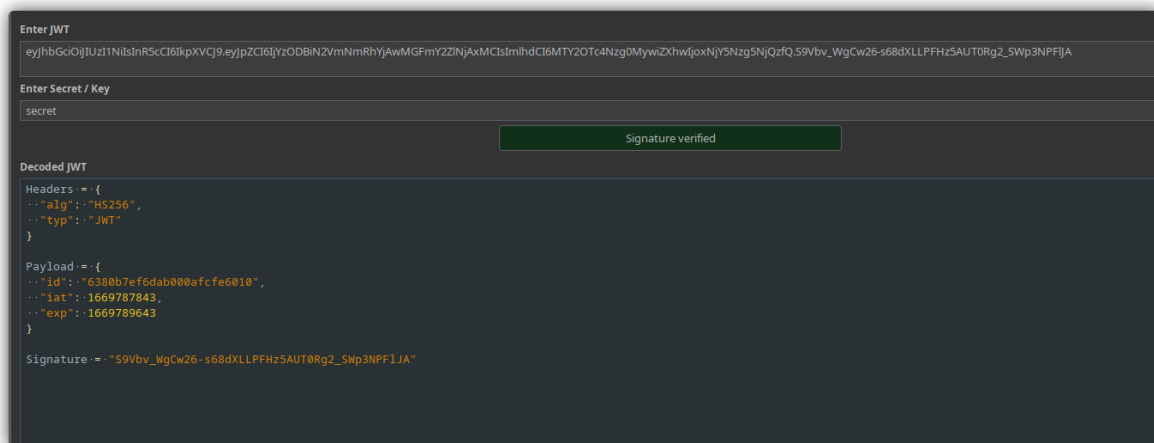
The dev environment is using JWT with HMAC signing key to sign the tokens. This key is weak and can be brute-forced offline to get the key in plaintext which can then be used to forge new JWT for other users using their account IDs.

Vulnerable Endpoint

- <https://dev-api.juno.finance/>*

PoC

1. Log in to the application and get a hold of the jwt generated.
2. This can be brute-forced to obtain the secret using tools like jwt-hack.
3. The secret key was found to be "secret".



Impacts

This could lead to account compromise by forging custom JWT. Even though this is not affecting production servers, since dev is publicly accessible, it may allow users to access anyone's testing accounts that could contain potentially sensitive users' PII.

Remediation

Keep a strong signing secret to sign the JWTs. It is also suggested to move to RS256 instead of HMAC.

Retest

This is fixed. The algorithm has been changed to RS256 which remediates this bug.

Bug ID#16 [Won't Fix]

Weak Card PIN

Vulnerability Type

Configuration - [CWE-16](#)

Severity

Medium

Description

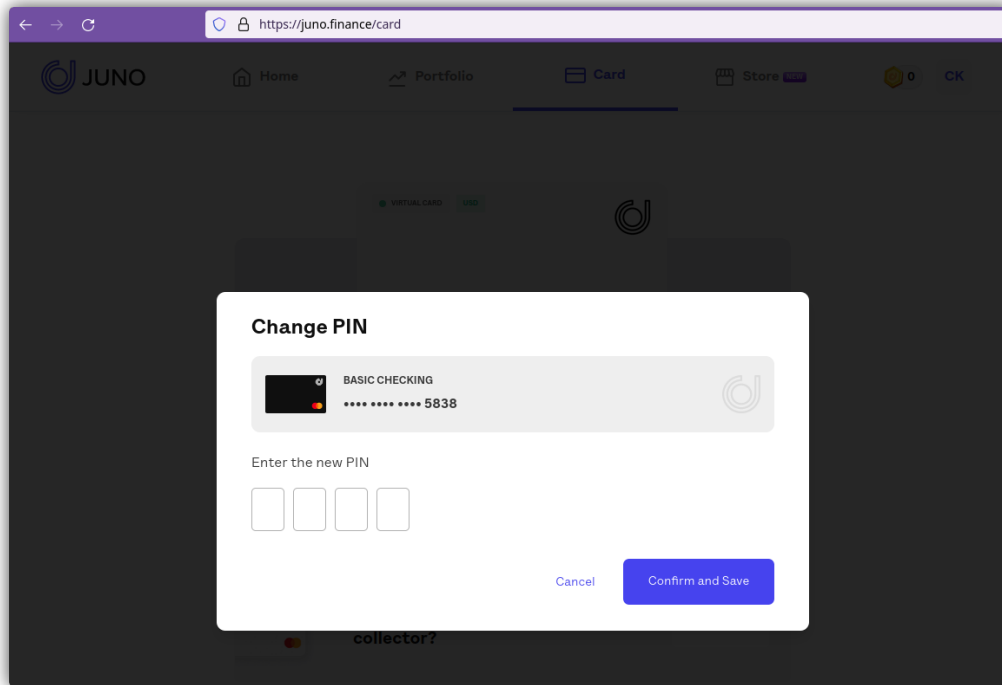
The cards in the Juno application allow users to set PINs to protect them against unauthorized usage. This PIN is only 4 digits in length which is not considered safe.

