hashlock.

2024

Penetration Testing

Vield (FinTech)

October 2024

info@hashlock.com.au

Hashlock Pty Ltd

Table of Contents

CAUTION

THIS DOCUMENT IS A SECURITY PENETRATION TEST REPORT AND MAY CONTAIN CONFIDENTIAL INFORMATION. THIS INCLUDES IDENTIFIED VULNERABILITIES AND MALICIOUS CODE THAT COULD BE USED TO COMPROMISE THE PROJECT. THIS DOCUMENT SHOULD ONLY BE FOR INTERNAL USE UNTIL ISSUES ARE RESOLVED. ONCE VULNERABILITIES ARE REMEDIATED, THIS REPORT CAN BE MADE PUBLIC. THE CONTENT OF THIS REPORT IS OWNED BY HASHLOCK PTY LTD FOR THE USE OF THE CLIENT.

Executive Summary

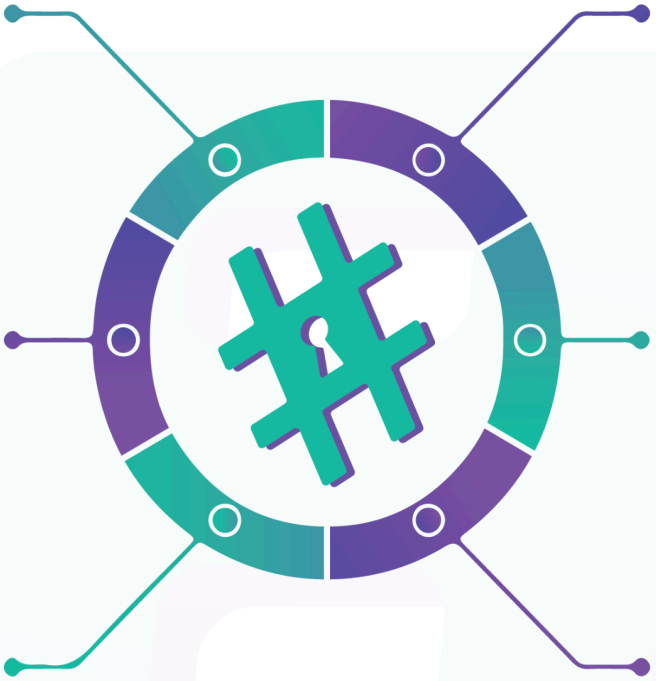
The Vield team partnered with Hashlock to conduct a security Penetration Test of their Vield Project code base. Hashlock manually and proactively reviewed the code to ensure the project's team and community that the deployed contracts were secure.

Project Context

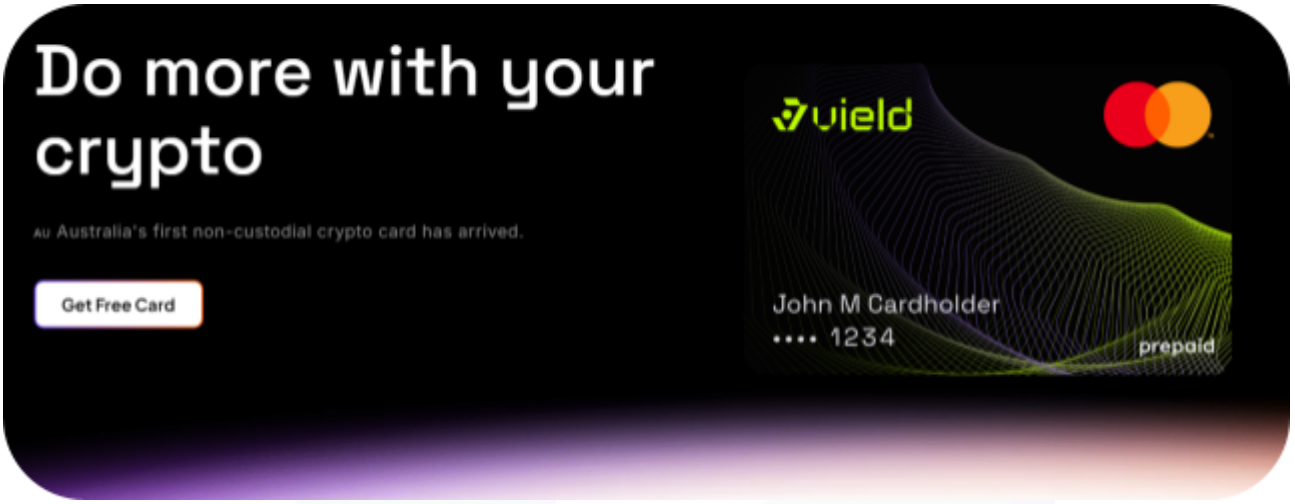
Vield offers Australia's first non-custodial crypto debit card, enabling users to spend stablecoins like USDC anywhere Mastercard is accepted. The platform also provides crypto-backed loans, allowing users to borrow against their BTC or ETH without selling their assets. Vield focuses on financial freedom for crypto natives, emphasizing control over digital assets and seamless stablecoin transactions. This project bridges the gap between Web3 and real-world financial use, making crypto more accessible through decentralized solutions.

