Security Review for HumbleSwap Smart Contract

Findings and Recommendations Report Presented to:

Reach

February 21, 2022

Version: 1.0

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

For Public Release

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

Overview

Reach engaged Kudelski Security to perform a Security Review for the HumbleSwap Smart Contract.

The assessment looked at the REACH code and the corresponding TEAL code. The REACH compiler itself was not in scope for the work performed.

The assessment was conducted remotely by the Kudelski Security Team. Testing took place on January 24 - February 11, 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.

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-REACH-01 – Non-Network Tokens

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

We conclude that the reviewed code implements the documented functionality based on formal verification.

Scope and Rules of Engagement

Kudelski performed a Security Review for HumbleSwap Smart Contract. 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 with the commit hash through a private repository at https://github.com/reach-sh/duoswap-core fa5114f54075a3a822f5c064c29c49ced8d6b0a1.

Files included in the code review

reach-rs-duoswap-core/
└── build/
    ├── index.main.appApproval.teal
    ├── index.main.mjs
    ├── n2nn.main.appApproval.teal
    └── n2nn.main.mjs

Table 1: Scope

TECHNICAL ANALYSIS & FINDINGS

During the Security Review for HumbleSwap Smart Contract, we discovered:

- 1 finding with an INFORMATIONAL severity rating.

The following chart displays the findings by severity.