Vulnerable Endpoint

- <https://juno.finance/>*

PoC

1. Log in to the application and head over to Cards.
2. Try to update the card pin and note that it has a limit of 4 digits.



Impacts

Having a short PIN could allow attackers to guess or bruteforce them.

Remediation

It is suggested to set the minimum limit to at least 6 digits to increase overall security around card usage.

Retest

This can not be fixed since the PIN is enforced by the banking partner to have 4 digits.

Bug ID#17 [Fixed]

Missing Access Control in Fetching Pay Allocations

Vulnerability Type

Improper Access Control - [CWE-284](#)

Severity

Medium

Description

The application lacks access control validations on the endpoint to fetch pay allocation details allowing users to get details for any user on the platform by sending requests to "/v2/deposit-switch/get?pay_allocation_id=<allocation_id>".

Vulnerable Endpoint

- https://dev-api.juno.finance/v2/deposit-switch/get?pay_allocation_id=639828fe8f6a196b3a02b3d2

PoC

1. Log in to the platform and create a direct deposit using Amazon on two different accounts.
2. Open the direct deposit you just created and you'll see a parameter called pay_allocation_id being passed in the URL.
3. Use the allocation ID from the other account using a different user's session token and the API will show their details.

```
Request
Pretty Raw Hex
1 GET /v2/deposit-switch/get?pay_allocation_id=639828fe8f6a196b3a02b3d2 HTTP/2
2 Host: dev-api.juno.finance
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:107.0) Gecko/20100101 Firefox/107.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: https://dev.juno.finance/
8 OnJuno-Platform: web
9 OnJuno-OS-Name: Linux
10 OnJuno-OS-Version: x86_64
11 OnJuno-Browser-Name: Firefox
12 OnJuno-Browser-Version: 107.0
13 OnJuno-Hardware-Type: computer
14 OnJuno-Session-Key: ZJE10TI2YQ2Y2NHN2RkZTVmNDZlNjI3NTU3ODFhODUgMTY3MDk5Mjk4MDA3Mg==
15 OnJuno-Fp: f15923b46cca7dde5f40e62755781a85
16 X-Client-Id: 49.37.73.72
17 Origin: https://dev.juno.finance
18 Sec-Fetch-Dest: empty
19 Sec-Fetch-Mode: cors
20 Sec-Fetch-Site: same-site
21 Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6ImlhbnR5cyJ9.eyJpIjoiYz00bWVhbnR5YAwMGFvY2ZlNjAxMCI6Imh0dCI6MTY3MDk5Mjk4MDA3Mg==MzZmSWZxhWjoxNjIwOTk0MDYxYQ.yd7_hayzTjwdb_L0fEvSoEz2weyellEwIjPz00VhL3M
22 OnJuno-Company-Name:
23 OnJuno-Model-Name:
24 If-None-Match: W/"5fd-/H1ta0zCAFTBEyaZlshLahEzx10"
25 Te: trailers
26
27

Response
Pretty Raw Hex Render
{"connection":{"status":"connected","error_code":null,"error_message":null,"updated_at":"2022-12-13T07:25:32.813748Z"},"pay_distribution":{"status":"success","error_code":null,"error_message":null,"updated_at":"2022-12-14T04:54:00.979717Z"},"employer":{"id":"amazon","name":"Amazon","url":"https://res.cloudinary.com/argyle-media/image/upload/v1598543068/partner-logos/amazon.png"},"link_item":{"id":"amazon","name":"Amazon","url":"https://res.cloudinary.com/argyle-media/image/upload/v1598543068/partner-logos/amazon.png"},"paycheck_info":{"id":"61418aadd48662b0786d45f","profile_name":"All Cash","split_type":"percent","distribution":{"currency_short_name":"USD","split":100,"beneficiary_id":null,"alternative_name":"Cash"}},{"status":1,"amount":550,"applied_on":"2022-12-14T04:57:30.995Z"}}
```

Impacts

This vulnerability allows authenticated users to get pay allocation details for other user accounts on the platform given that the allocation ID for the other user is known or brute-forced.

Remediation

Implement access control validation on the endpoint to make sure a user is only allowed to access and modify details belonging to their own account.

Retest

This is fixed. Access controls are implemented.

6. Disclosure

The Reports provided by CredShields is not an endorsement or condemnation of any specific project or team and do not guarantee the security of any specific project. The contents of this report are not intended to be used to make decisions about buying or selling tokens, products, services, or any other assets and should not be interpreted as such.

Emerging technologies such as Web3 carry a high level of technical risk and uncertainty. CredShields does not provide any warranty or representation about the quality of code, the business model or the proprietors of any such business model, or the legal compliance of any business. The report is not intended to be used as investment advice and should not be relied upon as such.

CredShields Audit team is not responsible for any decisions or actions taken by any third party based on the report.