URBIT STARDUST SMART CONTRACT AUDIT

July 19, 2021

MixBytes()

CONTENTS

1. INTRODUCTION

1.1 DISCLAIMER

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of Urbit. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

1.2 PROJECT OVERVIEW

Azimuth is a general-purpose PKI ("public key infrastructure") that Urbit uses as an identity system. This system is implemented as a suite of smart contracts on the Ethereum blockchain, and it determines which Ethereum addresses own which Urbit planets, stars, or galaxies. In Arvo, a single identity is called a "ship," whereas in Azimuth, a single identity is called a "point."

1.3 SECURITY ASSESSMENT METHODOLOGY

A group of auditors are involved in the work on the audit who check the provided source code independently of each other in accordance with the methodology described below:

01 Project architecture review:
> Reviewing project documentation
> General code review
> Reverse research and study of the architecture of the code based on the source code only
> Mockup prototyping

Stage goal:
Building an independent view of the project's architecture and identifying logical flaws in the code.

02 Checking the code against the checklist of known vulnerabilities:
> Manual code check for vulnerabilities from the company's internal checklist
> The company's checklist is constantly updated based on the analysis of hacks, research and audit of the clients' code
> Checking with static analyzers (i.e Slither, Mythril, etc.)

Stage goal:
Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flashloan attacks, etc.)

03 Checking the code for compliance with the desired security model:
> Detailed study of the project documentation
> Examining contracts tests
> Examining comments in code
> Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit
> Exploits PoC development using Brownie

Stage goal:
Detection of inconsistencies with the desired model

04 Consolidation of interim auditor reports into a general one:
> Cross-check: each auditor reviews the reports of the others
> Discussion of the found issues by the auditors
> Formation of a general (merged) report

Stage goal:
Re-check all the problems for relevance and correctness of the threat level and provide the client with an interim report.

05 Bug fixing & re-check:
> Client fixes or comments on every issue
> Upon completion of the bug fixing, the auditors double-check each fix and set the statuses with a link to the fix

Stage goal:
Preparation of the final code version with all the fixes

06 Preparation of the final audit report and delivery to the customer.

Findings discovered during the audit are classified as follows:

FINDINGS SEVERITY BREAKDOWN

Level	Description	Required action
Critical	Bugs leading to assets theft, fund access locking, or any other loss funds to be transferred to any party	Immediate action to fix issue
Major	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.	Implement fix as soon as possible
Warning	Bugs that can break the intended contract logic or expose it to DoS attacks	Take into consideration and implement fix in certain period
Comment	Other issues and recommendations reported to/acknowledged by the team	Take into consideration

Based on the feedback received from the Customer's team regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The project team is aware of this finding. Recommendations for this finding are planned to be resolved in the future. This finding does not affect the overall safety of the project.
No issue	Finding does not affect the overall safety of the project and does not violate the logic of its work.

1.4 EXECUTIVE SUMMARY

The audited scope implements the smart contracts which deposit/reedem user azimuth stars for star tokens.

1.5 PROJECT DASHBOARD

Client	Urbit
Initial version	c446b1f12f53fa75ea6c347daee1e15df562a81d
Final version	-
Date	July 05, 2021 - July 19, 2021
Auditors engaged	2 auditors

FILES LISTING

Treasury.sol	https://github.com/ransonhobbes/stardust/blob/c446b1f12f53fa75ea6c347daee1e15df562a81d/contracts/Treasury.sol
StarToken.sol	https://github.com/ransonhobbes/stardust/blob/c446b1f12f53fa75ea6c347daee1e15df562a81d/contracts/StarToken.sol
AzimuthWrapper.sol	https://github.com/ransonhobbes/stardust/blob/c446b1f12f53fa75ea6c347daee1e15df562a81d/contracts/wrapper/AzimuthWrapper.sol
ClaimsWrapper.sol	https://github.com/ransonhobbes/stardust/blob/c446b1f12f53fa75ea6c347daee1e15df562a81d/contracts/wrapper/ClaimsWrapper.sol
EclipticWrapper.sol	https://github.com/ransonhobbes/stardust/blob/c446b1f12f53fa75ea6c347daee1e15df562a81d/contracts/wrapper/EclipticWrapper.sol
PollsWrapper.sol	https://github.com/ransonhobbes/stardust/blob/c446b1f12f53fa75ea6c347daee1e15df562a81d/contracts/wrapper/PollsWrapper.sol
IAzimuth.sol	https://github.com/ransonhobbes/stardust/blob/c446b1f12f53fa75ea6c347daee1e15df562a81d/contracts/interface/IAzimuth.sol

IEcliptic.sol	https://github.com/ransonhobbes/stardust/blob/c446b1f12f53fa75ea6c347daee1e15df562a81d/contracts/interface/IEcliptic.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	0
Warning	0
Comment	1

CONCLUSION

Smart contract has been audited and no suspicious places have been spotted. During the audit no critical or major issues were found. One comment was marked and discussed with the client. After working on the reported finding it was acknowledged by the client, as the problem was not critical. So, the contract is assumed as secure to use according to our security criteria.Final commit identifier with all fixes: -

2. FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

Not Found

2.3 WARNING

Not Found

2.4 COMMENT

CMT-1	Use `external` instead of `public` modifier
File	Treasury.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At line:
Treasury.sol#L79
Treasury.sol#L120

RECOMMENDATION

We recommend using `external` instead of `public` modifier for `deposit` and `redeem` functions because they are not called internally.

3. ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build open-source solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, do research and tech consultancy.

BLOCKCHAINS

- Ethereum
- Cosmos
- EOS
- Substrate

TECH STACK

- Python
- Solidity
- Rust
- C++

CONTACTS

https://github.com/mixbytes/audits_public

https://mixbytes.io/

hello@mixbytes.io

https://t.me/MixBytes

https://twitter.com/mixbytes