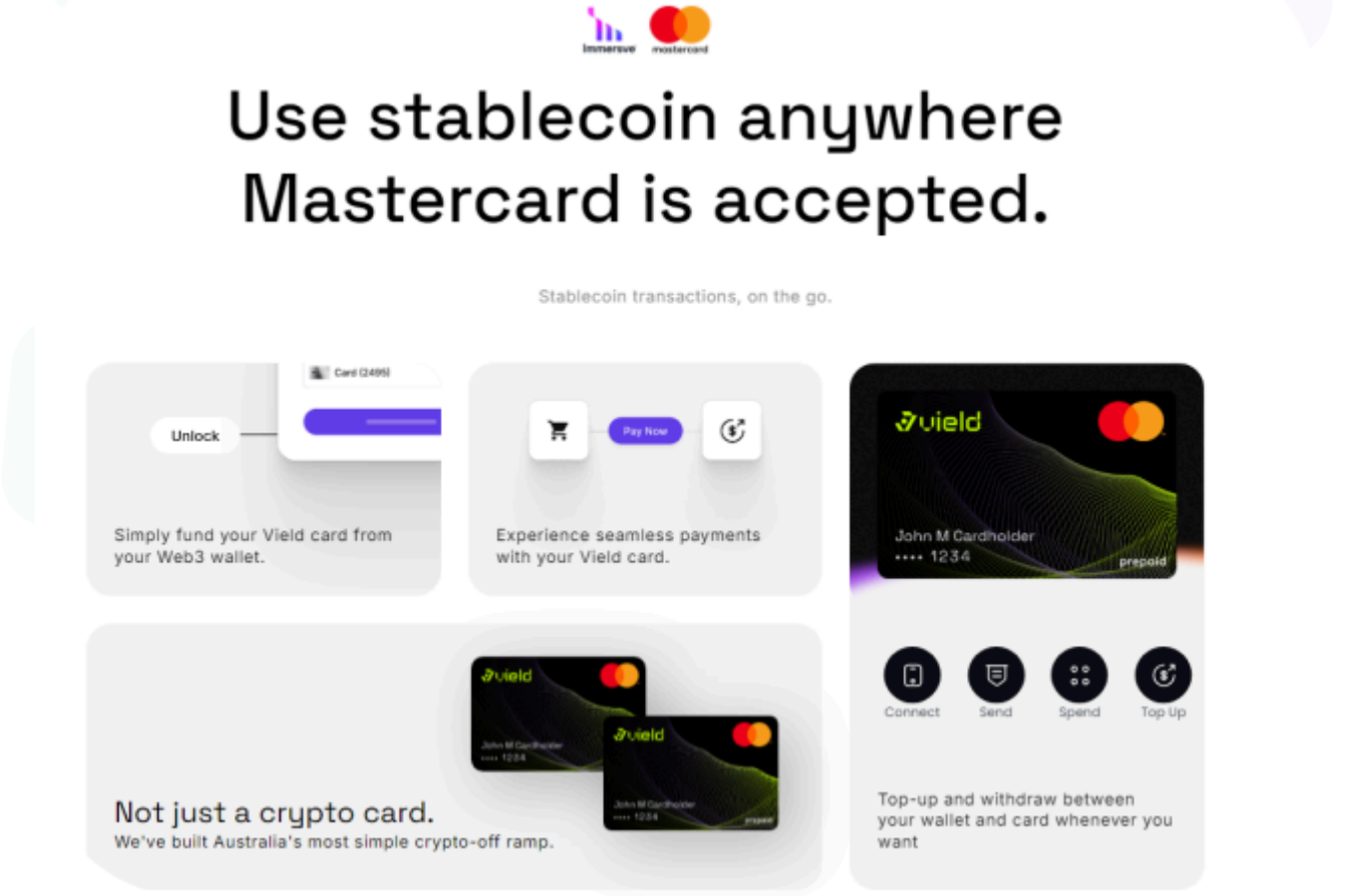
Project Name: Vield
Compiler Version: ^0.8.42
Website: www.vield.io
Logo:

Visualised Context:



Project Visuals:





Penetration Test scope

At Hashlock, we executed a comprehensive penetration test on the Vield project. Our scope included a detailed security assessment, where we rigorously examined the code to identify vulnerabilities and assess overall efficiency. The testing process involved a meticulous manual review, supplemented by advanced software-assisted techniques, to ensure thorough analysis and accuracy.

Description	**Vield Project code base**
Platform	**Typescript**
Penetration Test Date	**October, 2024**
Repository	https://github.com/Vield-Crypto/borrow-app-web
GitHub Commit Hash	6b871dcede9050a88d06ae78d506479ee8eb84a8

Security Rating

After Hashlock's Penetration Test, we found the code base to be **"Secure"**. The code follows simple logic, with correct and detailed ordering.

The 'Hashlocked' rating is reserved for projects that ensure ongoing security via bug bounty programs or on-chain monitoring technology.

All issues uncovered during automated and manual analysis were meticulously reviewed and applicable vulnerabilities are presented in the Penetration Test Findings section.

All vulnerabilities initially identified have now been resolved and acknowledged.

Hashlock found:

2 Medium-severity vulnerabilities

1 Low-severity vulnerabilities

Caution: *Hashlock's Penetration Tests do not guarantee a project's success or ethics, and are not liable or responsible for security. Always conduct independent research about any project before interacting.*

Code Quality

This Penetration Test scope involves the code bases of the Vield project, as outlined in the Penetration Test Scope section. All contracts, libraries, and interfaces mostly follow standard best practices to help avoid unnecessary complexity that increases the likelihood of exploitation, however, some refactoring was required.

The code is very well commented on and closely follows best practice nat-spec styling. All comments are correctly aligned with code functionality.

Penetration Test Resources

We were given the Vield projects code base code in the form of GitHub access.

As mentioned above, code parts are well-commented. The logic is straightforward, and therefore it is easy to quickly comprehend the programming flow as well as the complex code logic. The comments help understand the overall architecture of the protocol.

Dependencies

Per our observation, the libraries used in this code base's infrastructure are based on well-known industry-standard open-source projects.
Apart from libraries, its functions are used in external code base calls.

Severity Definitions

Significance	Description
High	High-severity vulnerabilities can result in loss of funds, asset loss, access denial, and other critical issues that will result in the direct loss of funds and control by the owners and community.
Medium	Medium-level difficulties should be solved before deployment, but won't result in loss of funds.
Low	Low-level vulnerabilities are areas that lack best practices that may cause small complications in the future.
Gas	Gas Optimisations, issues, and inefficiencies

Penetration Test Findings

Medium

[M-01] Backend API - 2FA Information is Public, which can assist attackers to target users who haven't enabled it

Description

The API endpoint for checking 2FA information returns the 2FA status for any user.

