Secure Code Review for Choice Coin

Findings and Recommendations Report Presented to:

Fortior Blockchain, LLP

February 25, 2022

Version: 2.0

Presented by:

Kudelski Security – Nagravision Sàrl
Route de Genève, 22-24
1033 Cheseaux sur Lausanne
Switzerland

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

Overview

Fortior Blockchain, LLP engaged Kudelski Security to perform a Secure Code Review for Choice Coin.

The assessment was conducted remotely by the Kudelski Security Team. Testing took place on January 20 – February 02, 2022, and focused on the following objectives:

- Provide the customer with an assessment of their overall security posture and any risks discovered within the environment during the engagement.
- To provide a professional opinion on the security measures' maturity, adequacy, and efficiency.
- To identify potential issues and include improvement recommendations based on the result of our tests.

This report summarizes the engagement, tests performed, and findings. It also contains detailed descriptions of the discovered vulnerabilities, steps the Kudelski Security Teams took to identify and validate each issue, as well as any applicable recommendations for remediation.

A re-review was conducted on February 23, 2022.

Key Findings

The following are the major themes and issues identified during the testing period. These, along with other items, within the findings section, should be prioritized for remediation to reduce the risk they pose

- KS-CHOICECOIN-01 – Vulnerable versions of third-party dependencies are used

During the test, the following positive observations were noted regarding the scope of the engagement:

- The team was very supportive and open to discussing the design choices made

Based on the account relationship graphs or reference graphs and the formal verification, we can conclude that the reviewed code implements the documented functionality.

At the re-review, all issues were remediated,

Scope and Rules of Engagement

Kudelski performed a Secure Code Review for Choice Coin. The following table documents the targets in scope for the engagement. No other systems or resources were in scope for this assessment.

The source code was supplied through a public repository with the commit hash c063800e56820204ca5b30042534e5ec111b2727 https://github.com/ChoiceCoin/Voting_DApp.

Files included in the code review

```
choicecoin/
├── backend/
│   ├── Procfile
│   ├── index.js
│   ├── package.json
│   └── yarn.lock
├── frontend/
│   ├── public/
│   │   ├── img/
│   │   │   ├── background.svg
│   │   │   └── background2.svg
│   │   ├── _redirects
│   │   ├── choice.png
│   │   ├── favicon.ico
│   │   ├── index.html
│   │   ├── logo192.png
│   │   ├── logo512.png
│   │   ├── manifest.json
│   │   └── robots.txt
│   ├── src/
│   │   ├── components/
│   │   │   ├── ScrollText.js
│   │   │   └── ScrollTextLand.js
│   │   ├── faq/
│   │   │   └── index.js
│   │   ├── statics/
│   │   │   ├── AlertModal.js
│   │   │   ├── BottomNavigationBar.js
│   │   │   ├── ScrollText.js
│   │   │   └── TopNavigationBar.js
│   │   ├── store/
│   │   │   ├── reducers.js
│   │   │   └── stores.js
│   │   ├── styles/
│   │   │   ├── electionlist.css
│    │    │    ├── faq.css
│    │    │    ├── landing.css
│    │    │    └── reset.css
│    │    ├── App.js
│    │    ├── ElectionList.js
│    │    ├── GetCommittedAmount.js
│    │    ├── Home.js
│    │    ├── Landing.js
│    │    ├── MainPage.js
│    │    ├── constants.js
│    │    ├── index.css
│    │    └── index.js
│    ├── package-lock.json
│    ├── package.json
│    └── yarn.lock
├── mainnet/
│    ├── ElectionList.js
│    └── constants.js
├── rewards/
│    └── rewards.py
├── License.txt
├── README.md
└── run.sh
```

Table 1: Scope

TECHNICAL ANALYSIS & FINDINGS

During the Secure Code Review for Choice Coin, we discovered:

- 1 finding with a LOW severity rating.

The following chart displays the findings by severity.