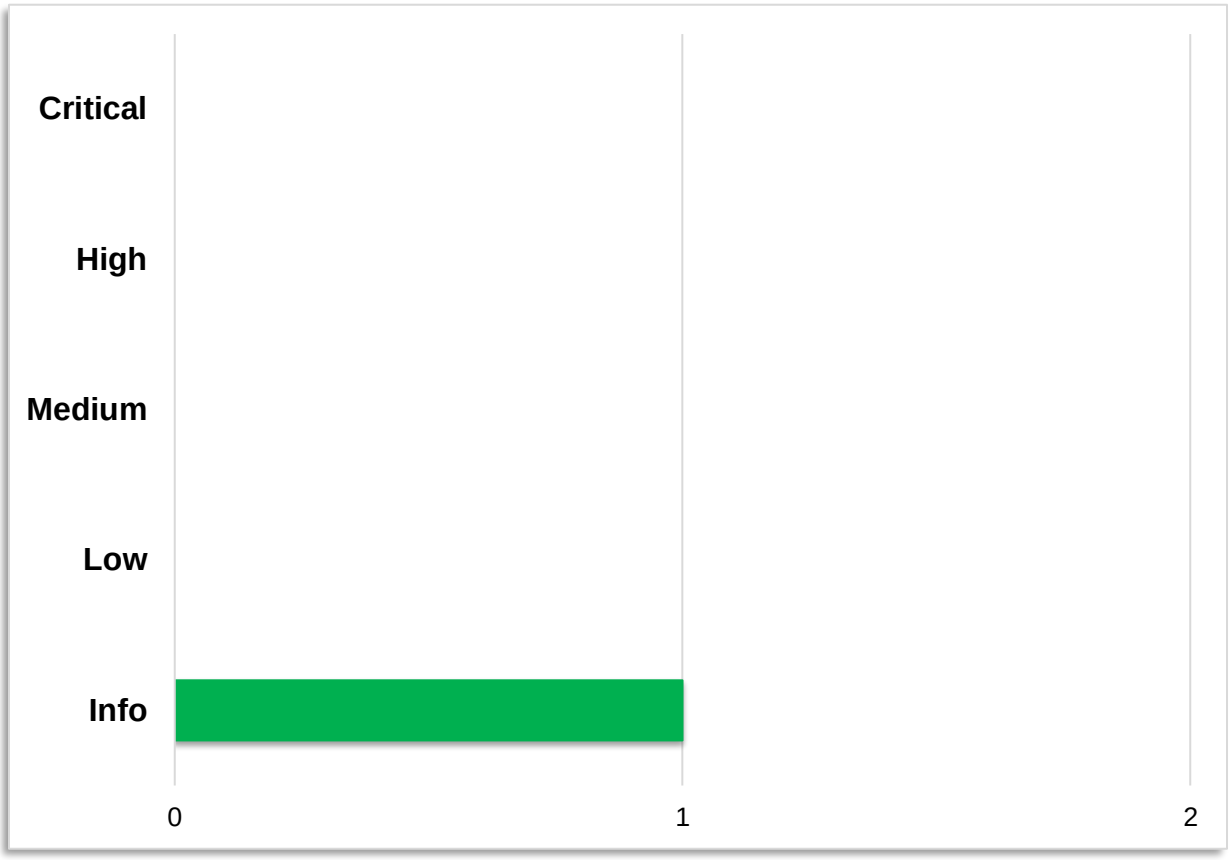

Figure 1: Findings by Severity

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-REACH-01	**Informational**	Non-Network Tokens

Table 2: Findings Overview

Technical Analysis

The source code has been manually validated to the extent that the state of the repository allowed. The validation includes confirming that the code correctly implements the intended functionality.

Further investigations were made, which concluded that they did not pose a risk to the application. They were:

- No potential panics were detected
- No potential errors regarding wraps/unwrap, expect, and wildcards
- No internal unintentional unsafe references

Conclusion

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

Technical Findings

General Observations

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

REACH

- Uses verifyArthimatic to prevent int overflow.
- Uses declassification to make the local information public.
- Uses Assume feature for local steps correctly.
- Uses Assert and Require functions correctly when making checks.
- Uses RequireIfVerify properly.
- Uses .only feature when needed.
- Uses Commit() and exit() correctly.
- Uses MulDiv when handling multiplication and division.

TEAL

- Sets Pragma version
- Handles assertions correctly
- The general flow of application states are correct
- Checks for int overflows
- Handles group transactions correctly
- Checks transactions in some fields (AssetAmount, AssetReceiver, etc.)
- Checks AssetAmount
- Uses Type and TypeEnum to check if something is a payment or asset
- Multiplies integers before division, reducing the chance of over-frequent 0 values

The Reach code is well structured, with proper variable and function names. However, the compiled Teal code does not contain the same design structures recommended by Algorand. Both sets of code also lack many comments to document what it is doing or its intention.

Non-Network Tokens

Finding ID: KS-REACH-01
Severity: **Informational**
Status: **Open**

Description

Non-Network tokens (NNTs) can have different behaviors from network tokens in various ways.

Severity and Impact Summary

NNTs might have the option to freeze all future transfers. NNTs must also be pre-approved before the transfer, meaning the receiver must also agree to the transfer beforehand. Tokens minted in Reach are compiled to Algorand Standard Assets (ASAs), with extra functionality disabled (freezing, clawback, reserves, and separate managers), and they will behave as closely to network tokens as possible. On Algorand, NNTs are also built into the network and generally have predictable and stable behavior.

Recommendation

While this does not currently seem to be a problem, it is a difference to be mindful of, especially if the organization decides to deploy to other blockchains in the future or if using non-Reach minted NNTs.

References

- Reach: How do network and non-network tokens differ?
- Algorand Developer Portal: Algorand Standard Assets (ASAs)

METHODOLOGY

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

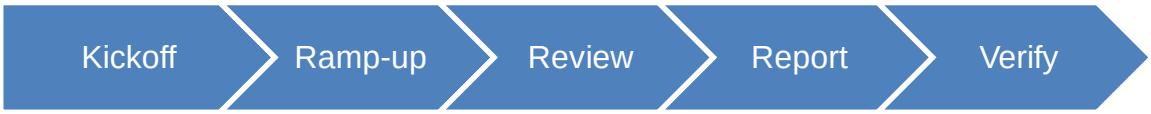

Figure 2: Methodology Flow

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. During this meeting, we verified the scope of the engagement 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