Vulnerability Details

A request to `/user/check/{email}/tfa` will return the 2FA information for other users:

```
GET /user/check/loan_johnny-vield_io/tfa HTTP/2

Host: api-staging.vield.io

Accept: application/json, text/plain, */*

X-Vield-Id: Ym9ycm93

X-Vield-Signature: <...>

Content-Type: application/x-www-form-urlencoded

Content-Length: 0

Accept-Encoding: gzip, deflate, br

User-Agent: okhttp/4.9.2



HTTP/2 200 OK

Date: Wed, 16 Oct 2024 22:27:08 GMT

Content-Type: application/json; charset=utf-8

Content-Length: 38
```

```
X-Amzn-Requestid: f7f61249-88c8-48de-91f4-8d03f78946f7

Access-Control-Allow-Origin: *

X-Amzn-Remapped-Content-Length: 38

X-Amz-Apigw-Id: fw6VcEl4SwMEiOQ=

Etag: W/"26-ek2KEdA90LwHe0ijrePI19kIgAA"

X-Powered-By: Express

X-Amzn-Trace-Id:
Root=1-67103dbc-18230182228c79ef5b46d7a8;Parent=629dd2af992ecde9;Sampled=0;Lineage=1:b4
79d74d:0



{"tfaEnabled":true,"totpEnabled":true}
```

Impact

2FA information being public can help attackers focus their efforts on users who have yet to enable it.

Recommendation

2FA information should be kept private.

Status

Resolved

[M-02] Backend API - Can upload non-allowed file types allowing attackers to infect the computers of operators

Description

The API endpoints for uploading files do not have a whitelist for allowed file types.

Vulnerability Details

A request to `/fiat/documents` that is uploading an executable file (file redacted for brevity):

```
POST /fiat/documents HTTP/2

Host: api-staging.vield.io

Accept: application/json, text/plain, */*

X-Vield-Id: Ym9ycm93

X-Vield-Signature:
eyJhbGciOiJSUzI1NiIsImNhdCI6ImNsX0I3ZDRQRDIyMkFBQSIsImtpZCI6Imluc18yWVB3anZRQk1BMVZoRm5
XVHZDUXo5WHZlWDYiLCJ0eXAiOiJKV1QifQ.eyJhcHBfbWV0YWRhdGEiOnt9LCJhdWQiOiJhdXRoZW50aWNhdGV
kIiwiZW1haWwiOm51bGwsImV4cCI6MTcyOTExNjExOSwiaWF0IjoxNzI5MTE1NTE5LCJpZCI6InVzZXJfMmxkQz
FnT0FCYU5tdTd5Z2YwVDNsNzF3dWl2IiwiaXNzIjoiaHR0cHM6Ly9lbmFibGVkLWFzcC00Mi5jbGVyay5hY2Nvd
W50cy5kZXYiLCJqdGkiOiIyNDEwNjQ5MGZmNjk0YTZjM2E1OSIsIm5iZiI6MTcyOTExNTUxNCwicm9sZSI6ImF1
dGhlbnRpY2F0ZWQiLCJzdWIiOiJ1c2VyXzJsZEMxZ09BQmFObXU3eWdmMFQzbDcxd3VpdiIsInVzZXJfbWV0YWR
hdGEiOnt9fQ.ay4OX1E2Fp_gDHoMHZgdmXLA3tNRb8wihhpxg9W6WCR9cad2sC939xMZSJT4uOMLCuCuwseWFZp
stJT-z21QjbOl8tEH-LrhxSzt4_iYZU0Hvcb-zM_14kZSTfT2pVgx1f44xV0Hklj0rxLPv_ZzeJkSYlsXEoR4QA
3ApdlDZ10D21WKOlmTAc3GftFITDFJalvRN3LbQ7pzOEENrhOny_s852lqEnV4K9qzjNGdkJBwghyYS5CZAnNEC
ERxr5FUWVqOt1pIlAyoAMjT9JsMIuPWqQ_HsXi-ZJaIYym8W7g0yAN_amPz_ct3oV_iRMd8-bBP4ZBLZWu0k_tn
ZvOXlw


Content-Type: application/x-www-form-urlencoded

Content-Length: 89931

Accept-Encoding: gzip, deflate, br

User-Agent: okhttp/4.9.2
```

```
files%5B0%5D%5Bcontent%5D=<....>&files%5B0%5D%5BmimeType%5D=application%2Foctet-stream&
files%5B0%5D%5Bname%5D=bad.exe
```