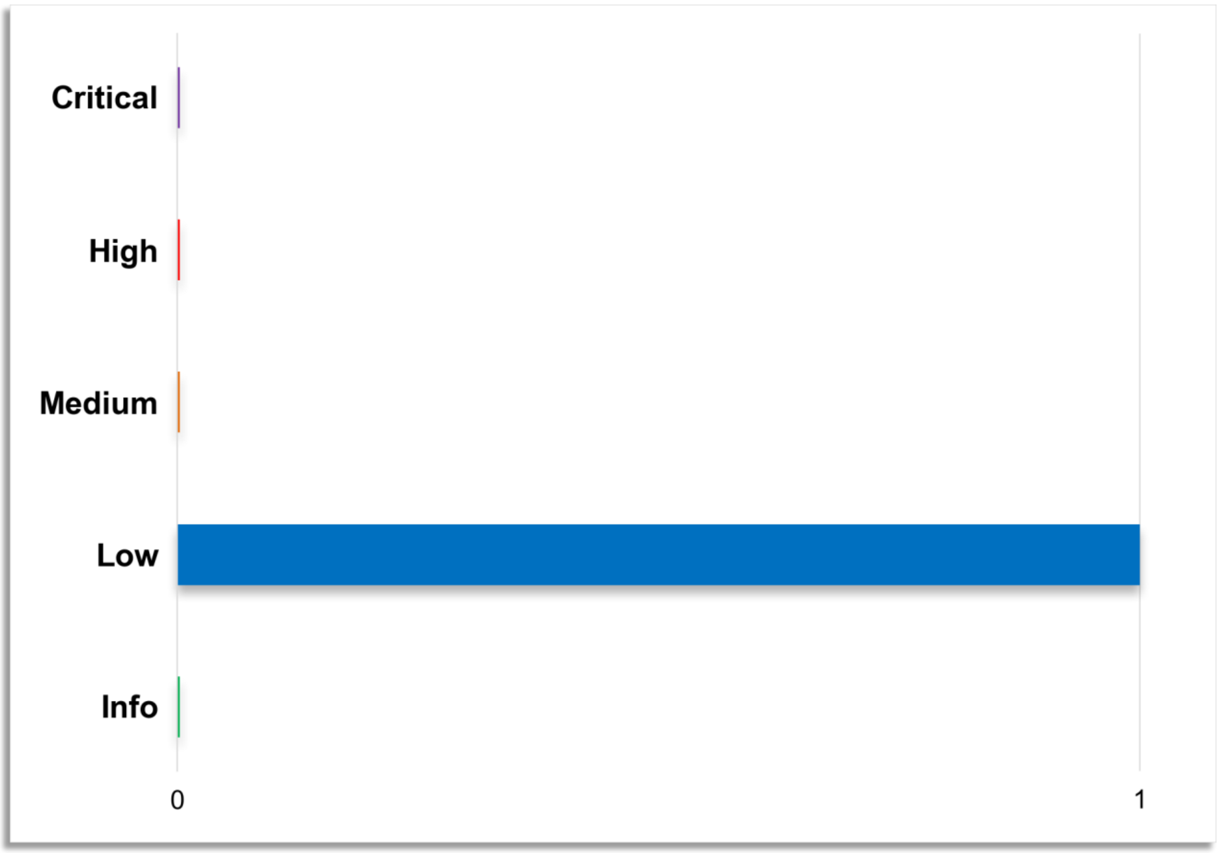

Figure 1: Findings by Severity

Findings

The *Findings* section provides detailed information on each finding, including discovery methods, explanation of severity determination, recommendations, and applicable references.

The following table provides an overview of the findings.

#	Severity	Description
KS-CHOICECOIN-01	**Low**	Vulnerable versions of third-party dependencies are used

Table 2: Findings Overview

Technical analysis

Based on formal verification, we can conclude that the code implements the documented functionality to the extent of the code reviewed.

Technical Findings

General Observations

ChoiceCoin DAP is the Decentralized Decisions software that allows for votes to be held on the Choice Coin DAO issues, enabling a new form of direct and decentralized democratic decision-making. The software leverages the Algorand blockchain to provide a new, scalable, secure voting technology. It is implemented in JavaScript. The application currently looks like a test as it contains hardcoded outdated voting forms and hardcoded links to the test Algorand network.

Summary of strengths

- The code is self-explanatory. The naming policy makes instructions understandable
- There are used secure mechanisms of the wallets and private key management

Summary of discovered vulnerabilities

- During the assessment, issues were found in third-party dependencies. There are not any found ways to exploit them.
- The code does not contain any tests of the code

Re-review comments

During the re-review, we acknowledge the addition of tests to confirm the correctness of the application. We see this issue as being remediated.

Vulnerable versions of third-party dependencies are used

Finding ID: KS-CHOICECOIN-01
Severity: **Low**
Status: **Remediated**

Description

There are 4 vulnerable versions of third-party dependencies used.

Vulnerability	Description	Reference	Impact
Prototype Pollution in immer	immer is vulnerable to Improperly Controlled Modification of Object Prototype Attributes ('Prototype Pollution')	CVE-2021-3757	Critical
Exposure of sensitive information in follow-redirects	follow-redirects are vulnerable to Exposure of Private Personal Information to an Unauthorized Actor	CVE-2022-0155	High
Uncontrolled Resource Consumption in ANSI-HTML	This affects all versions of package ansi-html. If an attacker provides a malicious string, it will get stuck processing the input for an extremely long time.	CVE-2021-23424	High
Regular expression denial of service	This affects the package glob-parent before 5.1.2. The enclosure regex used to check for strings ending in an enclosure containing a path separator.	CVE-2020-28469	High