The uploaded file list:

```
HTTP/2 200 OK

Date: Wed, 16 Oct 2024 21:53:10 GMT

Content-Type: application/json; charset=utf-8

Content-Length: 418

X-Amzn-Requestid: d83efafb-86b4-47dc-aea3-eb8676a12fac

Access-Control-Allow-Origin: *

X-Amzn-Remapped-Content-Length: 418

X-Amz-Apigw-Id: fw1XAHNRSwMEMhw=

Etag: W/"1a2-97zVXIj6oqS6X/LzB87t8osr074"

X-Powered-By: Express

X-Amzn-Trace-Id:
Root=1-671035c6-15d1f92a2222a5290c5d483f;Parent=538a31ca8401238f;Sampled=0;Lineage=1:f2
a79da5:0


[{"_id":"66fd3afe5908cf3a451103f2","name":"Vield_Transaction_Report_1723186528442.pdf"}
,{"_id":"66fe652a2718ecc4d0051c8c","name":"hpDoItYourselfFitnessProgram             (1)
(1).pdf"},{"_id":"66fe65672718ecc4d0051c97","name":"test.html"},{"_id":"66fe65712718ecc
4d0051c9b","name":"test.html"},{"_id":"6710358c19cd212c2c97b930","name":"hpDoItYourself
FitnessProgram (1) (1).pdf"},{"_id":"671035c019cd212c2c97b937","name":"bad.exe"}]
```

Impact

Allowing arbitrary file types can be dangerous as it can allow attackers to infect the computers of human operators that inspect these files. It might also allow attackers to disseminate malware if the direct links to files are discovered.

Recommendation

File upload functionality throughout the app should have a white list of file extensions and mime types.

Status

Resolved

Low

[L-01] Clerk API - Users can sign up using non-Australian Phone Numbers

Description

Currently, the UI for the mobile app and the website does not allow non-Australian numbers to sign up. However, it is possible to send non-Australian numbers to the API.

Impact

This discrepancy might result in unexpected behavior or unexpected charges when dealing with users.

Recommendation

The number country code should be verified on the server side.

Status

Resolved

Centralisation

The Vield project values security and utility over decentralisation.

The owner executable functions within the protocol increase security and functionality but depend highly on internal team responsibility.

Conclusion

After Hashlocks analysis, the Vield project seems to have a sound and well-tested code base, now that our vulnerability findings have been resolved and acknowledged. Overall, most of the code is correctly ordered and follows industry best practices. The code is well commented on as well. To the best of our ability, Hashlock is not able to identify any further vulnerabilities.

Our Methodology

Hashlock strives to maintain a transparent working process and to make our Penetration Tests a collaborative effort. The objective of our security Penetration Tests is to improve the quality of systems and upcoming projects we review and to aim for sufficient remediation to help protect users and project leaders. Below is the methodology we use in our security Penetration Test process.

Manual Code Review:

In manually analysing all of the code, we seek to find any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future Penetration Tests. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our methodologies include manual code analysis, user interface interaction, and white box penetration testing. We consider the project's website, specifications, and whitepaper (if available) to attain a high-level understanding of what functionality the code base under review contains. We then communicate with the developers and founders to gain insight into their vision for the project. We install and deploy the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other Penetration Test results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We undergo a robust, transparent process for analysing potential security vulnerabilities and seeing them through to successful remediation. When a potential issue is discovered, we immediately create an issue entry for it in this document, even though we still need to verify the feasibility and impact of the issue. This process is vast because we document our suspicions early even if they are later shown not to represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, and then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this, we analyse the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take and finally, we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinised by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the contract details are made public.

Disclaimers

Hashlock's Disclaimer

Hashlock's team has analysed the code bases in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in the code base source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

Due to the fact that the total number of test cases is unlimited, the Penetration Test makes no statements or warranties on the security of the code. It also cannot be considered a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this code base.

Hashlock is not responsible for the safety of any funds and is not in any way liable for the security of the project.

Technical Disclaimer

Code is deployed and executed on a platform. The platform, its programming language, and other software related to the code base can have their own vulnerabilities that can lead to attacks. Thus, the Penetration Test can't guarantee the explicit security of the Penetration Tested code bases.

About Hashlock

Hashlock is an Australian-based company aiming to help facilitate the successful widespread adoption of distributed ledger technology. Our key services all have a focus on security, as well as projects that focus on streamlined adoption in the business sector.

Hashlock is excited to continue to grow its partnerships with developers and other web3-oriented companies to collaborate on secure innovation, helping businesses and decentralised entities alike.

Website: hashlock.com.au
Contact: info@hashlock.com.au