Proof of issues

Filename: package-lock.json
Line number: 42397

```
"react-dev-utils": {
      "version": "11.0.4",
      "resolved": "https://registry.npmjs.org/react-dev-utils/-/react-dev-
utils-11.0.4.tgz",
      "integrity": "sha512-
dx0LvIGHcOPtKbeiSUM4jqpBl3TcY7CDjZdfOIcKeznE7BWr9dg0iPG90G5yfVQ+p/rGNMXdbfStv
zQZEVEi4A==",
      "requires": {
        "@babel/code-frame": "7.10.4",
        "address": "1.1.2",
        "browserslist": "4.14.2",
        "chalk": "2.4.2",
        "cross-spawn": "7.0.3",
        "detect-port-alt": "1.1.6",
        "escape-string-regexp": "2.0.0",
        "filesize": "6.1.0",
        "find-up": "4.1.0",
        "fork-ts-checker-webpack-plugin": "4.1.6",
        "global-modules": "2.0.0",
        "globby": "11.0.1",
        "gzip-size": "5.1.1",
        "immer": "8.0.1",
        "is-root": "2.1.0",
        "loader-utils": "2.0.0",
        "open": "^7.0.2",
        "pkg-up": "3.1.0",
        "prompts": "2.4.0",
        "react-error-overlay": "^6.0.9",
        "recursive-readdir": "2.2.2",
        "shell-quote": "1.7.2",
        "strip-ansi": "6.0.0",
        "text-table": "0.2.0"
        }
}
```

Filename: package-lock.json
Line number: 34804

```
"follow-redirects": {
      "version": "1.14.6",
      "resolved": "https://registry.npmjs.org/follow-redirects/-/follow-
redirects-1.14.6.tgz",
      "integrity": "sha512-
fhUl5EwSJbbl8AR+uYL2KQDxLkdSjZGR36xy46AO7cOMTrCMON6Sa28FmAnC2tRTDbd/Uuzz3aJBv
7EBN7JH8A=="
}
```

Filename: package-lock.json
Line number: 30495

```
"ansi-html": {
      "version": "0.0.7",
      "resolved": "https://registry.npmjs.org/ansi-html/-/ansi-html-
0.0.7.tgz",
      "integrity": "sha1-gTWEAhliqenm/QOflA0S9WynhZ4="
}
```

Filename: package-lock.json
Line number: 46359

```
"chokidar": {
          "version": "2.1.8",
          "resolved": "https://registry.npmjs.org/chokidar/-/chokidar-
2.1.8.tgz",
          "integrity": "sha512-
ZmZUazfOzf0Nve7duiCKD23PFSCs4JPoYyccjUFF3aQkQadqBhfzhjkwBH2mNOG9cTBwhamM37EIs
IkZw3nRgg==",
          "optional": true,
          "requires": {
            "anymatch": "^2.0.0",
            "async-each": "^1.0.1",
            "braces": "^2.3.2",
            "fsevents": "^1.2.7",
            "glob-parent": "^3.1.0",
            "inherits": "^2.0.3",
            "is-binary-path": "^1.0.0",
            "is-glob": "^4.0.0",
            "normalize-path": "^3.0.0",
            "path-is-absolute": "^1.0.0",
            "readdirp": "^2.2.1",
            "upath": "^1.1.1"
        }
}
```

Severity and Impact summary

These issues can lead to DoS attacks and Exposure of Private Personal Information. It is no found way to exploit these vulnerabilities, but any changes in the code could make them dangerous.

Details of the impact are here:

- Exposure of Private Personal Information to an Unauthorized Actor in follow-redirects/follow-redirects
- Regular Expression Denial of Service (ReDoS)

Recommendation

It is recommended to update the react-scripts package by calling:

```
npm install react-scripts@5.0.0
npm update follow-redirects --depth 5
```

References

- CVE-2020-28469: Regular expression denial of service
- CVE-2021-3757: Prototype Pollution in immer
- CVE-2021-23424: Uncontrolled Resource Consumption in ANSI-html
- CVE-2022-0155: Exposure of sensitive information in follow-redirects

Re-review comments

The remediation to the finding has been added as an essential update to the Decentralized Decisions repository on GitHub, which in essence solves the issue.

METHODOLOGY

Kudelski Security uses the following high-level methodology when approaching engagements. They are broken up into the following phases.

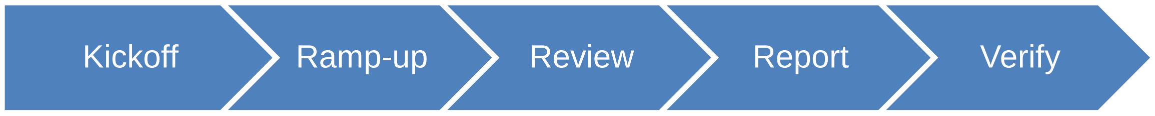

Figure 2: Methodology Flow

Kickoff

The project is kicked off as the sales process has concluded. We typically set up a kickoff meeting where project stakeholders are gathered to discuss the project and the responsibilities of participants. We verified the engagement's scope during this meeting and discussed the project activities. It's an opportunity for both sides to ask questions and get to know each other. By the end of the kickoff, there is an understanding of the following:

- Designated points of contact
- Communication methods and frequency
- Shared documentation
- Code and/or any other artifacts necessary for project success
- Follow-up meeting schedule, such as a technical walkthrough
- Understanding of timeline and duration

Ramp-up

Ramp-up consists of the activities necessary to gain proficiency on a particular project. This can include the steps needed for familiarity with the codebase or technological innovation utilized. This may include, but is not limited to:

- Reviewing previous work in the area, including academic papers
- Reviewing programming language constructs for specific languages
- Researching common flaws and recent technological advancements

Review

The review phase is where most of the work on the engagement is completed. This is the phase where we analyze the project for flaws and issues that impact the security posture. Depending on the project, this may include an architecture analysis, a code review, and a specification matching to match the architecture to the implemented code.

In this code audit, we performed the following tasks:

1. Security analysis and architecture review of the original protocol
2. Review of the code written for the project
3. Compliance with the code with the provided technical documentation

The review for this project was performed using manual methods and utilizing the reviewer's experience. No dynamic testing was performed. Only custom-built scripts and tools were used to assist the reviewer during the testing. We discuss our methodology in more detail in the following sections.

Code Safety

We analyzed the provided code, checking for issues related to the following categories:

- General code safety and susceptibility to known issues
- Poor coding practices and unsafe behavior
- Leakage of secrets or other sensitive data through memory mismanagement
- Susceptibility to misuse and system errors
- Error management and logging

This list is general and not comprehensive, meant only to understand the issues we are looking for.

Technical Specification Matching

We analyzed the provided documentation and checked that the code matches the specification. We checked for things such as:

- Proper implementation of the documented protocol phases
- Proper error handling
- Adherence to the protocol logical description

Reporting

Kudelski Security delivers a PDF draft report containing an executive summary, technical details, and observations about the project.

The executive summary contains an overview of the engagement, including the number of findings and a statement about our general risk assessment of the project. We may conclude that the overall risk is low, but depending on what was assessed, we may conclude that more scrutiny of the project is needed.

We not only report security issues identified but also informational findings for improvement categorized into several buckets:

- Critical
- High
- Medium
- Low
- Informational

The technical details are aimed more at developers, describing the issues, the severity ranking, and recommendations for mitigation.

As we perform the audit, we may identify issues that aren't security-related but are general best practices and steps that can be taken to lower the attack surface of the project. We will call those out as we encounter them and as time permits.

As an optional step, we can create a public report that can be shared and distributed to a larger audience.

Verify

After the preliminary findings have been delivered, this could be in the form of the approved communication channel or the delivery of the draft report. We will verify any fixes within a window of time specified in the project. After the fixes have been verified, we will change the finding status in the report from open to remediate.

The output of this phase will be a final report with any mitigated findings noted.

Additional Note

It is important to note that, although we did our best in our analysis, no code audit or assessment is a guarantee of the absence of flaws. Our effort was constrained by resource and time limits and the agreement's scope.

While assessing the severity of the findings, we considered the impact, ease of exploitability, and the probability of attack. This is a solid baseline for severity determination.

The Classification of identified problems and vulnerabilities

There are four severity levels of an identified security vulnerability.

Critical – a vulnerability that will lead to loss of protected assets

- This is a vulnerability that would lead to immediate loss of protected assets

- The complexity to exploit is low
- The probability of exploit is high

High - A vulnerability that can lead to loss of protected assets

- All discrepancies found where there is a security claim made in the documentation that can not be found in the code
- All mismatches from the stated and actual functionality
- Unprotected key material
- Weak encryption of keys
- Badly generated key materials
- Tx signatures not verified
- Spending of funds through logic errors
- Calculation errors overflow and underflows

Medium - a vulnerability that hampers the uptime of the system or can lead to other problems

- Insecure calls to third party libraries
- Use of untested or nonstandard, or non-peer-revied crypto functions
- Program crashes leaves core dumps or writes sensitive data to log files

Low - Problems that have a security impact but does not directly impact the protected assets

- Overly complex functions
- Unchecked return values from 3rd party libraries that could alter the execution flow

Informational

- General recommendations

KUDELSKI SECURITY CONTACTS

NAME	POSITION	CONTACT INFORMATION
Ramsey El-Khazen	**Blockchain Security**	**Ramsey.el-khazen@kudelskisecurity.com**